FAST AND ENERGY-EFFICIENT MOBILITY MANAGEMENT IN MOBILE EDGE COMPUTING NETWORKS

by

Haoxin Wang

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2020

Approved by:

Dr. Jiang (Linda) Xie

Dr. Tao Han

Dr. Yu Wang

Dr. Weichao Wang

ABSTRACT

HAOXIN WANG. Fast and Energy-Efficient Mobility Management in Mobile Edge Computing Networks. (Under the direction of DR. JIANG (LINDA) XIE)

The prevalence of computation-intensive and latency-sensitive mobile applications, such as mobile augmented reality (MAR) and autonomous driving, has an utmost effect on resource-limited mobile clients. Mobile edge computing (MEC) is proposed to be a promising paradigm to bridge the gap between the stringent computation and latency requirements of mobile applications and the constrained computation and battery capacity on mobile clients. The main feature of MEC is to push mobile computing, network control and storage to the network edges (e.g., base stations (BSs) and access points (APs)). However, prior work on MEC fails to achieve their expected performance in multiple practical cases, e.g., irreparable network disruptions caused by wireless link instability or user-mobility that is a critical characteristic of mobile applications.

In this dissertation, fast and energy-efficient mobility management in MEC networks is explored. Link-instability and user-mobility incurred challenges in MEC are addressed from four steps. (1) An intelligent handoff trigger mechanism is designed to achieve a fast and accurate trigger for seamless mobility support in MEC networks. (2) Fast and energy-efficient radio-service handoff protocols are established in order to rebuild offloading services on a new MEC server with low overhead after a handoff is triggered at a mobile client in MEC networks. (3) To minimize performance degradation during mobility caused by radio resource allocation unfairness, single and multiple edge server radio resource allocation protocols to impartially allocate the uplink and the downlink radio resources in MEC networks are proposed. (4) A dynamic configuration adaptation algorithm is proposed for mobile clients to achieve energy-efficient offloading in MEC networks while satisfying variant clients' user preferences.

In summary, this research is essential for providing fast and energy-efficient mobility support for mobile clients in MEC networks. In addition, this research provides critical insights for future designs of mobility management in MEC networks.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the direction of my advisor, advice from my dissertation committee members, assistance from my colleagues, help from my friends, and encouragement from my wife and family.

First and foremost, I wish to express my greatest and deepest gratitude to my advisor, Dr. Jiang (Linda) Xie, for her guidance and supervision throughout these years. The scientific methods and work ethic she taught me made where I am now. I am very honored to be her student. Her instructions will always inspire me to continue improving myself in the future.

I would like to thank my committee members: Dr. Tao Han, Dr. Yu Wang, and Dr. Weichao Wang for their great help and invaluable advice along this dissertation work. In addition, I appreciate the GASP grant from UNCC and Research Assistantships from the National Science Foundation (NSF) and Toyota Motor North America as the financial assistance for this dissertation.

I would also like to thank my colleagues and friends: Xingya Liu, Wahida Nasrin, Moinul Hossain, Siqi Huang and many others. I am so thankful and appreciative to have been able to receive help from them.

Thanks also to my parents and my parents-in-law, who have over the years supported and encouraged me with their best wishes.

In the end, I would like to thank my wife, Dairui Zhang. This dissertation would not have been possible without her support, patience, and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AC Assistant MEC

AE Auto-Exposure

AF Auto-Focus

AIFSN Arbitration Inter-Frame Space Number

AP Access Point

AR Augmented Reality

AWB Auto-White-Balance

BCD Block Coordinate Descent

BS Base Station

CC Color Correction

CNN Convolutional Neural Network

DC Dominating MEC

EE Edge Enhancement

FPS Frame Per Second

IEEE Institute of Electrical and Electronics Engineers

ISP Image Signal Processor

LTE Long-Term Evolution

mAP Mean Average Precision

MAR Mobile AR

MEC Mobile Edge Computing

MINLP Mixed-Integer Non-Linear Programming Problem

NR Noise Reduction

QoE Quality-of-Experience

QoS Quality-of-Service

RBF Radial Basis Function

RSSI Received Signal Strength Indicator

SINR Signal-to-Interference-plus-Noise Ratio

VM Virtual Machine

WAN Wide Area Network

CHAPTER 1: INTRODUCTION

1.1 Background on Mobility Management in MEC Networks

Mobile edge computing (MEC) has emerged as a promising technology to overcome the challenges of executing latency-sensitive and computation-intensive applications at resource-limited mobile devices, by pushing mobile computing, network control, and storage resources to the edge of mobile wireless networks (e.g., base stations (BSs) and access points (APs)) [1].

As compared to the cloud, MEC supports reduced network latency, which enabled a myriad of real-time mobile applications that require low latency and high computation power, including computational offloading, edge video caching, connected vehicles, and smart healthcare.

To achieve the goal of seamless mobility, first, services should experience very low-latency breaks so that mobility is transparent to applications. Second, the application end-to-end Quality-of-Service (QoS)/ Quality-of-Experience (QoE) should be maintained after the mobility to ensure smooth transition and minimum performance degradation. Although the mobility support issue has been extensively investigated in conventional wireless networks, mobility management schemes proposed in these networks consider the communication-only scenario, while MEC networks tie communications together with computing activities. Therefore, under MEC networks, there are some unique issues in the seamless mobility support.

First, in conventional wireless networks, user mobility causes the connectivity change between a user and its attachment point in the wireless networks (i.e., AP or BS). This triggers a radio handoff process in which a mobile user first identifies nearby candidate APs/BSs, and then switches to the best available AP/BS. However,

in MEC networks, a user's computation needs should also be considered in the handoff process when it roams to the service area of a new MEC server that is usually attached to a BS or AP. For instance, a mobile user's offloaded computational tasks at its current MEC server should be migrated to or rebuilt on a new MEC server, after a radio handoff is triggered. This service handoff process is new in MEC networks. The time period, from the time when a mobile user loses its computation service provided by its current MEC server to the time when the mobile user re-achieves the service provided by a new MEC server, is defined as *service rebuilding latency*. Seamless mobility requires a seamless service rebuilding process that should support very low-latency of combined radio handoff and service handoff. All existing mobility solutions consider the radio handoff and service handoff separately. In other words, studies in communications only focus on improving the radio handoff efficiency (e.g., reducing the delay of identifying the best target AP), while studies in computing only consider reducing the service handoff latency (e.g., reducing the data size needs to be migrated). However, this leads to the inefficiency and prolonged delay of the service rebuilding process; for example, the radio and service handoffs are always performed sequentially, instead of in parallel, thus seamless mobility cannot be guaranteed.

Second, the QoS/QoE performance of the mobility management mechanisms designed for conventional wireless networks cannot be guaranteed under MEC networks due to the neglect of the computation related metrics in handoff decisions. For example, in IEEE 802.11, the link quality connecting a mobile user and its associated AP is used as a metric for triggering a handoff and selecting the new target AP after a handoff. However, not considering any computation metric in handoff decisions may lead to poor performance at the MEC server side. For example, a roaming user in MEC networks will select the AP to associate with during a handoff if it obtains a good radio link quality (e.g., high received signal strength indicator (RSSI)), but the computation performance may be weak if the new MEC server has limited computing

resources, causing a high computation delay of the offloaded computing tasks thus, seamless mobility is not supported.

In addition, the QoS of computation services in MEC networks cannot always be guaranteed due to the legacy design of radio resource allocation in existing wireless networks. Traditionally, wireless networks allocate more radio resources to downlink, since downlink traffic volume is usually much higher than uplink. For example, downloading a video from the Internet consumes significantly more bandwidth than uploading a mobile user's GPS information for the navigation purpose. Therefore, downlink acquires higher throughput, peak data rate, and spectral efficiency than uplink. However, in MEC networks, uplink traffic (e.g., a mobile user who is enjoying an edge-based augmented reality (AR) application offloads real-time camera captured video frames to an MEC server) may be massive and very latency-sensitive, thus requiring even higher throughput than the downlink traffic (e.g., the MEC server sends back the computation results to the user). Such new traffic distribution creates unique challenges for seamless mobility support in MEC networks and requires the resource allocation issue in wireless networks to be revisited and redesigned.

Finally, the limited battery life of mobile clients becomes a bottleneck, which impedes the mobile clients to obtain better user experience in MEC environment. For example, the advancement in deep learning and edge computing has enabled intelligent mobile AR (MAR) on resource limited mobile clients. However, today very few deep learning based MAR applications are applied in mobile clients because they are significantly energy-guzzling. Although compared to running a deep learning algorithm locally on a mobile device, edge-based approach may extend the device's battery life to some extent, it is still considerably energy consuming due to conducting multiple pre-processes on the mobile device, such as camera sampling, screen rendering, image conversion, and data transmission [2]. For instance, based on the measurement from our developed MAR testbed, a 3000 mAh smartphone battery is

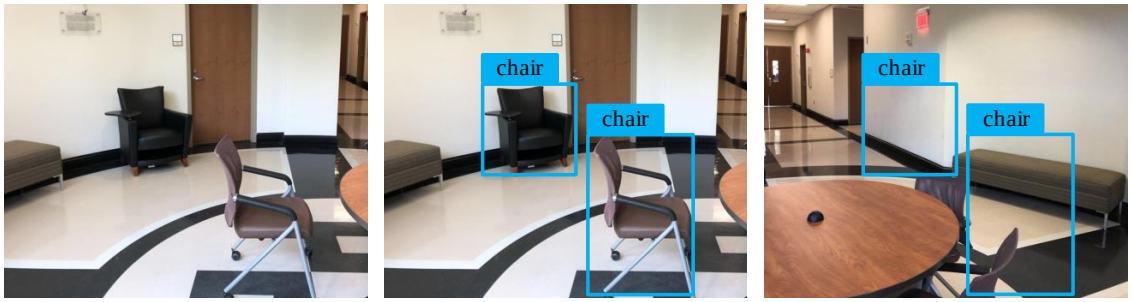
exhausted within approximately 2.3 hours for executing our developed MAR application which continuously transmits the latest camera sampled image frames to an edge server for object detection. Therefore, the energy efficiency of MAR devices becomes a bottleneck, which impedes MAR clients to obtain better MAR performance. For example, decreasing the energy consumption of an MAR device is always at the cost of reducing the object detection accuracy. Therefore, improving the energy efficiency of MAR devices and balancing the tradeoffs between energy efficiency and other MAR performance metrics are crucial to edge-based MAR systems.

1.2 Problem Statement

1.2.1 Inadequate Handoff Triggers in MEC Networks

Handoff triggers in conventional wireless networks is based on the radio link quality related metrics [3–7], such as RSSI (Received Signal Strength Indicator) [4, 7], the number of the lost beacons [4], and the number of re-transmitted packets at users. In IEEE 802.11, only one of above mentioned metrics measured on one link direction, usually the downlink direction, is used as the handoff trigger. However, these radio quality based triggers are no longer sufficient in MEC networks. First, downlink and uplink may be asymmetric, since mobile users and their associated AP may acquire different transmission/receiving capabilities or have different hardware/software implementations. Thus, it is possible the downlink quality stays good while the uplink quality is weak. Therefore, considering a single measurement metric at the downlink direction as the only metric for triggering handoffs is inadequate. In addition, as explained previously, not considering computation related metrics in handoff triggers will lead to poor performance at the MEC server side and severely affect the handoff trigger accuracy. For example, a handoff will not be triggered if a user obtains a good radio link quality (e.g., high RSSI or low packet re-transmission ratio), but a weak computation performance (e.g., the MEC server allocates insufficient computing resources to the user causing a high computation latency).

To study the impact of inadequate handoff triggers, experimental studies are conducted, based on an edge-based mobile AR application, to demonstrate the IEEE 802.11 standard handoff trigger is inadequate in MEC networks, as shown in Figures 1.1(a), 1.1(b), and 1.1(c). Edge-based AR applications are uplink traffic dominating, where users offload a large amount of camera captured video frames to edge servers for object analytics purposes in real-time. Figure 1.1(a) shows the video frame offloaded to the MEC server for processing. After a long delay, the user acquires the returned analytics results (i.e., labels) in Figure 1.1(c). From the figure we can see that the reported two chairs' locations are obviously wrong, compared with the correct results shown in Figure 1.1(b). This incorrect analytics result is due to the change of the scene captured by the user's camera because of user mobility within the long delay waiting for the returned analytics results. Therefore, without an adequate handoff triggering mechanism that considers both the computing and communication requirements of user applications, a user may experience poor performance in MEC and may remain poor for a lengthy duration.



(a) (b) (c)

Figure 1.1: Inadequate handoff triggers in MEC networks. (a) Offloaded video frame; (b) Correct analytics result; and (c) Result with staleness.

1.2.2 Inefficient Radio-Service Handoff Process in MEC Networks

When a mobile user moves away from its attached MEC server, its offloading service has to be migrated or rebuilt on a new nearby MEC server. This process is called service rebuilding and takes a long delay that may deteriorate user experience. The

service rebuilding process includes radio and service handoffs. Although many existing papers proposed different ways to reduce radio and service handoff latencies, the service rebuilding latency still cannot satisfy the requirements of latency-sensitive applications. First, all existing solutions consider radio and service handoffs separately and they are always performed sequentially. This leads to inefficient service rebuilding process. In addition, most existing work on service handoff is not suitable for practical wireless networks. For instance, virtual machine (VM) live migration is widely used in data centers, where the computing service is encapsulated in a VM. However, it is not suitable for the service handoff across MEC servers [8] because, unlike centralized data centers that are deployed with dedicated high-bandwidth networks, connectivity between MEC servers is subject to varying wide area network (WAN) latency, bandwidth, and jitter. Therefore, migrating the whole VM file system takes considerably longer.

Experimental measurement studies are conducted to demonstrate the inefficiency of service rebuilding process in existing MEC networks. The measurement results are shown in Figure 1.2 and Table 1.1. According to Table 1.1, the total service rebuilding latency is around 41 seconds, which is too much for latency-sensitive applications/services. Additionally, as shown in Figure 1.2, the radio handoff cannot run in parallel with the service handoff because the target MEC server has to be notified first if a mobile user is handed off to the corresponding AP before initiating a service handoff. Compiling the application code costs approximate 27.3 seconds that significantly delays the service rebuilding.

1.2.3 Impartial Resource Allocation in MEC Networks

Many current video streaming applications/services in MEC networks are no longer downlink traffic dominant only [9–19]. Thus, traditional radio resource allocation designs that favor the downlink traffic are no longer suitable for supporting QoS/QoE in MEC. Experimental measurement studies are conducted to explore the wireless

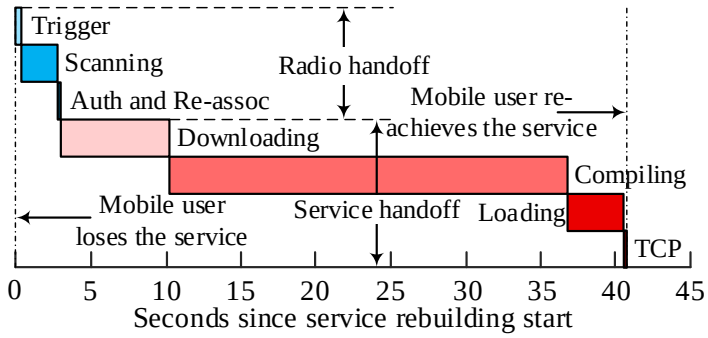


Figure 1.2: Traditional radio-service handoff process.

Table 1.1: Conventional radio-service handoff latency results

	Overall	Breakdown	
Radio handoff	3.06sec	Trigger	0.41sec
		Scanning	2.62sec
		Auth and Re-assoc	24ms
Service handoff	38.89sec	Downloading	7.1sec
		Compiling	27.3sec
		Loading	4.4sec
		Building TCP connection	90ms
Total latency	41.15sec		

network performance changes while heavy uplink and downlink traffic co-exist and demonstrate the unfairness in traditional IEEE 802.11 wireless networks, as shown in Figure 1.3.

Observations from Figure 1.3: The latency of downloading and offloading each video frame and the average latency are measured and calculated, where users do not have mobility. As shown in Figure 1.3(a), the average offloading latency per frame is dramatically increased by approximately 1600%, after a user with downloading traffic joined the network. As shown in Figure 1.3(b), the average downloading latency per frame is only increased by approximately 125% after a user with offloading traffic joined the network, which is much less than the increase of the average offloading latency. As shown in Figure 1.3(a), when the downloading video frame resolution increases (i.e., the total downloading data size increases), the average offloading latency per frame does not change much. The same observation is obtained in the downlink

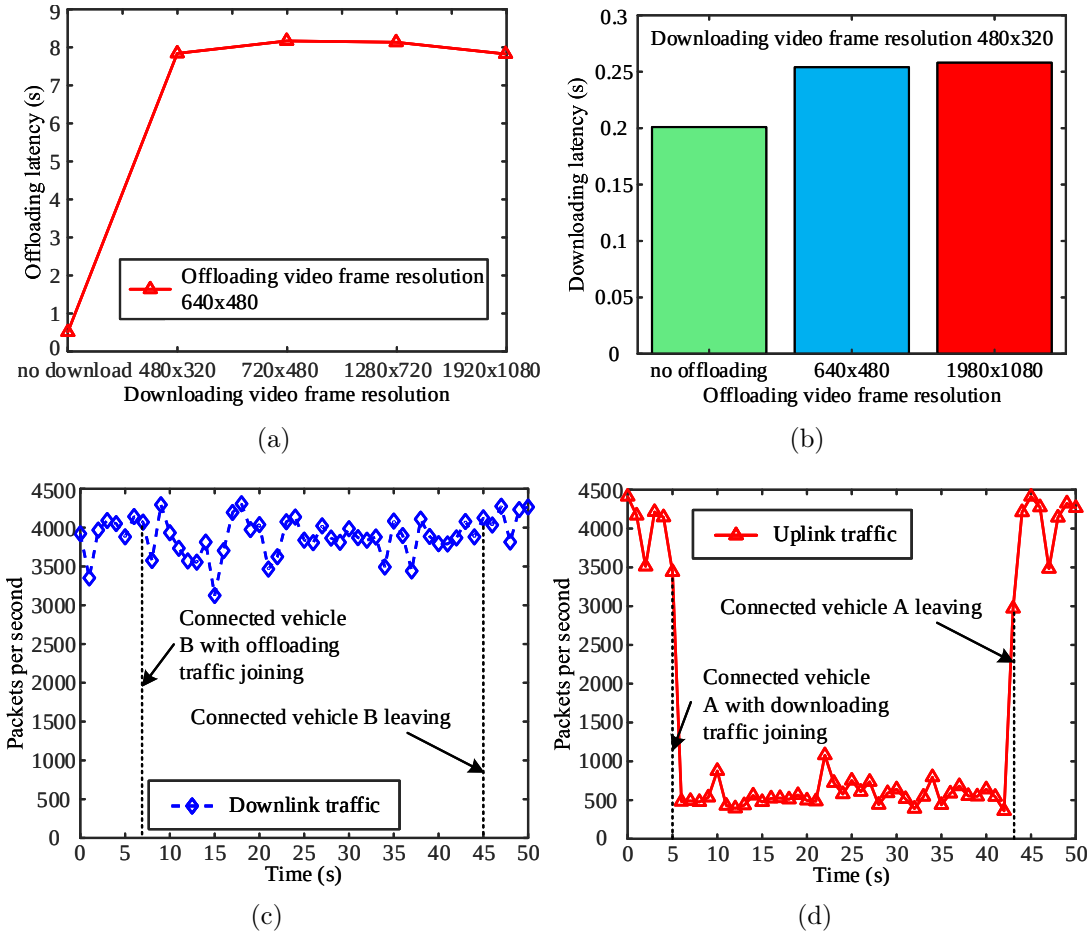


Figure 1.3: Experimental results for users with no mobility. (a) Average offloading latency per frame; (b) Average downloading latency per frame; (c) The number of downlink packets; and (d) The number of uplink packets.

scenario, as shown in Figure 1.3(b).

In order to find the reasons for the aforementioned observations, another experiment study is conducted. The traffic of downlink and uplink is measured, as shown in Figure 1.3(c) and 1.3(d), respectively. In Figure 1.3(d), a big uplink throughput drop occurs during the [5, 43] second period, because a user with downloading traffic joins the network. On the other hand, there is no obvious downlink throughput decline as shown in Figure 1.3(c), when a user with uplink traffic joins the network. This is because usually APs are configured with a smaller Arbitration Inter-Frame Space Number (AIFSN) value for voice and video traffic (e.g., AIFSN = 1) than that of

its associated stations (e.g., AIFSN = 2). Thus, an AP obtains a higher priority for channel contention than its associated stations (i.e., the downlink traffic has a higher priority than the uplink traffic during contention), which significantly impacts the transmission efficiency of the user offloading its camera captured video frames. Although the same AIFS value may be configured for both the AP and stations in order to eliminate the above unfairness, the complicated and fierce channel contention may still badly impact the throughput of both downlink and uplink.

1.2.4 Energy-Guzzling Offloading Services in MEC Networks

An accurate analytical energy model is significantly important for understanding how energy is consumed in an MAR device and for guiding the design of energy-aware MAR systems. However, to the best of our knowledge, there is no existing energy model developed for MAR devices or applications. Developing a comprehensive MAR energy model that is general enough to handle any MAR architecture and application is very challenging. This is because (i) interactions between MAR configuration parameters (e.g., client's CPU frequency and computation model size) and MAR device's energy consumption are complex and lack analytic understandings; (ii) interactions between these configurations and the device's energy consumption may also vary with different mobile architectures.

In addition, designing an energy-aware solution for mobile devices in edge-based MAR systems is also challenging, even after we obtain an analytical energy model. This is because: (i) complicated pre-processes on MAR devices increase the complexity of the problem. Compared to conventional computation offloading systems, besides data transmission, there are also a variety of pre-processing tasks (e.g., camera sampling, screen rendering, and image conversion) necessarily to be performed on MAR devices, which are also energy consuming. For example, over 60% of the energy is consumed by camera sampling and screen rendering, based on observations from our developed testbed. Therefore, we have to take into account the energy efficiency

of these pre-processing tasks while designing an energy-aware approach for MAR clients. (ii) Considering the user preference constraint of individual MAR clients also increases the complexity of the problem. For example, maintaining a high detection accuracy for a client who prefers a precise MAR while decreasing its energy consumption is very challenging. As stated previously, reducing the energy consumption of the MAR device without degrading other performance metrics is no easy task. (iii) In practical scenarios, an edge server is shared by multiple MAR clients. Individual client's energy efficiency is also coupled with the radio resource allocation at the edge server. Such a coupling makes it computationally hard to optimally allocate radio resources and improve each client's energy efficiency.

1.3 Overview of the Proposed Research

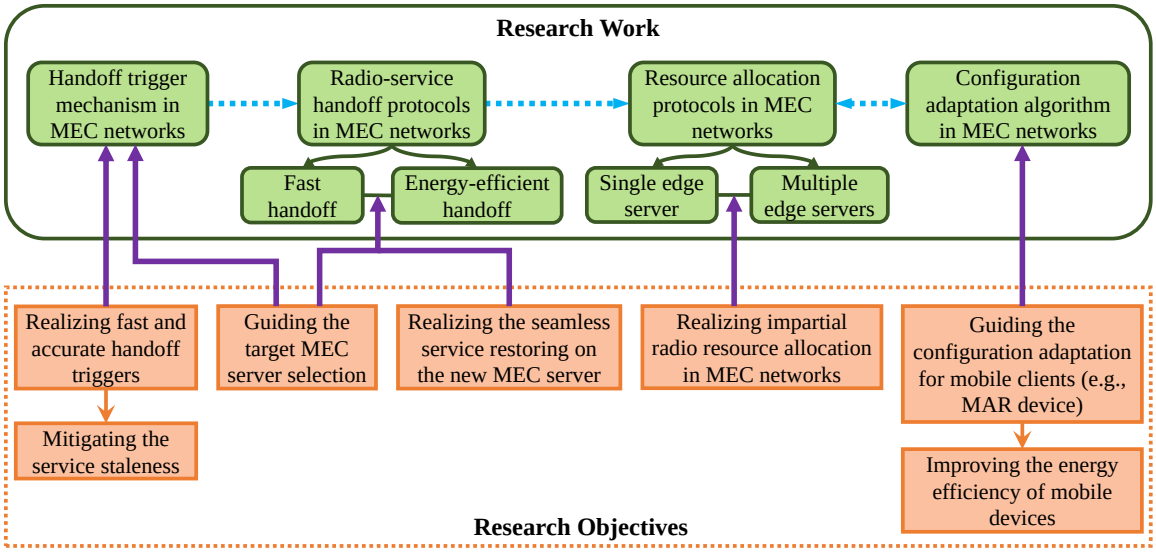


Figure 1.4: The overview of the proposed research.

Based on the analysis of the above four issues in MEC networks, to achieve the performance goals of seamless mobility, carefully and intelligently design the overall service rebuilding procedure that fully exploits the extracted application features and information that a user can possibly obtain to the advantages of seamless mobility support is proposed. Figure 1.4 shows the overview of the proposed research. Fol-

lowing four design strategies are proposed: **(1) smart handoff trigger** that covers both network-side link quality and server-side computation performance to achieve a fast and accurate trigger for seamless mobility support in MEC networks; **(2) radio-service handoff process** for seamlessly restoring offloaded services on the new MEC server after a handoff is triggered; **(3) impartial resource allocation** to minimize performance degradation during mobility caused by radio resource allocation unfairness; and **(4) dynamic configuration adaptation** to guide MAR configuration adaptations and radio resource allocations at the MEC server, and to minimize the MAR client's energy consumption while satisfying their user preferences.

First of all, a practical smart handoff trigger scheme is intelligently designed to achieve a fast and accurate handoff trigger. The proposed trigger scheme covers both network-side link quality and server-side computation performance for users with frequent mobility in MEC networks. In particular, (1) the factors that may cause service staleness in detail by establishing analytical models are first studied. Then, based on our analytical models, (2) a handoff trigger scheme to quickly and accurately discover the factors causing the service staleness is designed. Finally, (3) a method that can efficiently mitigate the service staleness after a mobile user detects the factors incurring service staleness is proposed.

Second, two protocols to mitigate the radio-service handoff latency and improve the handoff energy efficiency are proposed. (1) A fast radio-service handoff protocol is proposed, where radio and service handoffs are conduct in parallel. The proposed protocol seamlessly restores offloading services on the target MEC server after a radio handoff is triggered. The seamless service rebuilding process is achieved via predicting mobile user's target MEC server, before being triggered a radio handoff, and leveraging extracted features from the user's camera captured frames. Furthermore, based on the proposed fast radio-service handoff protocol, a feature database training strategy and feature mapping algorithm are proposed to achieve a high prediction

precision and a short prediction latency. (2) An energy efficient radio-service handoff protocol is proposed, where radio and service handoffs are conduct in sequence. The proposed protocol provides an energy-efficient and low-latency handoff service for its associated mobile users by reproducing and scheduling the beacons broadcast from APs.

In addition, two resource allocation protocols are proposed to mitigate the radio resource allocation unfairness for mobile users in MEC networks. The proposed protocols mitigate the unfairness caused by the design bias in conventional wireless networks. (1) For the single edge server scenario, the key factors that may impact the radio resource allocation in detail are first studied. Then, based on our analysis, a service period allocation protocol is proposed to impartially assign the radio resources to both users with downlink traffic and users with uplink traffic. In addition, a frame resolution selection algorithm is proposed to guarantee a better performance in terms of a higher frame rate. (2) For the multiple edge servers scenario, a user-friendly load-balancing protocol among multiple MEC servers is proposed. In addition, the proposed protocol can always find the optimal load-balancing solution is proved.

Last but not least, a user preference based energy-efficient optimization algorithm is proposed to reduce the per frame energy consumption of MAR clients without compromising their user preferences by dynamically selecting the optimal combination of MAR configurations and radio resource allocations according to user preferences, camera FPS, and available radio resources at the edge server. In particular, (1) an edge-based MAR system is designed and implemented to analyze the interactions between MAR configurations and the client's energy consumption. Based on the experimental study, several insights that can potentially guide the design of energy-aware MAR systems are summarized. (2) The first comprehensive energy model is proposed, which identifies the tradeoffs among the energy consumption, service latency, and detection accuracy, and the interactions among MAR configuration pa-

rameters (i.e., CPU frequency and computation model size), user preferences, camera sampling rate, network bandwidth, and per frame energy consumption for a multi-user edge-based MAR system. (3) An energy-efficient optimization algorithm, LEAF, which guides MAR configuration adaptations and radio resource allocations at the edge server, and minimizes the per frame energy consumption while satisfying variant clients' user preferences is developed.

1.4 Dissertation Organization

The rest of the dissertation is organized as follows. In Chapter 2, related work on the proposed research is introduced. In Chapter 3, a handoff trigger mechanism in MEC networks is proposed. In Chapter 4, two radio-service handoff protocols in MEC networks are proposed. In Chapter 5, two resource allocation protocols in MEC networks are given. In Chapter 6, an energy-aware configuration adaptation algorithm in MEC networks is proposed. Following that, the publications and remaining work are listed in Chapter 7.

CHAPTER 2: RELATED WORK

Currently, a significant amount of research work has addressed various issues in MEC networks [20–35], including system and network modeling [36–44], architecture design [45–66], offloading decision [67–82], virtual machine (VM) migration [8,83–91], path selection [92, 93], multi-user resource allocation [94–113], green MEC [114–118], security [119–122], and standardization [123–128]. *However, the mobility support issue in MEC networks is under-explored in the literature and it is becoming a serious barrier to the success of large-scale MEC deployment.* In addition, although mobility management has been extensively studied in conventional wireless networks [3–6], these works did not consider the unique issues caused by both the communication and computing needs of mobile users as well as their interrelationships. Therefore, they cannot achieve the goal of seamless mobility in MEC.

2.1 Existing Handoff Trigger Mechanisms in MEC Networks

Currently, most of papers studying the handoff trigger only focus on conventional wireless networks. Handoff triggers in conventional wireless networks is based on the radio link quality related metrics [3–7], such as RSSI [4, 7], the number of the lost beacons [4], and the number of re-transmitted packets at users. In IEEE 802.11, only one of above mentioned metrics measured on one link direction, usually the downlink direction, is used as the handoff trigger. However, these radio quality based triggers are no longer sufficient in MEC networks. First, downlink and uplink may be asymmetry, since mobile users and their associated AP may acquire different transmission/receiving capabilities, or have different hardware/software implementations. Thus, it is possible the downlink quality stays good while the uplink quality is ter-

rible. Therefore, considering a single measurement metric at the downlink direction as the only metric for triggering handoffs is inadequate. In addition, as explained previously, not considering computation related metrics in handoff trigger will lead to poor performance at the MEC server side and severely affect the handoff trigger accuracy. For example, a handoff will be not triggered if a user obtains a good radio link quality (e.g., high RSSI or low packet re-transmission ratio), but a terrible computation performance (e.g., the MEC server allocates insufficient computing resources to the user causing a high computation latency).

2.2 Existing Radio-Service Handoff Protocols in MEC Networks

Related work on radio-service handoffs in MEC networks has two major issues. First, all existing solutions consider radio handoff and service handoff separately and they are always performed sequentially [129–133]. This leads to inefficient service rebuilding process. In addition, most existing work on service handoff focused on virtual machine (VM) migration [8, 83, 134–136], where the offloading service is encapsulated in a VM. When a mobile user moves away from its original server, the VM is directly migrated from the original server to the target server. However, it is not suitable for the service handoff across MEC servers [8], since unlike centralized data centers which are deployed with dedicated high-bandwidth networks, connectivity between MEC servers is subject to widely-varying wide area network (WAN) latency, bandwidth, and jitter. Therefore, migrating the whole VM file system takes considerably long time.

2.3 Existing Radio Resource Allocation Protocols in MEC Networks

There are several related papers studying the radio resource allocation in MEC networks with various limitations. First of all, most of existing work does not consider the resource allocation unfairness between the downlink and uplink [137–139]. In conventional wireless communication scenarios, downlink (from the Internet/server to

mobile clients) often attracts much more attention than uplink (from mobile clients to the Internet/server), since comparing to uplink, downlink occupies a larger amount of traffic. Therefore, in conventional wireless communication technologies, for instance, downlink in Long-Term Evolution (LTE) networks acquires higher throughput, peak data rate, and spectral efficiency than uplink [140,141]; and this also happens in WiFi networks [141, 142]. However, many current video streaming applications/services in MEC networks are no longer downlink traffic dominant only [14, 16, 18]. Thus, conventional radio resource allocation designs that favor the downlink traffic are no longer suitable for supporting QoS/QoE in MEC. Second, most of existing papers do not consider the user-mobility-impact on the radio allocation.

2.4 Related Work on Computation Offloading in MEC Networks

Most existing research on computation offloading focuses on how to make offloading decisions. [143–145] coordinate the scheduling of offloading requests for multiple applications to further reduce the wireless energy cost caused by the long tail problem. [146] proposes an energy-efficient offloading approach for multicore-based mobile devices. However, these solutions cannot be applied to improving the energy efficiency of mobile devices in MAR offloading cases. This is because (i) a variety of pre-processing tasks in MAR executions, such as camera sampling, screen rendering, and image conversion, are not taken into account and (ii) besides the latency constraint that is considered in most existing computation offloading approaches, detection accuracy is also a key performance metric, which must be considered while designing an MAR offloading solution. In addition, although some existing work proposes to study the tradeoffs between the MAR service latency and detection accuracy [147–149], none of them considered (i) the energy consumption of the MAR device and (ii) the whole processing pipeline of MAR (i.e., starting from the camera sampling to obtaining detection results).

In addition, energy modeling has been widely used for investigating the factors that

influence the energy consumption of mobile devices. [7, 141, 150–155] propose energy models of WiFi and LTE data transmission with respect to the network performance metrics, such as data and re-transmission rates, respectively. [156–159] propose multiple power consumption models to estimate the energy consumption of mobile CPUs. However, none of them can be directly applied to estimate the energy consumed by MAR applications. This is because MAR applications introduce a variety of (i) energy consuming components (e.g., camera sampling and image conversion) that are not considered in the previous models and (ii) configuration variables (e.g., computation model size and camera sample rate) that also significantly influence the energy consumption of mobile devices.

CHAPTER 3: PROPOSED HANDOFF TRIGGER SCHEME IN MEC NETWORKS

A practical smart handoff trigger scheme in MEC networks, named as *Explorer*, is described in this chapter. The proposed handoff trigger scheme covers both wireless link quality and server-side computation performance for users with frequent mobility in MEC networks in order to achieve a fast and accurate handoff trigger. First, the factors that may cause service staleness are investigated in detail by establishing analytical models. Then, based on these analytical models, a handoff trigger scheme that can quickly and accurately discover the factors causing the service staleness is proposed. Finally, after a mobile user detects the factors incurring service staleness, an approach that can efficiently mitigate the service staleness is proposed. Parameter that will be used in analytical models and the proposed handoff trigger scheme are listed in Table 3.1.

Table 3.1: Notations used in the proposed handoff trigger scheme.

Variable	Description
k_m^2	Frame resolution of the mth video frame (pixels^2)
γ	The number of bits carried by one pixel (bits)
LA_m	Service latency of the mth video frame (s)
LA_m^{tr}	Wireless network latency of the mth video frame (s)
LA_m^{cp}	Computation latency of the mth video frame (s)
c_m	Computation complexity of the mth video frame (TFLOPS)
H	Channel gain
I	Interference power (watt)
σ^2	Background noise power (watt)
P	Transmission power (watt)
B	Network bandwidth (Hz)
f	Available computational resources on the server (TFLOPS)

3.1 Preliminary Experiments

In order to mitigate the staleness and guarantee the detection accuracy simultaneously during user-mobility, MAR systems need to process at high frame-rates. Therefore, having a stable wireless link in an energy-efficient way, especially the uplink, when MAR users move, is very important. It is natural to ask that whether WiFi or cellular networks, which are the two most commonly-used wireless technologies, can offer wireless connections with the aforementioned feature for MAR users. To answer this question, we conduct a series of experiments in an $8000m^2$ campus building with multiple APs. ASUS ZenFone AR is used as the testing platform. Two network candidates are chosen for test, i.e., 802.11ac (eduroam) for WiFi and LTE (supported by T-mobile) for cellular networks.

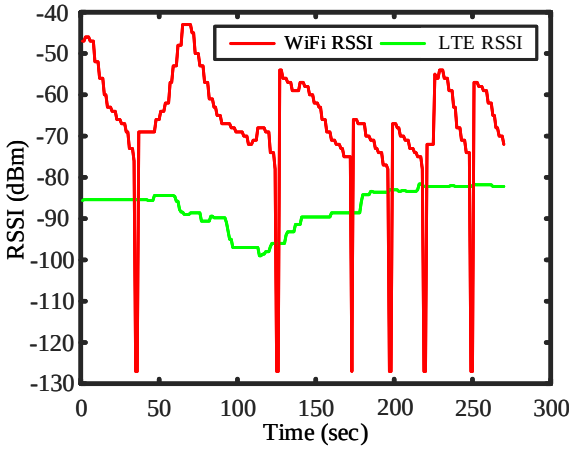


Figure 3.1: Signal strength fluctuation.

Signal strength fluctuation. First, as shown in Figure 3.1, we measure the signal strength fluctuation, indicated by RSSI, on the testing ZenFone AR. The data is generated from tracking the value of RSSI while the testing ZenFone AR is moving around inside the campus building. This measurement is repeated 5 times with the same moving trace, in order to get a statistical confidence in the experimental results. **WiFi:** user-mobility incurs violent signal strength fluctuations and frequent handoff attempts, approximately every 50s, which causes the MAR user losing the wireless

connection for more than 3 seconds. This will definitely cause a high staleness for MAR applications. **LTE:** the measured RSSI of LTE within the trace is relatively stable. Only when the testing platform is away from the building's window-side, the RSSI drops down.

