ABSTRACT

ON DISTRIBUTED MOBILE EDGE COMPUTING

by
Xiang Sun

Mobile Cloud Computing (MCC) has been proposed to offload the workloads of mobile applications from mobile devices to the cloud in order to not only reduce energy consumption of mobile devices but also accelerate the execution of mobile applications. Owing to the long End-to-End (E2E) delay between mobile devices and the cloud, offloading the workloads of many interactive mobile applications to the cloud may not be suitable. That is, these mobile applications require a huge amount of computing resources to process their workloads as well as a low E2E delay between mobile devices and computing resources, which cannot be satisfied by the current MCC technology.

In order to reduce the E2E delay, a novel cloudlet network architecture is proposed to bring the computing and storage resources from the remote cloud to the mobile edge. In the cloudlet network, each mobile user is associated with a specific Avatar (i.e., a dedicated Virtual Machine (VM) providing computing and storage resources to its mobile user) in the nearby cloudlet via its associated Base Station (BS). Thus, mobile users can offload their workloads to their Avatars with low E2E delay (i.e., one wireless hop). However, mobile users may roam among BSs in the mobile network, and so the E2E delay between mobile users and their Avatars may become worse if the Avatars remain in their original cloudlets. Thus, Avatar handoff is proposed to migrate an Avatar from one cloudlet into another to reduce the E2E delay between the Avatar and its mobile user. The LatEncy aware Avatar handDoff (LEAD) algorithm is designed to determine the location of each mobile user's Avatar in each time slot in order to minimize the average E2E delay among all the mobile

users and their Avatars. The performance of LEAD is demonstrated via extensive simulations.

The cloudlet network architecture not only facilitates mobile users in offloading their computational tasks but also empowers Internet of Things (IoT). Popular IoT resources are proposed to be cached in nearby brokers, which are considered as application layer middleware nodes hosted by cloudlets in the cloudlet network, to reduce the energy consumption of servers. In addition, an Energy Aware and latency guaranteed dynamic reSourcE caching (EASE) strategy is proposed to enable each broker to cache suitable popular resources such that the energy consumption from the servers is minimized and the average delay of delivering the contents of the resources to the corresponding clients is guaranteed. The performance of EASE is demonstrated via extensive simulations.

The future work comprises two parts. First, caching popular IoT resources in nearby brokers may incur unbalanced traffic loads among brokers, thus increasing the average delay of delivering the contents of the resources. Thus, how to balance the traffic loads among brokers to speed up IoT content delivery process requires further investigation. Second, drone assisted mobile access network architecture will be briefly investigated to accelerate communications between mobile users and their Avatars.

ON DISTRIBUTED MOBILE EDGE COMPUTING

by
Xiang Sun

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

May 2018

APPROVAL PAGE

ON DISTRIBUTED MOBILE EDGE COMPUTING

Xiang Sun

Dr. Nirwan Ansari, Dissertation Advisor — Date
Distinguished Professor, Helen and John C. Hartmann Department of Electrical and Computer Engineering, NJIT

Dr. Abdallah Khreishah, Committee Member — Date
Associate Professor, Helen and John C. Hartmann Department of Electrical and Computer Engineering, NJIT

Dr. Bipin Rajendran, Committee Member — Date
Associate Professor, Helen and John C. Hartmann Department of Electrical and Computer Engineering, NJIT

Dr. Roberto Rojas-Cessa, Committee Member — Date
Professor, Helen and John C. Hartmann Department of Electrical and Computer Engineering, NJIT

Dr. Chonggang Wang, Committee Member — Date
Member of Technical Staff, Convida Wireless, InterDigital Communications, Inc.

BIOGRAPHICAL SKETCH

Author:	Xiang Sun
Degree:	Doctor of Philosophy
Date:	May 2018
Date of Birth:	
Place of Birth:	

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2018

- Master of Science in Computer Applications Technology,
  Hebei University of Engineering, Hebei, P. R. China, 2011

- Bachelor of Science in Electronic Information,
  Hebei University of Engineering, Hebei, P. R. China, 2008

Major: Electrical Engineering

Presentations and Publications:

Journal articles:

X. Sun and N. Ansari, "Green Cloudlet Network: A Sustainable Platform for Mobile Cloud Computing," *IEEE Transactions on Cloud Computing*, doi: 10.1109/TCC.2017.2764463, early access.

X. Sun and N. Ansari, "Adaptive Avatar Handoff in the Cloudlet Network," *IEEE Transactions on Cloud Computing*, doi: 10.1109/TCC.2017.2701794, early access.

X. Sun and N. Ansari, "Dynamic Resource Caching in the IoT Application Layer for Smart Cities," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 606-613, April 2018.

X. Sun and N. Ansari, "Traffic Load Balancing among Brokers at the IoT Application Layer," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 489-502, March 2018.

N. Ansari and **X. Sun**, "Mobile Edge Computing Empowers Internet of Things," *IEICE Transactions on Communications,*, vol. E101.B, no. 3, pp. 604-619, 2018. (Invited paper)

X. Sun and N. Ansari, "Cloudlet Networks: Empowering Mobile Networks with Computing Capabilities," *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) Communications Frontiers*, vol. 12, no. 4, July 2017.

X. Sun and N. Ansari, "Latency Aware Workload Offloading in the Cloudlet Network," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1481-1484, July 2017.

Q. Fan, N. Ansari and **X. Sun**, "Energy Driven Avatar Migration in Green Cloudlet Networks," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1601-1604, July 2017.

X. Sun and N. Ansari, "Green Cloudlet Network: A Distributed Green Mobile Cloud Network," *IEEE Network*, vol. 31, no. 1, pp. 64-70, January/February 2017.

X. Sun and N. Ansari, "EdgeIoT: Mobile Edge Computing for Internet of Things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22-29, December 2016.

X. Sun, N. Ansari and R. Wang, "Optimizing Resource Utilization of a Data Center," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2822-2846, Fourth quarter 2016.

K. Guo, Y. Lu, Y. Li and **X. Sun**, "An Activity Recognition-Assistance Algorithm based on Hybrid Semantic Model in Smart Home," *International Journal of Distributed Sensor Networks*, vol. 12, no. 8, August 2016.

X. Guo, L. Chu and **X. Sun**, "Accurate Localization of Multiple Sources Using Semidefinite Programming based on Incomplete Range Matrix," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5319-5324, July, 2016.

J. Zhao, **X. Sun**, Z. Wei and Z. Li, "A New MAC Protocol for Moving Target in Distributed Wireless Sensor Networks," *Wireless Sensor Network*, vol. 3, no. 2, pp.62-72, 2011.

J. Zhao, Z. Wei, Z. Li and **X. Sun**, "An Adaptive MAC Protocol based on Spatial and Temporal Correlation in Wireless Sensor Networks," *Journal of Communications*, vol. 6, no. 3, pp.262-269, 2011.

Conference papers

X. Sun and N. Ansari, "Jointly Optimizing Drone-Mounted Base Station Placement and User Association in Heterogeneous Networks," in *IEEE International Conference on Communications (ICC 2018)*, Kansas City, MO, 2018, to be presented.

X. Sun and N. Ansari, "Latency Aware Drone Base Station Placement in Heterogeneous Networks," in *IEEE Global Communications Conference (GLOBECOM 2017)*, Singapore, 2017, pp. 1-6.

X. Sun and N. Ansari, "PRIMAL: PRofIt Maximization Avatar pLacement for Mobile Edge Computing," in *IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–6. (Best paper award)

X. Sun, N. Ansari and Q. Fan, "Green Energy Aware Avatar Migration Strategy in Green Cloudlet Networks," in *IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*, Vancouver, BC, 2015, pp. 139–146.

X. Sun and N. Ansari, "Energy-Optimized Bandwidth Allocation Strategy for Mobile Cloud Computing in LTE Networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, 2015, pp. 2120–2125.

X. Sun and N. Ansari, "Improving Bandwidth Efficiency and fairness in cloud computing," in *IEEE Global Communications Conference (GLOBECOM)*, Atlanta, GA, 2013, pp. 2313–2318.

J. Zhao, **X. Sun**, Z. Li, H. Hou, and P. Sun, "A Novel Date Collision Rate-Aware Energy Model Design with Single-Channel CSMA/CA MAC in Wireless Sensor Networks," in *IEEE 2nd International Conference on Computer Engineering and Technology (ICCET)*, Chengdu, China, 2010, pp. 483–486.

J. Zhao, H. Li and **X. Sun**, "Research on the Signal Random Attenuation Coefficient Based on RSSI in WSN Localization Technology," in *IEEE 5th International Conference on Wireless Communications, Networking and Mobile Computing*, Beijing, China, 2009, pp. 1–4.

J. Zhao and **X. Sun**, "MAC protocol based on T-MAC Multi-hop Reservation for Short-Latency Wireless Sensor Network," in *IEEE 11th International Conference on Communication Technology (ICCT 2008)*, Hangzhou, China, 2008, pp. 114–117.

Granted Patent

N. Ansari and **X. Sun**, "Management of Bandwidth Efficiency and Fairness in Cloud Computing," US Patent No. 9,742,675, issued on Aug. 22, 2017.

Patent Applications

X. Li, C. Wang, Q. Ly, and **X. Sun**, "Semantic Query Processing with Information Asymmetry," US/PCT Non-Provisional Patent Application no. PCT/US18/16103, filed on Sept. 29, 2017. (Some contents of this patent application have been adopted into oneM2M Release 2 Specifications TR007 2.11.1 and oneM2M Draft Release 3 Technical Specifications TS-0034.)

X. Sun, C. Wang, Q. Ly, and X. Li, "Dynamic Protocol Switching," US/PCT Non-Provisional Patent Application no. PCT/US18/13299, filed on Jan. 11, 2018.

Q. Ly, C. Wang, X. Li, **X. Sun**, and M. Guo, "Mechanisms for Resource-Directory to Resource-Directory Communications," US/PCT Non-Provisional Patent Application no. PCT/US17/65238, filed on Dec. 08, 2017.

X. Sun, C. Wang, X. Li, Q. Ly, D. Seed, and H. Li, "Enabling Semantic Mashup in Internet of Things," US/PCT Non-Provisional Patent Application no. PCT/US17/54274, filed on Sep. 29, 2017. (Some contents of this patent application have been adopted by oneM2M Release 2 Specifications TR007 2.11.1 and oneM2M Draft Release 3 Technical Specifications TS-0034.)

N. Ansari and **X. Sun**, "Virtual Machine Resource Utilization in a Data Center," US Non-Provisional Patent Application no. 15/280,713, filed on Sep. 29, 2016.

N. Ansari and **X. Sun**, "Virtual Machine Placement in a Heterogeneous Data Center," US Non-Provisional Patent Application no. 15/280,761, filed on Sep. 29, 2016.

To my beloved daughters, parents, and parents in law. Their encouragement, sacrifices, and love have meant more to me than they can imagine. To my dear wife, Xinjuan Xie, for her unwavering support. This thesis is dedicated to them.

Xiang Sun

ACKNOWLEDGMENT

My deepest gratitude is to my advisor, Dr. Nirwan Ansari. I have been amazingly fortunate to have him give me the freedom and encouragement to explore research ideas while providing excellent guidance. His persistent support and patience helped me overcome many difficult situations throughout my research. Without his continuous help, this dissertation would not have been possible.

To my committee members, Dr. Abdallah Khreishah, Dr. Bipin Rajendran, Dr. Roberto Rojas-Cessa, and Dr. Chonggang Wang, I thank them for their time and advisement.

I want to thank my friends: Tao Han, Yan Zhang, Mina Taheri, Thomas Lo, Xueqing Huang, Abbas Kiani, Xilong Liu, Qiang Fan, Ali Shahini, Liang Zhang, Di Wu, Jingjing Yao, Shuai Zhang, and many others, who have given me support and encouragement over the last six years. I would like to extend my gratitude to other faculty and staff members of the Department of Electrical and Computer Engineering for their support throughout my doctoral studies.

This dissertation is based upon work supported by the National Science Foundation under Grant No. CNS-1320468.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

The number of smartphone users is increasing tremendously over the years. According to Ericsson Mobility Report [1], there were more than 7 billion mobile subscriptions worldwide in 2016, and it is anticipated that the number of mobile subscriptions will be 9 billions by the end of 2022. Among these smartphone users, they more likely use their smartphones than desktops. According to Mobile App Report from comSore [2], people using smartphones on digital media accounts for over 50% of their total digital media time spent. The reasons of people preferring to use smartphones than desktops/laptops are 1) availability: smartphones are so tiny that you can take them everywhere; 2) connectivity: a huge number of Base Stations (BSs) have already been deployed everywhere, and so you can always use your smartphones to access the Internet by connecting with the nearby BSs; 3) versatility: a smartphone is considered as a swiss army knife, i.e., it can provide different kinds of functionalities. For instance, a smartphone can be a phone to make phone calls, a camera to take photos, a portable music player to play musics, a GPS device to guide you to the destination, etc.

Smartphones can provide various functions to ease people's lives. What is the most important smartphone feature to attract buyers? Table 1.1 lays out the motivating factors behind a smartphone purchase, and we can see that the battery life is always the most critical factor for buying a smartphone because a smartphone is not smart or useful once its battery is drained. In order to prolong the battery life of a smartphone, two solutions have been used recently. The first solution is to increase the battery capacity of a smartphone; however, the growth of a smartphone's battery capacity cannot catch up with the growth of the smartphone's energy demand

Table 1.1 Motivating Factors Behind a Smartphone Purchase.