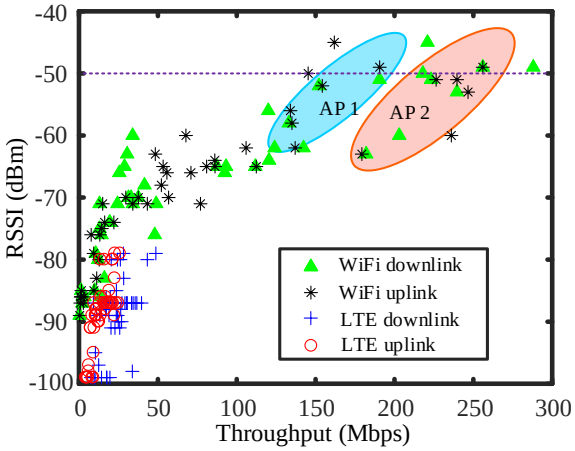


Figure 3.2: Throughput

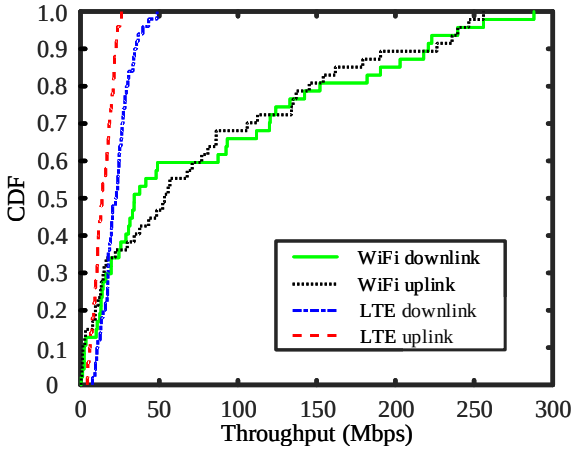


Figure 3.3: CDF of throughput

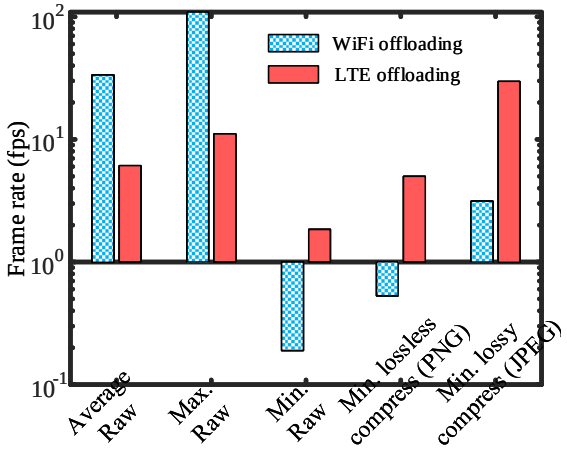


Figure 3.4: Frame rate.

Throughput/frame rate. Then, we measure the throughput of both WiFi and LTE in 50 different locations inside the campus building, and quantify the offloading frame rate, where the frame resolution of the offloaded streaming is 640×480 pixels. In order to mitigate the contextual discrimination on collected data, we repeat our measurement in the morning and evening, on weekdays and weekends. Measurement results are depicted in Figures 3.2, 3.3, and 3.4.

We observe that (1) although LTE can provide the downlink throughput in between 20 and 30 Mbps, its uplink throughput is still not good enough for supporting fast MAR offloading. This observation is also confirmed by the data rate report originated from *Speedtest* [160], where the average download and upload speed of LTE in the United States during the first half of 2018 are 27.33 Mbps and 8.63 Mbps, respectively. (2) Although WiFi obtains a higher average throughput than LTE on both downlink (80.21 Mbps vs. 22.80 Mbps) and uplink (78.36 Mbps vs. 14.26 Mbps), at over 32% and 24% testing locations, the throughput of the downlink and uplink through WiFi is lower than that of LTE, respectively. (3) The performance of the downlink and uplink is not always synchronized. For instance, sometimes LTE has a better downlink throughput than that of WiFi, however, its uplink throughput is lower than that of WiFi. (4) As shown in Figure 3.4, the testing platform suffers very low frame rate, approximately 0.19 fps, through WiFi at some locations, which is only 10% of the lowest frame rate through LTE. Even though the offloading frames are compressed by a lossy compression algorithm, such as JPEG, the frame rate through WiFi is still much lower than 10 fps. Even worse, such levels of compression result in an almost unusable reduction in the quantity of extractable keypoints [17], which leads to a terrible recognition accuracy. (5) As shown in Figure 3.2, although the hardware of each AP is the same, the throughput at the same RSSI of different APs might be different, which demonstrates that only collecting the raw RSSI can not directly imply the performance of throughput.

Energy efficiency. Lots of existing works propose to offload mobile network traffic from cellular to WiFi, since cellular networks incur a higher power consumption than WiFi. However, WiFi links are not always energy efficient. As demonstrated in many measurements [161, 162], the energy efficiency of WiFi is prone to multiple factors, such as multi-path and channel fading. Therefore, blindly offloading frames through WiFi during user mobility may not be beneficial or even steal more energy.

Summary. We find that both WiFi and LTE fail to offer satisfactory wireless links for MAR offloading services during user mobility, especially the uplink. Sometimes even compressing the video or decreasing the video frame resolution may not compensate the poor uplink throughput.

3.2 System Model

As we observed in our preliminary experiments, although WiFi can provide high downlink and uplink throughput in most locations, it is extremely volatile and unstable under user-mobility, which may incur terrible MAR performance even after applying video compression or frame resolution decrease. While cellular networks are more stable than WiFi when an MAR client is moving, the performance of MAR offloading is constrained by the low uplink throughput, monetary cost, and energy efficiency of cellular networks. Therefore, at the heart of the proposed *Explorer* is that MAR clients offload object analytics through WiFi at a higher priority and switch to cellular networks when their associated AP-attached MECs fail to offer high-quality MAR services. Thus, we consider all the AP-attached MECs as *Dominating MECs* (DCs) and the BS-attached MECs as the *Assistant MECs* (AC). However, unlike traditional work on WiFi and cellular integration, switching between DCs and AC requires more complicated estimation on the sources of performance decrease and sophisticated handoff trigger design.

We consider an MEC-enabled network environment with one AC and N densely deployed DCs. Denote $\mathcal{N}$ as the set of DCs and a as the AC. We focus on a representative MAR user moving in the above mentioned network environment. Denote $\mathcal{M}$ as the set of generated video frames. Let L_m denote the location where video frame $m \in \mathcal{M}$ is generated. Due to the dense deployment of APs, multiple DCs can provide service to the MAR user for each video frame m at location L_m. And these DCs are denoted as $\mathcal{D}(L_m) \subseteq \mathcal{N}$. Meanwhile, the MAR user can obtain the service from AC a at any location L_m. Object analytics of a particular video frame m is

performed at either the user associated DC $n \in \mathcal{D}(L_m)$ or the AC a without being further offloaded to other DCs or a remote cloud. Denote $(a^w_{m,n}, a^c_{m,a}) \in \{(1,0),(0,1)\}$ as the MEC association indicator which indicates the MAR user is served by DC n if $(a^w_{m,n}, a^c_{m,a}) = (1,0)$ and is served by AC a if $(a^w_{m,n}, a^c_{m,a}) = (0,1)$.

3.3 Analytical Models

In this section, analytical models are designed for analyzing the MEC-based MAR system. We consider how to model the wireless network latency, computation latency, and analytics staleness. The service latency of the mth video frame can be defined as,

$$LA_m = LA^{tr}_m + LA^{cp}_m, \tag{3.1}$$

where LA^{tr}_m is the wireless network latency incurred by transmitting the video frame m from the MAR user to its associated DC or AC; LA^{cp}_m is the computation latency of the object analytics on the server. In addition, since the data size of the analytics results is usually small, we do not include the latency caused by transmitting the analytics results.

Wireless Network Latency Model: the wireless network latency is determined by the user's video frame resolutions and wireless channel quality. We assume that the AR video generated by the MAR user is pre-processed into video frames with the resolution of $k_m \times k_m$ pixels. Thus, the data size of the mth video frame is calculated as $k^2_m\gamma$ bits. In addition, Shannon's Theorem is used to model the wireless uplink channel quality. Denote $H^\tau_{m,n}$ and $H^\tau_{m,a}$ as the channel gain when the MAR user transmits the mth video frame to DC $n \in \mathcal{D}(L_m)$ and AC a at time $\tau \in [0, LA^{tr}_m]$, respectively. The maximum achievable uplink transmission rate for transmitting the mth video frame at τ is given by

$$r^\tau_m = (a^w_{m,n}B_w + a^c_{m,a}B_c)\log_2\left(1 + a^w_{m,n}\frac{P_w H^\tau_{m,n}}{\sigma^2_w + I^\tau_{m,n}} + a^c_{m,a}\frac{P_c H^\tau_{m,a}}{\sigma^2_c + I^\tau_{m,a}}\right) \tag{3.2}$$

where B_w and B_c are the channel bandwidth of WiFi and cellular networks, respectively; σ_w^2 and σ_c^2 are the background noise power, and $I_{m,n}^{\tau}$ and $I_{m,a}^{\tau}$ are the interference power at DC n and AC a while transmitting the mth video frame at τ, respectively. Therefore, the wireless network latency experienced by the mth video frame is modeled as

$$LA_m^{tr} = \frac{k_m^2 \gamma}{R_m}, \tag{3.3}$$

where $R_m = \frac{\int_0^{LA_m^{tr}} r_m d\tau}{LA_m^{tr}}$, which is the average wireless uplink data rate for transmitting the mth video frame. Moreover, the energy consumption for transmitting the mth video frame can be modeled as

$$E_m^{tr} = (a_{m,n}^w P_w + a_{m,a}^c P_c) LA_m^{tr}. \tag{3.4}$$

Computation Latency Model: the computation latency is closely related to the computation complexity of a user's task and available computational resources at the user associated MEC server. Let $f_{m,n} \in (0, F_n]$ and $f_{m,a} \in (0, F_a]$ be the available computational resources at DC n and AC a, respectively, where F_n and F_a are the computation capacity of DC n and AC a, respectively. We assume that $f_{m,n}$ and $f_{m,a}$ do not change during the whole computation processing of the mth video frame. Therefore, the computation latency experienced by the mth video frame can be modeled as

$$LA_m^{cp} = \frac{c_m}{a_{m,n}^w f_{m,n} + a_{m,a}^c f_{m,a}}. \tag{3.5}$$

In addition, computation complexity c_m is closely related to the video frame resolution k_m^2. Since we focus on the wireless link level performance rather than the MEC server system level performance in this paper, we consider an existing computation complexity model [19] described as $c_m = \psi(k_m^2) = 7 \times 10^{-10} k_m^3 + 0.083$, where $\psi(k_m^2)$ is convex with respect to the mth video frame resolution k_m^2. Although this model is

built by implementing a special object recognition framework, YOLO [163], our work is applicable to any other computation complexity model.

Analytics Staleness Model: to evaluate the staleness, we use the Intersection over Union (IOU) metric, which is similar to [18]:

$$IOU = \frac{area|O \cap G|}{area|O \cup G|}, \tag{3.6}$$

where O and G are the bounding boxes of the detected object and the ground truth, respectively. The average of the object IOUs in a video frame gives the frame's IOU. The value of IOU highly depends on the service latency. A larger service latency usually results in a lower IOU, which denotes a higher staleness. Therefore, we model the IOU as a function of the service latency LA_m. To do so, we implement an object recognition framework, YOLOv3 [164], on a Nvidia Jetson TX2. Figure 3.5 shows that the IOU decreases when the service latency becomes larger. Such a relationship between the IOU and LA_m can be characterized by a convex function, e.g., the measurement data can be fitted by a convex function,

$$IOU_m(LA_m) = 0.1323 \times LA_m^3 - 0.02898 \times LA_m^2 - 0.9623 \times LA_m + 1.091, \tag{3.7}$$

with the root mean square error (RMSE) of 0.05.

3.4 The Proposed Handoff Trigger Scheme

The proposed handoff trigger scheme, *Explorer*, is shown in Figure 3.6. At the heart of the proposed scheme is that mobile users choose DCs for object analytics at a higher priority and switch to the AC when their associated DCs cannot offer small-staleness services. The proposed *Explorer*, as shown in Figure 3.6, consists of four components: the offloading engine, MEC resource tracker, tracking window controller, and data collector.

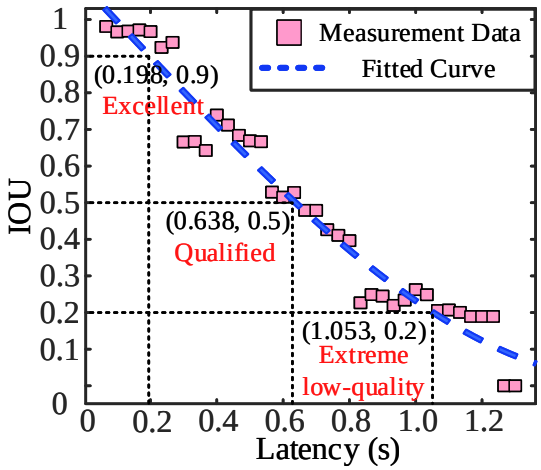


Figure 3.5: The service latency vs the IOU.

The proposed **Data Collector** is responsible for recording and storing the user-side, network-side, and server-side data, to facilitate the other three components. The user-side data required to be recorded and stored for the mth video frame includes k_m^2 and γ. The functions of the proposed **MEC Resource Tracker** are estimating the wireless link quality, $\overline{H_{m,n}}$, $\overline{I_{m,n}}$ and $\overline{H_{z,a}}$, $\overline{I_{z,a}}$, and the available computational resource, $\overline{f_{m,n}}$, using the information provided by the data collector. There are two components in the MEC resource tracker: the wireless link quality tracker and server resource tracker. Since sometimes a user may stay at a location for a while and the wireless link quality may not change much, executing the resource tracker and offloading engine frequently will drain the battery and consume lots of computational resources of the mobile device. To address this issue, a **Tracking Window Controller** is proposed to dynamically adjust the frequency of executing the MEC resource tracker (i.e., tracking window size) and the offloading engine. The core component of the proposed scheme is the **Offloading Engine** which determines when a user should switch its video frame analytics from its associated DC to the AC. Based on the different sources of the link quality decline, the offloading engine provides the corresponding offloading scheme. Besides determining the timing of switching, the offloading engine can also quantify the number of frames that need to be offloaded

to the AC based on the constraints of the LTE monetary cost and the battery drain. Therefore, the proposed offloading engine is able to not only dynamically mitigate the wireless link quality decline, but also guarantee a low energy consumption and monetary cost for the mobile device.

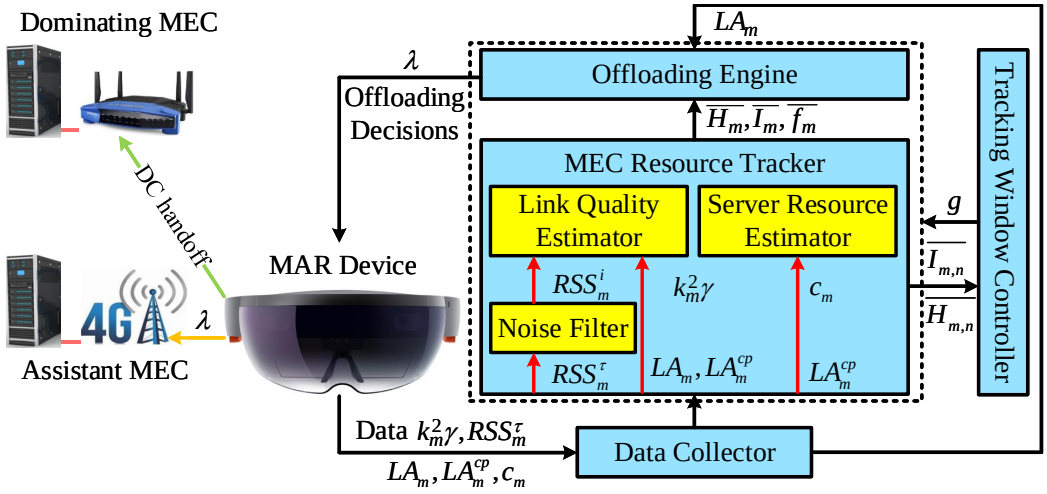


Figure 3.6: Overview of the proposed handoff trigger scheme.

3.4.1 Offloading Engine

The core component of our proposed handoff trigger scheme in MEC is the offloading engine which determines when an MAR device should switch its video frame analytics from its associated DC to the AC. And based on the different sources of the link quality decline, the offloading engine provides the corresponding offloading scheme. Besides determining the timing of switching, the offloading engine can also quantify the number of frames that need to be offloaded to the AC based on the constraints of the LTE monetary cost and the battery drain. Therefore, our proposed offloading engine can not only dynamically mitigate the wireless link quality decline, but also guarantee a low energy consumption and monetary cost for the mobile device.

The offloading engine is executed when LA_m is not smaller than a threshold LA_{tg}. We use three types of triggers to determine the major source of the wireless quality decline.

Case 1: user-mobility-incurred case. (Channel gain trigger H_{tg}) In wireless networks, channel gain is a critical metric to estimate the wireless link quality between two communication nodes. It may vary with time, due to fading and node mobility. The value of channel gain is an applicable metric to estimate the level of user-mobility-incurred latency. In our designed offloading engine, when $\overline{H_{m,n}} \leq H_{tg}$, the MAR device is triggered to switch its analytics tasks to the AC to maintain a small staleness. Meanwhile, the MAR user will re-associate with a nearby DC $i \in \mathcal{D}(L_m)$ for a better DC service, where we define this process as a *DC handoff*.

Case 2: temporary link quality decline case. (Radio interference trigger I_{tg}) Radio interference is another important metric to estimate the wireless link quality. We classify radio interference into two categories: *inter-DC* and *non-802.11 interference*. (i) Any traffic of the nearby DCs on the same channel or adjacent channels is defined as the inter-DC interference, where it may vary with time due to the user mobility. (ii) DCs are operated in the 2.4 or 5 GHz shared ISM band. The 2.4 GHz band is shared with other non-802.11 devices, such as microwave ovens and baby monitors, which could lead to a significant amount of interference. Therefore, the increase of the DC's radio interference may be permanent (inter-DC interference) or temporary (non-802.11 interference). In our designed offloading engine, when $\overline{I_{m,n}} \geq I_{tg}$, the MAR user is triggered for a switching but no DC handoffs.

Case 3: temporary server resources exhausted case. (Available DC computational resource trigger f_{tg}) The service latency also may be badly impacted by the computation latency because the available computational resources $f_{m,n}$ at the associated DC n is little. Similar to the non-802.11 interference, we design that when $\overline{f_{m,n}} \leq f_{tg}$, the MAR user is triggered for a switching but no DC handoffs.

After determining the timing of switching, we next quantify the number of frames that need to be offloaded to the AC, which is denoted as λ. As stated above, there are two different switching scenarios: (i) switching with DC handoffs and (ii) switching

without DC handoffs. For scenario (i), the MAR user keeps offloading video frames to the AC during a DC handoff. Since the latency of a DC handoff is long, around 3s, the MAR user suffers a long period without DC services during the DC handoff process. For scenario (ii), we build a model to calculate the number of frames that need to be offloaded to the AC, based on the constraints of the LTE monetary cost and the battery drain. The lower and upper bound of λ, (i.e., λ_{min} and λ_{max}) are calculated by:

$$P_c \frac{\lambda_{min}\overline{k^2\gamma}}{R(\overline{H_{z,a}}, \overline{I_{z,a}})} + 2e_{sw} \leq P_w \frac{\lambda_{min}\overline{k^2\gamma}}{R(\overline{H_{m,n}}, \overline{I_{m,n}})}, (z, m \in \mathcal{M}) \tag{3.8}$$

$$q\lambda_{max}\overline{k^2\gamma} \leq Q, \tag{3.9}$$

where **(3.8)** says that the battery drain of offloading after switching to the AC cannot exceed that of maintaining the link with the DC, and **(3.9)** says that the LTE monetary cost cannot exceed the maximum target Q (\$); P_w and P_c are the transmit power of the WiFi and cellular interface of MAR devices, respectively; $\overline{k^2\gamma}$ is the average data size of the video frames that have already been offloaded; $\overline{H_{z,a}}, \overline{I_{z,a}}$ and $\overline{H_{m,n}}, \overline{I_{m,n}}$ are the latest recorded estimation results of AC a and DC n, respectively, extracted from the MEC resource tracker (describe in Section 3.4.2); e_{sw} is the energy consumption of conducting a switching; and q is the LTE monetary cost (\$/bit). **If the calculated** $\lambda_{min} > 0$, $\overline{H_{m,n}}, \overline{I_{m,n}}$ are the current estimated wireless link quality of the associated DC n, and they are less accurate for representing the future link quality if the MAR user moves fast and frequently. In order to make the calculated λ more accurate, we choose λ based on the value of the tracking window size g which is extracted from the tracking window controller (described in Section 3.4.3):

$$\lambda = \lambda(g), \lambda \in [\lambda_{min}, \lambda_{max}]. \tag{3.10}$$

For example, when g is small indicating that the MAR user is moving fast and frequently and $R(\overline{H_{m,n}}, \overline{I_{m,n}})$ might change fast, we choose a relatively small value of λ; and vice versa. **If the calculated** $\lambda_{min} \leq 0$, the offloading engine reduces the transmit latency via decreasing the resolution of the next offloaded frame or executing existing computation-based solutions [18, 19]. The details of our designed offloading engine are given in Algorithm 1.

Algorithm 1: Offloading Engine

Input: LA_m, e_{sw}, $\overline{H_{m,n}}$, $\overline{I_{m,n}}$, $\overline{f_{m,n}}$, $\overline{H_{z,a}}$, $\overline{I_{z,a}}$, LA_{tg}, H_{tg}, I_{tg}, f_{tg}, P_w, P_c, $\overline{k_m^2\gamma}$, q, Q, g
Output: g and λ.

```
if Tracking Window Controller triggered = true then
    if LA_m ≥ LA_tg then
        j ← 0; /* Case 1. */
        if H̄_{m,n} ≤ H_tg then
            DC handoffs start← true; (a^w_{m,n}, a^c_{m,a}) ← (0, 1);
            while True do
                j ← j + 1; λ ← j; /* Offloading frames to AC a. */
                if DC handoffs complete = true then
                    /* Switch to DC i. */
                    (a^w_{m+λ,i}, a^c_{m+λ,a}) ← (1, 0), i ∈ D(L_m);
                    break;
        /* Case 2 and 3. */
        else if H̄_{m,n} > H_tg&&(Ī_{m,n} ≥ I_tg||f̄_{m,n} ≤ f_tg) then
            Calculate λ_min, λ_max, λ via (3.8), (3.9), (3.10);
            if λ_min > 0 then
                (a^w_{m,n}, a^c_{m,a}) ← (0, 1); /* Switch to AC a. */
                while True do
                    j ← j + 1; /* Offloading frames to AC a. */
                    if j = λ then
                        /* Switch back to DC n. */
                        (a^w_{m+λ,n}, a^c_{m+λ,a}) ← (1, 0);
                        break;
            else if λ_min ≤ 0 then
                k²_{m+1} ← k_min × k_min; /* Decrease the frame resolution or other existing
                    solutions. */
        g ← 1; /* Reset the tracking window size. */
    return g, λ
```

3.4.2 MEC Resource Tracker

The functions of the MEC resource tracker are estimating the wireless link quality, $\overline{H_{m,n}}$, $\overline{I_{m,n}}$ and $\overline{H_{z,a}}$, $\overline{I_{z,a}}$, and the available computational resource, $\overline{f_{m,n}}$, using the information provided by the data collector (describe in Section 3.4.4). There are two components in the MEC resource tracker: the wireless link quality tracker and server resource tracker.

Wireless link quality tracker: (1) Noise filter: raw RSS data are passed to a noise filter to filter out the noise first. Usually, RSS samples collected by the radio interfaces on smart devices contain noise due to reasons such as multi-path effects. Noise may introduce errors in estimating wireless link qualities and thus, needs to be filtered out. We use the Nadaraya-Watson estimator [162, 165], which is a kernel regression approach, for the noise filter. Typically, it models the RSS samples within a window as a series of random variables from a joint Probability Density Function (PDF): $RSS_m^i = \phi_m(i) + \epsilon_m^i, (i = 1, \cdots, n)$, where RSS_m^i is the collected RSS at time i within the mth video frame and $\phi_m(\cdot)$ is a function for the corrected RSS. The error $\{\epsilon_m^i\}$ satisfies: $E(\epsilon_m^i) = 0$, $V(\epsilon_m^i) = \sigma^2$, $Cov(\epsilon_m^i, \epsilon_m^j) = 0$ $(i \neq j)$. The estimation $\hat{RSS}_m^i$ after filtering can be expressed as follows:

$$\hat{RSS}_m^i = \frac{\sum_{j=1}^{n} K_h(i,j) RSS_m^i}{\sum_{j=1}^{n} K_h(i,j)}, \tag{3.11}$$

where $K_h(i,j)$ is the kernel function with a window size h for pairwise values (i,j). We choose Radial Basis Function (RBF) kernel for K_h and h is chosen adaptively. **(2) Link quality estimator:** this component is responsible for estimating the wireless link quality using the filtered RSS, the user-side data, and the network-side data in the data collector. We use two metrics to represent the wireless link quality: (i) channel gain H_m^τ and (ii) interference I_m^τ. We define $\overline{H_m}$ and $\overline{I_m}$ as the average channel gain and interference of the mth video frame, respectively. The filtered $\hat{RSS}_m^i$ is used for

calculating $\overline{H_m}$, where

$$\overline{H_m} = 10^{\frac{\sum_{i=1}^{n} \hat{RSS}_m^i}{10 \cdot n} - 3} \cdot (a_{m,n}^w P_w + a_{m,a}^c P_c)^{-1}. \tag{3.12}$$

Then based on Shannon's Theorem we have:

$$\overline{I_m} = \frac{(a_{m,n}^w P_w + a_{m,a}^c P_c) \cdot \overline{H_m}}{2^{\frac{k^2 \gamma}{\overline{LA_m^{tr} \cdot (a_{m,n}^w B_w + a_{m,a}^c B_c)}}} - 1} - (a_{m,n}^w \sigma_w^2 + a_{m,a}^c \sigma_c^2), \tag{3.13}$$

where k_m^2 and γ are extracted from the data collector; and $LA_m^{tr} = LA_m - LA_m^{cp}$. In addition, we assume that the background noise of DC, σ_w^2, and AC, σ_c^2, are constant.

Server resource tracker: this component is responsible for estimating the available computational resources of the MEC server. $\overline{f_m}$ is calculated as: $\overline{f_m} = \frac{c_m}{LA_m^{cp}}$, where LA_m^{cp} is extracted from the data collector and c_m is calculated via the computation latency model.

3.4.3 Tracking Window Controller

Since sometimes an MAR user may stay at a location for a while and the wireless link quality may not change much, executing the resource tracker and offloading engine frequently will drain the battery and consume lots of computational resources of the MAR device. To address this issue, we design a tracking window controller to dynamically adjust the frequency of executing the MEC resource tracker (i.e., tracking window size) and the offloading engine.

We choose to use the variation of the wireless link quality to dynamically determine the tracking window size, since the variation of the available computational resources of the MEC server is unpredictable (i.e., the number of MAR users who will connect with the server is unpredictable) and does not have a direct correlation with the user-mobility. In addition, according to our findings in Section 3.1, we only execute the tracking window controller when the MAR user is offloading its video frames to the

DC. We define $\overrightarrow{\Phi_{m,n}} = (H_{m,n}, I_{m,n})$ and $\overrightarrow{\Phi_{j,n}} = (H_{j,n}, I_{j,n})$ as the wireless link state vector of the MAR user connected with DC n at location L_m and L_j, respectively. $\Phi^{'}_{m,n} = \frac{\|\overrightarrow{\Phi_{m,n}} - \overrightarrow{\Phi_{j,n}}\|}{g}$ and $\Phi^{''}_{m,n} = \Phi^{'}_{m,n} - \Phi^{'}_{j,n}$ are defined for describing the wireless link quality variation rate and the variation trend, respectively, where $m - j = g$ and g is the tracking window size with an initial value 1.

Algorithm 2: Adjusting the Tracking Window Size g.

Input: The channel gain $\overline{H_{m,n}}$, the interference $overline I_{m,n}$, the wireless link state vector $\overrightarrow{\Phi_{j,n}}$, the link variation rate $\dot{\Phi}_{j,n}$, the tracking window size g, the MEC association indicator $(a^w_{m,n}, a^c_{m,a})$, the rate threshold δ.

Output: The updated tracking window size g, the wireless link state vector $\overrightarrow{\Phi_{m,n}}$, the link variation rate $\dot{\Phi}_{m,n}$.

/* Check if the mth video frame is offloaded to the DC. */

1 **if** $(a^w_{m,n}, a^c_{m,a}) = (1, 0)$ && $m - j = g$ **then**

/* Calculate the wireless link quality variation rate. */

2 $\overrightarrow{\Phi_{m,n}} \leftarrow (\overline{H_{m,n}}, \overline{I_{m,n}})$;

3 $\dot{\Phi}_{m,n} \leftarrow \frac{\|\overrightarrow{\Phi_{m,n}} - \overrightarrow{\Phi_{j,n}}\|}{g}$;

/* Check the variation rate. */

4 **if** $\dot{\Phi}_{m,n} \leq \delta$ **then**

5 $g \leftarrow g + 1$; /* Increase the tracking window size. */

6 **else**

/* Check the variation trend. */

7 $\ddot{\Phi}_{m,n} \leftarrow (\dot{\Phi}_{m,n} - \dot{\Phi}_{j,n})$;

8 **if** $\ddot{\Phi}_{m,n} \leq 0$ **then**

9 $g \leftarrow g$; /* Keep the tracking window size. */

10 **else if** $\ddot{\Phi}_n > 0$ && $g > 1$ **then**

11 $g \leftarrow \theta(g)$; /* Reduce the tracking window size. */

12 $\dot{\Phi}_{j,n} \leftarrow \dot{\Phi}_{m,n}$, $\ddot{\Phi}_{j,n} \leftarrow \ddot{\Phi}_{m,n}$;

13 **return** g, $\dot{\Phi}_{m,n}$, $\ddot{\Phi}_{m,n}$

The details of dynamically adjusting the tracking window size is shown in Algorithm 2. First, the tracking window controller checks if (i) the mth video frame is offloaded to DC n; (ii) $m - j$ is equal to the current window size g. The tracking window controller then extracts the channel gain $H_{m,n}$ and the interference $I_{m,n}$ from the data collector if the mth video frame satisfies the above two requirements. The tracking window size g is increased by 1 if the calculated wireless link quality variation rate $\Phi^{'}_{m,n}$ is not larger than a preset rate threshold δ, which indicates that the current

wireless link quality does not vary much. Increasing the window size will decrease the tracking frequency and thus, save the MAR device's battery and computational resource. Otherwise, we keep the window size if the current link variation trend $\Phi''_{m,n} \leq 0$ (i.e., the current link variation rate is decreasing) or we reduce g according to a preset regression function $\theta(g)$ if $\Phi''_{m,n} > 0$ (i.e., the current link variation rate is increasing).

3.4.4 Data Collector

The Data Collector is responsible for recording and storing the MAR user-side, network-side, and server-side data, to facilitate the other three components. The user-side data required to be recorded and stored for the mth video frame includes k_m^2 and γ. The network-side data includes (i) the continuous RSS of the wireless link within each video frame transmitting period $RSS_m^\tau, \tau \in [0, LA_m^{tr}]$ (e.g., from the moment that the MAR user starts transmitting the mth video frame to the moment of finishing the transmission); (ii) the service latency LA_m. The server-side data includes LA_m^{cp} and c_m.

3.5 The System Implementation and Experiments

In this section, we implement the cloudlet-based MAR system on a testbed, shown in Figure 3.7, which consists of three components: the MAR client with *Explorer*, cloudlet servers, and emulated network. We conduct experiments based on the implementation to validate the performance of *Explorer*.

3.5.1 The Cloudlet-based MAR System Implementation

MAR client with *Explorer*: The MAR client is built in a HP Z820 workstation. It sends real-world captured video frames to a cloudlet server and overlays the received information on its corresponding objects. The frame resolution of the video is 640×480. Four functional modules are implemented in the MAR client. The first one is a data collector that records the information of the MAR user, including the frame

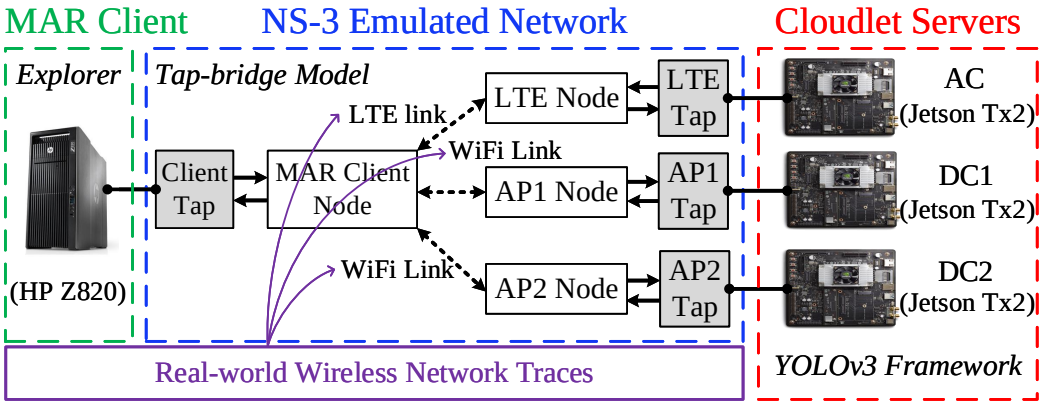


Figure 3.7: The cloudlet-based MAR testbed.

resolution, service latency, and transmission latency of each offloaded frame. The second one is a cloudlet resource tracker which dynamically estimates the wireless channel quality using the recorded service latency and transmission latency. The third module contains a tracking window controller and an offloading engine. It is responsible for triggering an AC switching or DC handoff according to Algorithm 1. The fourth module is a data communication module responsible for streaming the video frames to an assigned server selected by the offloading engine and receiving the analytics results from the server.

Cloudlet servers: There are three cloudlet servers in our testbed, one AC and two DCs. These three cloudlet servers are implemented on three NVIDIA Jetson TX development kits. They are developed to process the offloaded video frames and send the analytics results back to the MAR user. Two major modules are implemented on each server. The first one is the service connection module which establishes a socket connection with the MAR client and sends analytics results to the MAR client. The second one is the object analytics module performs the object analytics. The object analytics module is designed based on the YOLOv3 framework [164].

Emulated network: The wireless network connecting the cloudlet servers and the MAR client are emulated using NS-3. The MAR client and three cloudlet servers are connected to their corresponding tap nodes in the emulated network via the tap-

bridge model of NS-3. In order to emulate the variations of the wireless link quality when the MAR user is moving, we record 10 real-world wireless network traces in our campus building. The measured data includes the ping delay and throughput variations while user is moving. NS-3 takes the traces as input, and dynamically varies the quality of the emulated wireless network while transmitting video frames from the MAR client to the selected cloudlet server.

Reasons for using NS-3 emulated network: (i) emulation allows us to compare *Explorer* with other solutions under reproducible wireless network conditions; (ii) it is hard for us to implement a BS-attached cloudlet in real-world.

3.5.2 Performance Evaluation

Figure 3.8 shows the IOU comparison during a handoff process under *Explorer* and other solutions. The baselines include: (i) *LTE-only solution:* all frames are offloaded to the BS-attached cloudlet server (LTE-C); (ii) *WiFi-only solution:* all frames are offloaded to AP-attached cloudlet servers (WiFi-C); (iii) *groundtruth:* all frames are directly computed at server without offloading. The sampling period is every 6 frames. Handoff trigger in IEEE 802.11 commercial WiFi products is to count the number of continuously missed beacons, or when the RSSI is below a certain threshold [151]. However, as shown in Figure 3.8, this triggering mechanism is not sensitive to the performance of MAR and makes the IOU significant low during the handoff triggering, whereas the triggering mechanism in *Explorer*, as described in Algorithm 1, maintains good MAR and offloading performance, such as high IOU, low device's energy consumption, and low monetary cost. Table 3.2 shows the experimental results of the average IOU summarized from the experiments shown in Figure 3.8. *Explorer* achieves significant IOU improvement as compared to LTE-C and WiFi-C. Also, *Explorer* guarantees that qualified IOU (i.e., IOU > 0.5) is supported when the user is moving around.

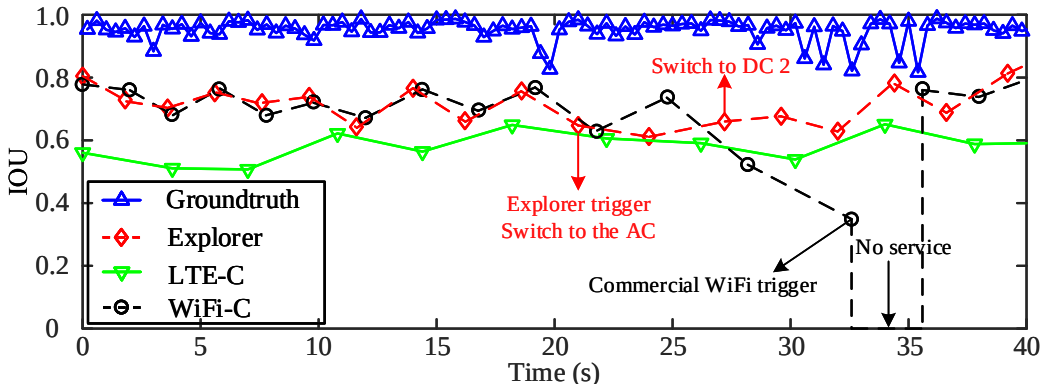


Figure 3.8: Sampled measurement IOU.

Table 3.2: Experimental results

	Average IOU	$IOU \leq 0.5$	$IOU \leq 0.2$
LTE-C	0.58	7.6%	0%
WiFi-C	0.62	17.5%	6.8%
Explorer	0.71	0%	0%
Groundtruth	0.94	0%	0%

3.5.3 Limitations of Our Testbed

The NS-3 emulated network only allows varying the wireless link quality every 1s. This limitation restricts us to evaluate *Explorer* in more complex and frequent link quality variation scenarios. However, implementing a large-scale deployment of cloudlets is very costly. Thus, some improvements obtained by *Explorer* cannot be validated through our testbed. Therefore, we resort to large-scale simulations in order to better understand our proposed *Explorer's* behavior.