	Android	iOS	Windows
Battery Life	56%	49%	53%
Ease of Use	33%	39%	38%
Operating System	37%	32%	40%
Touch Screen	34%	34%	37%
Screen Size	37%	22%	34%
Type of Network	27%	30%	20%
Brand	25%	32%	25%
Weight/Size	25%	21%	24%
Camera resolution	25%	19%	23%
Web Browsing Speed	23%	22%	22%

Source: [3].

because many mobile applications, such as augmented reality, image processing, and speech recognition, become resource intensive and drain mobile devices' batteries very quickly. Therefore, increasing the battery capacity of a smartphone is not enough to prolong the battery life of a smartphone. The second solution is to apply the Mobile Cloud Computing (MCC) technology. The concept of MCC [4, 5] is to offload computational intensive tasks from mobile devices to the remote cloud via the Internet, and so mobile devices only conduct some simple tasks, such as sensing the environment and displaying the contents; on the other hand, the remote cloud (which has been demonstrated to provision resources flexibly and efficiently [6]) would help the mobile devices execute offloaded tasks and return related results to the mobile devices. MCC can not only reduce energy consumption of mobile devices but also accelerate the execution time of the applications.

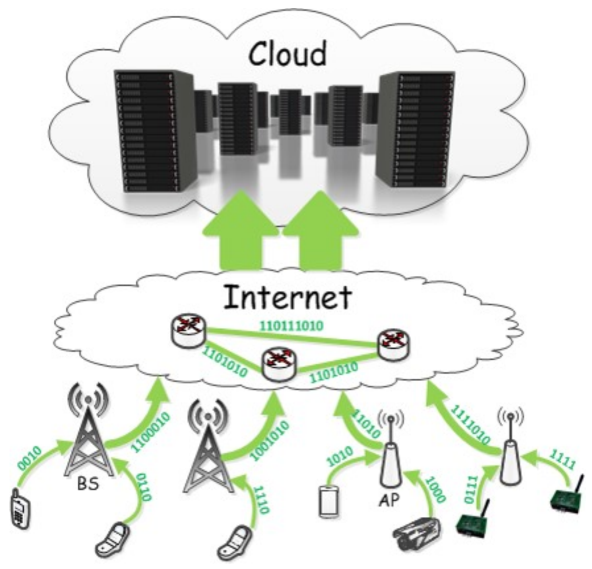

Figure 1.1 MCC platform.

Currently, many MCC platforms have already been developed to enable mobile devices to offload their tasks to the cloud. For instance, the MAUI project [7] is designed to provide method level code offloading based on the .NET framework. MAUI provides a method to enable each mobile device in determining whether to offload the source codes of an application (based on some context information, i.e., the computing resource demands, execution time, network condition, and state transfer requirements) such that the energy consumption of the mobile device is minimized. CloneCloud [8] and ThinkAir [9] are designed for Java applications written for Android based mobile devices in offloading their tasks to the cloud. ThinkAir focuses on how to efficiently and flexibly request computing resources in cloud, while CloneCloud takes the advantage of high compatibility, i.e., source codes of mobile applications can be executed in a Virtual Machine (VM) without any modification. Instead of offloading tasks from mobile devices to the cloud, Li and Wang [10] proposed that a mobile device (i.e., an initiator) can offload its tasks to nearby devices (i.e., devices within one wireless hop coverage to the initiator). However, owing to the mobility of devices and randomness of inter-contact time between the initiator and other devices, the initiator can only offload delay tolerant tasks to nearby devices.

MCC reduces the mobile devices' computational cost at the expense of the communications cost, i.e., mobile devices need to frequently interact with the remote cloud by offloading its computational intensive tasks via the Internet. Yet, the Internet is a very complicated network, and there is no control mechanism to manage the packet routing function in the Internet in order to provision guaranteed Quality of Service (QoS) [11–13]. Thus, offloading the tasks from the mobile devices to the remote cloud via the Internet may incur a long End-to-End (E2E). On the other hand, many interactive mobile applications require low network delay. For example, virtual reality applications requires 20 ms (or less) of motion-to-photon latency (i.e., the delay of a user movement to be fully reflected on a display screen) [14]; augmented reality applications also requires 20 ms (or less) of latency (between the perception of an action and image display) [15]; first person shooter games consider 80 ms end-to-end delay as an acceptable level [16].

In order to reduce the E2E delay between mobile devices and the computing resources, the cloudlet architecture is introduced by bringing computing and storage resources from the remote cloud close to mobile devices so that mobile devices can access the computing and storage resources with the low E2E delay [5, 17]. Note that a cloudlet is considered a micro data center, which comprises a number of interconnected physical machines and is deployed in a local area. Thus, mobile devices can actually offload their workloads to the nearby cloudlets without traversing the Internet.

Currently, many interactive mobile applications have been developed based on the cloudlet architecture to demonstrate that offloading compute-intensive tasks to nearby cloudlets can significantly reduce the response time of executing these tasks [18–22]. For example, the 2D Lego Assembly application [23], a type of cognitive assistance application (which is not only latency sensitive but also memory intensive), has been developed to help users assemble 2D Lego models. Essentially, a user wears a smart glass, equipped with a portable camera to capture the video streams of the current state of the Lego task. The captured video streams are uploaded to a cloudlet via the WiFi access point. The cloudlet would analyze the video streams by comparing the current state of the Lego task with the expected task state. Based on the comparison, user guidance for making incremental progress on the task is generated. This guidance has video and plain text components that are streamed back to the smart glass. The video guidance is shown on the display of the smart glass, and accompanying audio guidance is provided via the Android text-to-speech API. Nokia has also established a Multi-access Computing platform to attach a cloudlet to a BS. A use case named connected cars has been developed based on this platform, where each local cloudlet analyzes the data at the point of capture and feeds back the insights to the vehicles within the cloudlets coverage with extremely low latency (less than 20 ms) in order to improve road safety [24].

The cloudlet concept facilitates mobile users (MUs) in offloading their computational intensive tasks. However, many issues are still to be addressed in order to enable mobile devices efficiently utilize the resources in cloudlets.

1. Where to deploy cloudlets in the network?
   Microsoft proposed to build an extensive infrastructure of cloudlets, each of which has 1–10 interconnected physical servers with several TBs of storage, and place them everywhere [25]. But it is still not clear where to deploy these cloudlets such that mobile devices could always offload their workloads to the cloudlets with low E2E delay.

2. How to allocate/manage computing and storage resources in a cloudlet to different MUs?
   Many MUs may offload their workloads to the same cloudlet, which may not have enough resources to handle all the incoming workloads. Thus, it is necessary to design an efficient resource allocation strategy among different

MUs. Note that designing the efficient resource allocation strategy for MUs is challenging. This is because MUs are roaming over the network, and thus the previous resource allocation strategy may not be optimal once MUs roam away. On the other hand, a cloudlet is considered as a public resources available for all the MUs, and thus how to preserve privacy for MUs (i.e., the offloaded tasks from an MU will not be altered, eavesdropped, or replayed by other MUs) is a big challenge.

3. How does the cloudlet concept facilitate other applications?
   A cloudlet is originally used to execute the offloaded tasks from mobile devices. However, if cloudlets can also facilitate other applications, such as mobile crowd sensing, mobile data stream analysis, and Internet of Things (IoT), it is beneficial for the cloudlet providers to intrigue more application providers and MUs in utilizing the cloudlet infrastructure.

The rest of this thesis is organized as follows. In Chapter 2, we propose the cloudlet network architecture to provision mobile edge computing. Each MU is associated with a dedicated Avatar (i.e., a private VM executes tasks offloaded from its MU). In Chapter 3, we propose to hand off Avatars among cloudlets in order to maintain the low E2E delay between MUs and their Avatars when MUs roam away. We formulate the Avatar handoff problem to minimize the average E2E delay between MUs and their Avatars in each time slot and propose the LatEncy aware Avatar hanDoff (LEAD) algorithm to efficiently solve the problem. The performance of LEAD is demonstrated via simulations. In Chapter 4, we propose to empower the IoT system with the proposed cloudlet network. Popular IoT resources are cached in nearby brokers, which are considered as application layer middleware nodes hosted by cloudlets in the cloudlet network, to reduce the energy consumption of servers (which host these popular IoT resources). Also, we propose a novel Energy Aware and latency guaranteed dynamic reSourcE caching (EASE) strategy to enable each broker to cache suitable popular resources such that the energy savings from the servers are maximized, while guaranteeing the average delay for publishing the contents of the resources to the corresponding clients. The performance of EASE is demonstrated via extensive simulations. In Chapter 5, we briefly present two future research endeavors, i.e., traffic load balancing among brokers in the cloudlet network and drone aided mobile access networks. The conclusion is presented in Chapter 6.

CHAPTER 2

CLOUDLET NETWORK ARCHITECTURE

To reap benefits of the cloudlet, we propose the cloudlet network architecture, as shown in Figure 2.1, in order to provide ubiquitous computing and storage resources to MUs and at the same time maintain the low E2E delay between the MUs and computing and storage resources. Essentially, the cloudlet network architecture comprises three parts, i.e., distributed cloudlets in the mobile network, hierarchical structure of a cloudlet, and the Software Defined Networking (SDN) based mobile core network.

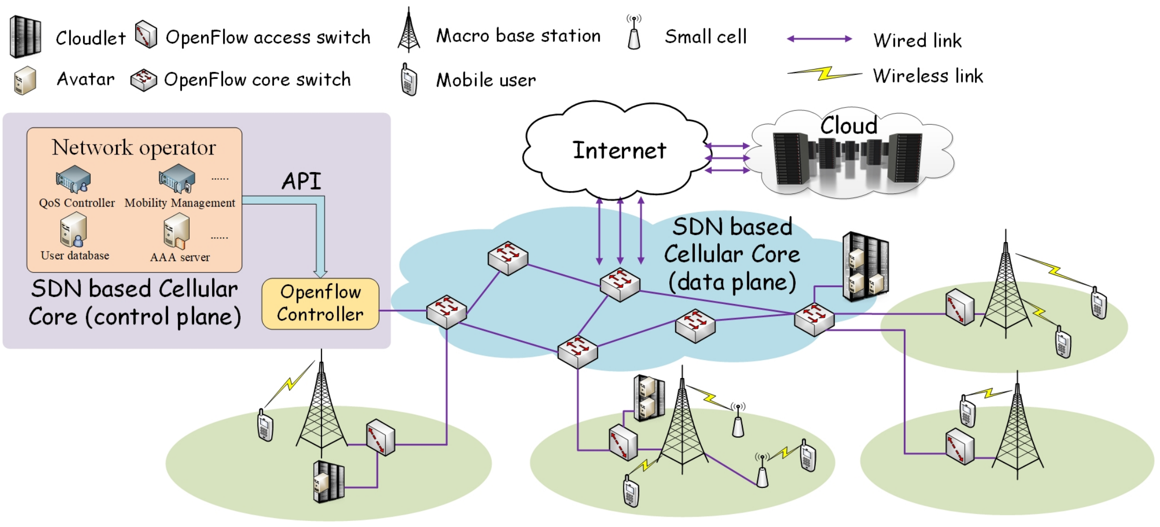

Figure 2.1 Cloudlet network architecture.

2.1 Distributed Cloudlets

The existing mobile network infrastructure can provide seamless connection between an MU and a Base Station (BS); thus, each BS is connected to a cloudlet such that MUs can always utilize computing and storage resources in the cloudlets with one wireless hop delay [26, 27]. The placement of a cloudlet is flexible, i.e., a cloudlet can be directly connected to a BS via the access switch such that the local MUs can access the computing and storage resources in the cloudlet with low latency; or a cloudlet can be attached to a switch at the edge of the mobile core network such that MUs in different BSs' coverage areas can share the computing and storage resources provided in the same cloudlet.

Each MU subscribes one Avatar, a high performance Virtual Machine (*VM*) in the cloudlet, which provides extra computing and storage resources. Avatars are software clones of their MUs [28, 29] (i.e., each Avatar and its MUs have the same operating system, installed apps, and contents) and always available to MUs when MUs are moving from one BS's coverage area to another. Assigning a specific Avatar to each MU in the cloudlet provides hardware isolation by securely running each MU's application workloads on a shared physical hardware [30]. Avatars (which contain much more computing and storage resources than their MUs) execute the offloaded tasks and return back the related results to their MUs, and so the task execution time and the energy consumption of MUs can be reduced significantly.

Moreover, every MU and its Avatar in the cloudlet can communicate with public cloud (e.g., Amazon EC2) via the Internet in order to provision scalability, i.e., if cloudlets are not available for MUs because of the capacity limitation, MUs' Avatars can be migrated to the remote data centers to continue serving their MUs.

2.2 Hierarchical Cloudlet Framework

In each cloudlet, as shown in Figure 2.2, there are two types of VMs. i.e., Avatars and Application VMs. Avatars in each cloudlet synchronize data with their associated MUs and execute tasks offloaded from their MUs. Application VMs in each cloudlet are maintained by different application providers. Applications VMs are used to retrieve contents from Avatars, analyze the retrieved contents to generate high-level knowledge, and provide the corresponding services to MUs.