3.6 Extensive Large-scale Simulations

In this section, we first seek to understand our proposed *Explorer*'s behavior in a large-scale simulation deployment. Second, we want to understand the impacts of various factors on the performance of *Explorer*. *To the best of our knowledge, this is the first work that explicitly considers decreasing the MAR offloading service latency from the communication perspective, as well as the impact of the user-mobility-incurred wireless network quality decline on the offloading strategy.*

3.6.1 Simulation Setup

Network setup. We simulate a cloudlet-based MAR network with 36 DCs that are regularly distributed in a square area, and one AC located at the center of the simulation area. The length of the simulation area is 840m. Each DC and the AC have circular transmission ranges, and their radii are 120m and 800m, respectively. In other words, the AC can provide services to MAR users at any location in this simulation area. In addition, the wireless channel gain of the WiFi and LTE are modeled as $H_{m,n} = -20.4 + 60 \times \log(d)$ and $H_{m,a} = 131.1 + 42.8 \times \log\left(\frac{d}{1000}\right)$, respectively. d is the distance between the AP/BS and the MAR user. The simulated channel interference is composed of two parts: one is the inter-cell interference which is proportional to the distance between the AP/BS and the MAR user; the other is a random instant interference which represents the instant channel quality decline or the non-802.11 interference. The data rate of DC is determined by the calculated Signal-to-Interference-plus-Noise Ratio (SINR). The channel quality is changed every 1ms. The maximum data rate of the LTE is set to 20 Mbps. Besides, channel bandwidth $B_w = B_c = 20$ MHz, noise power $\sigma_w^2 = \sigma_c^2 = 1 \times 10^{-12}$ watt, and transmit power $P_w = 0.1$ watt and $P_c = 0.5$ watt.

Servers and video frames setup. We adopt $\psi(k_m^2)$ as the computation complexity c_m of analyzing a $k_m \times k_m$ video frame. In simulations, the computing capacities of DCs and the AC are set to 2 and 4 TFLOPS, respectively. The default video frame resolution is 90000 pixels (300×300).

MAR user setup. 100 MAR users follow the Random Waypoint model. We select three average speed values: 1.4m/s, 2.8m/s, and 4.2m/s, to represent three states of the user-mobility: walking, walking fast, and running, respectively. Besides, LTE monetary cost $q = 1 \times 10^{-8}$bits/\$, $Q = 0.5$\$, $e_{sw} = 0.038$J, handoff energy consumption 4.366J, and handoff latency 3.06s [151].

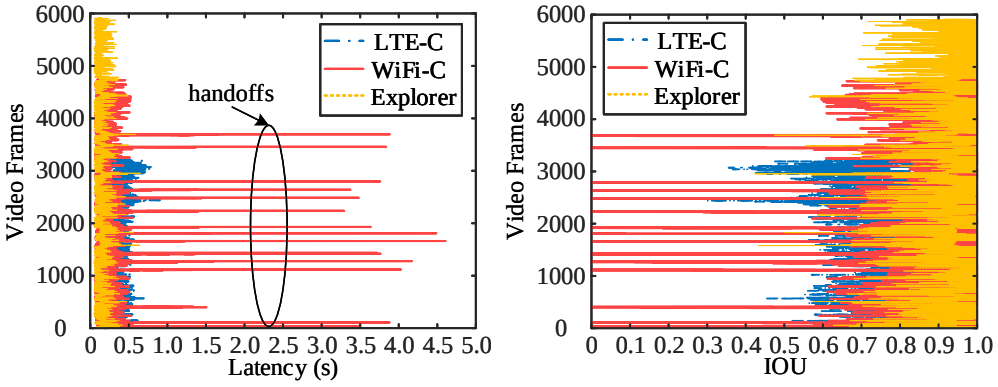


Figure 3.9: Comparison of the service latency and IOU. (a) Service latency; (b) IOU.

3.6.2 Performance Evaluation

3.6.2.1 *Explorer* vs. Baselines

Figure 3.9 illustrates the results of the service latency and IOU performance within 1000s. IOU is calculated based on the measurement results from our experiments, as shown in **(3.7)**. The Y-axis represents the offloaded video frame index within 1000s, and the X-axis represents the service latency and IOU of each offloaded video frame, respectively. As shown in Figure 3.9a, *Explorer* offloads around 5900 video frames within 1000s, which is significantly more than the frames offloaded through WiFi-C, 4742 frames, and LTE-C, 3218 frames. This denotes that *Explorer* obtains the lowest average service latency. Meanwhile, the user-mobility-incurred latency increasing (e.g., frequent handoffs occurred in WiFi-C, depicted as the big red spikes in Figure 3.9a) is efficiently mitigated in *Explorer*. Figure 3.9b shows the comparison of each frame's IOU. It is obvious that *Explorer* obtains a much better IOU performance than the other two baselines. The detailed results are recorded in Table 3.3. The *Explorer* decreases the unexcellent ratio from 86.9% (LTE-C) and 27.4% (WiFi-C) to 26.5%, as well as the unqualified ratio from 1.0% (LTE-C) and 4.4% (WiFi-C) to 0.02%.

Figure 3.10 evaluates the performance of *Explorer* under different frame resolutions. Figure 3.10a shows that *Explorer* obtains both the lowest average latency and latency

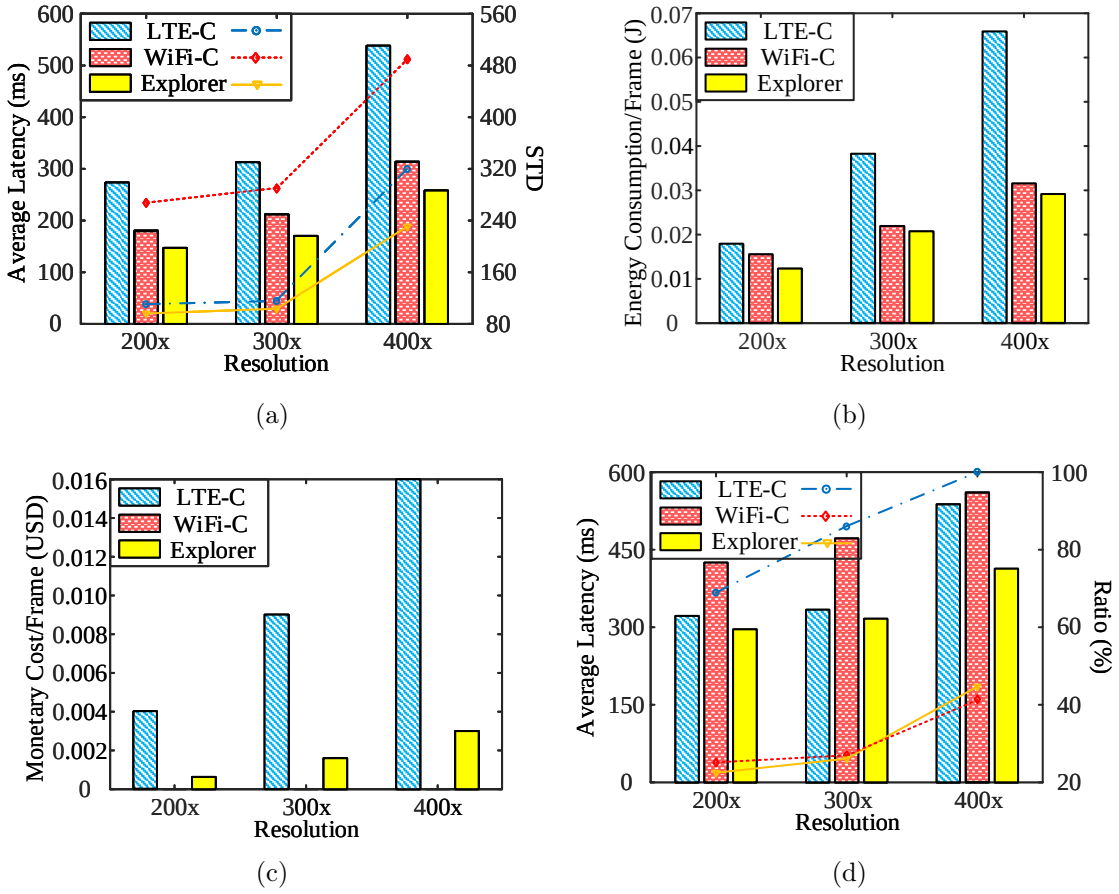


Figure 3.10: Performance comparisons among LTE-C, WiFi-C, and *Explorer*. (a) Average service latency and STD; (b) Average energy cost per frame; (c) Average monetary cost per frame; (d) Average service latency and the ratio of frames $LA_m \geq$ 200ms.

STD with these three frame resolutions. Also, as shown in Figure 3.10b, MAR devices offloading one video frame consumes much less energy than the other two baselines. In addition, as illustrate in Figure 3.5, only when the latency is less than 200ms (i.e., the IOU is larger than 0.9), the staleness performance is excellent. We extract the frames with the service latency larger than 200ms, and compare the average latency and the ratio of these frames to the total number of offloaded frames. As shown in Figure 3.10d, *Explorer* achieves the lowest average latency and ratio except when the frame resolution is 400×400.

Table 3.3: Staleness results (IOU)

	≤ 0.9	≤ 0.8	≤ 0.6	≤ 0.5	≤ 0.2	$= 0$
LTE-C	86.9%	43.1%	6.4%	1.0%	0%	0%
WiFi-C	27.4%	15.4%	5.7%	4.4%	2.1%	1.1%
Explorer	26.5%	11.0%	0.09%	0.02%	0%	0%

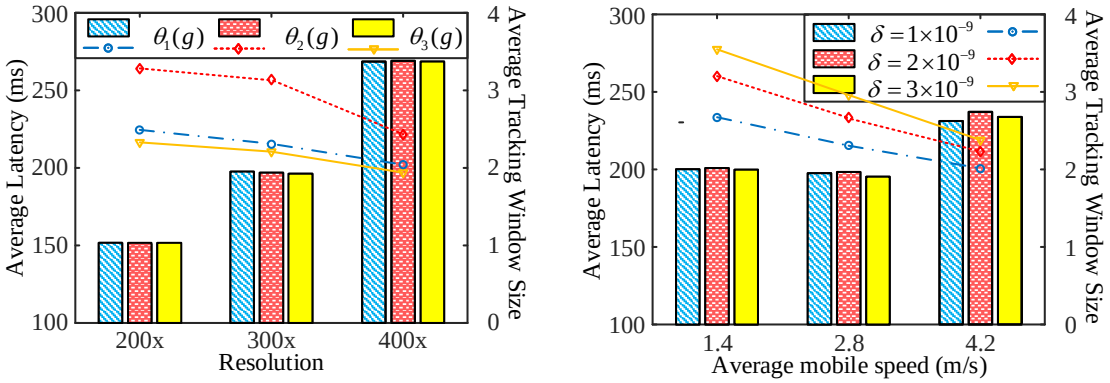


Figure 3.11: The impact of some factors on the system performance. (a) System performance vs $\theta(g)$; (b) System performance vs δ.

3.6.2.2 The Impact of Some Factors

First, we evaluate the impact of different function $\theta(g)$ on the latency and the average tracking window size g, since different $\theta(g)$ may vary the regression rate of the window size. We define three $\theta(g)$ functions in our simulations, where $\theta_1(g_{pre}) = g_{pre} - 1, (1 < g_{pre} \leq 2), = g_{pre} - 2, (2 < g_{pre} \leq 6), = g_{pre} - 6, (6 < g_{pre} \leq 10), = g_{pre} - 10, (g_{pre} > 10)$; $\theta_2(g_{pre}) = g_{pre} - 1$; and $\theta_3(g_{pre}) = 1$. As shown in Figure 3.11a, different $\theta(g)$ has little impact on the average service latency, however, it significantly varies the average tracking window size. As stated in Section 3.4.3, small window size will increase the frequency of executing the cloudlet resource tracker and the offloading engine, which incurs excessive battery and computation resources of mobile devices. Therefore, selecting a $\theta(g)$ with a lower regression rate, e.g., $\theta_2(g)$, is good for saving mobile devices' resources. Second, as shown in Figure 3.11b, we compare the system performance under different user-mobility speed and δ. We observe that higher

user-mobility speed increases the latency and reduces the average tracking window size. This is because a higher user-mobility speed may accelerate the wireless quality variation and thus, increases the frequency of switching to the AC and triggering the tracking window size regression. Therefore, since the average service latency varies little, selecting a relatively large δ, e.g., $\delta = 3 \times 10^{-9}$, will help the mobile devices save local resources.

CHAPTER 4: PROPOSED RADIO-SERVICE HANDOFF PROTOCOLS IN MEC NETWORKS

4.1 The Proposed Fast Radio-Service Hanoff Protocol

In this section, the proposed fast radio-service handoff protocol for MEC-based real-time mobile AR applications is described in detail. First, we introduce the key ideas of our protocol. Second, an overview of the proposed protocol is presented. Finally, we show the proposed feature database training strategy and feature mapping algorithm in the proposed protocol, respectively.

4.1.1 Key Ideas

During a radio-service handoff process, a mobile user has to identify the AP that it will hand off to first, in order to make a further decision on where its offloading service should be rebuilt. Thus, during a service rebuilding process, the radio handoff can not be performed in parallel with the service handoff, which makes the total service rebuilding time long. To overcome this problem, we can accurately predict the target MEC before the mobile user is triggered a radio handoff. Furthermore, in order to accelerate the service handoff speed, it is necessary to select an encapsulation mechanism which is lightweight enough for performing a fast service handoff and can provide an isolated running environment for the offloading service on the MEC.

The first key idea of our proposed protocol is to predict a mobile user's target MEC by leveraging the features that are extracted from the mobile device's camera captured frames. Almost all mobile users who play MEC-based mobile AR applications need to offload camera captured frames to the connected MEC, in order to match those frames against a pre-existing image database in real-time, or achieve the camera's 6D

transformation by computing the frame difference of two adjacent offloaded frames. In other words, after receiving offloaded frames, the MEC needs to extract every unique feature in each frame using a computer vision algorithm, such as SIFT. These extracted features are used for achieving different computation results based on user offloading services. Therefore, those already extracted features on the source MEC provide an excellent resource for us to predict the mobile user's target MEC without any additional cost at the mobile user side. Although some previous work [130–133] proposed several radio handoff prediction approaches in traditional wireless networks, they all are limited to the network environment without the aid from captured frames. To the best of our knowledge, our work is the first one that proposes using the captured frame features to predict mobile users' target MEC.

The second key idea of our proposed protocol is encapsulating offloading services via Docker which is a composing container engine on MEC. Docker satisfies the requirements for a suitable encapsulation mechanism in MEC. Firstly, it provides an isolated environment based on OS-level virtualization for running applications. Secondly, Docker is a lighter-weight encapsulation mechanism compared with VMs. Since Docker enables layered storage inside containers, which supports copy-on-write (CoW) strategy. Each Docker image references a list of read-only layers that represent file-system differences. Layers are stacked on top of each other to form a base for a containers root file-system. The Docker storage driver, e.g., advanced multi-layered unification file-system (AUFS), is responsible for stacking these layers and providing a single unified view. When we create a new Docker container, we add a new and thin writable layer on top of the underlying stack of layers present in the base docker image. All changes made to the running container, such as creating new files, modifying existing files or deleting files, are written in this thin writable container layer. Therefore, we only need to transfer this thin container layer and the run-time memory state during the service handoff. Furthermore, all the Docker images are available through

the public centralized image server, such as Docker Hub. Therefore, before initiating the service handoff, application container base image layers can be downloaded from the public image server first. Both of these Docker's properties may speed up the service rebuilding process. As far as we know, only one prior research paper [129] discussed the efficiency of using the Docker container for the MEC service handoff. However, this paper only considered the latency of the service handoff, not the whole service rebuilding process.

4.1.2 Overview of the Proposed Protocol

Figure 4.1 shows the overview of the proposed fast radio-service handoff protocol for MEC-based mobile AR applications, as well as multiple processing stages described below:

(S1) *Service rebuilding preparation triggered on source MEC.* Once a mobile user is associated with a MEC, a Docker container is created on this MEC. We call the MEC which processes the offloading workloads, the *source MEC*. Then the mobile user starts sending its captured frames and offloading computation to the Docker container running on the source MEC. When the mobile user moves away from the source MEC, the RSS at the mobile user gradually goes down. A request for service rebuilding preparation is then triggered at the source MEC when the user's RSS is below a threshold γ_1, i.e., the mobile user is at location a as shown in Figure4.1.

(S2) *Match extracted features against pre-categorized features in database.* After the source MEC is triggered a request for service rebuilding preparation, a feature mapping process is initiated. In the mapping process, features extracted from the latest frames that the mobile user offloads to the source MEC are matched against the pre-categorized features in database.

(S3) *Predict target MEC.* The source MEC achieves the prediction result through our proposed feature mapping algorithm. We explain the proposed feature mapping algorithm in detail in Section 4.1.4.

(S4) *Download base images through Docker Hub.* After achieving the predicted target MEC, the source MEC sends a downloading request to it. Then, the target MEC starts downloading the base images through Docker Hub.

(S5) *Radio handoff and service handoff triggered.* When the mobile user reaches location b, a radio handoff is triggered at the associated AP since the RSSI is below the threshold γ_2. Meanwhile, a service handoff is triggered at the source MEC.

(S6) *Snapshot run-time memory and stop container.* Upon initiating the service handoff, the source MEC snapshots the latest run-time memory states and stops the container. From this point, the container stops running on the source MEC and the mobile user loses the offloading service provided by the source MEC.

(S7) *Transfer container layer and run-time memory to the target MEC.* After we stop the container that runs on the source MEC, its writable container layer will not be changed. Thus, we transfer the container layer contents to the target MEC. Simultaneously, the run-time memory are also transferred to the target MEC.

(S8) *Restore the offloading service on the target MEC.* After finishing all transfers, the container is restored on the target MEC. In other words, the service handoff has finished in this moment. If the mobile user has been already associated with the AP that is attached to the target MEC, it can start to be served. Otherwise, the target MEC has to wait for the mobile user completing its radio handoff.

(S9) *Remove container on source MEC.* Finally, we need to clean up the footprint of the offloading service on the source MEC by removing the container. And the whole process of the service rebuilding is completed.

In addition, our proposed fast radio-service handoff protocol can be utilized in both stateful service rebuilding and stateless service rebuilding scenarios, where the former contains all stages in the proposed protocol, while the latter contains all stages except S6.

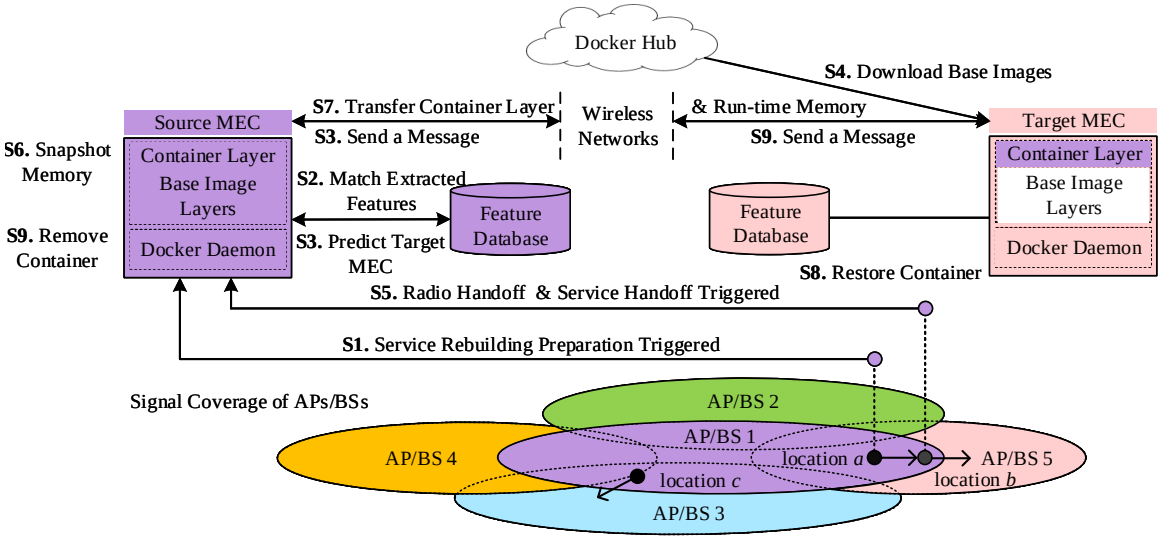


Figure 4.1: Overview of the proposed fast radio-service handoff protocol.

4.1.3 Strategy for Feature Database Training

Before the MEC is able to provide the target MEC prediction service for mobile users, we need to first train a feature database in each MEC. Our proposed feature database training strategy is shown in Figure4.2. We assume N target MECs are around a source MEC. The training process is divided into three stages, including triggering, caching, and categorizing.

Firstly, since RSSI on a mobile user indicates how well the mobile user can hear its remote connected AP. RSSI is an efficient resource for the source MEC to estimate if associated users have trends for requiring a service rebuilding service. Thus, we choose RSSI as the trigger for initiating the training service on the source MEC. In other words, when a mobile user's RSSI is below the threshold γ_1, the source MEC starts training its feature database. Secondly, the source MEC caches all the extracted features from mobile users' offloaded frames, until mobile users are triggered service handoffs (i.e., $RSSI \leq \gamma_2$). We define the area with RSSI in the range of $[\gamma_2, \gamma_1]$ as the *feature caching area* and the area with RSSI larger than γ_1 as the *safe area*. Thirdly, after mobile users complete service rebuilding and start being served on the

target MECs, the source MEC categorizes these cached features according to the corresponding mobile users' target MECs. The categorization result is described as a set $C = [c_1, ..., c_N]$.

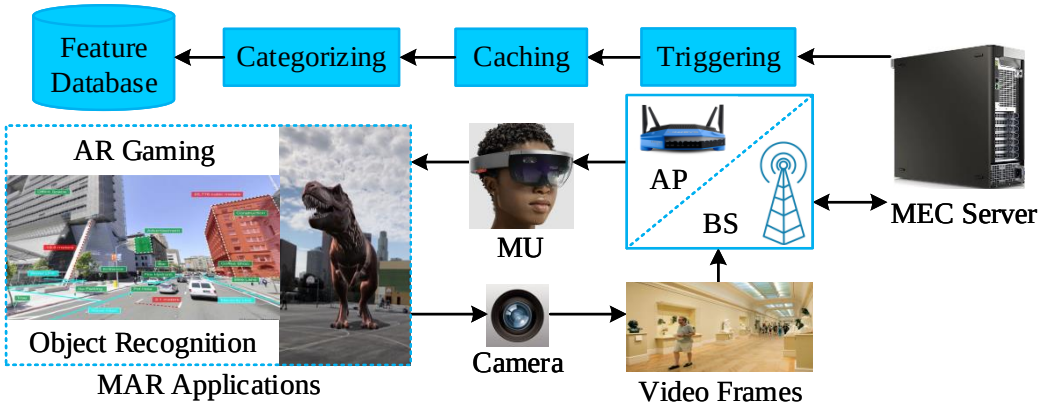


Figure 4.2: Feature database training process.

However, there exist some issues when implementing our proposed feature database training strategy in practical scenarios. Firstly, if a mobile user's feature caching time duration is too long, it may cause the training result being inaccurate. For example, as shown in Figure4.3, Path 1 is a moving path of a mobile user, where the user is associated with the source MEC originally. After being triggered a service rebuilding, its offloading service is finally handed off to target MEC 1. And all the cached features are categorized as the benchmark used for indicating that mobile users whose offloaded frames contain the same features have trends to handoff to target MEC 1. However, some features, such as feature 1, that are cached much earlier than mobile users being triggered service handoffs, might mislead target MEC predictions. To overcome this issue, we set a time period T_1, where database only categorizes features that are cached within this time period, depicted as the red dashed line in Figure4.3. Features that are cached T_1 period before triggering service handoffs would be abandoned. Secondly, as shown in Figure4.3, features in the purple area seem to belong to both target MEC 1 and target MEC 2, intuitively. But, because mobile devices usually are associated with the AP with higher RSSI and the feature categorization time is

limited by T_1. The actual training result of feature databases is shown as Figure4.4.

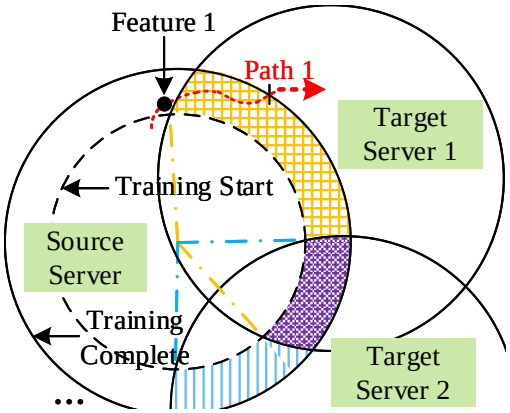


Figure 4.3: Basic training result

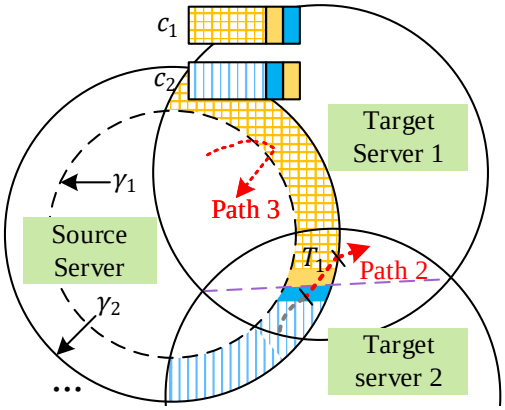


Figure 4.4: Advanced training result

4.1.4 Proposed Feature Mapping Algorithm

When a MEC has a completed trained feature database, it can offer target MEC prediction services to its attached mobile users. We design a feature mapping algorithm which is shown in Algorithm 3.

We assume the mapping iteration is executed every T_0 ms, periodically. At each iteration, the algorithm first checks the mobile user's RSSI. If it is below the threshold γ_1, the source MEC will trigger a target MEC prediction service for the mobile user. We use $flag_1$ to represent that the user is initiated a prediction service and α to represent the number of executed prediction iterations. Then, the source MEC collects a feature set F that are extracted in the latest user-offloading frames. F will be matched against the source MEC feature database D. After completing the matching process, the algorithm calculates a ratio matrix $R = [r_1, ..., r_N]$, which represents that, for example, the proportion of features that belong to c_1 in F is r_1. If there exists a MEC i, where r_i is not less than a fix ratio β, the prediction result in this iteration will be MEC i. Then i is recorded in the set Φ. Since we set $\beta \geq 0.5$, there is only one predicting target MEC in each iteration.

Furthermore, in our proposed mapping algorithm, prediction iteration would be stopped in two scenarios. The first one is the moment when the mobile user's RSSI

Algorithm 3: Feature Mapping

```
1  flag_1 = 0, flag_2 = 0, α = 0, Φ = ∅, φ = 0
2  for every T_0 ms do
       /* Check the RSSI of the mobile user */
3      if RSSI ≤ γ_1 ∧ flag_2 = 0 then
           /* Start a prediction */
4          if flag_1 = 0 then
5              flag_1 = 1
6          α = α + 1
7          Save the latest extracted features in F
8          Match F against database D
9          Calculate ratio matrix R = [r_1, ..., r_N]
10         if ∃ MEC i s.t. r_i ≥ β then
               /* Prediction result in this iteration is MEC i */
11             Φ(α + 1) = i
12         if RSSI ≤ γ_3 then
               /* 1st scenario of stopping prediction iterations */
13             flag_2 = 1
               /* Target MEC */
14             φ = j s.t. MEC j with the largest prediction number in Φ
15         else
16             if α ≥ λ_1 then
                   /* 2nd scenario of stopping prediction iterations */
17                 if ∃ MEC k s.t. the prediction number within Φ[α + 1 − λ_1, α + 1] is larger
                    than λ_2 then
18                     flag_2 = 1
                       /* Target MEC */
19                     φ = k
20     else if RSSI > γ_1 ∧ flag_1 = 1 then
           /* The mobile user goes back to the safe area */
21         flag_1 = 0, flag_2 = 0, α = 0, Φ = ∅, φ = 0
```

is below the threshold γ_3, where $\gamma_2 \leq \gamma_3 \leq \gamma_1$. And the algorithm choose the MEC with the largest prediction number in set Φ as the final prediction result. Another scenario is that the number of prediction iterations for the mobile user is larger than the threshold λ_1. And also, there exists a MEC k whose prediction number within $\Phi[\alpha + 1 - \lambda_1, \alpha + 1]$ is larger than the threshold λ_2. In order to achieve no more than one prediction result, we set $\lambda_2 \geq 0.5\lambda_1$. In addition, we use $flag_2$ to indicate the mobile user completes its prediction service on the source MEC.

There exists a special case that is depicted as Path 3 in Figure4.4. After the mobile

user being triggered a target MEC prediction service, it goes back to the safe area. We use the prediction initialized flag, $flag_1$, and $RSSI$ to identify mobile users in this special case. For these users, the source MEC just abandons its previous prediction result and waits for the next prediction service being triggered for them.

4.1.5 Analysis

In the proposed fast radio-service handoff protocol, some parameters might affect the performance of the service rebuilding process.

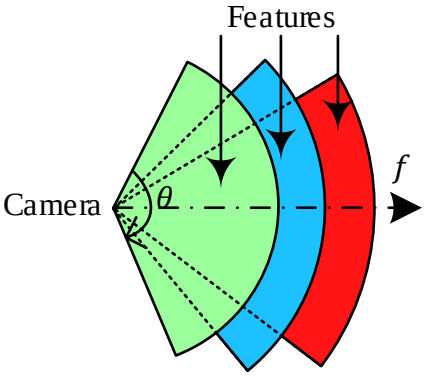


Figure 4.5: Effect of camera's properties

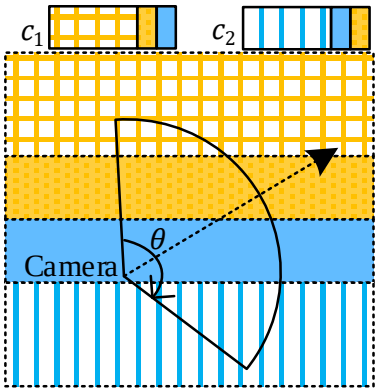


Figure 4.6: Effect of feature ratio β

First, currently most mobile devices support digital or optical zoom. Thus, in real world, what features and the number of features can be extracted from the user-offloading frames usually depends on the focus, ρ, and the angle of view, θ, of the mobile device's camera, as shown in Figure4.5. When a mobile user operates different AR applications or the same AR application running in different scenarios, camera's focus and angle of view might be changed. A good feature mapping algorithm should be able to maintain a high prediction precision and a short prediction time within different values of camera's focus and angle of view.

Second, the prediction time is influenced by the requirements of stopping prediction iterations, i.e., λ_1, λ_2, and γ_3. With larger λ_1 and λ_2, or γ_3, the mobile user may achieve a higher prediction precision, but it may spend much more prediction time. Therefore, even though we obtain a accurate predicted target MEC, the time left for

preparing the offloading service on the target MEC, e.g., downloading base images, may be not enough.

Third, the feature ratio β also may affect the prediction precision and latency in some cases. For example, in Figure4.4, a mobile user is moving through path 2. If β is set with a large value (e.g., 0.9), the mobile user may not get prediction results in some iterations or, even worse, in every iterations until reaching the threshold γ_3 and stopped prediction service. Since its extracted features contains very close number of features in both c_1 and c_2, as shown in Fig4.6, where neither r_1 nor r_2 is larger than β. Thus, the prediction time may be increased dramatically and the mobile user may not get a predicted target MEC in the end of the prediction. However, if β is set with a small value, the prediction precision may be lower.

4.1.6 Performance Evaluation

In this section, we evaluate the performance of the proposed service rebuilding scheme. We first implement our proposed scheme on a testbed and demonstrate the reduction of the service rebuilding latency. Since evaluating the performance of the proposed feature mapping algorithm in multiple different scenarios needs a relatively huge number of mobile users, which is very costly, we use extensive simulations to evaluate the prediction accuracy and latency of the proposed feature mapping algorithm.

4.1.6.1 Experimental Setup and Results

Our testbed consists of five components, a mobile user and two cloudlets attached with two APs, as shown in Figure4.7. Firstly, we use Intel NUC kits as the mobile user, which continues offloading a pre-captured video to the source cloudlet. Secondly, two Nvidia Jetson TX2 kits are conducted as a source cloudlet and a target cloudlet, respectively, and both of them are attached with a Linksys N900 access point for communicating with the mobile user. A stateless application, YOLO, runs in the

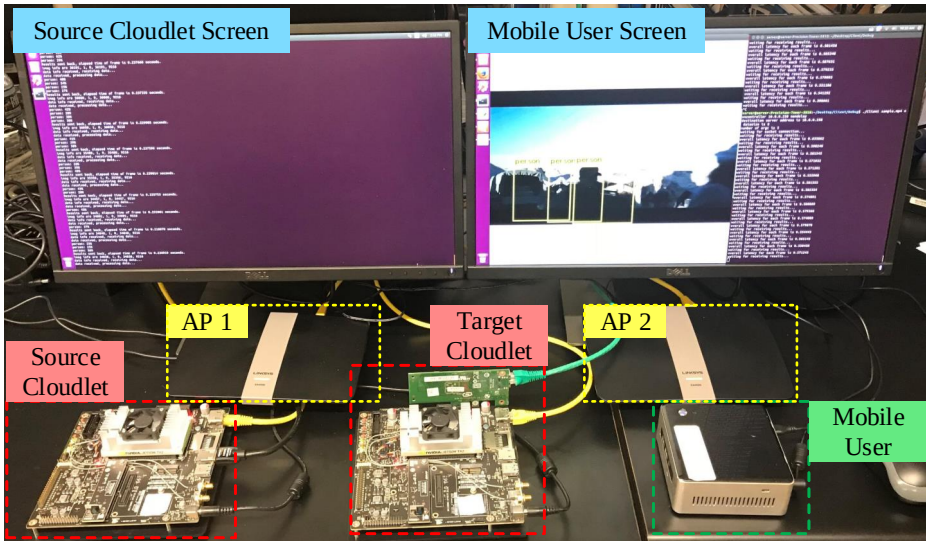


Figure 4.7: Testbed implementation.

source cloudlet which offers an object recognition service for the mobile user. We implement the proposed feature mapping algorithm in the Nvidia Jetson TX2 kits and pre-trained a small feature database in the source cloudlet. Because of the time limitation, we do not implement the Docker container in our testbed.

We plot the experimental results in Figure4.8. Figure4.8a and Figure4.8b show the service rebuilding latency of the conventional service rebuilding process and the service rebuilding process with our proposed scheme, respectively, where the latency of the mobile user re-achieving its offloading service on the target cloudlet is around 41.2sec and 14.1sec, respectively. Our proposed scheme decreases the whole service rebuilding latency by around 65.8% compared with the conventional way. As shown in Figure4.8b, since the source cloudlet gets a predicted target cloudlet ahead of the mobile user triggered a radio handoff, the target cloudlet could start preparing the service environment for the mobile user before connecting with it. And also, the service handoff is performed in parallel with the radio handoff, during the service rebuilding process.

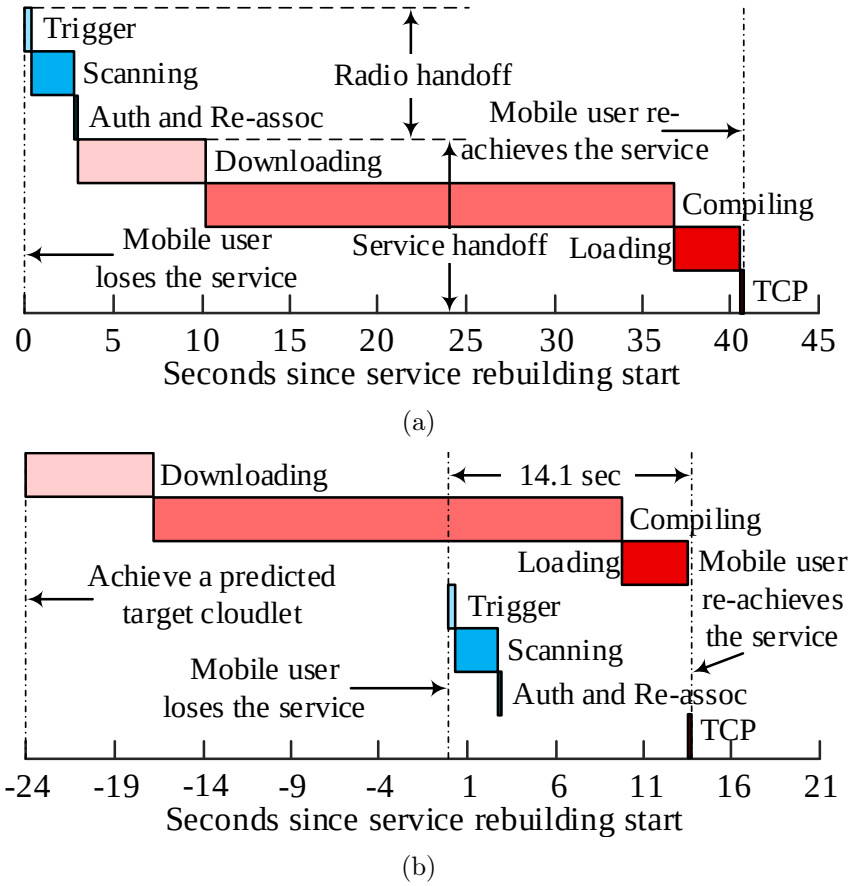


Figure 4.8: Experimental service rebuilding latency. (a) Conventional service rebuilding process; (b) Proposed service rebuilding scheme.

4.1.6.2 Simulation Setup and Results

The simulation setting is as follows. We consider a cloudlet-based mobile AR environment with 8 target cloudlets around a source cloudlet, where they are regularly distributed in a square area. The length of the simulation area is $450m$. 1000 mobile users who are playing cloudlet-based mobile AR applications move randomly within the area following the Random Waypoint model. The moving speed of each mobile users is in the range of $[1, 2]\, m/s$. In addition, we use the indoor path-loss model which is express as

$$P_L = 20 \log_{10} f + 10n \log_{10} d - 28(dB), \tag{4.1}$$

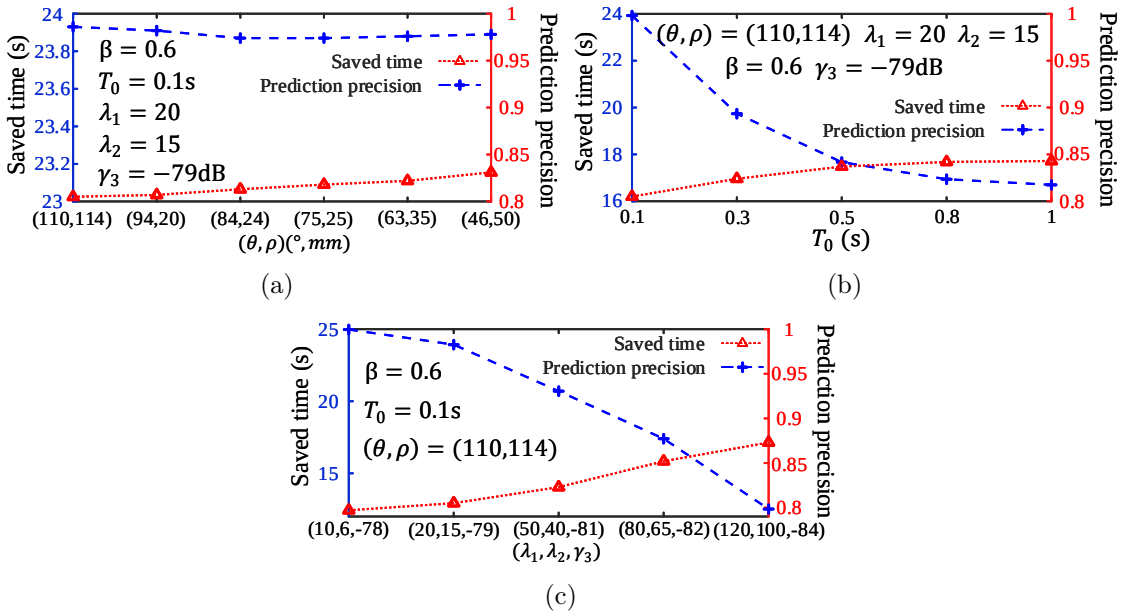


Figure 4.9: Simulation results. (a) Impact of the properties of device's camera; (b) Impact of the prediction iteration period; (c) Impact of the prediction iteration stop requirements.

where P_L is the RF signal propagation path-loss based on distance d between the source cloudlet and the mobile user, f is the carrier frequency in MHz, and n is the path-loss exponent. In our simulation environment, $n = 6$ and $f = 2400$ MHz. Furthermore, we trained a categorized feature database used for feature mapping in the source cloudlet, which contains 10 thousand unique features.