The placement of application VMs are flexible. For instance, if many MUs in a BS's coverage area require a specific service, the related application VM can be placed in the cloudlet (which is connected to the BS) to serve these MUs in order to reduce the network delay. We will provide three examples in Section 2.4 to further illustrate the relationship among MUs, their Avatars, and application VMs.

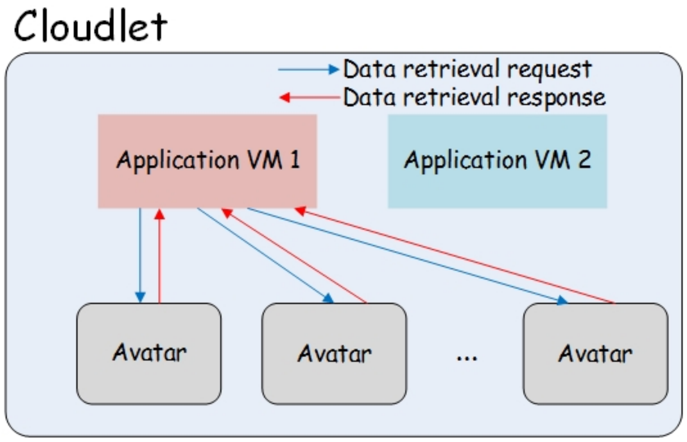

Figure 2.2 Hierarchical structure within a cloudlet.

2.3 Software Defined Network (SDN) based Cellular Core

The traditional cellular core network in terms of Evolved Packet Core (EPC) [31] can provide guaranteed services (i.e., ensuring the E2E delay between two end points in mobile networks to be less than a threshold), but it centralizes the data-plane and control-plane functionalities in the Packet data network GateWay (P-GW) and Serving GateWay (S-GW) [32, 33]. In other words, all the traffic flows including Device-to-Avatar (D2A) and Avatar-to-Application VM (A2A) should go through S-GW and P-GW, which is not efficient and increases the E2E delay. For instance, as shown in Figure 2.3, MU-1 is in BS-1's coverage area and its Avatar is placed in Cloudlet-2, which is attached to BS-2, and so the communications between MU-1 and its Avatar should go through the P-GS and S-GW. Similarly, if the Application

VM, which is placed in CLoudlet-1, needs to communicate with MU-1's Avatar, the communications path should also go over the P-GS and S-GW.

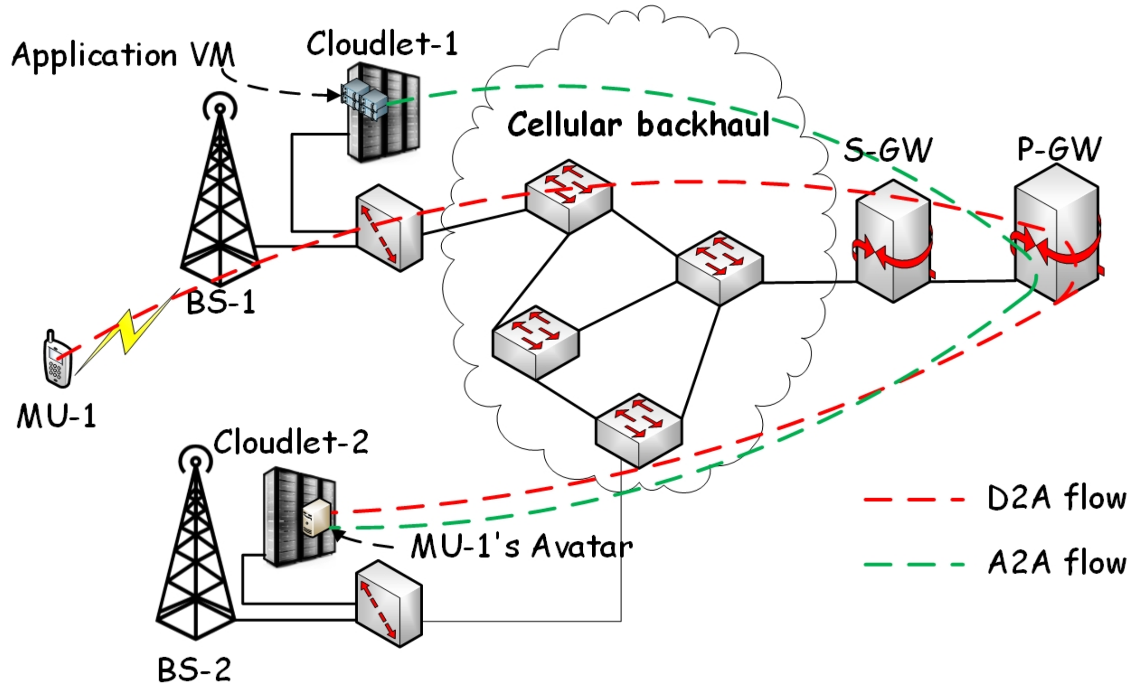

Figure 2.3 EPC architecture.

Applying the Software Defined Networking (SDN) technology to the cellular core network is one solution to provide a flexible and efficient networking solution [34–37]. The SDN based mobile core network is essentially decoupling the control plane from the switches, which only run data plane functionalities. The control plane is offloaded to a logical central controller, which transmits the control information (e.g., flow tables) to the OpenFlow switches by applying the OpenFlow protocol [38], monitors the traffic statistics of the network, and provides Application Programming Interfaces (APIs) to network management operators so that different mobile network functionalities, such as mobility management, user authentication, authorization and accounting, network virtualization, and QoS control, can be added, removed, and modified flexibly.

2.4 Cloudet Network based Mobile Applications

The proposed cloudlet network architecture can not only facilitate MUs to offloaded their tasks but also benefit other applications, such as mobile crowd sensing and big mobile data stream analysis, by analyzing mobile data streams at the mobile edges. This can substantially reduce the traffic load of the mobile core network and the application response time. Here, we provide three applications to illustrate how the cloudlet network architecture facilitate these applications.

Terrorist detection [36] helps MUs to identify terrorists to secure their societies. First, if MUs are interested in the service provided by the terrorist detection application, they can install the corresponding app in their Avatars. MUs upload their captured photos/videos to their Avatars over time. If the terrorist detecting application VM tries to locate a specific terrorist by conducting face matching over the captured photos/videos, instead of having each Avatar transmit its captured photos/videos to the application VM, the terrorist detecting application VM would send a data retrieval request containing the terrorist's photo to the Avatar (which have installed the corresponding app) among all the cloudlets. After receiving the request, the installed app in the Avatar would retrieve the videos/photos in the local storage, and conduct the face matching algorithm by comparing these videos/photos with the received terrorist's photo. If a match is detected, the Avatar would respond to the application VM with the related metadata, i.e., the location information and time stamps of the corresponding photos/videos.