Figure4.9 shows the simulation results. The left Y-axis represents the time period from the moment when a mobile user achieves a predicted target cloudlet to the moment when the mobile user is really triggered a radio handoff, and we call this time period as saved time. Obviously, the shorter prediction latency the longer saved time the target cloudlet obtains to prepare mobile users' offloading services. The right Y-axis represents the prediction precision.

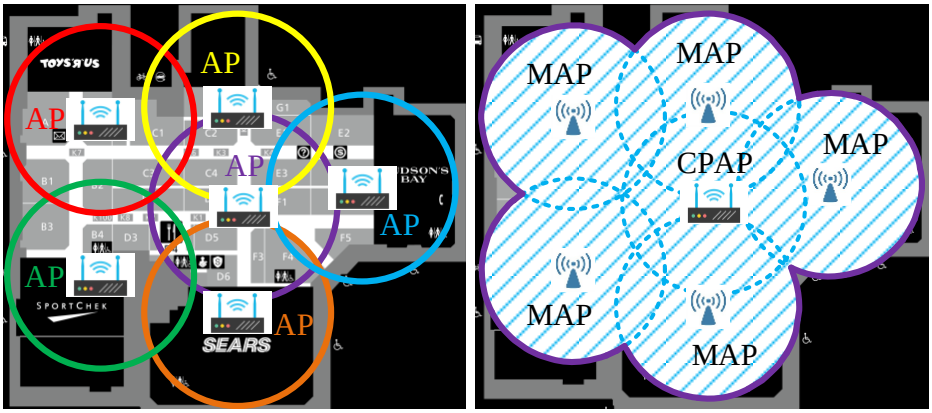
Figure4.9a illustrates the robustness of the proposed algorithm under different mobile device's camera properties, focus ρ and angle of view θ. when ρ and θ are changed, our proposed algorithm can still achieve a high prediction precision, over 80% and

a long saved time, around $24sec$. Figure4.9b depicts the impact of the prediction iteration period T_0 on the performance of the proposed prediction algorithm. When the period of the prediction iteration increases, the prediction precision increases but the length of the saved time decreases. Since, under the same prediction stop requirements, increasing the period of the prediction iteration indicates that the source cloudlet might spend a longer time for prediction and achieve more efficient captured features. Figure4.9c shows the impact of the prediction iteration stop requirements. Simulation results confirm our analyses in Section 4.2.4.

4.2 The Proposed Energy-Efficient Radio-Service Hanoff Protocol

In this section, an energy-efficient and seamless radio-service handoff protocol is proposed for MEC-based real-time mobile AR applications is described in detail. Ultimately, the reason for MUs consuming a lot of energy for mobility management service is performing frequent and periodic scans. Since the signal coverage and capacity of a single AP is considerably limited, MUs have to frequently and periodically update their knowledge of surrounding APs and switch between different APs to maintain their wireless connections. For example, consider that an MU is hanging out in a shopping mall, and a WLAN is deployed as shown in Figure 4.10(a). A mobility management service is triggered when the MU walks across the signal boundary of APs, which is depicted as colored circles in Figure 4.10(a). When the MU moves, for instance, walking from the TOYS R US store (up left corner) to the SEARS store (down right corner), the MU has to perform at least two handoffs, including channel scans, to maintain its wireless connection. However, in practical scenarios, users' path may not just follow a straight line. They may erratically visit the stores they are interested in one by one. Hence, MUs might perform much more number of scans and handoff attempts, which may consume a lot of energy of mobile devices and badly affect user experience because of the high handoff latency and load transfer delay.

We have conducted an energy consumption measurement study of mobile devices



(a) (b)

Figure 4.10: Comparison of WLAN deployments. (a) Conventional WLAN deployment; (b) Proposed BELL deployment.

during roaming, including scanning and handoff, using Motorola Nexus 6p smart phones. The measurement shows that performing a successful handoff with one scanning attempt consumes approximate 1.5 Joule. This energy consumption may dramatically increase if the Nexus 6p attempts to scan channels for more than once, i.e., the first scanning attempt fails to capture an appropriate AP to connect to. We then utilize this measurement result to conduct a trace-driven simulation, where 500 MUs walk in an area with 36 APs deployed for 5.5 hours. Each mobile device is attached with a 3000mAh battery and follows the Random Waypoint model. Figure 4.11 depicts the energy consumption of performing mobility management services. Each device's total energy consumption is sorted from low to high as shown in the X-axis. From the figure we can see that most devices consume approximate 8% of the total battery life for performing mobility management. Some mobile devices with high mobility consume a whopping 13.4% of the total battery life, 400mAh, for these two WLAN management services, which is equivalent to the battery drain for watching video around 1 hour. Therefore, reducing the energy consumption of mobility management services is crucial and will help MUs survive longer within the WLAN environment.

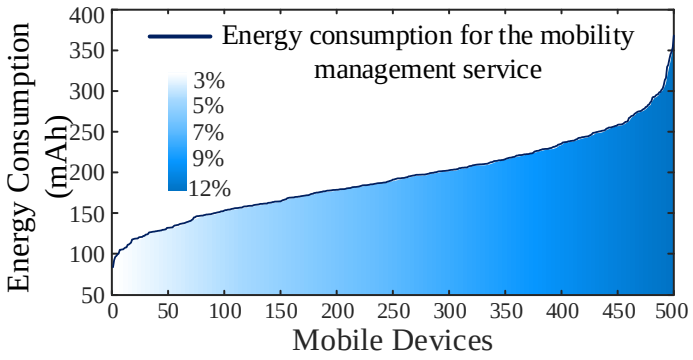


Figure 4.11: Mobile devices' energy consumption for mobility management services.

To overcome the above issue, we propose BELL that virtually extends the signal coverage of a single AP. As shown in Figure 4.10(b), the signal coverage of the AP located in the center is extended by the support of other APs around it. These APs are called *mirror access points* (MAPs) for the central one. They have the same SSID and MAC address with the center AP, named *copied physical access point* (CPAP). APs that generate these MAPs are called host physical access points. MAPs and the CPAP compose the proposed novel WLAN system, BELL. Since an MU would not filter out beacons that are generated from APs obtaining the same SSID and MAC address with its associated AP, the MU can receive beacons broadcast by MAPs and the CPAP in a BELL without scanning. In other words, MUs in a BELL would consider they are always in a single AP. Based on BELL, we propose an energy-efficient mobility management service, *BELL-handoff*.

4.2.1 AP-side Handoff Protocol

At the heart of the proposed BELL-handoff is to create an evenly spaced periodic schedule of beacon periods for MAPs and the CPAP. Consider a BELL system that has $(N+1)$ number of APs (i.e., N MAP and a CPAP). The CPAP is operating in channel v, and its beacon is broadcast at time $\phi_{cpap} \in [0, T]$, periodically, where T is the beacon interval. Each MAP has an index number $i \in \{1, \ldots, N\}$ which is allotted clockwise. Let MAPs with an odd index operate in channel p and MAPs with an even

index operate in channel q. Channel v, p, q are three non-overlapping channels (e.g., $\{v, p, q\} = \{1, 6, 11\}$ in the 2.4 GHz band). If N is even, then

$$\phi_i = \begin{cases} \phi_{cpap} + \frac{T}{\chi}, & \text{if } i = 2n + 1 \\ \phi_{cpap} + (\chi - 1) \cdot \frac{T}{\chi}, & \text{otherwise} \end{cases}$$

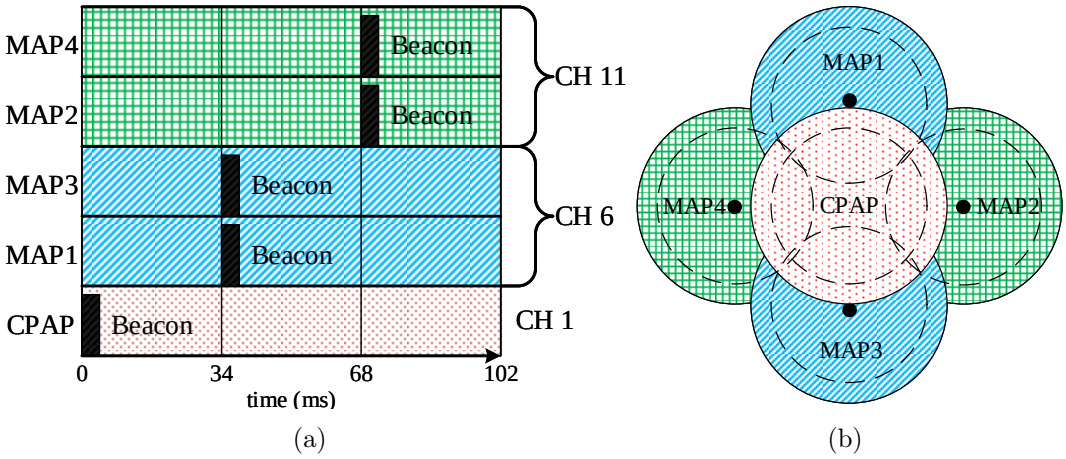
where $n \in \{0, \ldots, \frac{N-1}{2}\}$, ϕ_i denotes the beacon broadcasting time of MAP i, and χ denotes the number of segments in the beacon interval. If N is odd,

$$\phi_i = \begin{cases} \phi_{cpap} + \frac{T}{\chi}, & \text{if } i = 2n + 1 \\ \phi_{cpap} + (\chi - 1) \cdot \frac{T}{\chi}, & \text{if } i = N \\ \phi_{cpap} + (\chi - 2) \cdot \frac{T}{\chi}, & \text{otherwise} \end{cases}$$

where $n \in \{0, \ldots, \frac{N-2}{2}\}$. Furthermore, when N is even, $\chi = 3$ (i.e., the total number of operating channels in the BELL), otherwise $\chi = 4$.

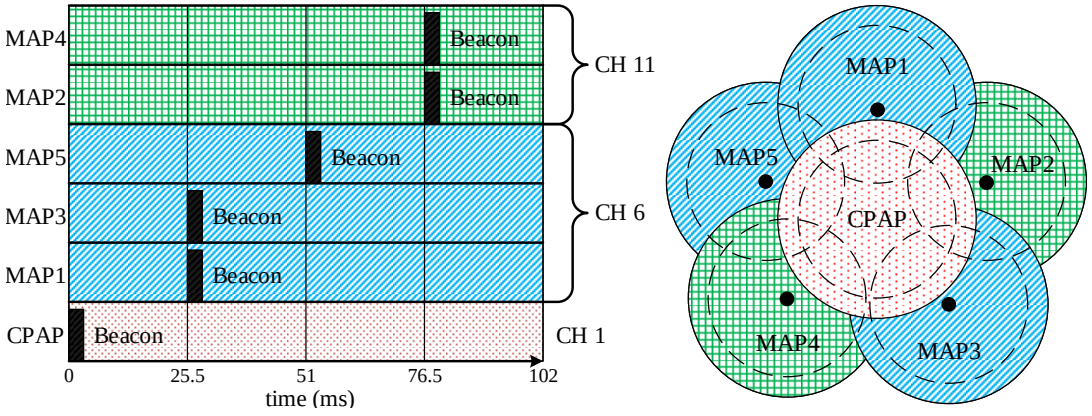
Example 1. Consider a BELL system with even number of MAPs (e.g., $N = 4$ in this example) operating in the 2.4 GHz band, as shown in Figure 4.12(b). APs' physical signal transmission range is depicted as the solid circles, and BELL-handoff triggering range is depicted as the dashed circles. Assume the CPAP, MAP $\{1, 3\}$, and MAP$\{2, 4\}$ are operating in channel 1, 6, and 11, respectively. $\chi = 3$, $\phi_{cpap} = 0$. All the APs in the BELL are set for $T = 102ms$. Therefore, their beacon broadcasting time is set to be $\{34, 68\}ms$, as shown in Figure 4.12(a). Furthermore, there would be no collision for different MAPs broadcasting beacons at the same time, since they do not have overlapping coverage area.

Example 2. Consider a BELL system with odd number of MAPs (e.g., $N = 5$ in this example) operating in the 2.4 GHz band, as shown in Figure 4.13(b). Assume that the CPAP, MAP $\{1, 3, 5\}$, and MAP$\{2, 4\}$ are operating in channel 1, 6, and 11,



(a) (b)

Figure 4.12: Example of a BELL with even number of MAPs. (a) BELL deployment; (b) Beacon broadcast schedule.



(a) (b)

Figure 4.13: Example of a BELL with odd number of MAPs. (a) BELL deployment; (b) Beacon broadcast schedule.

respectively. $\chi = 4$, $\phi_{cpap} = 0$, and $T = 102ms$. Therefore, their beacon broadcasting time is set to be $\{25.5, 51, 76.5\}ms$, as shown in Figure 4.13(a).

4.2.2 User-side Protocol

MUs in BELL perform our proposed energy-efficient BELL-handoff. Consider an MU u currently located in MAP i $(1 < i < N)$ and a BELL-handoff is triggered since the received signal strength (RSS) at the MU is below a threshold γ. The possible candidate AP set for MU u to connect to is {MAP $i+1$, MAP $i-1$, CPAP}. Therefore, MU u only needs to switch to the corresponding channels one by one,

and waits for the beacon at each beacon broadcasting time according to the BELL beacon schedule. MU u stops this channel switching when successfully receives a beacon frame. For example, consider a BELL system as deployed in Figure 4.12. Assume a BELL-handoff is triggered when MU u is in MAP 3 at time t_0. It first switches its channel to CH 11 and completes its BELL-handoff if it receives a beacon frame at $(t_0 + 34)ms$. Otherwise, it switches its channel again to CH 1 and waits for receiving the beacon broadcast at $(t_0 + 68)ms$. Therefore, MUs can successfully roam in BELL without performing scanning, achieving energy-efficient handoffs. Note that although after a BELL-handoff, an MU may be handed off to an MAP with RSSI below γ, it can trigger another BELL-handoff soon and connect to an MAP with better RSSI.

4.2.3 Implementation Challenges

There exist some issues when implementing our presented BELL system both in APs and in user devices.

Challenge 1. *To have an accurate beacon schedule over long time scales, it is necessary to overcome the clock drift that exists in each AP.*

We address this issue by proposing an AP *clock synchronization* (CS) protocol. In the proposed CS protocol, all the MAPs in a BELL adjust their clock every ψ, by synchronizing its local Timing Synchronization Function (TSF) timer with a common clock which is the local TSF timer of the CPAP. ψ is an integral multiple of the beacon interval T. Assume that the CPAP always broadcast its beacons first in every beacon interval. In addition, at time σ, MAP i and the CPAP have a local TSF value of $t_i(\sigma)$ and $t_{cpap}(\sigma)$, respectively. Therefore, if N is even, then

$$t^i_{offset}(\sigma) = \begin{cases} |t_i(\sigma) - t_{cpap}(\sigma)| - \frac{T}{\chi}, & \text{if } i = 2n+1 \\ |t_i(\sigma) - t_{cpap}(\sigma)| - \frac{(\chi-1)\cdot T}{\chi}, & \text{otherwise} \end{cases}$$

where $n \in \{0, \ldots, \frac{N-1}{2}\}$, $t^i_{offset}(\sigma)$ denotes the timing offset value for MAP i at time σ. If N is odd, then

$$t^i_{offset}(\sigma) = \begin{cases} |t_i(\sigma) - t_{cpap}(\sigma)| - \frac{T}{\chi}, & \text{if } i = 2n+1 \\ |t_i(\sigma) - t_{cpap}(\sigma)| - \frac{(\chi-1)\cdot T}{\chi}, & \text{if } i = N \\ |t_i(\sigma) - t_{cpap}(\sigma)| - \frac{(\chi-2)\cdot T}{\chi}, & \text{otherwise} \end{cases}$$

where $n \in \{0, \ldots, \frac{N-2}{2}\}$. Then,

$$\bar{t_i}(\sigma) = t_i(\sigma) - t^i_{offset}(\sigma), \tag{4.2}$$

where $\bar{t_i}(\sigma)$ denotes the corrected TSF for MAP i at time σ.

Challenge 2. *How does an MU acquire enough MAPs' information to achieve a successful BELL-handoff?*

In the proposed BELL system, MUs do not need to know the index of a particular MAP. Instead, acquiring BELL's beacon broadcasting schedule and its corresponding channels is enough for performing a successful BELL-handoff. For example, consider a BELL as deployed in Figure 4.12 and an MU is in MAP 3. The MU only knows it is currently in channel 6. After a BELL-handoff is triggered, the MU just jump to channel 11 and channel 1 in order according to the beacon broadcasting schedule. Therefore, we only ask each MU acquiring a beacon broadcasting schedule while joining BELL, which is simple enough to implement.

Furthermore, the deployment of the BELL presented above is based on an assumption that, MAPs with an even index or an odd index can operate in the same channel and have the same beacon broadcast time, only if they do not have overlapping coverage area. For example, in Figure 4.12, MAP1 and MAP3 operating in the same channel and having the same beacon broadcast time do not have any overlapping cov-

erage area. However, if the density of MAPs in a BELL is increased, two neighboring MAPs with an even index or an odd index may have overlapping coverage area. To avoid the interference, their beacons need to broadcast at different time or they need to operate in different channels. Therefore, there exists a limitation of the maximum number of MAPs in a BELL: the minimal gap in time between two beacons that broadcast by different MAPs has to be larger than the MU's channel switching delay, i.e., $\frac{T}{\chi} > t_{cs}$, where t_{cs} is the MU's channel switching delay.

4.2.4 Analysis

1. Energy efficiency: MUs roaming in BELL do not perform scanning, which indicates that they do not need to send or receive probe request and response for the purpose of scanning. Therefore, the proposed BELL provides a more energy-efficient mobility management service than conventional WLAN (C-WLAN).

2. Handoff latency: the handoff latency in BELL is equal to the delay of waiting for the beacon frame, plus the delay for authentication and re-association which is the same as in traditional handoff. Excluding authentication and re-association delay,

$$PLS = \begin{cases} \{\frac{T}{\chi}, (\chi-1)\cdot\frac{T}{\chi}\}, & \text{if } N \text{ is even} \\ \{\frac{T}{\chi}, (\chi-2)\cdot\frac{T}{\chi}, (\chi-1)\cdot\frac{T}{\chi}\}, & \text{otherwise} \end{cases}$$

where PLS is the the possible handoff latency set. In our cases, T is set to be $102ms$. Thus, ideally, the maximum handoff latency, excluding authentication and re-association delay, is $68ms$, if N is even, and it is $76.5ms$, if N is odd.

4.2.5 Mutiple BELL Zones Scenario

In the mutiple BELLs scenario, several BELL WLANs coexist in a certain area. Each AP can be a CPAP, which indicates that a host physical AP would generate multiple different MAPs in it, as illustrated in Figure 4.14. The number of MAPs generated in a host physical AP is equal to the number of MAPs in this BELL.

Different BELL systems can have the same channel set. In addition, MUs roaming between different BELL WLANs perform conventional handoffs.

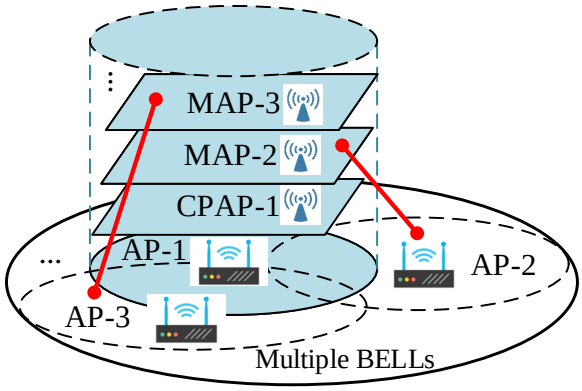


Figure 4.14: Multiple BELL zones.

4.2.6 Performance Evaluation

In this section, we evaluate the performance of the proposed BELL. We first implement our proposed BELL-handoff mobility management service on a testbed and demonstrate MUs' roaming performance improvement in BELL, including handoff energy consumption and latency, as compared to the conventional WLANs (C-WLANs). Since implementing a large-scale deployment of MAPs is very costly, we conduct extensive simulations to evaluate the performance of BELL-2M and mobile devices' battery life within a large-scale deployment of BELL.

4.2.6.1 Testbed and Experimental Setup

Our testbed is composed of three subsystems, MU and energy measurement system, BELL including a CPAP and an MAP, and AP clock synchronization system. Firstly, in the MU and energy measurement system, we use KEYSIGHT N6705B DC Power Analyzer for powering and measuring the energy consumption of a Raspberry Pi Model 3 connected with an ALFA AWUS036NHA USB Wi-Fi adapter, which is conducted as a test MU. The Wi-Fi adapter is driven by an open-source hardware driver, ath9k, and its MAC functionality is handled by the protocol driver mac80211,

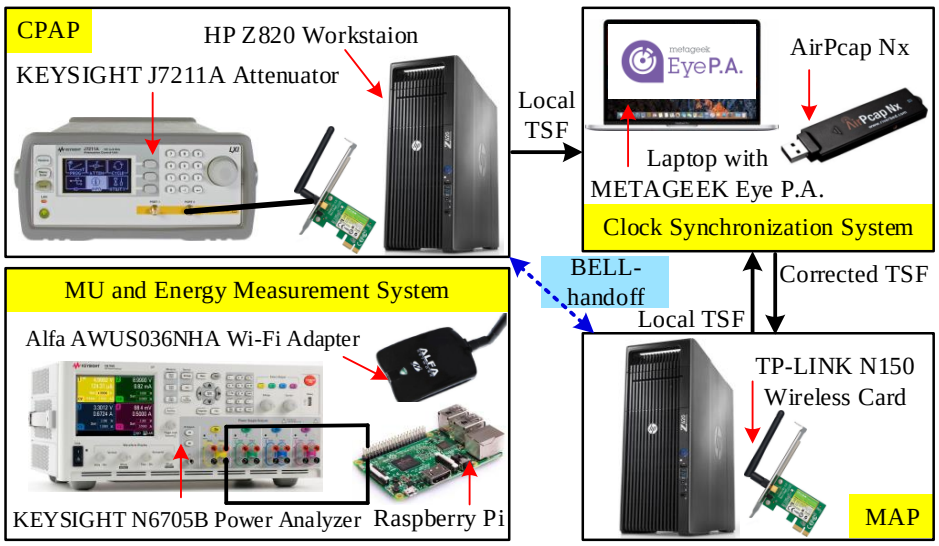


Figure 4.15: Testbed hardware architecture.

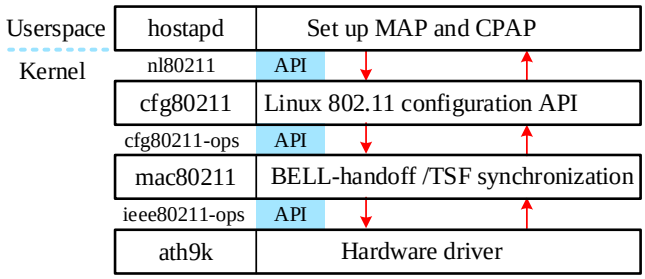


Figure 4.16: Testbed driver software structure.

which is where we implement BELL-handoff. We update our modified mac80211 to the Linux kernel which is then embedded into the test Raspberry Pi. Secondly, the tested BELL is setup with a CPAP and an MAP. These two are built in two workstations with the Linux operating system. Both workstations are connected with a TP-LINK N150 PCI-Express wireless card driven by ath9k and mac80211. Then, we use hostapd in userspace to set the configuration of these two APs. KEYSIGHT J7211A attenuator is used for emulating the mobility of the MU, e.g., decreasing the transmission signal strength of the CPAP emulates the case when an MU gradually moves away from the CPAP. Furthermore, to implement our AP's CS protocol, we modify ath9k by adding a function interface to rewrite the register that stores the dynamic local TSF of APs. Lastly, we use Airpcap Nx and its client software,

METAGEEK Eye P.A., for capturing control frames at the MAC layer and extracting local TSF values of the CPAP and the MAP to conduct the clock synchronization. The testbed hardware architecture and driver software structure are shown in Figure 4.15 and Figure 4.16, respectively.

We set the beacon interval of APs in BELL to be $102ms$. Thus, the beacon broadcast time of the CPAP and the MAP are 0 and $51ms$, respectively. Both APs operate in the 2.4 GHz band. The CPAP operates in channel 1 and the MAP operates in channel 6. Furthermore, the test Raspberry Pi is powered by 5.2V and its BELL-handoff threshold is set to be -80dB.

4.2.6.2 Mobility Management Service Performance

Energy Efficiency. We plot experimental results in Figure 4.17. The measurement current of the test Raspberry Pi without connecting any peripherals is approximately fixed at 0.21A. After connecting a USB Wi-Fi adapter, the Raspberry Pi's current is increased to around 0.3A, as shown in Figure 4.17a. This indicates that the Wi-Fi adapter consumes an approximately fixed current of 0.09A in idle state. Figure 4.17a shows the current measurement result of the Raspberry Pi within 250sec. We find that in approximately every 120sec, the mac80211 driver will trigger a periodic full channel scanning, which lasts around 9sec, as shown in Figure 4.17b. Each spike represents one scan on a channel. Since the MU still has a good-quality connection with its current associated AP, it switches back to the original channel to maintain its connection after scanning each channel. Figure 4.17c and Figure 4.17d compare the current of performing the 802.11 standard handoff within C-WLAN and the proposed BELL-handoff within BELL. Obviously, both the current value and duration of performing the proposed BELL-handoff is much smaller than that of performing the 802.11 standard handoff, which indicates that the proposed BELL provides energy-efficient mobility management service for MUs.

The calculated energy consumption during handoffs is depicted in Table 4.1. The

Table 4.1: Handoff energy consumption results

	Overall	Breakdown	
C-WLAN (periodic scanning)	10.138J		
C-WLAN (802.11 standard handoff)	4.366J	Trigger	0.902J
		Scan	2.809J
		Auth and re-assoc	0.655J
BELL (BELL-handoff)	0.748J	Channel switching	0.038J
		Idle	0.050J
		Auth and re-assoc	0.660J

Table 4.2: Handoff latency Results

	Overall	Breakdown	
C-WLAN (802.11 standard handoff)	3.06sec	Trigger	0.41sec
		Scan	2.62sec
		Auth and Re-assoc	24ms
BELL (BELL-handoff)	75.6ms	Channel switching	$<$ 1ms
		Idle	51ms
		Auth and Re-assoc	24ms

proposed BELL-handoff decreases the energy consumption by around 82.9% as compared with the 802.11 standard handoff protocol. Furthermore, we find that, in 802.11 standard handoff protocol, the handoff trigger is a number of continuously missed beacon frames instead of a fixed RSSI value. Therefore, it would consume an additional energy of 0.902J for triggering. The periodic scanning in 802.11 standard protocol consumes 10.138J energy. This would aggravate MUs' energy drain. However, in BELL, MUs perform neither periodic scanning nor scanning within mobility management services, which significantly reduces the handoff energy consumption of MUs.

Latency. The experimental results are shown in Table 4.2. The proposed BELL-handoff decreases the overall handoff latency from 3.06sec to 75.6ms, as compared with C-WLANs. We can see that, in C-WLANs, the scanning latency, 2.62sec, is the largest portion of the total handoff latency. In addition, since the handoff trigger

mechanism for the 802.11 standard handoff is to count the number of continuously missed beacons, the trigger delay, which is measured as 0.41sec, increases the total handoff latency. However, the delay of BELL-handoff is from three parts: the MU's channel switching delay, which is less than 1ms, the time for waiting for the new beacon, around 51ms, which depends on the beacon scheduling in BELL, and authentication and re-association delay, around 24ms. In total, the proposed BELL-handoff decreases the handoff latency by around 97.5%, as compared with the C-WLAN.

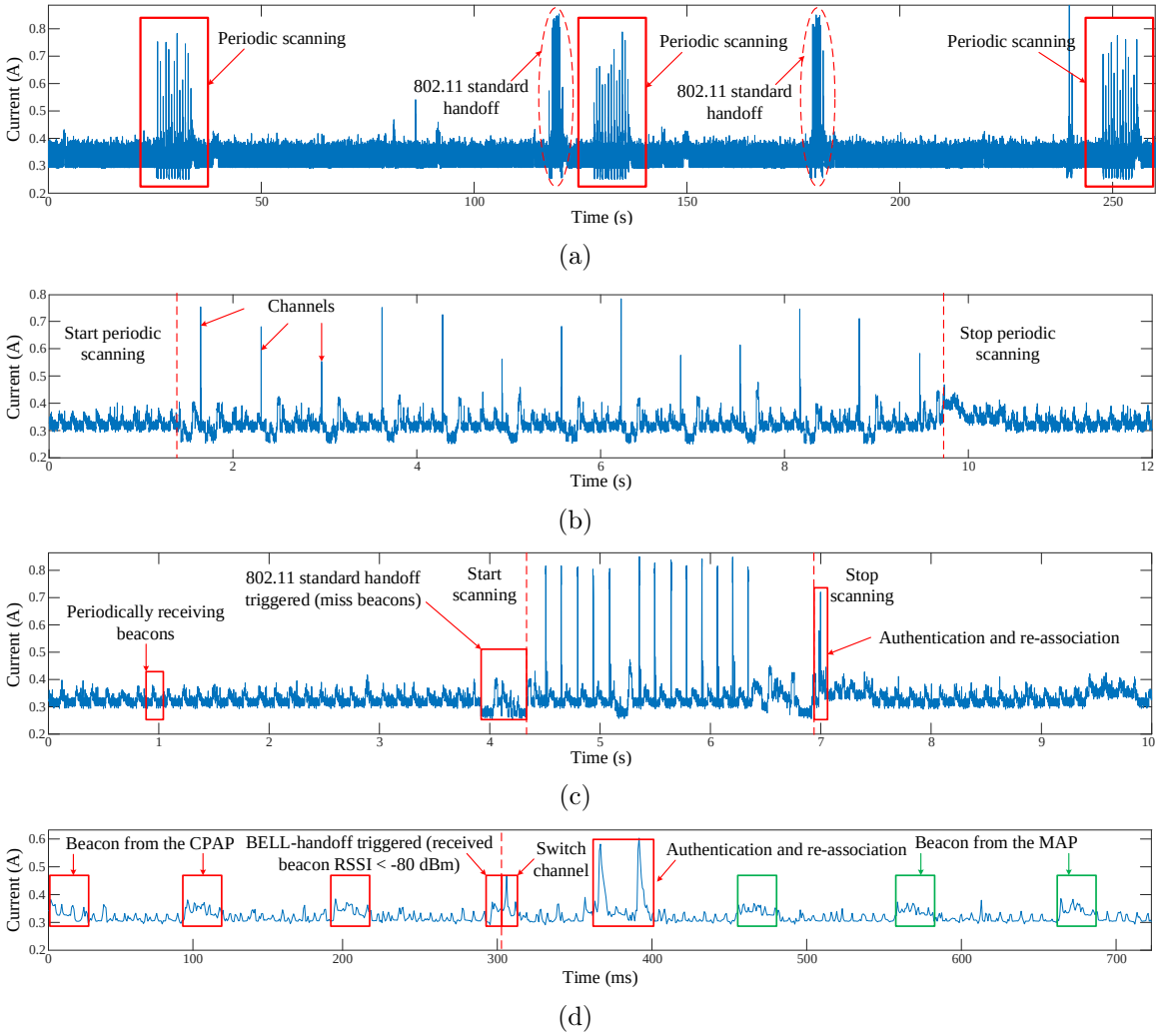


Figure 4.17: Measured current of MU. (a) Current of the test Raspberry Pi connecting a USB Wi-Fi adapter within C-WLAN; (b) Current of the test Raspberry Pi for performing a periodic full channel scanning within C-WLAN; (c) Current of the test Raspberry Pi for performing one 802.11 standard handoff within C-WLAN; (d) Current of the test Raspberry Pi for performing one BELL-handoff within BELL.

CHAPTER 5: PROPOSED RESOURCE ALLOCATION PROTOCOLS IN MEC NETWORKS

5.1 The Proposed Single Edge Server Resource Allocation Protocol

A single edge server radio resource allocation protocol, *E-Auto*, based on IEEE 802.11 to enable fast, stable, and accurate edge-assisted autonomous driving services for connected vehicles within any road conditions (e.g., high speed or traffic congestion) is proposed in this section. The parameters used in our analysis are listed in Table 5.1. We consider an edge-based connected vehicle network environment with an RSU server a and N connected vehicles that always have network activities. Denote N_{down} as the number of connected vehicles with downloading traffic (DCVs) (i.e., downloading entertainment videos) and N_{up} as the number of connected vehicles with uploading traffic (UCVs) (i.e., offloading their real-time camera captured video frames for object detection purpose), respectively. In addition, $N_{down} + N_{up} \geq N$, and a connected vehicle may have downlink and uplink traffic simultaneously. $\mathcal{N}$, $\mathcal{N}_{down}$, and $\mathcal{N}_{up}$ are denoted as sets of all connected vehicles, DCVs, and UCVs, respectively.

5.1.1 Key Factors in Edge-assisted Autonomous Driving

In this subsection, we define three key factors in an edge-assisted autonomous driving system. They can help us design the proposed *E-Auto* scheme.

Speed of Connected Vehicles: consider a UCV $n \in \mathcal{N}_{up}$, and denote $V_n^{t_i}$ as its moving speed at time t_i, as shown in Figure 5.1. UCVs are all equipped with a video camera with θ° horizontal field of view and l meter 3-D measurement range. When the UCV moves after a time period τ, the area of the surroundings that its camera can capture will change, which is denoted as $\Delta S_n^{t_i} = (2l - \theta \cdot d_n^{t_i}) \cdot V_n^{t_i}$. Obviously, the

Table 5.1: Notations used in E-Auto

Variable	Description
V	Speed of a connected vehicle (m/s)
fps	Frame rate (frames/sec)
ΔS	The area of camera captured different surroundings (m^2)
$k \times s$	Frame resolution of the offloaded video frame (pixels)
$\bar{k} \times \bar{s}$	Frame resolution of the downloaded video frame (pixels)
γ	The number of bits carried by one pixel (bits)
LU	Wireless network latency for offloading a video frame (s)
LD	Wireless network latency for downloading a video frame (s)
P_{tr}	Wireless transmit power of a connected vehicle (watt)
P_{rev}	Wireless receive power of a connected vehicle (watt)
EU	Energy consumption for offloading a video frame (J)
ED	Energy consumption for downloading a video frame (J)

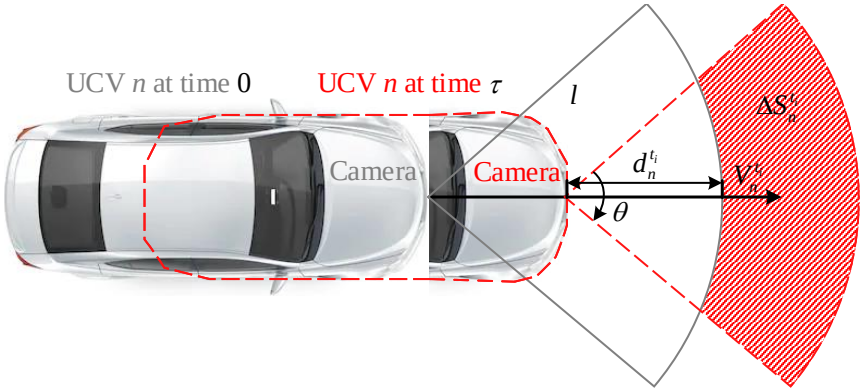


Figure 5.1: The area change of camera captured video frames during autonomous driving.

area of the camera newly captured surroundings is determined by the speed of the UCV. Consider two UCVs $n, m \in \mathcal{N}_{up}$ at time t_i. We define that

$$\sigma_{(n,m)}^{t_i} = \frac{\Delta S_n^{t_i}}{\Delta S_m^{t_i}} = \frac{(2l - \theta \cdot d_n^{t_i}) \cdot V_n^{t_i}}{(2l - \theta \cdot d_m^{t_i}) \cdot V_m^{t_i}}, \tag{5.1}$$

where, $d_n^{t_i} = V_n^{t_i} \cdot \tau$, $d_m^{t_i} = V_m^{t_i} \cdot \tau$.

Frame Rate: the frame rate, denoted as fps, is the number of frames that a UCV/DCV offloads/downloads to/from the RSU server per second for object detection/entertainment. **For UCVs:** When the speed of a UCV is high, the UCV must

offload more captured frames for a fast and accurate autonomous driving, since it obtains a large $\Delta S_n^{t_i}$. Consider two UCVs $n, m \in \mathcal{N}_{up}$ at time t_i, and let $fps_n^{t_i}$ and $fps_m^{t_i}$ be the minimum acceptable frame rate for maintaining their autonomous driving services, respectively. Thus, if we know the value of $fps_n^{t_i}$, $fps_m^{t_i}$ can be calculated by

$$fps_m^{t_i} = \frac{1}{\sigma_{(n,m)}^{t_i}} \cdot fps_n^{t_i}. \tag{5.2}$$

For DCVs: A higher frame rate can maintain a smoother watching experience. Denote $\bar{fps}_0$ as the lowest frame rate that human beings can accept. A higher data rate is needed in order to download more frames in a second.