ParkNet [39] helps MUs locate available parking spots in the urban area. Specifically, each Avatar collects the sensed data streams from the smart car via the MU. Note that each smart car is equipped with a GPS receiver and a passenger-side-facing ultrasonic rangefinder to generate the location and parking spot occupancy data streams. Each Avatar analyzes the data sensed, generates the high level knowledges (which identify the available parking spots), and forwards the knowledges to the ParkNet application VM. The ParkNet application VM will inform and assign the available parking spots to the MUs upon requests.

FaceDate [40] is to find and date nearby people based on their face preference in real-time. Specifically, each MU uploads a profile photo and provides basic information (such as date of birth, gender, and a brief write-up) about himself/herself into its Avatar. In addition, each MU uploads a set of preference photos (i.e., the photos of a boy/girl whom she/he wants to date. For instance, if a man wants to date a woman who resembles Marilyn Monroe, he would upload the photos of Marilyn Monroe into its Avatar) to identify his dream date partner. If a man tries to find a nearby date partner, his Avatar (i.e., request Avatar) would send a request, which contains its preference photos, to the FaceDate application VM. The FaceDate application VM forwards the request to other Avatars, which conduct the face recognition algorithm by comparing the preference photos in the request with their MUs' profile photos. If the photos are highly matched, the Avatars (i.e., response Avatars) would respond to the FaceDate application VM with the metadata (similarity of the photos) as well as the preference photos in these response Avatars. The FaceDate application VM would forward these preference photos (from the response Avatars) to the request Avatar, which conducts the face recognition algorithm by comparing the received preference photos with its profile photo and responds to the FaceDate application VM with the metadata (similarity of the photos). Finally, the FaceDate application VM would pick the best matched candidate and enable the chatting accordingly.

CHAPTER 3

AVATAR HANDOFF IN THE CLOUDLET NETWORK

MUs are roaming among BSs over time and so the E2E delay between MUs and their Avatars may become worse if the Avatars remain in their original cloudlets. For instance, as shown in Figure 3.1, if an MU roams from BS 1's coverage area into BS 2's coverage area and its Avatar still resides in Cloudlet 1, the communications path between the MU and its Avatar should traverse the SDN based cellular core, which may incur a long E2E delay as well as increase the traffic load of the SDN based cellular core. As we mentioned before, the E2E delay is critical for many interactive mobile application in conducting task offloading.

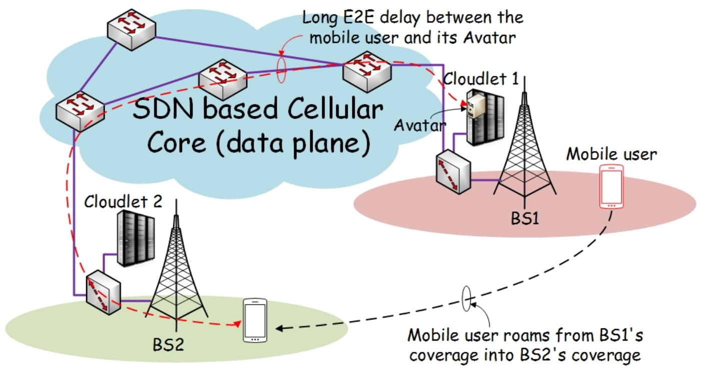
Long E2E delay between the mobile user and its Avatar
SDN based Cellular Core (data plane)
Cloudlet 1
Avatar
Mobile user
BS1
Cloudlet 2
BS2
Mobile user roams from BS1's coverage into BS2's coverage

Figure 3.1 Long E2E delay between an MU and its Avatar when the MU roams away.

In order to preserve the low E2E delay between MUs and their Avatar when their MUs roam away, Avatars can actually be handed off among cloudlets based on their MUs' locations. For example, as shown in Figure 3.1, if the MU roams into BS 2's coverage area, its Avatar can be handed off into Cloudlet 2 in order to reduce the E2E delay between the MU and its Avatar. Note that the Avatar handoff process is considered as the live VM migration process [41–44], where the vCPU, virtual memory, and virtual disk of an Avatar are transmitted from the source cloudlet into destination cloudlet over the SDN based cellular core. Note that the Avatar in the source cloudlet is still serving its MU during the Avatar handoff process, and so the MU would not be aware of the Avatar handoff process.

The Avatar handoff process is a very expensive operation, i.e., the Avatar handoff process incurs a huge volume of data transmission (i.e., the contents of the vCPU, virtual memory, and virtual disk of the Avatar as well as the dirty blocks of the virtual memory and the virtual disk generated during the Avatar handoff process should be transmitted from the source cloudlet to the destination cloudlet) over the SDN based cellular core, and thus significantly increases the traffic load of the SDN based cellular core as well as the Avatar handoff time[1]. Note that increasing the Avatar handoff time may increase the average E2E delay between the MU and its Avatar because the long average E2E delay between the MU and its Avatar persists until the Avatar handoff process is finished (i.e., the Avatar is located in the destination cloudlet and start to serve its MU), and so a shorter Avatar handoff time will produce a lower average E2E delay [37]. For example, assume that the average E2E delay between an MU and its Avatar is 100 ms if the Avatar is placed in the source cloudlet, and the average E2E delay between the MU and its Avatar is 50 ms if the Avatar is placed in the destination cloudlet. As shown in Figure 3.2, if the Avatar handoff process is finished in t_1, and so the average E2E delay between the MU and its Avatar during time period T is $\frac{100t_1+50(T-t_1)}{T}$; on the other hand, if the Avatar handoff process is finished in t_2, where $t_2 < t_1$, the average E2E delay between the MU and its Avatar during time period T is $\frac{100t_2+50(T-t_2)}{T}$. Obviously, $\frac{100t_2+50(T-t_2)}{T} < \frac{100t_1+50(T-t_1)}{T}$, which indicates that shorter Avatar handoff time incurs lower average E2E delay between the MU and its Avatar.