Frame Resolution: for UCVs: We assume that, at time t_i, the camera captured video of UCV n is preprocessed into video frames with the resolution of $k_n^{t_i} \times s_n^{t_i}$ pixels. γ is denoted as the color depth of a frame which is the number of bits required to represent the information carried by one pixel. The data size of the offloaded video frame at time t_i can be calculated as $k_n^{t_i} \times s_n^{t_i} \times \gamma$ bits. Thus, the offloading latency experienced by the video frame at time t_i is modeled as

$$LU_n^{t_i} = \frac{k_n^{t_i} \times s_n^{t_i} \times \gamma}{R_n^{t_i}}, \tag{5.3}$$

where $R_n^{t_i}$ is the offloading data rate of UCV n at time t_i. Therefore, the offloading latency is determined by the frame resolution and wireless data rate. For instance, if the wireless data rate is low, the UCV can reduce the frame resolution of its offloaded camera captured video frames, in order to acquire a faster object detection response. However, the accuracy of the object detection highly depends on the resolutions of UCV's offloaded video frames. A lower video frame resolution always results in a worse mean average precision of an object recognition function [166]. Thus, a UCV with a high moving speed must keep offloading captured frames with a high frame resolution, in order to guarantee the accuracy of object detection and the safety of

autonomous driving. In addition, since the data size of object detection results is usually small, we do not consider the latency caused by transmitting the results in this paper.

For DCVs: Consider a DCV $g \in \mathcal{N}_{down}$. We assume that, at time t_i, the frame resolution of the requested entertainment video is $\bar{k_g^{t_i}} \times \bar{s_g^{t_i}}$ pixels. Similar to the UCV case, the downloading latency experienced by the video frame at time t_i can be modeled as

$$LD_g^{t_i} = \frac{\bar{k_g^{t_i}} \times \bar{s_g^{t_i}} \times \gamma}{R_g^{t_i}}, \tag{5.4}$$

where $R_g^{t_i}$ is the downloading data rate of DCV g at time t_i.

Offloading/Downloading Energy Consumption: the energy consumption for UCV $n \in \mathcal{N}_{up}$ or DCV $g \in \mathcal{N}_{down}$ offloading/downloading a frame at time t_i can be modeled as

$$EU_n^{t_i} = P_{tr} \cdot LU_n^{t_i}, \tag{5.5}$$

$$ED_g^{t_i} = P_{rev} \cdot LD_g^{t_i}, \tag{5.6}$$

where P_{tr} and P_{rev} are the wireless transmit power and receive power of a connected vehicle, respectively.

5.1.2 Key Idea

The moving speed of UCVs determines the amount of data (i.e., the number of captured frames and the frame resolution) that needs to be offloaded to the RSU server, in order to maintain an accurate, fast, and safe autonomous driving service. Thus, the key idea of our proposed *E-Auto* edge-assisted autonomous driving scheme is that **the RSU server periodically allocates a dedicated offloading period (DOP) to each associated UCVs by leveraging the values of their real-time moving speeds.** The length of UCV n's ($n \in \mathcal{N}_{up}$) allocated DOP is with respect to

its instantaneous speed $V_n^{t_i}$, where $DOP_n = \psi(V_n^{t_i})$. In other words, the UCV with a higher instantaneous speed will be allocated a larger DOP due to the requirements of high frame rate and high frame resolution. In addition, since the instantaneous speed of connected vehicles varies frequently, the RSU server conducts the allocation once every T ms. In *E-Auto*, we set T to be a Beacon Interval, which is the period for an access point (AP) broadcasting its beacon frames (usually its value is a multiple of 102 ms). Furthermore, each associated DCV is also allocated a dedicated downloading period (DDP). Since the download data size is irrelevant to the instantaneous speed of the corresponding DCV, every allocated DDP within T is of the same length and can be calculated by

$$DDP_g = \frac{T - \sum_{n=1}^{N_{up}}(DOP_n)}{N_{down}}, n \in \mathcal{N}_{up}, g \in \mathcal{N}_{down}. \tag{5.7}$$

5.1.3 Proposed Algorithms in E-Auto

Based on the above presented key idea, we design the *E-Auto* scheme. As shown in Figure 5.2, there are two components in the proposed *E-Auto*. One is the *RSU-server-side* algorithm, which determines the DOPs and DDPs of each connected vehicle in $\mathcal{N}$ periodically. The other is the *connected-vehicle-side* algorithm, which selects the frame resolution of the offloaded camera captured frames or the downloaded entertainment videos based on the length of the DOPs or DDPs.

RSU-server-side algorithm. We assume that the instantaneous speed of each UCV is constant within T ms. The *UCV Speed Vector*, $\vec{v} = \{v_1^x, \ldots, v_{N_{up}}^y\}$ $(x, y \in \mathcal{N}_{up})$, is defined as an N_{up}-tuple consisting of the instantaneous speed of each UCV in $\mathcal{N}_{up}$ within T, where UCVs are sorted by their instantaneous speed values in a decreasing order (i.e., $v_1^x = V_x$). Within each T, the RSU server allocates DOPs according to the order of UCVs in $\vec{v}$. For instance, the RSU server first allocates DOP_x to UCV x who has the highest instantaneous speed v_1^x. Furthermore, as mentioned in

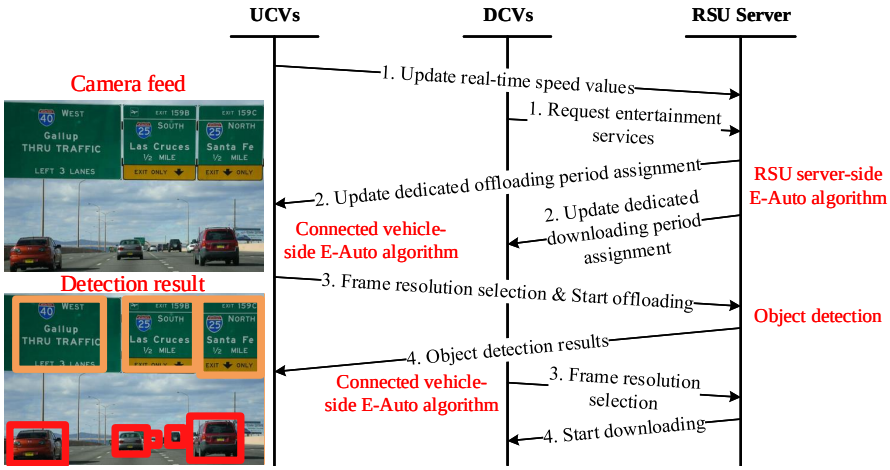


Figure 5.2: Overview of the proposed E-Auto scheme.

Section 5.1.1, the UCV with a higher instantaneous speed has tougher autonomous driving requirements, such as a higher frame rate, frame resolution, and object detection accuracy. Therefore, our proposed *E-Auto* scheme guarantees that the UCV with the highest instantaneous speed occupies the best offloading resources including the longest allocated DOP, denoted as $DOP_x = DOP_{max}$, the highest offloading frame resolution, denoted as $k_x \times s_x = k_{max} \times s_{max}$, and a guaranteed frame rate, denoted as fps_0. We define that DOP_{max} is calculated as

$$DOP_{max} = DOP_x = \frac{k_{max} \times s_{max} \times \gamma}{R_n} \cdot \frac{fps_0}{T}. \tag{5.8}$$

The rest of UCVs in $\mathcal{N}_{up}$ are allocated with their own DOPs based on the order in $\vec{v}$. For a particular UCV $n \in \mathcal{N}_{up}$, its allocated DOP can be calculated by

$$DOP_n = \psi(V_n) = \frac{1}{\sigma^{t_i}_{(x,n)}} \cdot DOP_{max}. \tag{5.9}$$

Therefore, the *E-Auto* RSU server-side algorithm guarantees that the UCV with a higher instantaneous speed will always be allocated a longer DOP. We define $\overrightarrow{DOP}$ as the set of RSU server assigned DOPs of each UCVs in $\mathcal{N}_{up}$ within T. In addition, in order to better classify the UCVs under complex speed conditions, we define three

Algorithm 4: E-Auto RSU-server-side algorithm

Input: The UCV speed vector $\vec{v}$, κ_1, κ_2, κ_3, the data rate of each connected vehicle R, N_{up}, N_{down}, T, fps_0, $k_{max} \times s_{max}$.

Output: The set of RSU server allocated DOPs $\overrightarrow{DOP}$, the set of RSU server allocated DDPs $\overrightarrow{DDP}$.

$T_0 \leftarrow T$;
if $\kappa_2 > 0$ **then**
    /* Special case 2) */
    $DOP_{n \in \mathcal{K}_2} \leftarrow$ calculate via (5.11); $T_0 \leftarrow T_0 - \sum_1^{\kappa_2} DOP_{n \in \mathcal{K}_2}$;
if $\kappa_1 \geq 2$ **then**
    Find R_{min}; /* Special case 1) */
    $DOP_{max} \leftarrow$ calculate via (5.10); $T_0 \leftarrow T_0 - \kappa_1 \cdot DOP_{max}$;
    if $\kappa_3 > 0$ **then**
        $DOP_{n \in \mathcal{K}_3} \leftarrow$ calculate via (5.9); /* Special case 3) */
        $T_0 \leftarrow T_0 - \sum_1^{\kappa_3} DOP_{n \in \mathcal{K}_3}$;
    for $n = \kappa_1 + 1$ **to** N_{up} **do**
        /* Allocate DOPs based on the order in $\vec{v}$. */
        $DOP_n \leftarrow$ calculate via (5.9);
        if $T_0 \leq DOP_n$ **then**
            $DOP_n \leftarrow T_0$; $T_0 = 0$; **break**;
        $T_0 \leftarrow T_0 - DOP_n$;
else
    $DOP_{max} \leftarrow$ calculate via (5.8); $T_0 \leftarrow T_0 - DOP_{max}$;
    if $\kappa_3 > 0$ **then**
        $DOP_{n \in \mathcal{K}_3} \leftarrow$ calculate via (5.9); /* Special case 3) */
        $T_0 \leftarrow T_0 - \sum_1^{\kappa_3} DOP_{n \in \mathcal{K}_3}$;
    for $n = 2$ **to** N_{up} **do**
        $DOP_n \leftarrow$ calculate via (5.9);
        if $T_0 \leq DOP_n$ **then**
            $DOP_n \leftarrow T_0$; $T_0 = 0$; **break**;
        $T_0 \leftarrow T_0 - DOP_n$;
if $T_0 > 0$ **then**
    for $g = 1$ **to** N_{down} **do**
        $DDP_g \leftarrow$ calculate via (5.7);
$\overrightarrow{DOP} \leftarrow DOP_n$; $\overrightarrow{DDP} \leftarrow DDP_g$;
return $\overrightarrow{DOP}$, $\overrightarrow{DDP}$

special cases:

1. $V_{highway}$ is the speed trigger to judge if a UCV is driving on highway (e.g., usually $V_{highway} = 60$ mph in the United States of America). Denote κ_1 as the number of UCVs whose instantaneous speeds are over $V_{highway}$, and $\mathcal{K}_1$ as the set of these UCVs. Considering the safety for driving on highway, we allocate

all the UCVs in $\mathcal{K}_1$ with the same length of DOP, DOP_{max}. Additionally, in order to guarantee that UCVs in $\mathcal{K}_1$ can complete offloading camera captured frames with conditions of $k_n \times s_n = k_{max} \times s_{max}$ and fps_0 within a DOP_{max}, the DOP_{max} is calculated as

$$DOP_{max} = \frac{k_{max} \times s_{max} \times \gamma}{R_{min}} \cdot \frac{fps_0}{T}, \tag{5.10}$$

where R_{min} is the minimum data rate among all UCVs in $\mathcal{K}_1$.

2. $V = 0$ denotes that the corresponding UCVs are stopping due to traffic congestion or the red traffic light. Denote κ_2 as the number of UCVs under this case, and $\mathcal{K}_2$ as the set of these UCVs. In addition, for each UCV under this special case, it just needs to periodically offload a frame to check if the vehicle in front of it starts moving or the traffic light turns green, where the checking period is defined as $\alpha \cdot T$. The allocated DOP is calculated by

$$DOP_{n \in \mathcal{K}_2} = \frac{k_n \times s_n \times \gamma}{R_n}. \tag{5.11}$$

3. Some UCVs are not allocated with one DOP within a continuous $\beta \cdot T$. Denote κ_3 as the number of UCVs under this case, and $\mathcal{K}_3$ as the set of these UCVs. For UCVs in this case, *E-Auto* gives them the highest allocation priority, except the UCV that has the highest speed or the UCVs in $\mathcal{K}_1$.

Furthermore, allocated DDP is calculated by (5.7). We define $\overrightarrow{DDP}$ as the set of RSU server assigned DDPs of each DCV in $\mathcal{N}_{down}$ within T. The details of *E-Auto* RSU-server-side algorithm are shown in Algorithm 4.

Connected-vehicle-side algorithm. As mentioned above, *E-Auto* guarantees that the UCV with the highest instantaneous speed or the UCVs with instantaneous speed over $V_{highway}$ can obtain the highest offloading frame resolution $k_{max} \times s_{max}$.

For other UCVs/DCVs, their offloading/downloading frame resolutions are selected between the higher one $k_{max} \times s_{max}/\bar{k_{max}} \times \bar{s_{max}}$ and the lower one $k_{low} \times s_{low}/\bar{k_{low}} \times \bar{s_{low}}$, via the connected-vehicle-side algorithm based on the length of the allocated DOPs/DDPs. In addition, a lower frame resolution is selected if a UCV is under the special cases 2) and 3). The details are given in Algorithm 5.

Algorithm 5: E-Auto connected-vehicle-side algorithm

Input: The set of RSU server allocated DOPs $\overrightarrow{DOP}$, the set of RSU server allocated DDPs $\overrightarrow{DDP}$, $k_{max} \times s_{max}$, $k_{low} \times s_{low}$, $\bar{k_{max}} \times \bar{s_{max}}$, $\bar{k_{low}} \times \bar{s_{low}}$, fps_0, $\bar{fps}_0$, R.

Output: The selected offloading frame resolution $k_n \times s_n$, the selected downloading frame resolution $\bar{k_g} \times \bar{s_g}$.

```
for n = 1 to N_up do
    if R_n × DOP_n ≥ fps_0 × k_max × s_max × γ then
        k_n × s_n ← k_max × s_max;
    else
        k_n × s_n ← k_low × s_low;
for g = 1 to N_down do
    if R_g × DDP_g ≥ fps̄_0 × k̄_max × s̄_max × γ then
        k̄_g × s̄_g ← k̄_max × s̄_max;
    else
        k̄_g × s̄_g ← k̄_low × s̄_low;
return k_n × s_n, k̄_g × s̄_g
```

Therefore, our proposed *E-Auto* scheme not only guarantees the variant requirements of autonomous driving based on UCVs' speed, but also improves the channel utilization comparing to IEEE 802.11 Enhanced Distributed Channel Access (EDCA), as shown in Figure 5.3.

5.1.4 Performance Evaluation

In this section, we evaluate our proposed $E-Auto$ scheme through network simulations. We simulate a connected vehicle network with a RSU server and 10 connected vehicles that have network activities during the whole simulations. Connected vehicles follow the Random Waypoint model, where the instantaneous speed of each vehicles is in the range of $[20, 65]$ mph, and each vehicles may stop a period in the

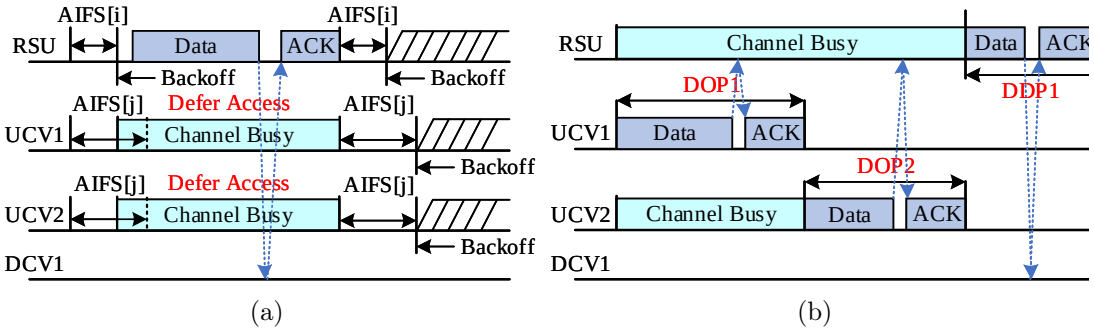


Figure 5.3: Comparison of channel access. (a) EDCA channel access; (b) E-Auto channel access.

range of $[2, 10]$ sec. In addition, we use the path-loss model which is expressed as

$$P_L = 20 \log_{10} f + 10n \log_{10} d - 28(dB), \tag{5.12}$$

where P_L is the RF signal propagation path-loss based on distance d between the RSU server and a connected vehicle, f is the carrier frequency in MHz, and n is the path-loss exponent. In our simulation environment, $n = 6$ and $f = 2400$ MHz. To determine the data rate of the connection between the RSU server and a connected vehicle, the Signal-to-Noise Ratio (SNR) is calculated and the data rate is chosen according to Table 5.2. The parameters of the stereo video camera deployed on connected vehicles are $\theta = 50°$, $l = 55$ m. $fps_0 = 10$ Hz [167, 168]. $k_{max} \times s_{max} = 640 \times 480$ pixels, $\bar{k_{max}} \times \bar{s_{max}} = 480 \times 320$ pixels, and $\bar{k_{low}} \times \bar{s_{low}} = 320 \times 240$ pixels.

Table 5.2: Data rate table of 802.11n (4 spatial streams)

Min. SNR (dBm)	2	5	9	11	15	18	20	25
Data rate (Mbps)	29	58	87	116	173	231	260	289

In the simulation, we compare the proposed *E-Auto* with two categories of schemes summarized in Table 5.3. 1) **Same AIFS (SA):** In this category, the RSU and connected vehicles acquire the same AIFS. 2) **Different AIFS (DA):** The RSU acquires a shorter AIFS than connected vehicles. We simulate maximum accuracy (maxA) and minimum latency (minL) schemes which adopt the highest frame resolution and the

lowest frame resolution, respectively, in both SA and DA.

Table 5.3: Scheme comparison

	frame resolution		channel access	
	fixed	dynamically selected	contend	scheduled
E-Auto		x		x
SA (MaxA)	x		x	
SA (MinL)	x		x	
DA (MaxA)	x		x	
DA (MinL)	x		x	

Simulation results of connected vehicles acquired average frame rate. We evaluate the performance of average frame rate through varying the value of low frame resolution, $k_{low} \times s_{low}$, in this simulation. As shown in Figure 5.4a, our proposed *E-Auto* scheme obtains a higher average frame rate for both UCVs and DCVs. Since varying $k_{low} \times s_{low}$ does not impact the DDP and download latency, the average frame rate of DCVs will not change as $k_{low} \times s_{low}$ increases, as shown in Figure 5.4b.

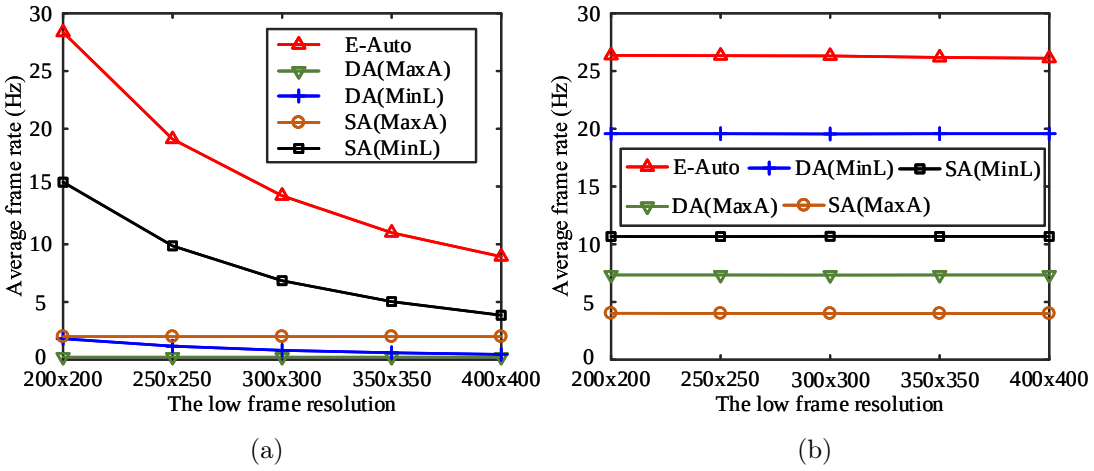


Figure 5.4: Acquired average frame rate vs. $k_{low} \times s_{low}$. (a) UCVs acquired average offloading frame rate; (b) DCVs acquired average downloading frame rate.

Simulation results of connected vehicles' energy efficiency. In addition, we evaluate the energy efficiency of connected vehicle's wireless interface. Three different states of the connected vehicle's wireless interface are define as follows:

1. *TX/RX.* The connected vehicle is transmitting (offloading) or receiving (downloading) a frame.

2. *Sleep.* The connected vehicle's wireless interface is put to sleep.

3. *Idle listening.* A state other than the above two. This includes sensing the channel, waiting for incoming frames, etc.

The power consumption for connected vehicle in these three states follows: P_{tx} = 127 mW, P_{rx} = 223.2 mW, P_{Il} = 219.6 mW, and P_{sleep} = 10.8 mW [169]. As depicted in Figure 5.3, Since *E-Auto* let the RSU schedule a fixed communication period for each associated connected vehicles, a connected vehicle can directly go to the sleep mode when it is not in its DOP/DDP. However, connected vehicle with other two schemes, DA and SA, have to keep in the idle listening state for channel sensing and contention. Therefore, our proposed *E-Auto* obtains higher energy efficiency. The simulation results are shown in Table 5.4, where our proposed *E-Auto* dramatically improves the energy efficiency of both DCV's and UCV's wireless interfaces. For example, as compared with the SA (MaxA), *E-Auto* reduces about 97.8% energy consumption for DCVs, and 98.8% energy consumption for UCVs.

Table 5.4: Energy efficiency results

	DCV (J/frame)	UCV (J/frame)
E-Auto	0.0012	0.0009
SA (MaxA)	0.0546	0.0771
SA (MinL)	0.0205	0.0100
DA (MaxA)	0.0300	0.6540
DA (MinL)	0.0112	0.0852

5.2 The Proposed Multiple Edge Servers Resource Allocation Protocol

5.2.1 Problem Statement

Balancing the load on APs is a primary way for MUs obtaining a fair service. However, the definition of the load of an AP is fuzzy. Thus, determining an appropriate

load definition is necessary. *Intuitively, the load of an AP needs to reflect its inability to satisfy the requirements of its associated users and as such it should be inversely proportional to the average bandwidth that they use.* We present our load definition in WLANs which captures the above summary.

Assume that the proposed BELL system has a set of MAPs, denoted by A. N denotes the number of MAPs in the system. The MAPs having the maximal load are called the load-heaviest MAPs and their load is denoted as y_{max}. In addition, without loss of generality, in our BELL-2M, we only reserve one MAP as the load-heaviest one at each iteration. Let U denote the set of all MUs in the BELL coverage area and M denote the number of MUs.

Definition 1 (The Load of an MAP)**.** *Consider an MAP $a \in A$, and let U_a be the set of MUs associated with MAP a. The load of MAP a, denoted by y_a, is the aggregate period of time that takes MAP a to provide a unit of traffic volume to all associated MUs $u \in U_a$. Thus,*

$$y_{a\in A}^{U_a} = \sum_{u \in U_a} (\frac{1}{r_{a,u}}), \tag{5.13}$$

where $r_{a,u}$ is the wireless link bit rate between MAP a and MU u.

Definition 2 (BELL-2M Load-Balanced Vector)**.** *We define the BELL load vector as $\vec{Y} = \{y_1^{U_1}, \ldots, y_N^{U_N}\}$, which is BELL-2M balanced when the load of any MAP can not be reduced only if increasing another MAP load with the same or higher load.*

We can show that the problem of finding a BELL-2M load-balanced vector is NP-hard, by proving that even the simplest scenario, e.g., only two MAPs in the BELL, is NP-hard.

Proof. Consider a case with only two MAPs. Each MU $u \in U$ can be covered by both MAPs. Therefore, to obtain the BELL-2M load-balanced vector $\vec{Y}$ equals to find a subset $U' \subseteq U$ that satisfies

$$y_1^{U'} = y_2^{U-U'}. \tag{5.14}$$

By restricting the problem to $\frac{1}{r_{1,u}} = \frac{1}{r_{2,u}}$, the problem can be a reduction from the partition problem. *The partition problem is the task of deciding whether a given multiset S of positive integers can be partitioned into two subsets S_1 and S_2 such that the sum of the numbers in S_1 equals the sum of the numbers in S_2*, which is a known NP-hard problem. □

BELL-2M aims to iteratively find an optimal BELL-2M load-balanced vector $\vec{Y}$ in polynomial time.

5.2.2 BELL-2M Algorithm

The basic idea is that, at each iteration, g, we reduce the load of the load-heaviest MAP a at this stage by transferring some of its associated MUs to its neighboring MAPs. Furthermore, MUs with lower RSSI have a higher priority to be transferred. In other words, we define the RSSI vector $\overrightarrow{UR_a} = \{\mu_1, \ldots, \mu_m\}$ to be an m-tuple consisting of the RSSI of each MU in MAP a sorted in an increasing order, where m is the total number of MUs in MAP a. MAP a always first transfers the MU with the lowest order in $\overrightarrow{UR_a}$.

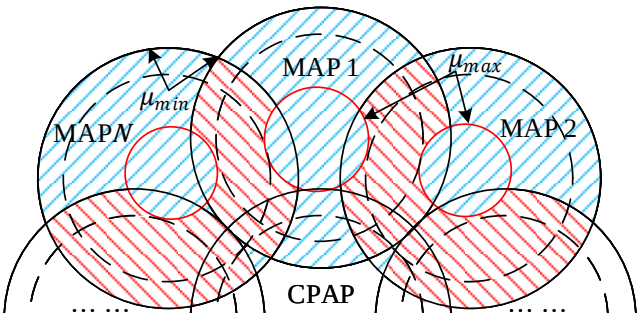


Figure 5.5: Only MUs with RSSI in $[\mu_{min}, \mu_{max}]$ might have chance to be transferred.

However, there are two algorithmic challenges for the above problem:

Challenge 3. *Not every MU in MAP a is covered by another neighboring MAP. In BELL, as shown in Figure 5.5, MUs located within the blue shadow area have no chance to be successfully transferred to neighboring MAPs.*

Challenge 4. *The Ping-Pong effect of load-balancing decision.*

Example 3. Consider a BELL with two MAPs, as shown in Figure 5.6, denoted as v_1 and v_2, and two MUs u_1 and u_2. u_1 can only connect to v_1 and it yields a load of 0.2. u_2 can connect to both v_1 and v_2 and yields a load of 0.6 on its connected MAP. Both u_1 and u_2 first connect to v_1, and the RSSI vector of v_1 is $\overrightarrow{UR_{v_1}} = \{\mu_{u_2}, \mu_{u_1}\}$. As shown in Figure 5.6(a), the loads of v_1 and v_2 are 0.8 and 0, respectively. To balance the load, a greedy algorithm first lets v_1 transfer u_2 to v_2. Now the loads of the two MAPs are 0.2 and 0.6, respectively, as depicted in Figure 5.6(b). Then, the algorithm lets v_2 reduce its load, since it is the load-heaviest MAP at this stage. Thus, v_2 transfers u_2 back to v_1, and the algorithm repeats the above indefinite iterations, which obviously can not obtain an optimal solution.

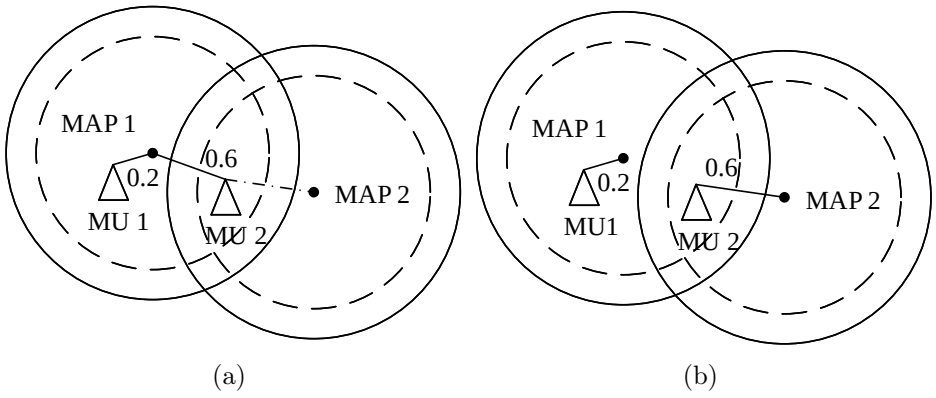


Figure 5.6: Example of the Ping-Pong effect of a simple greedy algorithm. (a) Original MU-MAP connection; (b) MU-MAP connection after an iteration.

Our BELL-2M algorithm resolves the above two challenges by, firstly, introducing a maximal transfer RSSI μ_{max} and a minimal transfer RSSI μ_{min}. As shown in Figure 5.5, we ask the MU with RSSI larger than μ_{max} (those MUs only covered by its associated MAP) not to be transferred. On the other hand, let μ_{min} be the RSSI corresponding to the MAP's transmission range. Thus, only MUs with RSSI in the range of $[\mu_{min}, \mu_{max}]$ might have chances to be triggered with a load transfer. Secondly, to overcome Challenge 4, we define a set of fixed MAPs, D, whose loads

have already been determined by previous iterations. At the beginning, the set D is empty, and after each iteration, a new MAP is added to it, until $D = A$. Therefore, BELL-2M only searches the set of the non-fixed MAPs, $\{A-D\}$, for the load-heaviest MAP at each stage.

The proposed BELL-2M load-balancing algorithm is shown in Algorithm 6. At each iteration, the algorithm first finds the load-heaviest MAP α_0 and preserves its load y_{max} in θ. Then MAP α_0 checks MUs from its farthest possible MUs whose RSSI are in the range of $[K_{\alpha_0}, K_{\alpha_0} + \beta]$, where K_{α_0} is initialized with μ_{min} and β is a step value. If these MUs exist, MAP α_0 will evaluate the load transfer result of them one by one: i.e., if after transferring an MU, none of the fixed MAPs' load increases, and the new load-heaviest MAP's load is smaller than that of the previous stage, the checked MU will be confirmed to transfer. Otherwise, the MU will not be transferred. After all MUs are checked, K_{α_0} is incremented by β in each iteration. The evaluation stops if $K_{\alpha_0} = \mu_{max}$. Then, MAP α_0 joins the fixed MAP set as well as its load value. Finally, the algorithm stops when $D = A$, which means that all the MAPs' load has been fixed.

We now prove that the proposed algorithm always finds a BELL-2M load-balanced vector for BELL.

Proof. Each iteration starts with the load-heaviest MAP with $K_{\alpha_0} = \mu_{min}$ and stops when $K_{\alpha_0} = \mu_{max}$. MUs are transferred only if y_{max} is decreased after this transfer. Therefore, at the end of the gth iteration, y_{max} is not larger than its value at the beginning of the gth iteration. The corresponding MAP is added in the fixed MAP set D after the gth iteration. Since the definition of the load-heaviest MAP is the non-fixed MAP that has the maximal load at this stage, y_{max} at the $(g+1)$th iteration is smaller than that at the gth iteration. Also, MUs can only be transferred to the non-fixed MAPs, which means that the load of MAPs in D is not affected in the $(g + 1)$th iteration. Therefore, the algorithm keeps reducing y_{max} and stops at an

Algorithm 6: BELL-2M Load Balancing Algorithm

```
1  D = ∅
2  while D ≠ A do
       /* Find and record the load-heaviest MAP α_0 and its load y_{α_0} */
3      α = α_0 s.t. y_{α_0} = max_{α_0 ∈ {A−D}} {y_{α_0}}
4      θ = y_max = y_{α_0}
5      K_{α_0} = μ_min
6      while K_{α_0} < μ_max do
7          c = the number of MUs with RSSI in [K_{α_0}, K_{α_0} + β]
8          K_{α_0} = K_{α_0} + β
           /* Check if exist MUs with RSSI in this range */
9          if c > 0 then
10             for i = 1 to c do
11                 MUi performs BELL-handoff
                   /* Find and record the load-heaviest MAP α_0 and its load y_{α_0} */
12                 α_0 s.t. y_{α_0} = max_{α_0 ∈ {A−D}} {y_{α_0}}
13                 y_max = y_{α_0}
                   /* Check if fixed MAPs' load was not increased and a better load vector was
                      found */
14                 if (∄α_0 ∈ D s.t. y_{α_0} > ȳ_{α_0}) ∧ (y_max < θ) then
15                     θ = y_max
16                     α = α_0
17     D = D ∪ {α}
18     ȳ_α = y_α
```

optimal solution. The algorithm stops when $D = A$, in other words, all the MAPs become load fixed MAPs. Therefore, y_{max} cannot be reduced further. According to Definition 2, the BELL-2M load-balanced vector is found. □

5.2.3 Performance Evaluation

5.2.3.1 Simulation Setup

The simulation setting is as follows. 9 APs are regularly distributed in a square are. The length of the simulation area is $600m$. Each AP has a circular transmission range, and its radius is $145m$. In BELL, the operating channel of the CPAP, MAPs with odd index, and MAPs with even index are 1, 6, and 11 in 2.4 GHz band, respectively. In C-WLAN, the operating channel of the APs are chosen randomly in 2.4 GHz band. To determine the bit rate of the connection between an MU and an AP, Signal-to-

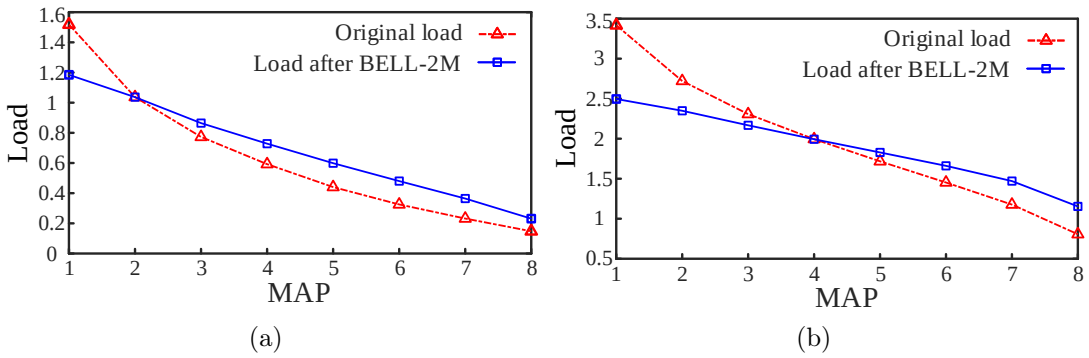


Figure 5.7: Load comparison. (a) $M = 100$; (b) $M = 300$.

Noise Ratio (SNR) is computed and the bit rate is chosen accordingly. We set each AP's transmission power to be 20 dBm and the value of Δ to be 5. To simulate an indoor environment, we use the indoor path-loss model which is expressed as

$$P_L = 20 \log_{10} f + 10n \log_{10} d - 28(dB), \tag{5.15}$$

where P_L is the RF signal propagation path-loss based on distance d between the AP and the MU, f is the carrier frequency in MHz, and n is the path-loss exponent. In our simulation, $n = 6$ and $f = 2400$ MHz. The background noise level is set to be -90 dBm. In the simulation of evaluating BELL-2M load balancing performance, the number of M MUs are randomly positioned in the simulation area and they are assumed to follow the saturated traffic model and quasi-static. In the simulation of evaluating the battery drain of mobile devices, 500 MUs follow the Random Waypoint model and walk in the simulation area for 5.5 hours.

5.2.3.2 Simulation Results

Load Balancing Performance. Figure 5.7 shows the simulation results of the load balancing performance. The Y-axis represents the load of MAPs y_a and the X-axis represents MAP index. MAPs are sorted by their load values in a decreasing order. In Figure 5.7a and Figure 5.7b, we obtain their load values by running simulations at least 500 times. In other words, each load value, y_a is obtained by

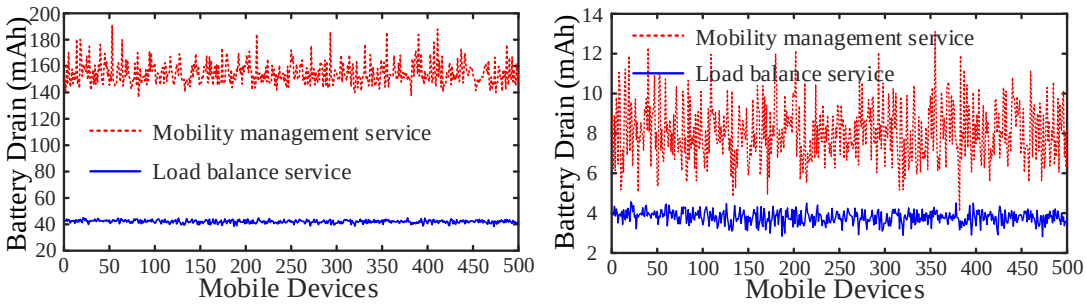


Figure 5.8: Battery drain comparison. (a) Battery drain within C-WLAN; (b) Battery drain comparison.

averaging the 500 simulation load results. The red dotted line represents the original load of BELL, and the blue solid line represents the load of BELL after performing our BELL-2M load-balancing protocol. It is confirmed that BELL-2M can improve the load-balancing performance of BELL.

Mobile Devices' Battery Drain. Figure 5.8 illustrates the simulation results of mobile devices' battery drain for WLAN management services within 5.5 hours. The Y-axis represents the battery drain of mobile devices and the X-axis represents MU index. Comparing Figure 5.8a and Figure 5.8b, it is shown that MUs in BELL drain 94% and 90% less battery life for mobility management service and load balancing service, respectively, than MUs in C-WLAN.

CHAPTER 6: PROPOSED ENERGY-AWARE CONFIGURATION ADAPTATION ALGORITHM IN MEC NETWORKS

6.1 A Comprehensive Experimental Study

In order to better investigate and understand the relationship between the energy consumption and the performance of deep learning-based applications, following questions are proposed:

1. *How is energy consumed when a deep learning-based application is executed locally?*

2. *Does smartphone's computation capacity impact the energy consumption when a deep learning-based application is executed locally?* It is intuitive that the smartphone with higher computation capacity can achieve a lower inference latency. However, the energy consumption is more complicated to analyze. This is because, for example, the smartphone with more powerful processors may drain its battery faster.

3. *Does transferring all the computation data to a powerful infrastructure significantly decrease the energy consumption and latency?* When a deep learning-based application is executed remotely, communication latency is non-negligible and unstable, especially in wireless networks. Previous work [151] shows that smartphone's radio interfaces account for up to 50% of the total power budget. In addition, improved communication speeds generally come at the cost of higher power consumption [170].

4. *Besides the network condition, what impact the energy consumption and latency*

Table 6.1: Smartphones used in our study.

Manufacturer	Samsung	Google	Asus
Model	Galaxy S5	Nexus 6	ZenFone AR
OS	Android 6.0.1	Android 5.1.1	Android 7.0
SoC	Snapdragon 801 (28 nm)	Snapdragon 805 (28 nm)	Snapdragon 821 (14 nm)
CPU	32-bit 4-core 2.5GHz Krait 400	32-bit 4-core 2.7GHz Krait 450	64-bit 4-core 2.4GHz Kryo
GPU	578MHz Adreno 330	600MHz Adreno 420	653MHz Adreno 530
RAM	2GB	3GB	6GB
WiFi	802.11n/ac	802.11n/ac	802.11n/ac/ad
Release date	April 2014	November 2014	July 2017

Table 6.2: Classifications of the tested smartphones.

Smartphone	S5	Nexus 6	ZenFone AR
CPU score	36871	37521	58531
GPU score	6678	18063	67286
Image processing score	3103	6862	11321
Total score	66414	80047	173472
Class	Low-end	Low-end	High-end

when executed remotely, and how?

6.1.1 Experimental Methodology

6.1.1.1 Hardware Setup

Our study was performed using three different smartphones. We summarize their characteristics in Table 6.1. We classify them into two classes, *low-end* and *high-end* smartphones, according to their general hardware performance tested by using an Antutu benchmark [171]. The testing results are shown in Table 6.2. In addition, we emulate an edge server with an Nvidia Jetson AGX Xavier, which connects to a WiFi access point (AP) through a 1Gbps Ethernet cable. Details of our equipped edge server are shown in Table 6.1.

6.1.1.2 Software Implementation

Edge server side. The edge server is developed to process the video frames and send the detection results back to smartphones. We implement two major mod-

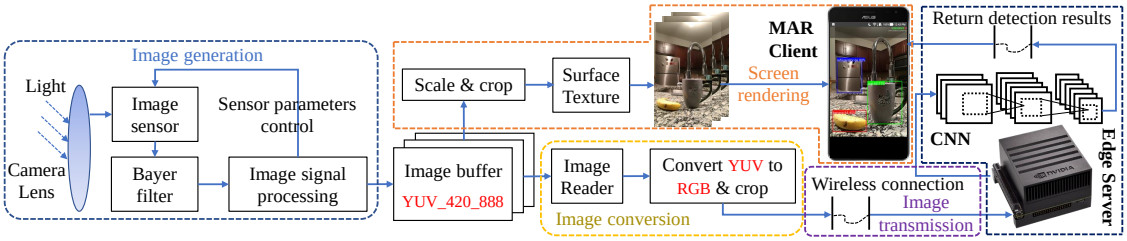


Figure 6.1: Processing pipeline of the deep CNN optimized object detection application implemented in this section.

ules on the edge server. The first one is the communication service handler module which performs authentication and establishes a socket connection with smartphones. This module is also responsible for dispatching the detection results to corresponding smartphones. The second one is the object detection module which is designed based on a custom framework called Darknet [172] with GPU acceleration and runs YOLOv3 [164], a large neural network model with 24 convolutional layers. The YOLOv3 model used in our experiments is trained on COCO dataset [173] and can detect 80 classes.

Smartphone side. We implement two scenarios for our experimental study. The first one is executing deep learning on smartphones, defined as *local execution*. In this scenario, the Android implementation is based on a light framework called Tensorflow Lite [174] which is TensorFlow's lightweight solution for embedded and mobile devices. It runs a small neural network model, called MobileNetv1 [175]. In order to run MobileNetv1 with different frame resolutions in Tensorflow Lite on smartphones, we convert a pre-trained MobileNetv1 SSD model to the FlatBuffers format. The second one is executing deep learning on our equipped edge server, defined as *remote execution*. In this scenario, a smartphone transfers the converted RGB frames to the edge server through a socket connection in real time. To avoid having the server process stale frames, the smartphone sends the latest captured frame to the server and waits to receive the detection result before sending the next frame for processing.

The detailed processing pipeline is shown in Figure 6.1.

6.1.1.3 Power Measurement Setup

To measure the power consumption, we use an external power monitor, a Monsoon Power Monitor, to provide power supply for the smartphone. Different from old smartphone models, modern smartphones like Nexus 6 have very tiny battery connectors, making it very challenging to connect the power monitor to them. To solve this problem, we modify the battery connection of Nexus 6 by designing a customized circuit and soldering it to the smartphone's power input interface. In addition, the power measurements are taken with the screen on, with the Bluetooth/LTE radios disabled, and with minimal background application activity, ensuring that the smartphone's *base power* is low and does not vary unpredictably over time. For the measurements of the power consumption in local execution, base power is defined as the power consumed by the smartphone when its WiFi interface is turned off. For the measurements of the power consumption in remote execution, base power is defined as the power consumed when the smartphone is connected to the AP without any data transmission activity [151, 152].

6.1.2 Experimental Results

In this section, we describe our efforts towards measuring and understanding the energy consumption and the performance of running deep CNNs on both high-end and low-end smartphones.

6.1.2.1 Key metrics

Currently, object detection applications focus on the following two critical metrics:

Latency/frames per second (FPS). Latency is the total time needed to obtain the detection results on one video frame (i.e., usually shown as one or multiple bounding boxes that identify the location and classification of the objects in a frame). In this paper, it is defined as the time period from the moment the *Image Reader*

acquiring one camera captured image frame to the moment the bounding boxes are drawn on the smartphone's screen, as depicted in Figure 6.1. In local execution, the per frame total latency includes the time used for converting the YUV frame to the RGB frame, cropping the frame to the fitted resolution $k \times k$, and executing deep learning, defined as *inference latency*, on the smartphone. In remote execution, the per frame total latency includes, besides the convert and crop latency that are both executed locally on the smartphone, the communication latency (i.e., transmitting the frame and receiving the results) and the inference latency on the edge server.

Accuracy. The mean average precision (mAP) is a commonly used performance metric in object detection. Better performance is indicated by a higher mAP value. Specifically, the average precision [176] is computed as the area under the precision/recall curve through numerical integration. The mAP is the mean of the average precision across all classes.

6.1.2.2 Local Execution vs. Remote Execution

We first evaluate the object detection performance of both local execution and remote execution in terms of latency, FPS, accuracy, and energy consumption, as shown in Figure 6.2 and 6.3. The preview resolution is set to $k_1 \times k_2 = 640 \times 480$ pixels. In remote execution, we use a WiFi 5 GHz channel and TCP socket connection to transfer data between smartphones and the edge server.

Local execution. First, we examine the total latency of executing object detection with different frame resolutions, from 100×100 to 600×600 pixels, in our three smartphones. The experimental results are shown in Figure 6.2a. We find that (1) *a higher frame resolution always results in a higher per frame total latency.* For example, for Nexus 6, the per frame total latency surges from 569.8 ms to 2378.7 ms when the frame resolution increases from 100×100 to 600×600 pixels. (2) *The high-end smartphone achieves a significantly lower per frame total latency compared to the low-end smartphones.* For example, when the frame resolution is 300×300

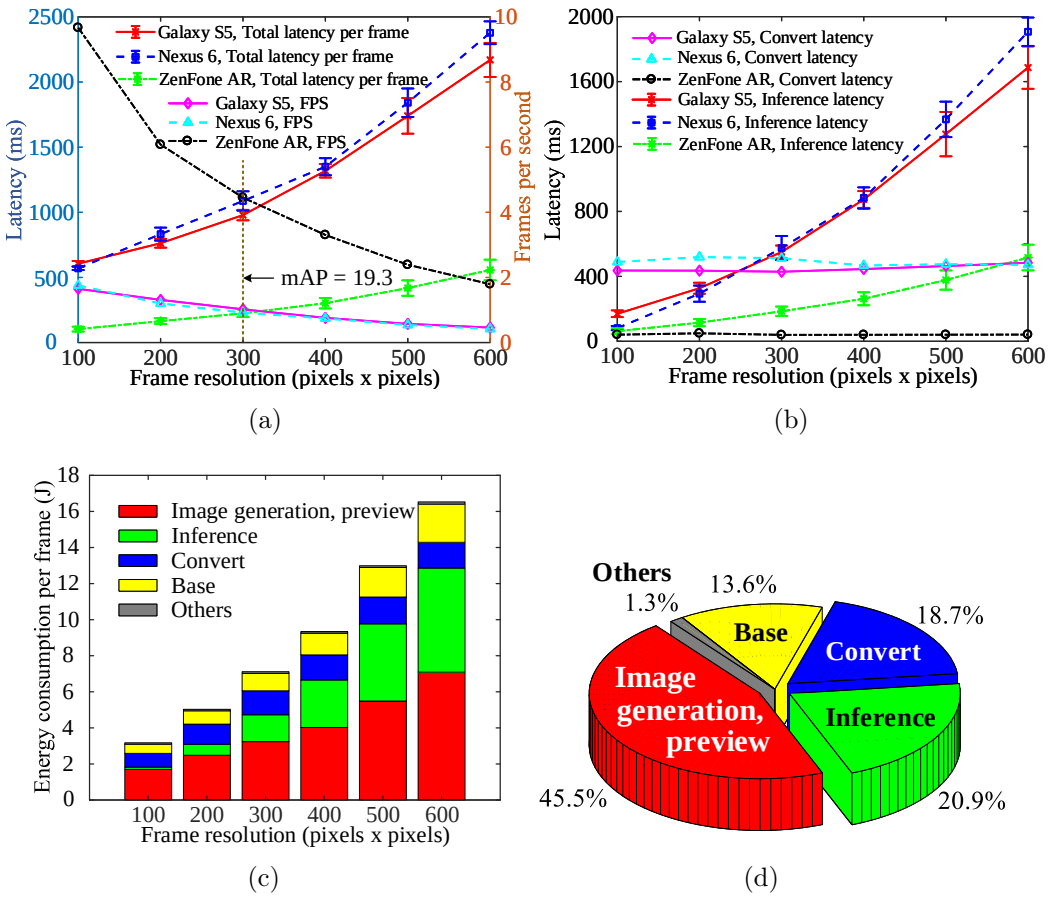


Figure 6.2: Experimental results for local execution. (a) Total latency per frame and FPS; (b) Convert and inference latency per frame; (c) Average energy consumption per frame breakdown (Nexus 6); and (d) Average percentage breakdown of energy consumed in executing 300×300 MobileNetv1 SSD model (Nexus 6).

pixels, the per frame total latency of ZenFone AR is only 20.7% and 22.9% of that of Nexus 6 and Galaxy S5, respectively.

Second, we measure the latency of each phase in the processing pipeline. We show the latency of the two highest time-consuming phases, convert and inference latency, in Figure 6.2b, which comes up to 95% of the per frame total latency. We find that (1) *for both high-end and low-end smartphones, the convert latency does not vary much when the frame resolution increases.* This is because no matter what the frame resolution $k \times k$ is configured, every YUV frame is converted to an RGB frame with the preview resolution $k_1 \times k_2$ first. After the convert is completed, the RGB frame will be

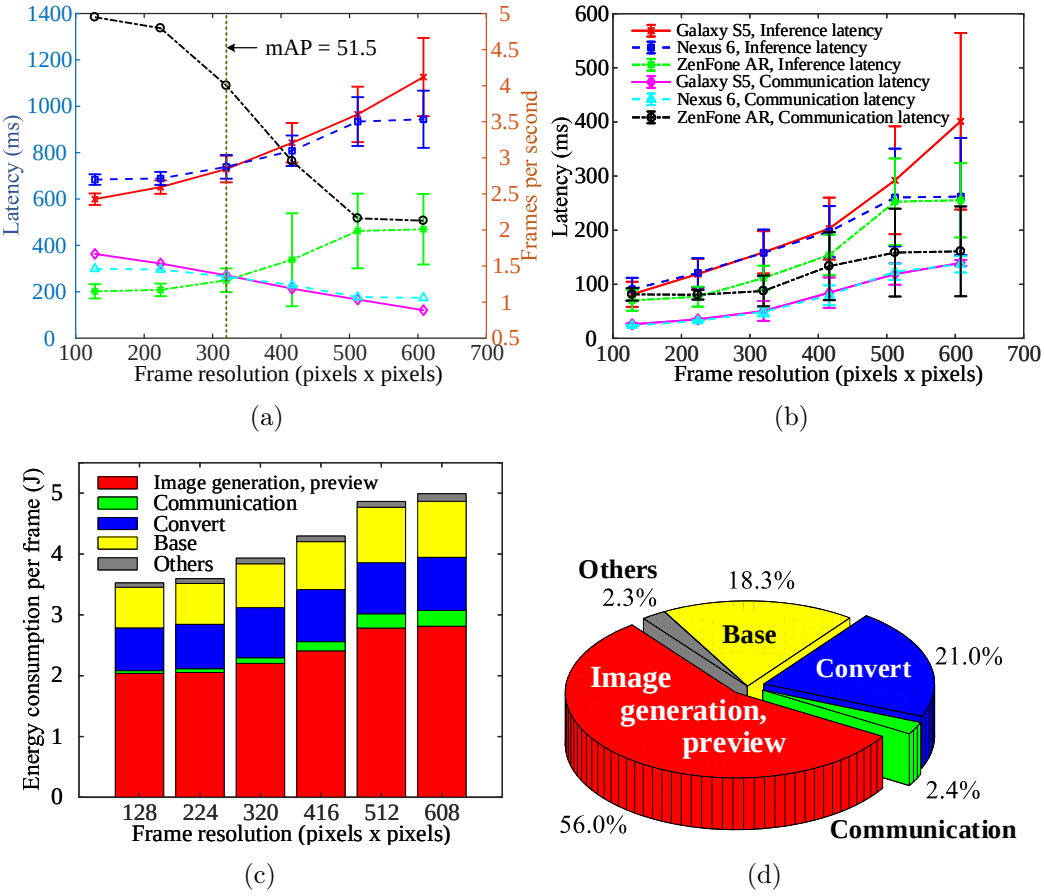


Figure 6.3: Experimental results for remote execution. (a) Total latency per frame and FPS; (b) Inference and communication latency per frame; (c) Average energy consumption per frame breakdown (Nexus 6); and (d) Average percentage breakdown of energy consumed in executing 320×320 YOLOv3 model (Nexus 6).

resized to $k \times k$ pixels. (2) *For low-end smartphones, the largest time-consuming phase is converting a YUV frame to an RGB frame when the frame resolution is smaller than* 300×300 *pixels. In contrast, when the frame resolution is larger than* 300×300 *pixels, inference becomes the largest latency source.* For example, for Nexus 6, the convert latency is 85.6% of the per frame total latency when the frame resolution is 100×100 pixels; whereas the inference latency is 80.2% of the per frame total latency when the frame resolution is 600×600 pixels. This is rather significant because most of the previous work that evaluates the per frame latency of object detection executed in smartphones only consider the inference latency. However, our experimental

results indicate that the convert latency is non-negligible and sometimes larger than the inference latency. (3) Interestingly, *when the frame resolution is small, the inference latency of the high-end and low-end smartphones are comparable. However, the convert latency of the high-end smartphone is always considerably smaller than that of the low-end smartphones.* For example, when the frame resolution is 100×100 pixels, the inference latency of ZenFone AR and Nexus 6 are 62.5 ms and 81.3 ms, respectively, whereas the convert latency of ZenFone AR is 41.2 ms which is only 8.4% of the convert latency of Nexus 6. This result indicates that when the frame resolution is low, converting a YUV frame to an RGB frame is more computation-intensive than running a small CNN in smartphones and low-end smartphones are unable to generate the same convert performance as high-end smartphones in terms of the convert latency.

Third, to dissect the energy drain through different processing pipeline phases, we first break down the per frame total energy consumption as follows: image generation, preview, inference, convert, base, and others. We make the following observations from Figure 6.2c and 6.2d. (1) *The image generation and the preview always contribute the highest energy consumption in the smartphone and grows significantly as the frame resolution increases.* Specifically, in Nexus 6, when the frame resolution is 300×300 pixels, on average a whopping 45.5% of the per frame total energy consumption is from the image generation and preview. The reason why the image generation process consumes considerably high energy is executing the 3A (i.e., auto-focus (AF), auto-exposure (AE), and auto-white-balance (AWB)) and multiple fine-grained image post processing algorithms (e.g., noise reduction (NR), color correction (CC), and edge enhancement (EE)) on image signal processor (ISP). These sophisticated algorithms are designed to make an image that is captured by the smartphone camera look perfect. *However, is it always necessary for the camera captured frame to be processed by all of those energy-hungry image processing algorithms in order to achieve a*

successful object detection result? In addition, the number of frames captured by the camera per second is a fixed value (e.g., 24 or 30 frames/second) or in a range (e.g., $[7, 30]$ frames/second), which is controlled by the AE algorithm. However, for low-end smartphones, even if the frame resolution is small, the detection FPS is still less than 2, as shown in Figure 6.2a, which is far slower than the camera capture frame rate. Furthermore, the CNN always extracts the latest captured frame, which indicates that, *from the perspective of the energy efficiency of the object detection pipeline, capturing frames with a fast rate is unnecessary and energy-inefficient.* (2) *The inference energy consumption grows dramatically as the frame resolution increases.* For example, it accounts for 4.1% and 33.9% of the per frame total energy consumption when the frame resolution is 100×100 and 600×600 pixels, respectively.

Remote execution. We next compare against the remote execution scenario where the CNN is run on the implemented edge server with a 5GHz WiFi link to the smartphone. Note that we conduct our measurements in different network conditions (e.g., the Received Signal Strength Indicator (RSSI) at the tested smartphones or the network bandwidth gradually drops down). However, due to the page limitation, we only present our experimental results obtained in an excellent network condition (i.e., the RSSI is in the range of -15 and -20 dBm). First, we compare the latency and FPS, as shown in Figure 6.3a and 6.3b, and make the following observations. (1) *For the high-end smartphone, the per frame total latency (FPS) is larger (lower) than that of the local execution scenario when the frame resolution is smaller than* 512×512 *pixels* (note that this observation may differ depending on how powerful the server's GPU is). This observation supports the fact that lots of recently released smartphones with high computation power possess the capability to run a small CNN model. However, the mAP of the large CNN model on the server is better than that of the small CNN model on the smartphone (e.g., $mAP = 51.5$ on the server and $mAP = 19.3$ on the smartphone when the frame resolution is around 300×300

pixels). Generally, different implementation cases have variant latency/accuracy requirements. For example, the AR cognitive assistance case where a high-end wearable device helps visually impaired people to navigate on a street may need a low latency but can tolerate a relatively high number of false positives (i.e., false alarms are fine but missing any potential threats on the street is costly) [177]. In contrast, an AR used for recommending products in shopping malls or supermarkets may tolerate a long latency but require high detection accuracy. Therefore, choosing the appropriate execution approach (i.e., local or remote) in different implementation cases is critical.

Furthermore, (2) *for the low-end smartphones, the per frame total latency (FPS) is slightly lower (higher) than that of the local execution scenario.* The reason why the latency does not decrease significantly is the high convert latency which is executed by the smartphone in both local and remote execution cases. This observation is rather significant for deciding what computation tasks should be transferred to the server. Most of the existing works simply consider transferring the converted RGB frames to the server. *However, for low-end smartphones, only executing the CNN in the server is inadequate to achieve an acceptable FPS. Converting YUV to RGB frames remotely is also desirable.* (3) Interestingly, as shown in Figure 6.3c, *the frames transmitted by ZenFone AR obtain less inference latency than the frames transmitted by Galaxy S5 under the same conditions (i.e., the same frame resolution and camera view).* We repeated these measurements several times and got the same results although, until now, we do not have a definite explanation for this result.

In addition, Figure 6.3c and 6.3d analyze the energy drain of the smartphone by different processing pipeline phases in the remote execution case, including image generation, preview, communication, convert, base, and others. Compared to the local execution scenario, we have the following observations. (1) *Similar to the local execution, image generation and preview are the biggest energy consuming phases.* For example, it comes up to 56.0% of the per frame total energy consumption when

the frame resolution is 320×320 pixels. (2) *Transmitting one frame and receiving the result consume less energy than our expectation, when the wireless network condition is excellent.* For example, on average it only accounts for 2.4% of the per frame total energy consumption when the frame resolution is 320×320 pixels. (3) *The remote execution saves approximately 53% energy per frame on average when the frame resolution is larger than* 128×128 *pixels.* However, it consumes 12.9% more energy per frame than the local execution when the frame resolution is 128×128 pixels. This observation is rather significant, which demonstrates that *running deep learning remotely does not always consume less energy than the local execution, even when the network quality is excellent.*

6.1.2.3 Power Consumption of the Image Generation and Preview

As we observed above, the image generation and the preview are the most energy-consuming phases in both local and remote execution cases. Thus, to reduce the energy consumption of the object detection processing pipeline, we must improve the energy efficiency of these two phases. We seek to understand the interactions between the power consumption and various factors (e.g., the preview resolution, 3A, and several image post processing algorithms) as follows.

Preview resolution vs. power consumption. We first examine how the preview resolution influences the power consumption of the image generation and preview phases, as shown in Figure 6.4a. We find that *as the preview resolution grows, the power consumption increases dramatically.* Therefore, a preview with a higher frame resolution on the smartphone provides a better quality preview for users, but at the expense of battery drain, which is applicable for both local and remote execution cases.

Image post processing and 3A algorithms vs. power consumption. We next examine the effect of multiple image post processing and 3A algorithms on the power consumption of the image generation and preview phases, as shown in Figures

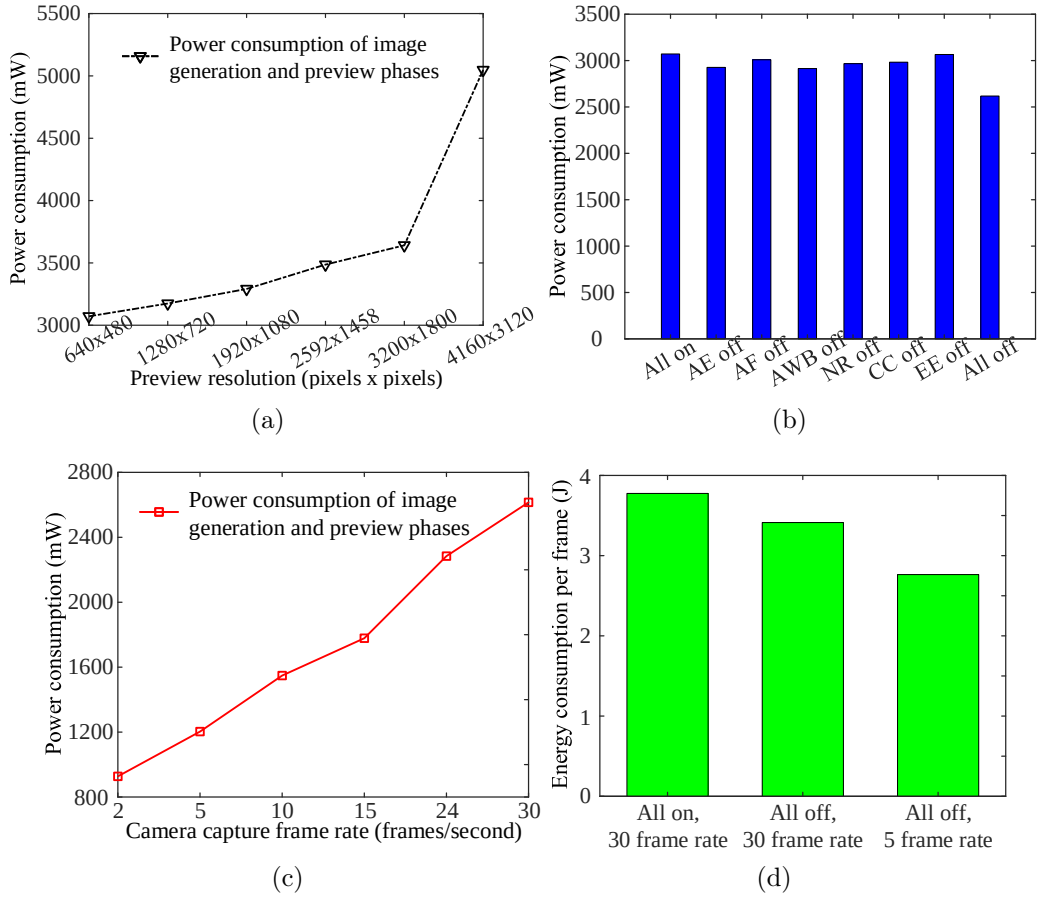
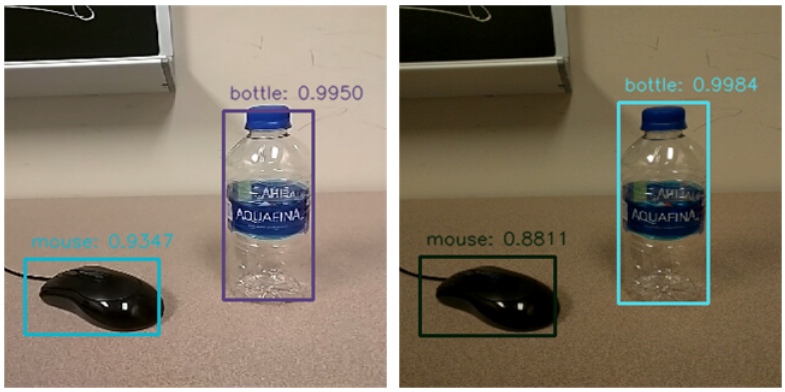


Figure 6.4: Power consumption analyses of the image generation and preview phases. (a) Preview resolution vs. power consumption; (b) 3A and image post processing algorithms vs. power consumption; (c) Camera capture frame rate vs. power consumption; and (d) Comparison of the energy consumption per frame (remote execution).

6.4b and 6.4c. Note that when the AE is disabled, we manually set the camera ISO and exposure time to 400 and 20 ms, respectively. We observe that (1) *disabling the 3A, NR, CC, and EE algorithms decreases the power consumption by* 14.8%. We conduct another experiment to understand if disabling these algorithms would impact the object detection performance. As shown in Figure 6.5, the detection performance does not degrade. (2) *Accelerating the camera capture frame rate significantly increases the power consumption.* As we discussed above, the maximum detection FPS that the low-end smartphones can obtain is around 2; a capture rate larger than 2 frames/second is unnecessary and energy-inefficient from the perspective of energy



(a) (b)

Figure 6.5: Comparison of the object detection results (remote execution). (a) All enabled; (b) All disabled.

efficiency. Furthermore, we compare the per frame energy consumption among three cases, as depicted in Figure 6.4d: all enabled with camera capture frame rate 30, all disabled with camera capture frame rate 30, and all disabled with camera capture frame rate 5. We find that (3) *the per frame energy consumption of the second and the third cases decreases by approximately* 10% *and* 27%*, respectively, compared to the first case.*

6.2 The Proposed Energy-Aware Configuration Adaptation Algorithm

In this section, a user preference based energy-aware edge-based MAR system is designed [178]. The novel contributions are summarized as follows:

1. An edge-based MAR system to analyze the interactions between MAR configurations and the client's energy consumption is designed and implemented. Based on the experimental study, several insights that can potentially guide the design of energy-aware MAR systems are summarized.

2. A comprehensive energy model is proposed, which identifies (i) the tradeoffs among the energy consumption, service latency, and detection accuracy, and (ii) the interactions among MAR configuration parameters (i.e., CPU frequency and computation model size), user preferences, camera sampling rate, network

bandwidth, and per frame energy consumption for a multi-user edge-based MAR system.

3. An energy-efficient optimization algorithm named LEAF is proposed, which guides MAR configuration adaptations and radio resource allocations at the edge server, and minimizes the per frame energy consumption while satisfying variant clients' user preferences.

6.2.1 Experimental Results on Factors Affecting MAR Client Energy Efficiency

In this section, we describe our preliminary experiments to evaluate the impact of various factors on the energy efficiency of an MAR client, service latency, and detection accuracy in an edge-based MAR system. Specifically, these experimental results provide (i) observations on interactions between energy consumption and MAR configuration parameters, such as MAR client's CPU frequency, computation model size, camera sampling rate, and user preference, (ii) bases of modeling the energy consumption of an MAR client, and (iii) insights on designing an energy-efficient optimization algorithm.

6.2.1.1 Testbed Setup

Our testbed consists of three major components: MAR client, edge server, and power monitor. Note that this paper focuses on the MAR application in which an MAR client captures physical environmental information through the camera and sends the information to an edge server for object detection. The detailed processing pipeline is shown in Figure 6.1.

Edge Server. The edge server is developed to process received image frames and to send the detection results back to the MAR client. We implement an edge server on an Nvidia Jetson AGX Xavier, which connects to a WiFi access point (AP) through a 1Gbps Ethernet cable. The transmission latency between the server and AP can be ignored. Two major modules are implemented on the edge server:

(i) the *communication handler* which establishes a TCP socket connection with the MAR device and (ii) the *analytics handler* which performs object detection for the MAR client. In this paper, the analytics handler is designed based on a custom framework called Darknet [172] with GPU acceleration and runs YOLOv3 [164], a large Convolutional Neural Networks (CNN) model. The YOLOv3 model used in our experiments is trained on COCO dataset [173] and can detect 80 classes.

MAR Client. We implement an MAR client on a rooted Nexus 6 smartphone which is equipped with Qualcomm Snapdragon 805 SoC (System-on-Chip). The CPU frequency ranges from 0.3 GHz to 2.649 GHz. The MAR client transfers the converted RGB frames to the edge server through a TCP socket connection. To avoid the processing of stale frames, the MAR client sends the latest camera captured frame to the server and waits for the detection result before sending the next frame for detection.

Power Monitor. The power monitor is responsible for measuring the power consumption of the MAR client. We use Monsoon Power Monitor [179], which can sample at 5000 Hz, to provide power supply for the MAR device.

Key Performance Metrics. We define three performance metrics to evaluate the MAR system:

- *Per frame energy consumption:* The per frame energy consumption is the total amount of energy consumed in an MAR client by successfully performing the object detection on one image frame. It includes the energy consumed by camera sampling (i.e., image generation), screen rendering (i.e., preview), image conversion, communication, and operating system.

- *Service latency:* The service latency is the total time needed to derive the detection result on one image frame. It includes the latency of image conversion, transmission, and inference.

- *Accuracy:* The mean average precision (mAP) is a commonly used performance metric to evaluate the detection accuracy of a visual object detection algorithm [176], where a greater accuracy is indicated by a higher mAP.

6.2.1.2 The Impact of CPU Frequency on Power Consumption and Service Latency

In this experiment, we seek to investigate how the CPU frequency impacts the power consumption of the MAR device and the service latency. We set the test device to the *Userspace* Governor and change its CPU frequency manually by writing files in the `/sys/devices/system/cpu/[cpu#]/cpufreq` virtual file system with root privilege. The results are shown in Figure 6.6. The lower the CPU frequency, the longer service latency the MAR client derives and the less power it consumes. However, the reduction of the service latency and the increase of the power consumption is disproportional. For example, as compared to 1.03 GHz, 1.72 GHz reduces about 2% service latency but increases about 15% power consumption. As compared to 0.3 GHz, 0.72 GHz reduces about 60% service latency, but only increases about 20% power consumption.

Insight: *This result advocates adapting the client's CPU frequency for the service latency reduction by trading as little increase of the per frame energy consumption as possible, where the per frame energy consumption is calculated by the power multiplies the service latency.*

6.2.1.3 The Impact of Computation Model Size on Energy Consumption and Service Latency

In this experiment, we implement six object detection algorithms based on the YOLOv3 framework [164] with different computation model sizes. The test device works on the default CPU governor, *Interactive*. Increasing the model size always results in a gain of mAP. However, the gain on mAP becomes smaller as the increase

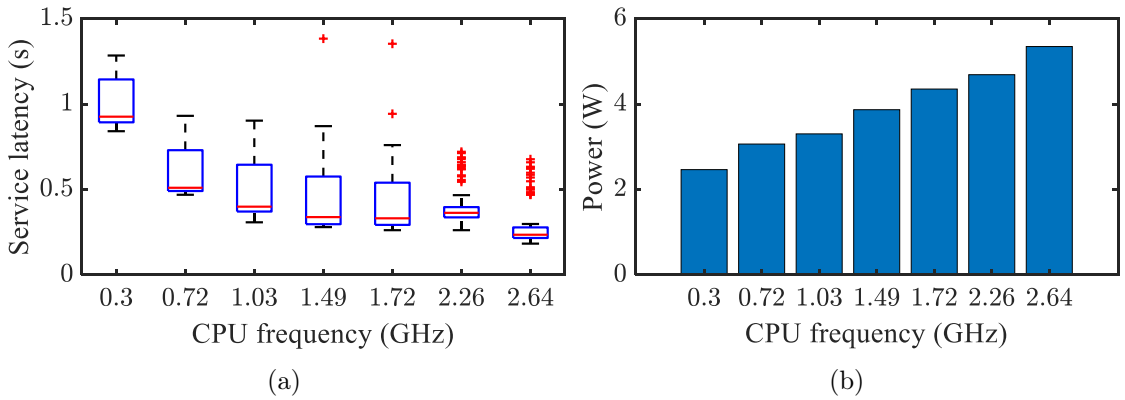


Figure 6.6: CPU frequency vs. power and service latency (computation model size: 320^2 pixels).

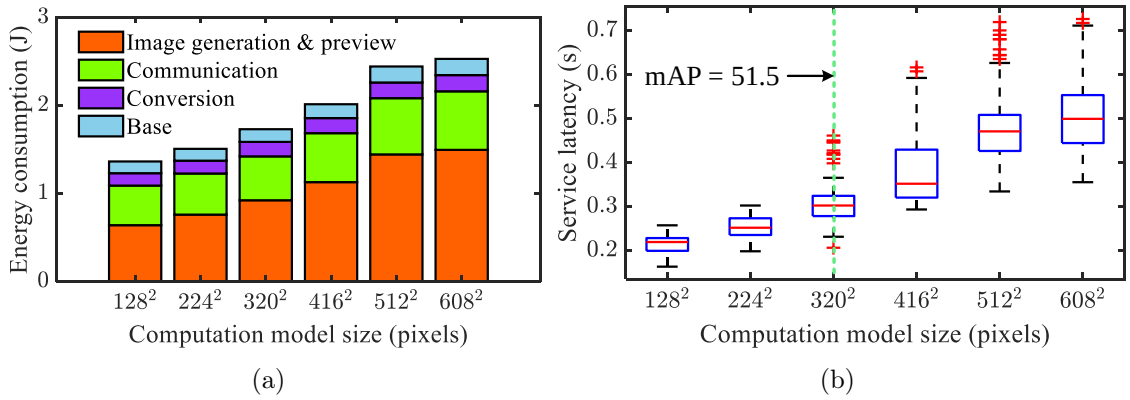


Figure 6.7: Computation model size vs. energy consumption and service latency.

of the model sizes [148]. In addition, the per frame energy consumption and the service latency boost 85% and 130%, respectively, when the model size increases from 128^2 to 608^2 pixels, as shown in Figure 6.7.

Insight: *This result inspires us to trade mAP for the per frame energy consumption and service latency reduction when the model size is large.*

6.2.1.4 The Impact of Camera FPS on Power Consumption

In this experiment, we vary the MAR client's camera FPS to explore how it impacts the device's power consumption, where the camera FPS is defined as the number of frames that the camera samples per second. Figure 6.8a shows that a large camera FPS leads to a high power consumption. However, as shown in Figure 6.1, not every

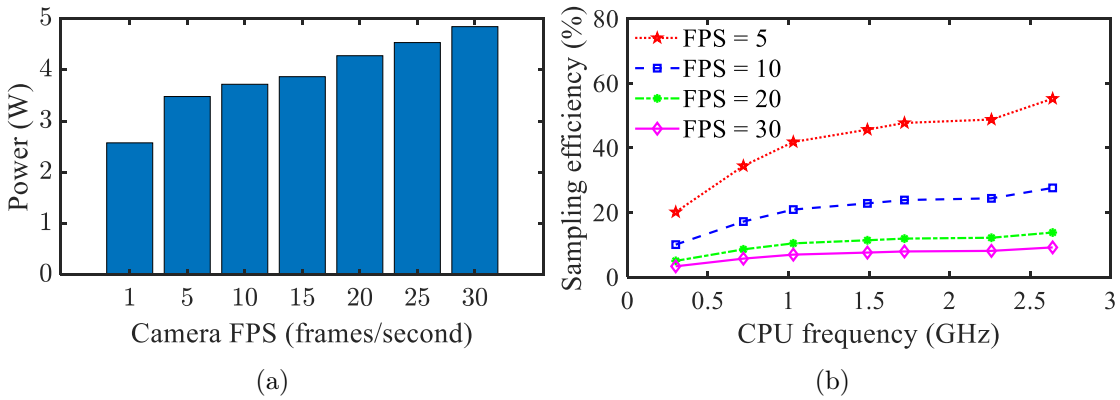


Figure 6.8: Camera FPS vs. power and sampling efficiency (computation model size: 320^2 pixels).

camera captured image frame is sent to the edge server for detection. Because of the need (i) to avoid the processing of stale frames and (ii) to decrease the transmission energy consumption, only the latest camera sampled image frame is transmitted to the server. This may result in the MAR client expending significant reactive power for sampling non-detectable image frames. In Figure 6.8b, we quantify the sampling efficiency with the variation of the camera FPS. As we expected, a large camera FPS leads to a lower sampling efficiency (e.g., less than 2% of the power is consumed for sampling the detectable image frames when the camera FPS is set to 30). However, in most MAR applications, users usually request a high camera FPS for a smoother preview experience, which is critical for tracking targets in physical environments. Interestingly, increasing CPU frequency can reduce the reactive power for sampling, as shown in Figure 6.8b.

Insight: *This result demonstrates that when a high camera FPS is requested, increasing CPU frequency can promote the sampling efficiency but may also boost the power consumption. Therefore, finding a CPU frequency that can balance this tradeoff is critical.*

6.2.1.5 User Preference

An MAR client may have variant preferences in different implementation cases, including:

- **Latency-preferred.** The MAR application of cognitive assistance [15], where a wearable device helps visually impaired people to navigate on a street, may require a low service latency but can tolerate a relatively high number of false positives (i.e., false alarms are fine but missing any potential threats on the street is costly).