[1]The Avatar handoff time refers to the overall time of transmitting the whole Avatar (i.e., the vCPU, virtual memory, and virtual disk of the Avatar as well as the dirty blocks of the virtual memory and the virtual disk generated during the Avatar handoff process) from the source cloudlet to the destination cloudlet

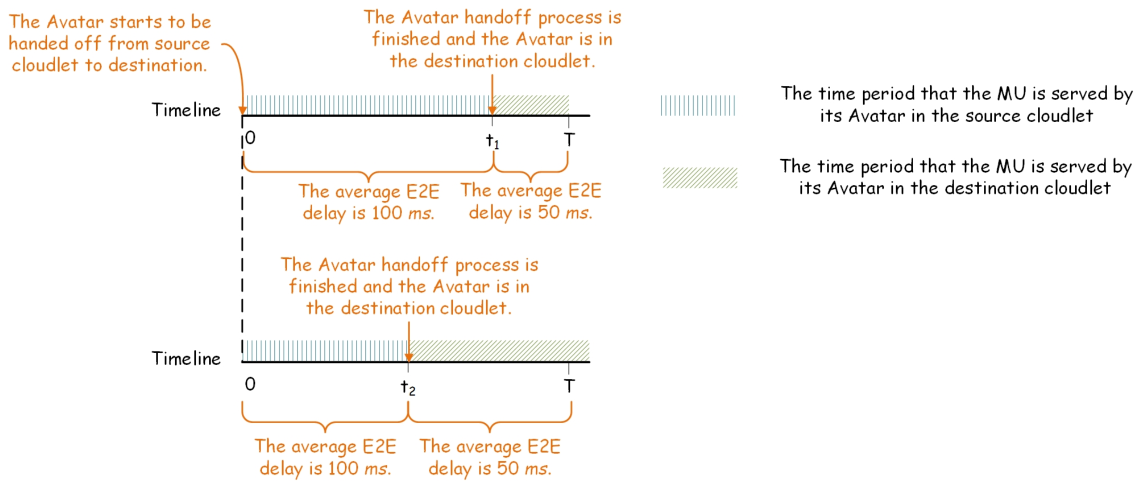
The Avatar starts to be handed off from source cloudlet to destination.
The Avatar handoff process is finished and the Avatar is in the destination cloudlet.
Timeline
0
t_1
T
The average E2E delay is 100 *ms*.
The average E2E delay is 50 *ms*.
The time period that the MU is served by its Avatar in the source cloudlet
The time period that the MU is served by its Avatar in the destination cloudlet
The Avatar handoff process is finished and the Avatar is in the destination cloudlet.
Timeline
0
t_2
T
The average E2E delay is 100 *ms*.
The average E2E delay is 50 *ms*.

Figure 3.2 Illustration of the relationship between Avatar handoff time and the average E2E delay between an MU and its Avatar.

Also, the Avatar handoff process consumes extra resources of the Avatar (especially the bandwidth resource) [45–47], and thus degrades the performance of tasks (which are offloaded from its MU) currently running in the Avatar. Therefore, short handoff time can potentially reduce the task executing time, i.e., more resources of an Avatar can be allocate to executing the offloaded tasks.

However, it is reported that migrating the whole Avatar between two cloudlets over a network with stable 10 $Mbps$ bandwidth and 50 ms Round Trip Time (RTT) consumes over two hours [48]; this indicates that handing off a whole Avatar between cloudlets cannot maintain the low E2E delay between the Avatar and its MU but exhausts the resource of the network and the Avatar. The main reason for incurring the unacceptable handoff time is to migrate the large volume of the Avatar's virtual disk over the network [48].

In order to significantly reduce the Avatar handoff time by avoiding virtual disk migration during the handoff process, we propose to place a number of replicas of an Avatar's virtual disk in the suitable cloudlets [49]. The replicas of an Avatar[2] are synchronized with the Avatar's virtual disk during a fixed time period (e.g., 5 min).

[2]The replicas of an Avatar are referred to as the replicas of the Avatar's virtual disk.

Thus, if an MU's Avatar tries to hand off to the cloudlet which contains one of the Avatar's replicas, only the memory and the *pre-handoff virtual disk dirty blocks* of the Avatar are needed to be transmitted to the destination cloudlet. The *pre-handoff virtual disk dirty blocks* of the Avatar means the virtual disk dirty blocks that are generated after the last replica synchronization process. For instance, as shown in Figure 3.3, the Avatar's replicas are synchronized at t_1; meanwhile, the Avatar handoff is triggered at t_2, and thus the number of the virtual disk blocks, which are modified during the interval between t_1 and t_2, are defined as the *pre-handoff virtual disk dirty blocks*, which need to be migrated during the handoff process. Since the dirty block generation rate of the virtual disk is relatively low, only very small portion of the virtual disk is needed to be transmitted to the destination cloudlet, which will significantly reduce the handoff time.

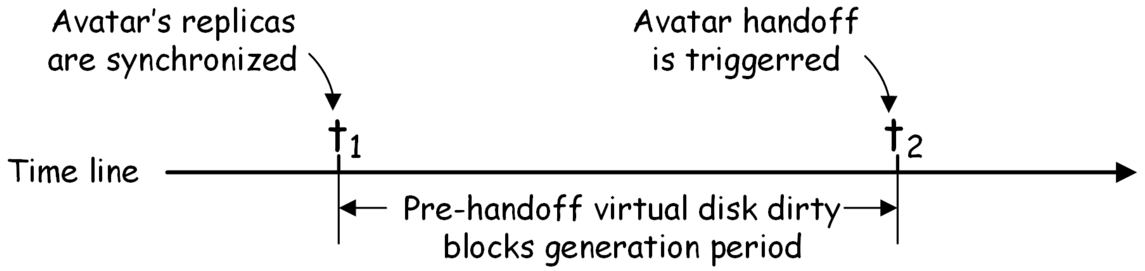
Avatar's replicas are synchronized
Avatar handoff is triggerred
t1
t2
Time line
Pre-handoff virtual disk dirty blocks generation period

Figure 3.3 Illustration of pre-handoff virtual disk dirty blocks.

Essentially, the Avatar handoff problem can be decomposed into two subproblems, i.e., the Avatar replica placement problem and the adaptive Avatar handoff problem. The Avatar replica placement is to determine the location of each Avatar's replica based on its MU's historical movement trace. The Avatar replica placement problem is solved offline; for example, the placement of each Avatar's replica can be updated during the midnight. After the replicas of Avatars have been deployed, the adaptive Avatar handoff problem is used to determine the locations of all MUs' Avatars based on the MUs' real-time movements as well as the network status (i.e., the average

E2E delay among cloudlets and BSs). The adaptive Avatar handoff problem is solved online, i.e., the locations of all MUs' Avatars can be updated in each time slot (e.g., 30 minutes).

It is worth noting that the Avatar handoff problem is not to determine the suitability of offloading tasks from an MU to its Avatar. Many methods have been investigated to solve the task offloading decision problem [50–52], i.e., MUs would determine whether a tasks is suitable to be offloaded based on the wireless channel condition, complexity of the task, and the configuration of the mobile device such that the energy consumption of the mobile device is minimized and the task execution time is reduced. The Avatar handoff problem is used to determine the location of each Avatar in order to further improve the performance of task offloading by reducing the network delay incurred in the wired networks.