- **Accuracy-preferred.** An MAR application for recommending products in shopping malls or supermarkets may tolerate a long latency but requires high detection accuracy and preview smoothness.

- **Preview-preferred.** The MAR drawing assistant application [180], where a user is instructed to trace virtual drawings from the phone, may tolerate a long latency (i.e., only needs to periodically detect the position of the paper where the user is drawing on) but requires a smooth preview to track the lines that the user is drawing.

Insight: *This observation infers that the user preference's diversity may significantly affect the tradeoffs presented above. For instance, for the accuracy-preferred case, trading detection accuracy for the per frame energy consumption or service latency reduction works against the requirement of the user.*

6.2.2 Proposed System Architecture

Based on the above insights, we propose an edge-based MAR system that can reduce the per frame energy consumption of MAR clients by dynamically selecting the optimal combination of MAR configurations (i.e., CPU frequency and computation model size) and radio resource allocations according to user preferences, camera FPS,

and available radio resources at the edge server. To derive the optimal MAR configurations and radio resource allocations, we propose an optimization algorithm (LEAF) that supports low-energy, accurate, and fast MAR applications. LEAF can jointly optimize the CPU frequency, computation model size, and radio resource allocation (explained in detail in Section 6.2.4).

Figure 6.9 shows the overview of our proposed system. In the first step, MAR clients send their service requests and selected camera FPS and user preferences to an edge server. In the second step, according to the received camera FPS and user preferences, the edge server determines the optimal CPU frequency, computation model size, and allocated radio resource for each MAR client using our proposed LEAF algorithm. The determined CPU frequency and computation model size are then sent back to corresponding MAR clients as MAR configuration messages. In the third step, MAR clients set their CPU frequency to the optimal value and resize their latest camera sampled image frames based on the received optimal computation model size. After the CPU frequency adaptation and image frame resizing, MAR clients transmit their image frames to the edge server for object detection. In the final step, the edge server returns detection results to corresponding MAR clients.

However, designing such a system is challenging. From the presented insights in the previous section, the interactions among the MAR system configuration variables, user preference, camera FPS, and the per frame energy consumption are complicated. (i) Some configuration variables improve one performance metric but impair another one. For example, a lower computation model size reduces the service latency but decreases the detection accuracy. (ii) Some configuration variables may affect the same metric in multiple ways. For example, selecting a higher CPU frequency can decrease the per frame energy consumption by increasing the sampling efficiency, but it increases the CPU power, which conversely increases the per frame energy consumption. Unfortunately, there is no analytical model for characterizing these interactions

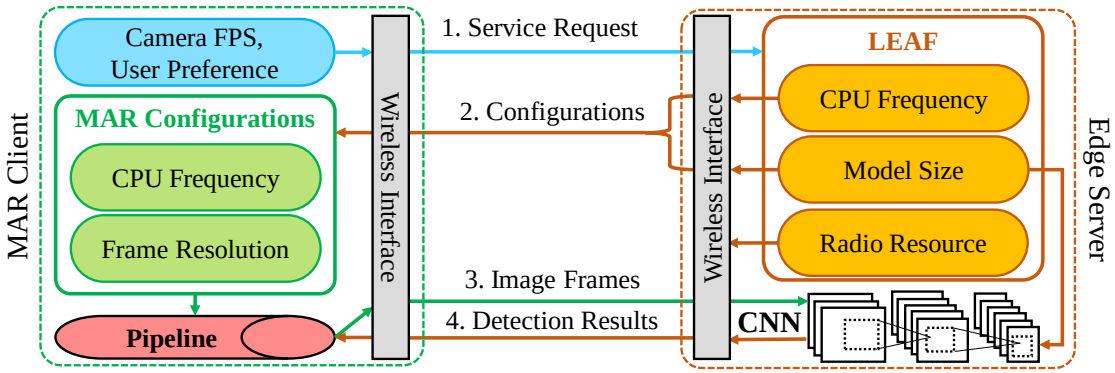


Figure 6.9: Overview of the proposed edge-based MAR system.

in the MAR system and it is not possible to design a prominent optimization algorithm without thoroughly analyzing these interactions.

6.2.3 Proposed Analytical Model and Problem Formulation

In this section, we thoroughly investigate the complicated interactions among the MAR configuration parameters, user preference, camera FPS, and the key performance metrics presented in Section 6.2.1. We first propose a comprehensive analytical model to theoretically dissect the per frame energy consumption and service latency. The proposed model is general enough to handle any MAR device and application. Then, using the proposed model, we further model multiple fine-grained interactions, whose theoretical properties are complex and hard to understand, via a data-driven methodology. Finally, based on the above proposed models, we formulate the MAR reconfiguration as an optimization problem.

6.2.3.1 Analytics-based Modeling Methodology

We consider an edge-based MAR system with K MAR clients and one edge server, where clients are connected to the edge server via a single-hop wireless network. Denote $\mathcal{K}$ as the set of MAR clients. The per frame service latency of the kth MAR client can be defined as

$$L^k = L^k_{cv} + L^k_{tr} + L^k_{inf}, \tag{6.1}$$

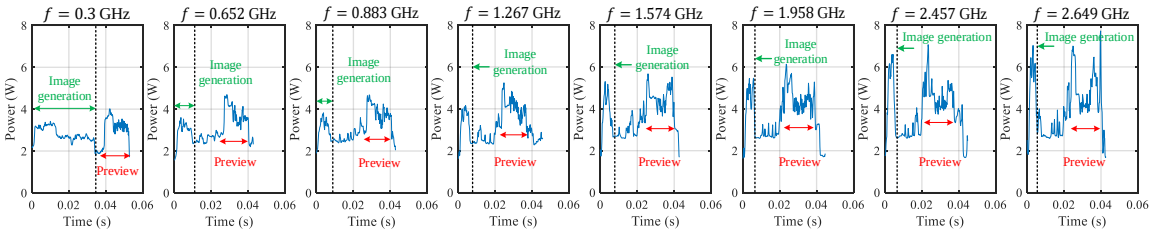


Figure 6.10: The impact of CPU frequency on the power consumption of image generation and preview.

where L^k_{cv} is the image conversion latency caused by converting a buffered camera captured image frame from YUV to RGB; L^k_{tr} is the transmission latency incurred by sending the converted RGB image frame from the kth client to its connected edge server; and L^k_{inf} is the inference latency of the object detection on the server. According to the MAR pipeline depicted in Figure 6.1, the per frame energy consumption of the kth MAR client can be defined as

$$E^k = E^k_{img} + E^k_{cv} + E^k_{com} + E^k_{bs}, \tag{6.2}$$

where E^k_{img} is the image generation and preview energy consumption incurred by image sampling, processing, and preview rendering; E^k_{cv} is the image conversion energy consumption; E^k_{com} is the wireless communication energy consumption, which includes four phases: *promotion, data transmission, tail*, and *idle*; and E^k_{bs} is the MAR device base energy consumption.

The Model of Image Generation and Preview. Image generation is the process that an MAR client transfers its camera sensed continuous light signal to a displayable image frame. Preview is the process of rendering the latest generated image frame on the client's screen. As these two processes are executed in parallel with the main thread, their execution delays are not counted in the per frame service latency.

As depicted in Figure 6.7a, the energy consumption of image generation and preview is the largest portion of the per frame energy consumption. To understand how

energy is consumed in image generation and preview and what configuration variables impact it, we conduct a set of experiments. We find that *the power consumption of image generation and preview highly depends on the CPU frequency.* Figure 6.10 shows the power consumption of image generation and preview under different CPU frequencies, where the camera FPS is set to 15. A higher CPU frequency results in a higher average power consumption. In addition, the image generation delay is also closely related to the CPU frequency, where a higher CPU frequency always leads to a shorter delay. However, the delay of rendering a preview is only related to the GPU frequency, which is out of the scope of this paper. Thus, we consider the preview delay as a fixed value with any CPU frequencies. We model the energy consumption of the kth MAR client's image generation and preview within a service latency as

$$E^k_{img} = \left(\int_0^{t^k_{gt}(f_k)} P^k_{gt}(f_k)\, dt + \int_0^{t_{prv}} P^k_{prv}(f_k)\, dt \right) \cdot fps_k \cdot L^k, \tag{6.3}$$

where P^k_{gt}, P^k_{prv}, t^k_{gt}, t_{prv} are the power consumption of image generation, preview, the delay of image generation, and preview, respectively; f_k is the CPU frequency; fps_k is the camera FPS; P^k_{gt}, P^k_{prv}, and t^k_{gt} are functions of f_k.

The Model of Image Conversion. Image conversion is processed through the MAR client's CPU; hence, the conversion latency and power consumption highly depend on the CPU frequency. We define L^k_{cv} and E^k_{cv} a function of f_k. Therefore, the major source of the power consumption of the image conversion is the CPU computation. The power consumption of mobile CPUs can be divided into two components, $P^k_{cv} = P_{leak} + P^k_{dynamic}$ [158], where P_{leak} is independent and $P^k_{dynamic}$ is dependent upon the CPU frequency. (i) P_{leak} is the power originating from leakage effects and is in essence not useful for the CPU's purpose. In this paper, we consider P_{leak} a constant value ϵ. (ii) $P^k_{dynamic}$ is the power consumed by the logic gate switching at f_k and is proportional to $V_k^2 f_k$, where V_k is the supply voltage for the CPU. Due to the DVFS for the power saving purpose, e.g. a higher f_k will be supplied by a larger V_k,

each f_k matches with a specific V_k, where $V_k \propto (\alpha_1 f_k + \alpha_2)$; α_1 and α_2 are two positive coefficients. Thus, the energy consumption of converting a single image frame of the kth MAR client can be modeled as

$$E_{cv}^k = P_{cv}^k L_{cv}^k = (\alpha_1^2 f_k^3 + 2\alpha_1\alpha_2 f_k^2 + \alpha_2 f_k + \epsilon) \cdot L_{cv}^k(f_k). \tag{6.4}$$

The Model of Wireless Communication and Inference. Intuitively, the wireless communication latency is related to the data size of the transmitted image frame (determined by the frame resolution) and wireless data rate. As the data size of detection results is usually small, we do not consider the latency caused by returning the detection results [148]. In this paper, we use s_k^2 (pixels) to represent the computation model size of the kth MAR client. The client must send image frames whose resolutions are not smaller than s_k^2 to the edge server to obtain the corresponding detection accuracy. Thus, the most efficient way is to transmit the image frame with the resolution of s_k^2 to the server. Denote σ as the number of bits required to represent the information carried by one pixel. The data size of an image frame is calculated as σs_k^2 bits. Let B_k be the wireless bandwidth derived by the kth MAR client. We model the transmission latency of the kth client as

$$L_{tr}^k = \frac{\sigma s_k^2}{R_k}, \tag{6.5}$$

where R_k is the average wireless data rate of the kth client, which is a function of B_k.

In addition to the computation model size and wireless bandwidth, the transmission latency is also determined by the MAR client's CPU frequency. This is because the image transmission uses TCP as the transport layer protocol, and TCP utilizes substantial CPU capacity to handle congestion avoidance, buffer, and re-transmission requests. For example, when the CPU frequency is low, the remaining CPU capacity may not be adequate to process the TCP task; thus, the TCP throughput is decreased.

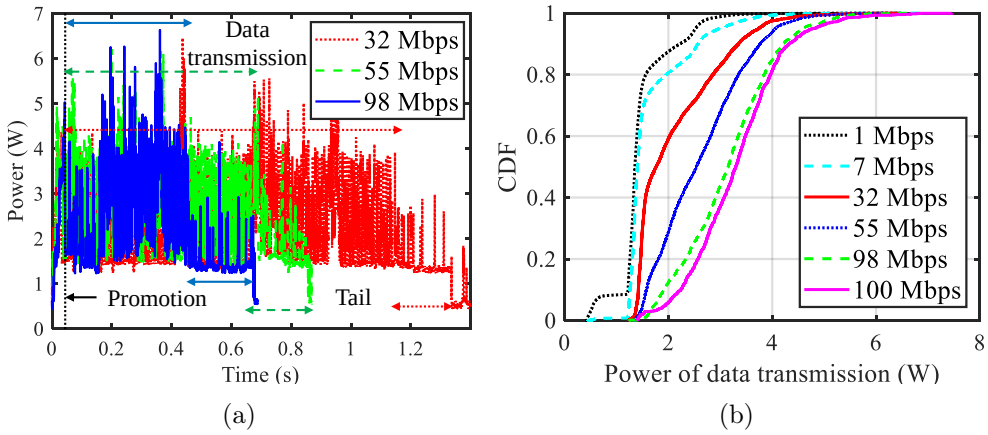


Figure 6.11: MAR client's wireless interface power consumption.

Therefore, R_k is also a function of f_k, i.e., $R_k(B_k, f_k)$. In this paper, $R_k(B_k, f_k)$ is defined as

$$R_k(B_k, f_k) = r_k^{max}(B_k) \cdot r_k^*(f_k), \tag{6.6}$$

where $r_k^{max}(B_k)$ is the network throughput, which is not affected by the variation of the MAR client's CPU frequency, and is only determined by the bandwidth (more comprehensive model of this part can be found in [150], which is out of the scope of this paper); $r_k^*(f_k)$ represents the impact of the CPU frequency on the TCP throughput.

In WiFi networks, when transmitting a single image frame, the MAR client's wireless interface experiences four phases: promotion, data transmission, tail, and idle. When an image transmission request comes, the wireless interface enters the promotion phase. Then, it enters the data transmission phase to send the image frame to the edge server. After completing the transmission, the wireless interface is forced to stay in the tail phase for a fixed duration and waits for other data transmission requests and the detection results. If the MAR client does not receive the detection result in the tail phase, it enters the idle phase and waits for the feedback from its associated edge server. Figure 6.11 depicts the measured power consumption of the MAR client that transmits a 3840×2160 pixel image with different throughput. We

find that the average power consumption of the data transmission phase increases as the throughput grows. However, the average power consumption and the duration of promotion and tail phases are almost constant. Therefore, we model the energy consumption of the kth MAR client in the duration that starts from the promotion phase to obtaining the object detection result as

$$E^k_{com} = P^k_{tr}(R_k(B_k, f_k))L^k_{tr} + P^k_{idle}t^k_{idle} + P_{pro}t_{pro} + P_{tail}t_{tail}, \tag{6.7}$$

where P^k_{tr}, P^k_{idle}, P_{pro}, and P_{tail} are the average power consumption of the data transmission, idle, promotion, and tail phases, respectively; t^k_{idle}, t_{pro}, and t_{tail} are the durations of the idle, promotion, and tail phases, respectively;

$$P^k_{idle}t^k_{idle} = \begin{cases} 0, & L^k_{inf}(s^2_k) \leq t_{tail}, \\ P^k_{bs} \cdot (L^k_{inf}(s^2_k) - t_{tail}), & L^k_{inf}(s^2_k) > t_{tail}, \end{cases} \tag{6.8}$$

where P^k_{bs} is the MAR device's base power consumption; $L^k_{inf}(s^2_k)$ is the inference latency on the edge server, which is determined by the computation model size [148]. Note that our proposed wireless communication model can also be used in other wireless networks (e.g., LTE).

The Model of Base Energy. In this paper, the base energy consumption is defined as the energy consumed by the MAR clients' CPU without any workloads, except running its operating system, and the energy consumed by the screen without any rendering. Because the screen's brightness is not a critical factor that affects the object detection performance, it is considered as a constant value in our proposed power model. Thus, the base power consumption is only a function of the CPU frequency. We model the base energy consumption of the kth MAR client within a service latency as

$$E_{bs}^{k} = \begin{cases} P_{bs}^{k}(f_k) \cdot L^k, & L_{inf}^{k}(s_k^2) \leq t_{tail}, \\ P_{bs}^{k}(f_k) \cdot (L^k - L_{inf}^{k}(s_k^2) + t_{tail}), & L_{inf}^{k}(s_k^2) > t_{tail}. \end{cases} \tag{6.9}$$

6.2.3.2 Regression-based Modeling Methodology

As shown above, some interactions or functions in our proposed analytical models still cannot be expressed clearly in an analytic form. This is because of (i) the lack of analytic understandings of some interactions and (ii) specific coefficients/functions that may vary with different MAR device models. For example, in (6.4), the specific coefficients in $P_{cv}^{k}(f_k)$ are unknown due to the lack of theoretical knowledge and vary with different MAR device models.

Therefore, we propose a data-driven methodology to address the above challenge, where those interactions with inadequate analytic understandings can be modeled and trained offline via empirical measurements and regression analyses. Note that regression-based modeling methodology is one of the most widely used approaches in developing mobile CPU's property models (e.g., CPU power and temperature variation modeling) and has shown to be effective in estimating CPU properties [143, 156, 157]. We use our testbed to collect measurements. The test MAR device is selected to work at 18 different CPU frequencies ranging from 0.3 to 2.649 GHz. In addition, in order to obtain fine-grained regression models and eliminate the interference among different workloads on the device power consumption, we develop three Android applications; each is applied with a specific function of the MAR client, which includes image generation and preview, image conversion, and image transmission applications. The developed regression models are shown in Figure 6.12 and Table 6.3. Note that to obtain a statistical confidence in the experimental results, each data point in Figure 6.12 is derived by generating, transmitting, and detecting 1000 image frames and calculating the average values. The root mean square error

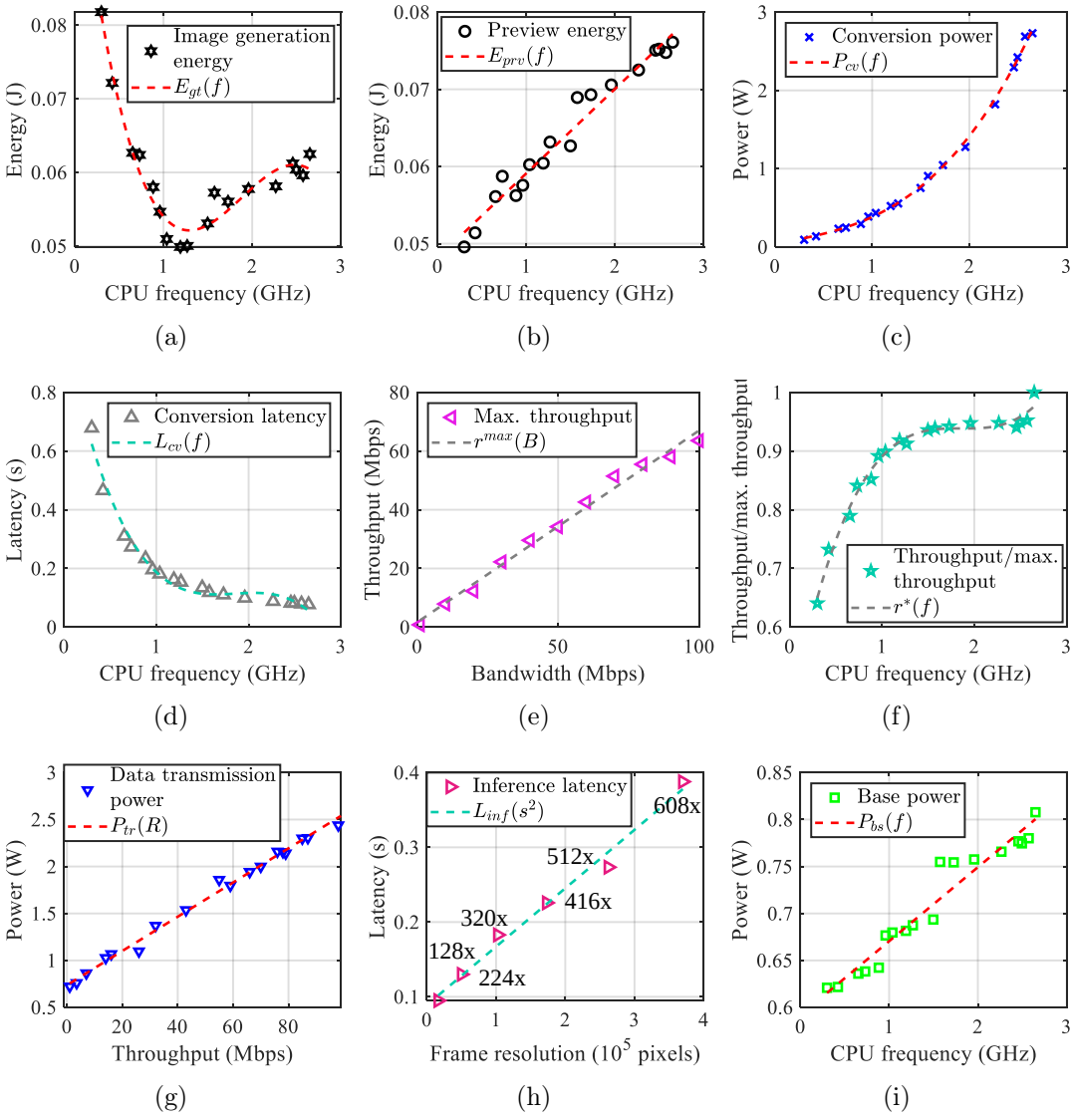


Figure 6.12: The proposed regression-based models.

(RMSE) is applied for calculating the average model-prediction error in the units of the variable of interest [181].

6.2.3.3 Problem Formulation

Based on the above proposed models, we formulate the MAR reconfiguration as a multi-objective optimization problem [182]. We aim to *minimize the per frame energy consumption of multiple MAR clients in the system while satisfying the user preference of each*. We introduce two positive weight parameters λ_1^k and λ_2^k to characterize the

Table 6.3: The proposed regression-based models.

	Proposed models	RMSE
$E_{gt}(f)$	$-0.01071f^3 + 0.06055f^2 - 0.1028f + 0.107$	0.002
$E_{prv}(f)$	$0.01094f + 0.04816$	0.002
$P_{cv}(f)$	$0.1124f^3 + 0.01f^2 + 0.2175f + 0.04295$	0.041
$L_{cv}(f)$	$-0.145f^3 + 0.8f^2 - 1.467f + 0.996$	0.025
$r^{max}(B)$	$0.677B$	2.403
$r^*(f)$	$0.07651f^3 - 0.4264f^2 + 0.7916f + 0.4489$	0.013
$P_{tr}(R)$	$0.01821R + 0.7368$	0.052
$L_{inf}(s^2)$	$0.07816s^2 + 0.08892$	0.838
$P_{bs}(f)$	$0.07873f + 0.5918$	0.015

user preference of the kth MAR client, where λ_1^k and λ_2^k can be specified by the client. We adopt the weighted sum method [183] to express the multi-object optimization problem as

$$\begin{aligned} \mathscr{P}_0 : \min_{\{f_k, s_k, B_k, \forall k \in \mathcal{K}\}} \quad & Q = \sum_{k \in \mathcal{K}} (E^k + \lambda_1^k L^k - \lambda_2^k A_k) \\ s.t. \quad & C_1 : \sum_{k \in \mathcal{K}} B_k \leq B_{max}; \\ & C_2 : L^k \leq L_{max}^k, \forall k \in \mathcal{K}; \\ & C_3 : F_{min} \leq f_k \leq F_{max}, \forall k \in \mathcal{K}; \\ & C_4 : s_k \in \{s_{min}, ..., s_{max}\}, \forall k \in \mathcal{K}; \end{aligned} \tag{6.10}$$

where A_k is an object detection accuracy function in terms of the kth MAR client selected computation model size s_k^2 (e.g., $A(s_k^2) = 1 - 1.578e^{-6.5 \times 10^{-3} s_k}$ [148]); L_{max}^k is the maximum tolerable service latency of the kth client; B_{max} is the maximum wireless bandwidth that an edge server can provide for its associated MAR clients. In practical scenarios, an edge server may simultaneously offer multiple different services for its associated users, e.g., video streaming, voice analysis, and content caching. Hence, the edge server may reallocate its bandwidth resource based on the user distribution. In this paper, we assume that B_{max} varies with time randomly. The constraint C_1

represents that MAR clients' derived bandwidth cannot exceed the total bandwidth allocated for the MAR service on the edge server; the constraint C_2 guarantees that the service latency of MAR clients are no larger than their maximum tolerable latency; the constraints C_3 and C_4 are the constraints of the MAR device's CPU frequency and computation model size configurations, where s_k is a discrete variable and its values depend on the available computation models in the MAR system.

6.2.4 Proposed LEAF Optimization Algorithm

As shown in the previous section, problem $\mathscr{P}_0$ is a mixed-integer non-linear programming problem (MINLP) which is difficult to solve [184]. In order to solve this problem, we propose the LEAF algorithm based on the block coordinate descent (BCD) method [185].

To solve problem $\mathscr{P}_0$, we relax the discrete variable s_k into continuous variable $\hat{s_k}$. The problem is relaxed as

$$\begin{aligned} \mathscr{P}_1 : \min_{\{f_k, \hat{s_k}, B_k, \forall k \in \mathcal{K}\}} \quad & Q = \sum_{k \in \mathcal{K}} (E^k + \lambda_1^k L^k - \lambda_2^k A_k) \\ s.t. \quad & C_1, C_2, C_3 \\ & \hat{C}_4 : s_{min} \leq \hat{s_k} \leq s_{max}, \forall k \in \mathcal{K}. \end{aligned} \tag{6.11}$$

According to the BCD method, we propose the LEAF algorithm which solves Problem $\mathscr{P}_1$ by sequentially fixing two of three variables and updating the remaining one. We iterate the process until the value of each variable converges.

$\nabla y(x)$ is denoted as the partial derivative of function y corresponding to variable x. Denote $\text{Proj}_{\mathcal{X}}(x)$ as the Euclidean projection of x onto $\mathcal{X}$; $\text{Proj}_{\mathcal{X}}(x) \triangleq \arg\min_{v \in \mathcal{X}} \|x - v\|^2$.

The procedure of our proposed solution is summarized as:

- Given $\hat{s_k}$ and B_k, we can derive a new f_k according to

$$f_k^{(j+1)} = \text{Proj}_{\mathcal{X}_f}\left(f_k^{(j)} - \gamma_k \nabla Q_k\left(f_k^{(j)}\right)\right), \forall k \in \mathcal{K}; \tag{6.12}$$

where $\gamma_k > 0$ is a constant step size and $\mathcal{X}_f$ is the bounded domain constrained by C_3. Based on the BCD method, we repeat (6.12) until the derived f_k is converged and then update f_k.

- Given f_k and B_k, we can derive a new $\hat{s_k}$ according to

$$\hat{s_k}^{(j+1)} = \text{Proj}_{\mathcal{X}_{\hat{s}}}\left(\hat{s_k}^{(j)} - \eta_k \nabla Q_k\left(\hat{s_k}^{(j)}\right)\right), \forall k \in \mathcal{K}; \tag{6.13}$$

where $\eta_k > 0$ is a constant step size and $\mathcal{X}_{\hat{s}}$ is the bounded domain constrained by $\hat{C}_4$. Based on the BCD method, we repeat (6.13) until the derived $\hat{s_k}$ is converged and then update $\hat{s_k}$.

- Given f_k and $\hat{s_k}$, the problem is simplified to

$$\begin{aligned} \min_{\{B_k, \forall k \in \mathcal{K}\}} \quad & Q = \sum_{k \in \mathcal{K}} (E^k + \lambda_1^k L^k - \lambda_2^k A_k) \\ s.t. \quad & C_1 : \sum_{k \in \mathcal{K}} B_k \leq B_{max}; \\ & C_2 : L^k \leq L_{max}^k, \forall k \in \mathcal{K}; \end{aligned} \tag{6.14}$$

where constraints C_3 and $\hat{C}_4$ are irrelevant to this problem.

The Lagrangian dual decomposition method is utilized to solve the above problem, where the Lagrangian function is

$$\begin{aligned} \mathcal{L}(B_k, \mu, \boldsymbol{\beta}) = & \sum_{k \in \mathcal{K}} (E^k + \lambda_1^k L^k - \lambda_2^k A_k) \\ & + \mu(\sum_{k \in \mathcal{K}} B_k - B_{max}) + \sum_{k \in \mathcal{K}} \beta_k (L^k - L_{max}^k), \end{aligned} \tag{6.15}$$

where μ and $\boldsymbol{\beta}$ are the Lagrange multipliers, (i.e., $\boldsymbol{\beta}$ is a Lagrange multiplier vector), corresponding to constraints C_1 and C_2, respectively. The Lagrangian dual problem can therefore be expressed as

$$\begin{aligned} \max_{\{\mu,\boldsymbol{\beta}\}} \quad & g(\mu,\boldsymbol{\beta}) = \min_{\{B_k,\forall k\in\mathcal{K}\}} \mathcal{L}(B_k,\mu,\boldsymbol{\beta}) \\ & s.t. \quad \mu \geq 0, \boldsymbol{\beta} \geq 0. \end{aligned} \tag{6.16}$$

Here, $g(\mu,\boldsymbol{\beta})$ is concave with respect to B_k.

Lemma 1. *The problem $\mathscr{P}_1$ is convex with respect to B_k.*

Proof. For any feasible $B_i, B_j, \forall i,j \in \mathcal{K}$, we have

$$\frac{\partial^2 Q}{\partial B_i \partial B_j} = \begin{cases} 0, & i \neq j, \\ \Psi_i \cdot \frac{\partial^2 (1/r^{max})}{\partial B_i \partial B_j}, & i = j, \end{cases} \tag{6.17}$$

where $\Psi_i = \frac{\left[fps_i(E_{gt}(f_i)+E_{prv}(f_i))+P_{tr}^i(0)+P_{bs}(f_i)+\lambda_1^i\right]\sigma s_i^2}{r_i^*(f_i)}$ which is positive, and $\frac{\partial^2(1/r^{max})}{\partial B_i \partial B_j} = \frac{2}{0.677B_i^3} > 0$. Thus, the Hessian matrix $\mathbf{H} = \left(\frac{\partial^2 Q}{\partial B_i \partial B_j}\right)_{K\times K}$ is symmetric and positive definite. Constraint C_1 is linear and C_2 is convex with respect to B_k. Constraints C_3 and C_4 are irrelevant to B_k. Therefore, $\mathscr{P}_1$ is strictly convex with respect to B_k. □

Therefore, based on the Karush-Kuhn-Tucker (KKT) condition [186], the sufficient and necessary condition of the optimal allocated bandwidth for the kth MU can be expressed as

$$B_k^* = \sqrt{\frac{\Phi(f_k, s_k, \beta_k)}{0.677\mu}}, \tag{6.18}$$

where $\Phi_k = \frac{\left[fps_i(E_{gt}(f_i)+E_{prv}(f_i))+P_{tr}^i(0)+P_{bs}(f_i)+\lambda_1^i+\beta_k\right]\sigma s_i^2}{r_i^*(f_i)}$.

Next, the sub-gradient method [186] is used to solve the dual problem. Based on the sub-gradient method, the dual variables of the kth MAR clients in the $(j+1)$th iteration are

$$\begin{cases} \mu_k^{(j+1)} = \max\left\{0, \left[\mu^{(j)} + \vartheta_k^{\mu} \nabla g(\mu^{(j)})\right]\right\}, \forall k \in \mathcal{K}; \\ \beta_k^{(j+1)} = \max\left\{0, \left[\beta_k^{(j)} + \vartheta_k^{\beta} \nabla g(\beta_k^{(j)})\right]\right\}, \forall k \in \mathcal{K}; \end{cases} \tag{6.19}$$

where $\vartheta_k^{\mu} > 0$ and $\vartheta_k^{\beta} > 0$ are the constant step sizes.

Based on the above mathematical analysis, we propose an MAR optimization algorithm, LEAF, which can dynamically determines the CPU frequency of multiple MAR devices, selects the computation model sizes, and allocates the wireless bandwidth resources. The pseudo code of the proposed LEAF MAR algorithm is presented in Algorithm 7. First, the LEAF is initialized with the lowest CPU frequency, the smallest computation model size, and evenly allocated bandwidth resources among MAR devices. We then iteratively update f_k, $\hat{s_k}$, and B_k until the LEAF converges (i.e., line 7-8 in Algorithm 7). In addition, $\hat{s_k}$ is a relaxed value of the computation model size. Thus, it may not match any pre-installed computation model in a real system. In this case, the LEAF selects the computation model size s_k that is the closest to the relaxed one $\hat{s_k}$ (i.e., line 10 in Algorithm 7). Since the LEAF MAR algorithm is developed based on the BCD method and follows the convergence results in [185], we claim that the LEAF converges to a local optimal solution.

Algorithm 7: The LEAF MAR Algorithm

Input: λ_1^k, λ_2^k, L_{max}^k, B_{max}, fps_k, and τ, $\forall k \in \mathcal{K}$.
Output: f_k, s_k, and B_k, $\forall k \in \mathcal{K}$.

1 $B_k \leftarrow B_{max}/|\mathcal{K}|$, $\hat{s_k} \leftarrow s_{min}$, $\forall k \in \mathcal{K}$, $i \leftarrow 1$;
2 **while** *True* **do**
3 $\quad f_k \leftarrow$ solving $\mathscr{P}_1$ with fixed $\hat{s_k}$ and B_k;
4 $\quad \hat{s_k} \leftarrow$ solving $\mathscr{P}_1$ with fixed f_k and B_k;
5 $\quad B_k \leftarrow$ solving $\mathscr{P}_1$ with fixed f_k and $\hat{s_k}$;
6 $\quad Q_i \leftarrow \sum_{k \in \mathcal{K}} (E^k + \lambda_1^k L^k - \lambda_2^k A_k)$
7 $\quad$ **if** $|(Q_i - Q_{i-1})/Q_i| \leq \tau$ **then**
8 $\quad\quad$ **break;** ▷ Converges
9 $\quad i \leftarrow i + 1$;
10 $s_k = \underset{s \in \{s_{min} \ldots s_{max}\}}{\arg\min} |s - \hat{s_k}|$, $\forall k \in \mathcal{K}$;
11 **return** f_k, s_k, and B_k, $\forall k \in \mathcal{K}$.

Table 6.4: Power and duration of promotion & tail phases.

P_{pro} (W)	t_{pro} (s)	P_{tail} (W)	t_{tail} (s)
1.97 ± 0.08	0.034 ± 0.004	1.61 ± 0.15	0.21 ± 0.02

6.2.5 Performance Evaluation

In this section, we evaluate both the proposed MAR analytical energy model and LEAF algorithm. We first validate our analytical model by comparing the estimated energy consumption with the physical energy measurement (obtained from our developed testbed described in Section 6.2.1). The Mean Absolute Percentage Error (MAPE) is used for quantifying the estimation error. Then, we evaluate the per frame energy consumption, service latency, and detection accuracy of the proposed LEAF algorithm under variant bandwidth and user preferences through data-driven simulations.

6.2.5.1 Analytical Model Validation

The measured power and duration of promotion and tail phases in WiFi are shown in Table 6.4 (note that LTE has different values [187]). As shown in Figure 6.13, we validate the proposed analytical model with respect to MAR client's CPU frequency, computation model size, allocated bandwidth, and camera FPS. Each measured data is the average of the per frame energy consumption of 1000 image frames. The calculated MAPE of these four cases are $6.1\% \pm 3.4\%$, $7.6\% \pm 4.9\%$, $6.9\% \pm 3.9\%$, and $3.7\% \pm 2.6\%$, respectively. Therefore, our proposed energy model can estimate the MAR per frame energy consumption very well.

6.2.5.2 Performance Evaluation of LEAF

We simulate an edge-based MAR system with an edge server and multiple MAR clients. Each MAR client may select a different camera FPS, which is obtained randomly in the range of $[1, 30]$ frames. The default user preference is $\lambda_1 = 0.3$ and

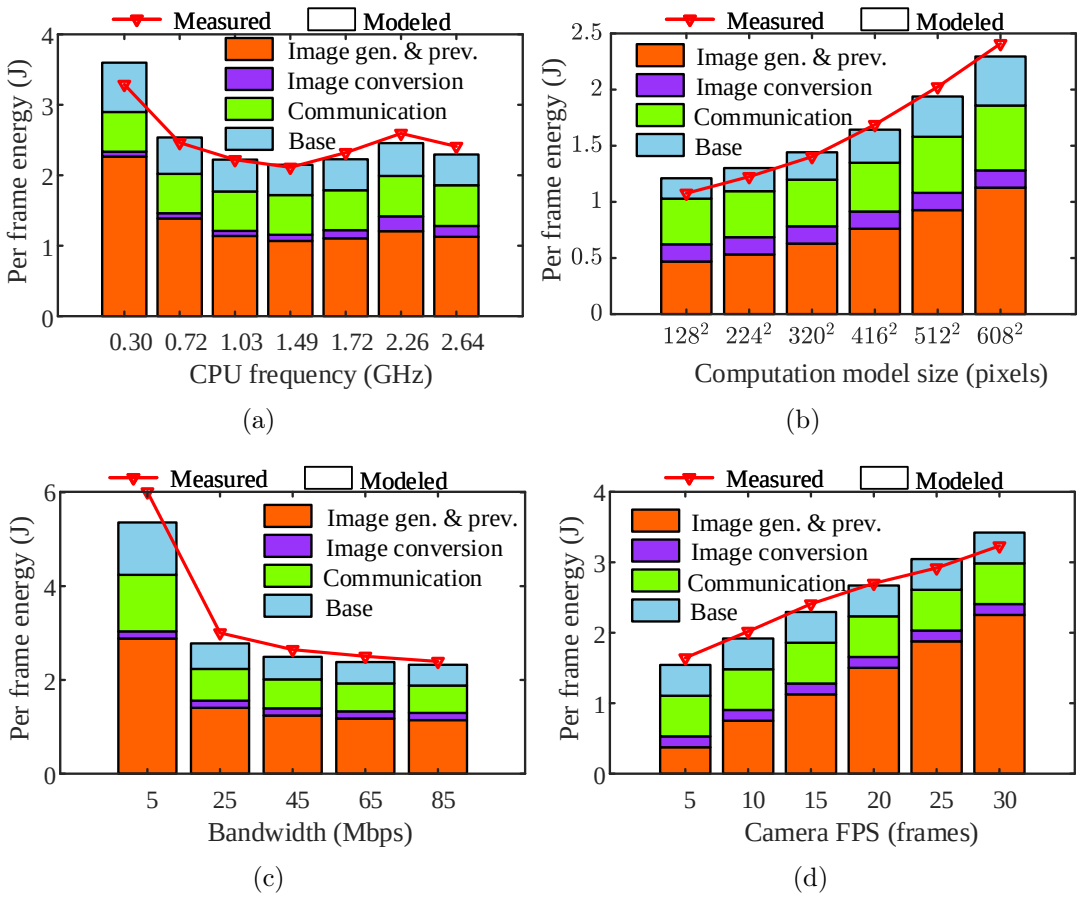


Figure 6.13: Measured data vs. estimated data from our proposed analytical model.

$\lambda_2 = 1.8$. We compare our proposed LEAF algorithm with two other algorithms summarized as follows:

- **FACT + Interactive:** It uses FACT [148] to select the computation model size, which is optimized for the tradeoff between the service latency and the detection accuracy. As FACT does not consider the MAR client's CPU frequency scaling and radio resource allocation at the edge server, we use *Interactive* to conduct CPU frequency scaling and the radio resource is allocated evenly. Note that FACT does not consider the energy efficiency of MAR clients either.

- **Energy-optimized only solution:** It selects the optimal CPU frequency, computation model size, and bandwidth allocation by minimizing the per frame energy consumption of MAR clients in the system without considering user

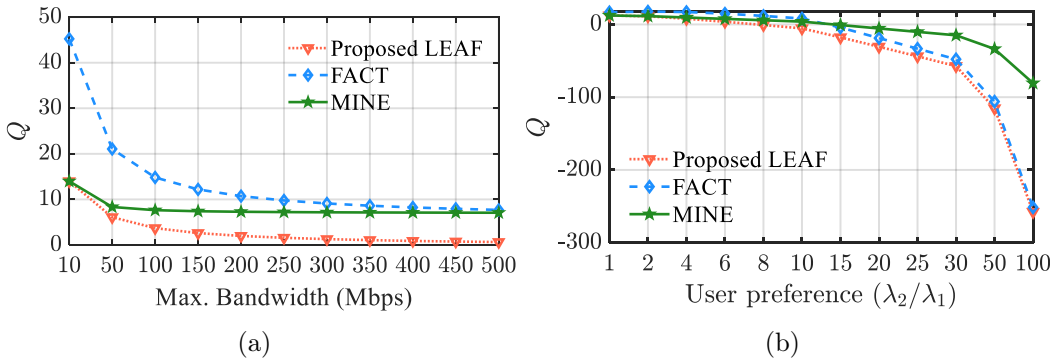


Figure 6.14: Optimality. (a) Q vs. Max. bandwidth; (b) Q vs. user preference.

preferences, which is named as MINE.

Optimality. We first validate the optimality of our proposed LEAF algorithm. As shown in Figure 6.14, LEAF always obtains the minimal Q compared to the other two algorithms under variant maximum available bandwidth and user preference.

Comparison under Variant Max. Bandwidth. We then evaluate the impact of the maximum available bandwidth on the performance of the proposed LEAF. In practical environments, the maximum bandwidth at an edge server for serving its associated MAR clients may vary with the user distribution. For each MAR client, the value of the allocated bandwidth directly impacts not only the service latency and the per frame energy consumption but also the detection accuracy. The evaluation results are depicted in Figure 6.15. (i) Compared to FACT, the proposed LEAF decreases up to 40% per frame energy consumption and 35% service latency with less than 9% loss of object detection accuracy when the Max. bandwidth is 300 Mbps. The performance gap between LEAF and FACT is due to the gain derived through optimizing the clients' CPU frequency and the server radio resource allocation. (ii) Compared to MINE, the proposed LEAF significantly improves the detection accuracy at the cost of a slightly increase of the service latency and per frame energy. The performance gap between LEAF and MINE reflects the gain derived through considering the user preference.

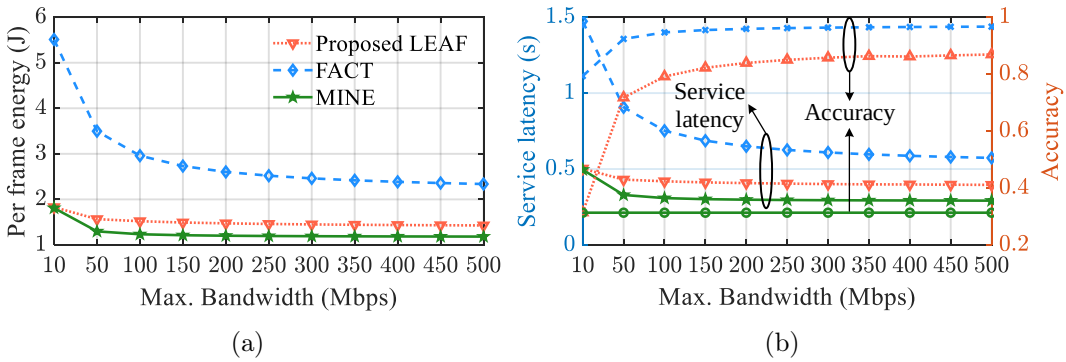


Figure 6.15: System performance vs. Max. bandwidth.

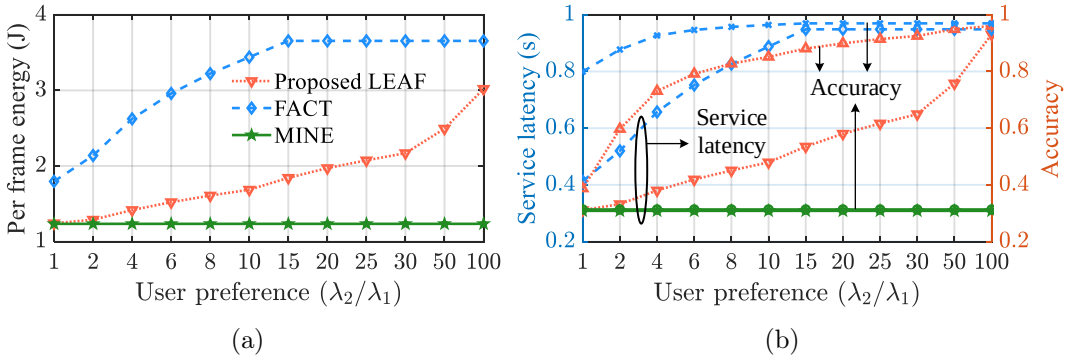


Figure 6.16: System performance vs. user preference.

Comparison under Variant User Preferences. Finally, we evaluate the impact of the user preference on the performance of the proposed LEAF by varying the value of λ_2/λ_1, as shown in Figure 6.16. User preference impacts the tradeoffs among the per frame energy consumption, service latency, and detection accuracy. When λ_2/λ_1 grows, the MAR client emphasizes on the detection accuracy by trading the service latency and per frame energy. Since MINE does not consider the user preference, the variation of λ_2/λ_1 does not change its performance. (i) Compared to FACT, the proposed LEAF reduces over 20% per frame energy consumption while maintaining the same detection accuracy ($\lambda_2/\lambda_1 = 100$). (ii) Compared to MINE, the proposed LEAF is able to enhance over 50% accuracy while ensuring similar per frame energy and service latency ($\lambda_2/\lambda_1 = 2$). Figure 6.16 also shows that, as compared to FACT, the proposed LEAF offers more fine-grained and diverse user preference options for

MAR clients.

CHAPTER 7: CONCLUSION

7.1 Completed Work

In this dissertation, fast and energy-efficient mobility management in MEC networks is investigated. Link-instability and user-mobility incurred challenges in MEC are addressed from four steps. First, an intelligent handoff trigger mechanism is designed to achieve a fast and accurate trigger for seamless mobility support in MEC networks. Second, after a handoff is triggered at a mobile client in MEC networks, the mobile client's application/service that is under processing on an MEC server must be migrated or rebuilt on a new MEC server. The service rebuilding process in MEC networks has two components: radio handoff and service handoff. Most existing mobility management solutions in MEC investigate radio and service handoffs separately. Thus, these handoffs are performed in sequence, which leads to long service rebuilding delay with high energy consumption. Therefore, a service rebuilding design jointly considering radio and service handoffs is proposed. In addition, to minimize performance degradation during mobility caused by radio resource allocation unfairness, single and multiple edge servers radio resource allocation protocols to impartially allocate the uplink and the downlink radio resources in MEC networks are proposed. Lastly, a dynamic configuration adaptation algorithm is proposed for mobile clients to achieve energy-efficient offloading in MEC networks while satisfying variant clients' user preferences.

Explorer, a novel MAR offloading solution for cloudlet-based MAR systems, is proposed to mitigate the analytics staleness caused by the wireless link quality variation, especially the user-mobility-incurred wireless network quality decline. To the best of our knowledge, this is the first formal study of reducing the MAR service latency

from the communication perspective. In addition, *Explorer* is complementary to all existing computation-based MAR offloading solutions to further reduce the MAR service latency. The performance of *Explorer* is validated by both experiments and large-scale simulations.

A smart service rebuilding scheme is proposed, which seamlessly restores offloading services on the target cloudlet while the mobile user is moving. The service rebuilding process includes the radio handoff stage and service handoff stage. A seamless service rebuilding process is achieved via predicting user's target cloudlet before being triggered a radio handoff, by leveraging extracted features from the captured frames of the mobile user's camera. Furthermore, based on the proposed service rebuilding scheme, we design a feature mapping algorithm to achieve a high prediction precision and a short prediction latency. We implement our scheme on a testbed and conduct experiments using real world AR applications. The experimental results show that our proposed scheme decreases the service rebuilding latency by around 65.8%, as compared with the conventional rebuilding process. In addition, we conduct extensive simulations to evaluate the performance of our proposed feature mapping algorithm. Simulation confirms that our algorithm is robust and can predict users' target cloudlet with high precision and low latency.

A novel WLAN system, BELL, is proposed to reduce mobile devices' energy consumption for performing management services. First, an energy-efficient mobility management service, BELL-handoff, is proposed under BELL deployment. We implemented BELL-handoff on a testbed and evaluated its performance. Experimental results show BELL-handoff decreases not only the handoff energy consumption but latency, as compared with C-WLAN. Then, we proposed a user-friendly load balancing service, BELL-2M. We conducted extensive simulations to evaluate BELL-2M performance and mobile devices' battery life within a large-scale deployment of BELL. Simulation results show that BELL-2M can balance APs' load in BELL and MUs in

BELL drain less battery life than in C-WLAN.

E-Auto, a novel communication scheme, is proposed to enable fast, stable, and accurate edge-assisted autonomous driving service for connected vehicles within any road types (e.g., driving on highway with very high speed or local roads with slow speed due to the traffic congestion). In addition, as two key components of the proposed *E-Auto* scheme, a service period allocation algorithm and a frame resolution selection algorithm are designed to guarantee a sufficient frame rate for connected vehicles acquiring either uplink application (offload camera captured frames to the edge server) or downlink application (download entertainment videos). Through network simulations, we evaluate the performance of the proposed *E-Auto* scheme. Simulation results demonstrate that *E-Auto* can provide a high frame rate and low energy consumption autonomous driving service for connected vehicles.

The first detailed experimental study of the energy consumption and the performance of a deep CNN optimized object detection application on a variety of smartphones is conduct. Both local and remote execution cases are examined. We found that the performance of the object detection is heavily affected by different generations of smartphones. Although executing deep learning on remote edge servers is one of the most commonly used approaches to assist low-end smartphones in improving their energy efficiency and performance, contrary to our expectation, remote execution does not always consume less energy and obtain lower latency, as compared to local execution, even when the network quality is excellent. Overall, we believe that our findings give great insights and guidelines to the future design of energy-efficient processing pipeline of CNN optimized object detection.

A user preference based energy-aware edge-based MAR system is proposed, which can reduce the per frame energy consumption of MAR clients without compromising their user preferences by dynamically selecting the optimal combination of MAR configurations and radio resource allocations according to user preferences, camera

FPS, and available radio resources at the edge server. To the best of our knowledge, this is the first analytical energy model for thoroughly investigating the interactions among MAR configuration parameters, user preferences, camera sampling rate, and per frame energy consumption in edge-based MAR systems. Based on the proposed analytical model, the LEAF optimization algorithm is proposed to guide the optimal MAR configurations and resource allocations. The performance of the proposed analytical model is validated against real energy measurements from our testbed and the LEAF algorithm is evaluated through extensive data-driven simulations.

7.2 Future Work

Based on the proposed mobility management designs, two issues can be considered in the future work:

- Current CPU governors cannot achieve energy-efficient object detection on smartphones. A CPU governor specifically designed for CNN-based object detection applications is critical. I plan to well utilize the proposed analytic models to design a specific CPU governor for CNN-based object detection applications.

- A special offloading protocol for MAR applications will be proposed to cooperate with the proposed mobility management protocols. This protocol is used to mitigate the trade-off between the energy consumption and service staleness for MAR offloading in MEC networks.

7.3 Published and Submitted Work

The following list is a summary of my publications.

1. Haoxin Wang and Jiang Xie, “User Preference Based Energy-Aware Mobile AR System with Edge Computing," in *Proc. IEEE International Conference on Computer Communications (INFOCOM),* Apr. 2020, pp.1379–1388.

2. Haoxin Wang and Jiang Xie, “Energy Drain of the Object Detection Processing

Pipeline: Analysis and Implications," *submitted to IEEE Transactions on Green Communications and Networking*, 2020.

3. Haoxin Wang, Tingting Liu, BaekGyu Kim, Chung-Wei Lin, Shinichi Shiraishi, Jiang Xie, and Zhu Han, "Architectural Design Alternatives based on Cloud/Edge/Fog Computing for Connected Vehicles," *accepted by IEEE Communications Surveys and Tutorials*, 2020.

4. Haoxin Wang, Siqi Huang, BaekGyu Kim, Jiang Xie, Han Zhu, and Tao Han, "You Can Enjoy Augmented Reality While Moving Around: A Cloudlet-based Mobile AR System," *submitted to IEEE Transactions on Mobile Computing*, 2020.

5. Haoxin Wang and Jiang Xie, "Edge-based Energy-aware Object Detection for Mobile Augmented Reality," *submitted to IEEE Transactions on Networking.*

6. Haoxin Wang, BaekGyu Kim, Jiang Xie, and Zhu Han, "How is Energy Consumed in Smartphone Deep Learning Apps? Executing Locally vs. Remotely," in *Proc. IEEE Global Communications Conference (GLOBECOM),* Dec. 2019, pp.1-6.

7. Haoxin Wang, BaekGyu Kim, Jiang Xie, and Zhu Han "E-Auto: A Communication Scheme for Connected Vehicles with Edge-Assisted Autonomous Driving," *Proc. IEEE International Conference on Communications (ICC)* Feb. 2019, pp.1-6.

8. Haoxin Wang, Jiang Xie, and Xingya Liu "Rethinking Mobile Devices' Energy Efficiency in WLAN Management Services," *Proc. IEEE SECON,* Jun. 2018, pp.1-9.

9. Haoxin Wang, Jiang Xie, and Tao Han, "A Smart Service Rebuilding Scheme

Across Cloudlets via Mobile AR Frame Feature Mapping," *Proc. IEEE ICC,* May 2018, pp.1-6.

10. Haoxin Wang, Jiang Xie, and Tao Han, "V-Handoff: A Practical Energy Efficient Handoff for 802.11 Infrastructure Networks," *Proc. IEEE ICC,* May 2017, pp.1-6.

REFERENCES

[1] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, and A. Neal, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-edge Computing (MEC) industry initiative*, 2014.

[2] H. Wang, B. Kim, J. Xie, and Z. Han, "How is energy consumed in smartphone deep learning apps? Executing locally vs. remotely," in *Proc. IEEE Globecom*, pp. 1–6, 2019.

[3] I. . W. Group, "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.

[4] V. Mhatre and K. Papagiannaki, "Using smart triggers for improved user performance in 802.11 wireless networks," in *Proc. 2006 ACM the 4th international conference on Mobile systems, applications and services*, pp. 246–259, 2006.

[5] H. Wu, K. Tan, Y. Zhang, and Q. Zhang, "Proactive scan: Fast handoff with smart triggers for 802.11 wireless LAN," in *Proc. IEEE INFOCOM*, pp. 749–757, 2007.

[6] P. Khadivi, T. D. Todd, and D. Zhao, "Handoff trigger nodes for hybrid IEEE 802.11 WLAN/cellular networks," in *Proc. 2004 IEEE First International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QSHINE)*, pp. 164–170, 2004.

[7] H. Wang, J. Xie, and T. Han, "V-handoff: A practical energy efficient handoff for 802.11 infrastructure networks," in *Proc. IEEE ICC*, 2017.

[8] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, and M. Satyanarayanan, "Adaptive VM handoff across cloudlets," tech. rep., CMU-CS-15-113, CMU School of Computer Science, 2015.

[9] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can WiFi deliver?," *IEEE/ACM Transactions on Networking (ToN)*, vol. 21, no. 2, pp. 536–550, 2013.

[10] A. Detti, M. Pomposini, N. Blefari-Melazzi, S. Salsano, and A. Bragagnini, "Offloading cellular networks with information-centric networking: The case of video streaming," in *Proc. 2012 IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–3, 2012.

[11] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "Microcast: Cooperative video streaming on smartphones," in *Proc. 2012 ACM*

the 10th international conference on Mobile systems, applications, and services, pp. 57–70, 2012.

[12] S. Dimatteo, P. Hui, B. Han, and V. O. Li, "Cellular traffic offloading through WiFi networks," in *Proc.2011 IEEE 8th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, pp. 192–201, 2011.

[13] V. A. Siris, M. Anagnostopoulou, and D. Dimopoulos, "Improving mobile video streaming with mobility prediction and prefetching in integrated cellular-WiFi networks," in *Proc. International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pp. 699–704, Springer, 2013.

[14] C. Vallati, A. Virdis, E. Mingozzi, and G. Stea, "Mobile-edge computing come home connecting things in future smart homes using LTE device-to-device communications," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 77–83, 2016.

[15] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. ACM SenSys*, 2014.

[16] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: latency-aware video analytics on edge computing platform," in *Proc. ACM/IEEE SEC*, p. 15, 2017.

[17] P. Jain, J. Manweiler, and R. Roy Choudhury, "Low bandwidth offload for mobile AR," in *Proc. ACM CoNEXT*, pp. 237–251, 2016.

[18] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE Infocom*, 2018.

[19] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE Infocom*, 2018.

[20] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[21] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[22] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[23] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[24] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *arXiv preprint arXiv:1702.05309*, 2017.

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

[26] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet*, pp. 48–55, Citeseer, 2014.

[27] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[28] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Proc. IEEE Global Internet of Things Summit (GIoTS)*, (Geneva, Switzerland), Jun. 2017.

[29] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, pp. 54–61, Apr. 2017.

[30] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[31] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. the 2015 ACM workshop on mobile big data*, pp. 37–42, 2015.

[32] G. I. Klas, "Fog computing and mobile edge cloud gain momentum open fog consortium, ETSI MEC and cloudlets," 2015.

[33] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: A key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[34] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[35] ETSI, "Mobile edge computing (MEC); technical requirements," Mar. 2016.

[36] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 271–278, 2015.

[37] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *Proc. 2017 IEEE 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pp. 165–172, 2017.

[38] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. Lau, "Dynamic virtual machine management via approximate markov decision process," in *Proc. 2016 IEEE The 35th Annual International Conference on Computer Communications (INFOCOM)*, pp. 1–9, 2016.

[39] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. the thirteenth ACM international symposium on Mobile Ad Hoc Networking and Computing*, pp. 145–154, 2012.

[40] A. Mtibaa, K. A. Harras, and A. Fahim, "Towards computational offloading in mobile device clouds," in *Proc. 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, pp. 331–338, 2013.

[41] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, "Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud," in *Proc. the first international workshop on Mobile cloud computing & networking*, pp. 19–26, ACM, 2013.

[42] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. 2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, pp. 9–16, 2015.

[43] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wireless communications*, vol. 20, no. 3, pp. 14–22, 2013.

[44] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *Proc. 2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 1291–1296, 2013.

[45] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. 2016 IEEE The 35th Annual International Conference on Computer Communications (INFOCOM)*, pp. 1–9, 2016.

[46] C. M. Huang, M. S. Chiang, D. T. Dao, W. L. Su, S. Xu, and H. Zhou, "V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture," *IEEE Access*, vol. 6, pp. 17741–17755, 2018.

[47] S. Nunna, A. Kousaridas, M. Ibrahim, M. Dillinger, C. Thuemmler, H. Feussner, and A. Schneider, "Enabling real-time context-aware collaboration through 5G and mobile edge computing," in *Proc. IEEE 12th International Conference on Information Technology - New Generations*, (Las Vegas, NV, USA), pp. 601–605, Apr. 2015.

[48] C.-Y. Chang, K. Alexandris, N. Nikaein, K. Katsalis, and T. Spyropoulos, "MEC architectural implications for LTE/LTE-A networks," in *Proc. 2016 ACM the Workshop on Mobility in the Evolving Internet Architecture*, pp. 13–18, 2016.

[49] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consumer Electronics Magazine*, vol. 5, no. 4, pp. 84–91, 2016.

[50] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *arXiv preprint arXiv:1502.01815*, 2015.

[51] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. 2014 IEEE Australasian Telecommunication Networks and Applications Conference (ATNAC)*, pp. 117–122, 2014.

[52] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, and P. Narasimhan, "The case for mobile edge-clouds," in *Proc. 2013 IEEE 10th International Conference on Ubiquitous Intelligence & Computing and 10th International Conference on Autonomic & Trusted Computing (UIC/ATC)*, pp. 209–215, 2013.

[53] I. Giannoulakis, E. Kafetzakis, I. Trajkovska, P. S. Khodashenas, I. Chochliouros, C. Costa, I. Neokosmidis, and P. Bliznakov, "The emergence of operator-neutral small cells as a strong case for cloud computing at the mobile edge," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1152–1159, 2016.

[54] F. Lobillo, Z. Becvar, M. A. Puente, P. Mach, F. L. Presti, F. Gambetti, M. Goldhamer, J. Vidal, A. K. Widiawan, and E. Calvanesse, "An architecture for mobile computation offloading on cloud-enabled LTE small cells," in *Proc. 2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 1–6, 2014.

[55] M. A. Puente, Z. Becvar, M. Rohlik, F. Lobillo, and E. C. Strinati, "A seamless integration of computationally-enhanced base stations into mobile networks towards 5G," in *Proc. 2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pp. 1–5, 2015.

[56] Z. Becvar, M. Rohlik, P. Mach, M. Vondra, T. Vanek, M. A. Puente, and F. Lobillo, "Distributed architecture of 5G mobile networks for efficient computation management in mobile edge computing," *5G Radio Access Networks: Centralized RAN, Cloud-RAN and Virtualization of Small Cells*, vol. 29, 2017.

[57] S. Wang, G.-H. Tu, R. Ganti, T. He, K. Leung, H. Tripp, K. Warr, and M. Zafer, "Mobile micro-cloud: Application classification, mapping, and deployment," in *Proc. Annual Fall Meeting of ITA (AMITA)*, 2013.

[58] K. Wang, M. Shen, J. Cho, A. Banerjee, J. Van der Merwe, and K. Webb, "Mobiscud: A fast moving personal cloud in the mobile network," in *Proc. the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pp. 19–24, ACM, 2015.

[59] T. Taleb and A. Ksentini, "Follow me cloud: Interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[60] T. Taleb, A. Ksentini, and P. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, 2016.

[61] A. Aissioui, A. Ksentini, and A. Gueroui, "An efficient elastic distributed SDN controller for follow-me cloud," in *Proc. 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 876–881, 2015.

[62] J. Liu, T. Zhao, S. Zhou, Y. Cheng, and Z. Niu, "CONCERT: a cloud-based architecture for next-generation cellular systems," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 14–22, 2014.

[63] A. Ceselli, M. Premoli, and S. Secci, "Mobile edge cloud network design optimization," *IEEE/ACM Transactions on Networking (TON)*, vol. 25, no. 3, pp. 1818–1831, 2017.

[64] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE ISCC*, pp. 59–66, 2012.

[65] Y. Liu, J. E. Fieldsend, and G. Min, "A framework of fog computing: Architecture, challenges, and optimization," *IEEE Access*, vol. 5, 2017.

[66] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: architecture and service modes," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 18–24, 2015.

[67] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, 2017.

[68] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. 2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, 2016.

[69] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Communications Letters*, vol. 18, no. 10, pp. 1779–1782, 2014.

[70] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2516–2529, 2015.

[71] 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.

[72] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1481–1484, 2017.

[73] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, pp. 4514–4526, May 2018.

[74] S. Melendez and M. P. McGarry, "Computation offloading decisions for reducing completion time," in *Proc. 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 160–164, 2017.

[75] S. E. Mahmoodi, R. Uma, and K. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Transactions on Cloud Computing*, 2016.

[76] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Joint allocation of radio and computational resources in wireless application offloading," in *Proc. 2013 Future Network and Mobile Summit (FutureNetworkSummit)*, pp. 1–10, 2013.

[77] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE access*, vol. 4, pp. 5896–5907, 2016.

[78] J. Dolezal, Z. Becvar, and T. Zeman, "Performance evaluation of computation offloading from mobile device to the edge of mobile network," in *Proc. 2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, pp. 1–7, 2016.

[79] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-optimal resource scheduling and computation offloading in small cell networks," in *Proc. 2015 22nd International Conference on Telecommunications (ICT)*, pp. 313–318, 2015.

[80] W. Labidi, M. Sarkiss, and M. Kamoun, "Joint multi-user resource scheduling and computation offloading in small cell networks," in *Proc. 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 794–801, 2015.

[81] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[82] O. Muñoz, A. P. Iserte, J. Vidal, and M. Molina, "Energy-latency trade-off for multiuser wireless computation offloading," in *Proc. 2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 29–33, 2014.

[83] K. Ha, Y. Abe, T. Eiszler, Z. Chen, W. Hu, B. Amos, R. Upadhyaya, P. Padmanabhan, and M. Satyanarayanan, "You can teach elephants to dance: Agile VM handoff for edge computing," in *Proc. ACM SEC*, 2017.

[84] A. Ksentini, T. Taleb, and M. Chen, "A markov decision process-based service migration procedure for follow me cloud," in *Proc. 2014 IEEE International Conference on Communications (ICC)*, pp. 1350–1354, 2014.

[85] X. Sun and N. Ansari, "PRIMAL: Profit maximization avatar placement for mobile edge computing," in *Proc. 2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2016.

[86] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. 2014 IEEE Military Communications Conference (MILCOM)*, pp. 835–840, 2014.

[87] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *Proc. 2015 IEEE IFIP Networking Conference (IFIP Networking)*, pp. 1–9, 2015.

[88] A. Nadembega, A. S. Hafid, and R. Brisebois, "Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE," in *Proc. 2016 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2016.

[89] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2017.

[90] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.

[91] S. Secci, P. Raad, and P. Gallard, "Linking virtual machine mobility to user mobility," *IEEE Transactions on Network and Service Management*, vol. 13, no. 4, pp. 927–940, 2016.

[92] Z. Becvar, J. Plachy, and P. Mach, "Path selection using handover in mobile networks with cloud-enabled small cells," in *Proc. 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, pp. 1480–1485, 2014.

[93] J. Plachy, Z. Becvar, and P. Mach, "Path selection enabling user mobility and efficient distribution of data for computation at the edge of mobile network," *Computer Networks*, vol. 108, pp. 357–370, 2016.

[94] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[95] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398–401, 2017.

[96] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, no. 5, pp. 2795–2808, 2016.

[97] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[98] P. Di Lorenzo, S. Barbarossa, and S. Sardellitti, "Joint optimization of radio resources and code partitioning in mobile edge computing," *arXiv preprint arXiv:1307.3835*, 2013.

[99] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738–4755, 2015.

[100] K. Wang, K. Yang, and C. S. Magurawalage, "Joint energy minimization and resource allocation in C-RAN with mobile cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2018.

[101] M. Molina Pena, O. Muñoz Medina, A. Pascual Iserte, and J. Vidal Manzano, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *Proc. 2014 IEEE International Symp. on Personal Indoor and Mobile Radio Comm. (PIMRC)*, pp. 1093–1098, 2014.

[102] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and cpu time allocation for mobile edge computing," in *Proc. 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

[103] A. Al-Shuwaili and O. Simeone, "Optimal resource allocation for mobile edge computing-based augmented reality applications," *arXiv preprint arXiv:1611.09243*, 2016.

[104] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," in *Proc. 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

[105] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

[106] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *Proc. 2016 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2016.

[107] V. Di Valerio and F. L. Presti, "Optimal virtual machines allocation in mobile femto-cloud computing: An MDP approach," in *Proc. 2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 7–11, 2014.

[108] S. S. Tanzil, O. N. Gharehshiran, and V. Krishnamurthy, "Femto-cloud formation: A coalitional game-theoretic approach," in *Proc. 2015 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2015.

[109] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, "Small cell clustering for efficient distributed fog computing: A multi-user case," in *Proc. 2015 IEEE 82nd Vehicular Technology Conference (VTC Fall)*, pp. 1–5, 2015.

[110] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. 2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pp. 1–6, 2015.

[111] M. Vondra and Z. Becvar, "QoS-ensuring distribution of computation load among cloud-enabled small cells," in *Proc. 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, pp. 197–203, 2014.

[112] S. Wang, M. Zafer, and K. K. Leung, "Online placement of multi-component applications in edge computing environments," *IEEE Access*, vol. 5, pp. 2514–2533, 2017.

[113] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Proc. 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2016.

[114] X. Sun, N. Ansari, and Q. Fan, "Green energy aware avatar migration strategy in green cloudlet networks," in *Proc. 2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 139–146, 2015.

[115] X. Sun and N. Ansari, "Green cloudlet network: A distributed green mobile cloud network," *IEEE Network*, vol. 31, no. 1, pp. 64–70, 2017.

[116] J. Xu and S. Ren, "Online learning for offloading and autoscaling in renewable-powered mobile edge computing," in *Proc. 2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

[117] P. Mach and Z. Becvar, "Cloud-aware power control for cloud-enabled small cells," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, pp. 1038–1043, 2014.

[118] P. Mach and Z. Becvar, "Cloud-aware power control for real-time application offloading in mobile edge computing," *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 5, pp. 648–661, 2016.

[119] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *International conference on wireless algorithms, systems, and applications*, pp. 685–695, Springer, 2015.

[120] B. Liang, *Mobile edge computing*. Cambridge University Press, 2017.

[121] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. 2014 IEEE Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 1–8, 2014.

[122] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[123] ETSI, "Executive briefing - mobile edge computing (MEC) initiative."

[124] ETSI, "Mobile-edge computing (MEC): Service scenarios."

[125] ETSI, "Mobile-edge computing (MEC): Framework and reference architecture."

[126] S. Antipolis, "ETSI first mobile edge computing proof of concepts at MEC world congress," Sep. 2016.

[127] N. Sprecher, J. Friis, R. Dolby, and J. Reister, "Edge computing prepares for a multi-access future," in *Proc. MEC World Congress*, 2016.

[128] G. Brown, "Mobile edge computing use cases and deployment options," *Juniper White Paper*, pp. 1–10, 2016.

[129] L. Ma, S. Yi, and Q. Li, "Efficient service handoff across edge servers via docker container migration," in *Proc. ACM SEC*, 2017.

[130] L. Dimopoulou, G. Leoleis, and I. Venieris, "Fast handover support in a WLAN environment: challenges and perspectives," *IEEE network*, vol. 19, no. 3, 2005.

[131] B.-J. Chang and J.-F. Chen, "Cross-layer-based adaptive vertical handoff with predictive RSS in heterogeneous wireless networks," *IEEE Transactions on vehicular technology*, vol. 57, no. 6, 2008.

[132] B.-J. Chang, J.-F. Chen, C.-H. Hsieh, and Y.-H. Liang, "Markov decision process-based adaptive vertical handoff with RSS prediction in heterogeneous wireless networks," in *Proc. IEEE WCNC*, pp. 1–6, 2009.

[133] W. Wanalertlak, B. Lee, C. Yu, M. Kim, S.-M. Park, and W.-T. Kim, "Behavior-based mobility prediction for seamless handoffs in mobile wireless networks," *Wireless Networks*, vol. 17, pp. 645–658, 2011.

[134] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc. ACM NSDI*, pp. 273–286, 2005.

[135] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast Transparent Migration for Virtual Machines," in *Proc. USENIX ATC*, pp. 391–394, 2005.

[136] L. Ma, S. Yi, N. Carter, and Q. Li, "Efficient live migration of edge services leveraging container layered storage," *IEEE Transactions on Mobile Computing*, 2018.

[137] X. Xu, J. Liu, and X. Tao, "Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation," *IEEE Access*, vol. 5, pp. 16406–16415, 2017.

[138] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255–11268, 2017.

[139] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, 2017.

[140] M. J. Chang, Z. Abichar, and C.-Y. Hsu, "WiMAX or LTE: Who will lead the broadband mobile Internet?," *IEEE IT professional*, vol. 12, no. 3, pp. 26–32, 2010.

[141] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. ACM the 10th international conference on Mobile systems, applications, and services*, (Low Wood Bay, Lake District, UK), pp. 225–238, June 2012.

[142] W. L. Tan, W. C. Lau, O. Yue, and T. H. Hui, "Analytical models and performance evaluation of drive-thru internet systems," *IEEE Journal on Selected Areas in Communications*, vol. 29, pp. 207–222, January 2011.

[143] W. Hu and G. Cao, "Energy-aware CPU frequency scaling for mobile video streaming.," in *Proc. IEEE ICDCS*, pp. 2314–2321, 2017.

[144] W. Hu and G. Cao, "Energy optimization through traffic aggregation in wireless networks," in *Proc. IEEE INFOCOM*, pp. 916–924, 2014.

[145] H. Wang, B. Kim, J. Xie, and Z. Han, "E-auto: A communication scheme for connected vehicles with edge-assisted autonomous driving," in *Proc. IEEE ICC*, pp. 1–6, 2019.

[146] Y. Geng, Y. Yang, and G. Cao, "Energy-efficient computation offloading for multicore-based mobile devices," in *Proc. IEEE INFOCOM*, pp. 46–54, 2018.

[147] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "Deepdecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE INFOCOM*, pp. 1421–1429, 2018.

[148] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pp. 756–764, 2018.

[149] J. Hanhirova, T. Kämäräinen, S. Seppälä, M. Siekkinen, V. Hirvisalo, and A. Ylä-Jääski, "Latency and throughput characterization of convolutional neural networks for mobile computer vision," in *Proc. 9th ACM Multimedia Systems Conference*, pp. 204–215, 2018.

[150] Y. Xiao, Y. Cui, P. Savolainen, M. Siekkinen, A. Wang, L. Yang, A. Ylä-Jääski, and S. Tarkoma, "Modeling energy consumption of data transmission over Wi-Fi," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1760–1773, 2013.

[151] H. Wang, J. Xie, and X. Liu, "Rethinking mobile devices' energy efficiency in WLAN management services," in *Proc. IEEE SECON*, pp. 1–9, 2018.

[152] A. M. Srivatsa and J. Xie, "A performance study of mobile handoff delay in IEEE 802.11-based wireless mesh networks," in *Proc. IEEE ICC*, pp. 2485–2489, 2008.

[153] X. Liu and J. Xie, "A practical self-adaptive rendezvous protocol in cognitive radio ad hoc networks," in *Proc. IEEE INFOCOM*, pp. 2085–2093, 2014.

[154] U. Narayanan and J. Xie, "Signaling cost analysis of handoffs in a mixed IPv4/IPv6 mobile environment," in *Proc. IEEE GLOBECOM*, pp. 1792–1796, 2007.

[155] H. Wang, J. Xie, and T. Han, "A smart service rebuilding scheme across cloudlets via mobile AR frame feature mapping," in *Proc. IEEE ICC*, pp. 1–6, 2018.

[156] A. Shye, B. Scholbrock, and G. Memik, "Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures," in *Proc. IEEE/ACM International Symposium on Microarchitecture*, pp. 168–178, 2009.

[157] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and stable run-time power modeling for mobile and embedded CPUs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, 2016.

[158] K. DeVogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "Modeling the temperature bias of power consumption for nanometer-scale CPUs in application processors," in *Proc. IEEE International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV)*, pp. 172–180, 2014.

[159] F. Xu, Y. Liu, Q. Li, and Y. Zhang, "V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics," in *Proc. USENIX Symposium on Network Systems Design and Implementation (NSDI)*, pp. 43–55, 2013.

[160] Speedtest. `https://www.speedtest.net/reports/united-states/2018/Mobile/`.

[161] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," in *Proc. ACM SIGMETRICS*, 2013.

[162] Y. Zhang, J. Wang, Y. He, X. Ji, Y. Kang, D. Liu, and B. Li, "Furion: Towards energy-efficient WiFi offloading under link dynamics," in *Proc. 13th IEEE SECON 2016*, pp. 1–9, 2016.

[163] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE CVPR*, 2016.

[164] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[165] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[166] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[167] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. ACM the 23th International Conference on Architectural Support for Programming Languages and Operating Systems*, (Williamsburg, VA), pp. 751–766, March 2018.

[168] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Providence, RI), pp. 3354–3361, June 2012.

[169] X. Zhang and K. G. Shin, "E-mili: energy-minimizing idle listening in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 1441–1454, Sept. 2012.

[170] S. K. Saha, P. Deshpande, P. P. Inamdar, R. K. Sheshadri, and D. Koutsonikolas, "Power-throughput tradeoffs of 802.11 n/ac in smartphones," in *Proc. IEEE INFOCOM*, pp. 100–108, 2015.

[171] "Antutu benchmark." `https://www.antutu.com/en/`.

[172] J. Redmon, "Darknet: Open source neural networks in C." `http://pjreddie.com/darknet/`, 2013–2016.

[173] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. European Conference on Computer Vision*, 2014.

[174] "Tensorflow lite." `https://www.tensorflow.org/lite/`.

[175] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[176] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[177] X. Ran, H. Chen, Z. Liu, and J. Chen, "Delivering deep learning to mobile devices via offloading," in *Proc. ACM Workshop on Virtual Reality and Augmented Reality Network*, pp. 42–47, 2017.

[178] H. Wang and J. Xie, "User preference based energy-aware mobile AR system with edge computing," in *Proc. IEEE INFOCOM*, pp. 1379–1388, 2020.

[179] "Monsoon power monitor." `https://www.msoon.com/`.

[180] "SketchAR."

[181] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005.

[182] K. Deb, *Multi-objective Optimization Using Evolutionary Algorithms*, vol. 16. John Wiley & Sons, 2001.

[183] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Structural and Multidisciplinary Optimization*, vol. 41, no. 6, pp. 853–862, 2010.

[184] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, pp. 1–131, 2013.

[185] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear gauss–seidel method under convex constraints," *Operations Research Letters*, vol. 26, no. 3, pp. 127–136, 2000.

[186] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[187] W. Hu and G. Cao, "Energy-aware video streaming on smartphones," in *Proc. IEEE INFOCOM*, pp. 1185–1193, 2015.