3.1 Avatar Replica Placement

Migrating the whole virtual disk of an Avatar during the Avatar handoff process incurs unbearable handoff time and increases the traffic load of the SDN-based cellular core significantly. Thus, in order to avoid the virtual disk migration during the Avatar handoff process, we propose to deploy a number of the Avatar's replicas among the cloudlets. As we mentioned before, an Avatar's replica refers to a software copy of the Avatar's virtual disk. Note that the initial purpose of creating replicas of a virtual disk is to avoid data losses owing to hard disk failures [53, 54]. For example, in the Hadoop distributed file system, each block has three replicas by default [55]. Here, we try to place the replicas of an Avatar in suitable cloudlets in order to avoid virtual disk migration during the Avatar handoff process. For instance, as shown in Figure 3.4, if an Avatar's replica has already been deployed in Cloudlet 2, handoff the Avatar from Cloudlet 1 into Cloudlet 2 only needs to migrate virtual memory, virtual disk dirty blocks, and vCPU states of the Avatar. This can significantly reduce the volume of data need to be migrated, thus tremendously reducing the Avatar handoff time. Here, we define the cloudlet, which contains one of an Avatar's replicas, as the Avatar's *available cloudlet*. Thus, we say that an Avatar can only be handed off to its *available cloudlets* in order to avoid virtual disk migration during the Avatar handoff process.

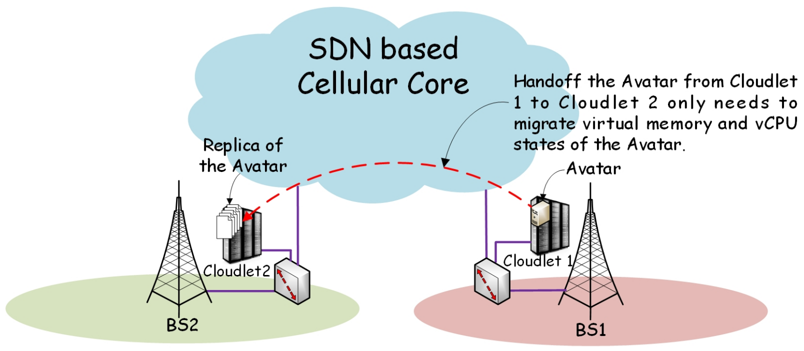
SDN based Cellular Core
Handoff the Avatar from Cloudlet 1 to Cloudlet 2 only needs to migrate virtual memory and vCPU states of the Avatar.
Replica of the Avatar
Avatar
Cloudlet2
Cloudlet 1
BS2
BS1

Figure 3.4 Handoff an Avatar into a cloudlet

Note that it is unnecessary and inefficient to place the Avatar's replicas in all the cloudlets in the network because increasing the number of replicas for each Avatar increases the capital expenditure (CAPEX) of the cloudlet provider (by implementing more storage space in the cloudlets) as well as the synchronization traffic (i.e., the traffic incurred by synchronizing the contents among the replicas of each Avatar) in the SDN based cellular core. Meanwhile, placing the Avatar's replicas in the cloudlets, which are never visited by the MU, cannot benefit the communications between the MU and its Avatar. Therefore, it is important to optimally place a limited number of replicas for each Avatar among the cloudlets so that the average E2E delay during a period ΔT (e.g., one day) between the MU and its Avatar can be minimized (by utilizing Avatar handoff) when the MU roams in the network.

Normally, the Avatar's replicas will be placed where its MU will commonly visit (it has been demonstrated that about 10% to 30% of all human movement can be explained by their social relationship, while 50% to 70% is attributed to periodic behaviors [56]; thus, we believe that the dynamics of future human movement can

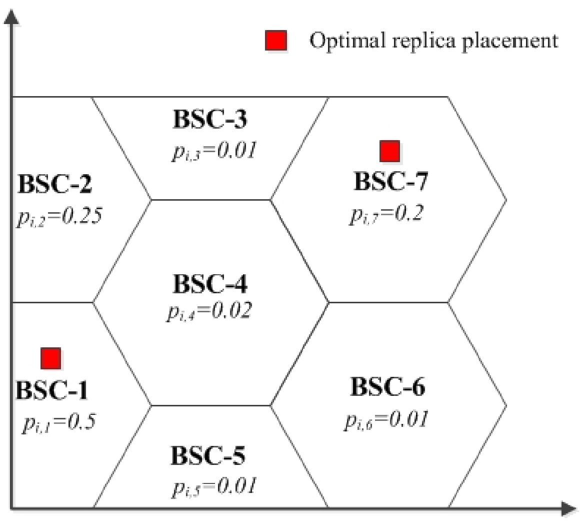

Figure 3.5 Illustration of the optimal Avatar replica placement.

be reliably predicted based on the mathematical models [56–58]), such as home and workplace. However, this is not the optimal Avatar replica placement strategy. For instance, suppose the cloudlet network topology is shown in Figure 3.5, which contains 7 BS-Cloudlet (BSC) combinations[3], and two replicas of MU i's Avatar need to be placed. Meanwhile, suppose the occurrence probability of MU i in BS j's coverage area, denoted as p_{ij}[4] (where $j = 1, 2, \cdots, 7$), is also shown in Figure 3.5. Thus, traditionally, two replicas will be placed in BSC-1 and BSC-2 (because $p_{i,1}$ and $p_{i,2}$ are the two largest values, implying that MU i will most commonly visit BSC-1 and BSC-2). Yet, deploying the two replicas in BSC-1 and BSC-7 may be the optimal solution for MU i. This is because, first, the value of $p_{i,2}$ and $p_{i,7}$ are close; second, BSC-1 and BSC-2 are adjacent to each other, and so the E2E delay between MU i and its Avatar is low even if MU i is in the BSC-2's coverage area and its Avatar is in BSC-1. On the contrary, since BSC-7 is far away from BSC-1, the E2E delay may be unbearable if MU i is in the BSC-7's coverage area and its Avatar is in

[3]A BSC combination indicates that a BS is attached to a dedicated cloudlet.

[4]$p_{ij} = \frac{\textit{the total time that MU i stays in the BS j's coverage area}}{\textit{the total time period (i.e., one day)}}$

BSC-1, and thus placing the 2nd replica in BSC-7 may improve the average E2E delay significantly.

Therefore, we conclude that the value of p_{ij} is not the only determinant to affect the performance of the Avatar replica placement. The E2E delay among different BSCs can also affect the performance of Avatar replica placement. Table 3.1 summarizes the main notations applied in the Avatar replica replacement problem.

Table 3.1 List of Important Notations in Avatar Replica Replacement

Notation	**Definition**
$\boldsymbol{\mathcal{I}}$	Set of all the MUs.
$\boldsymbol{\mathcal{J}}$	Set of all the BSs.
$\boldsymbol{\mathcal{K}}$	Set of all the cloudlets.
x_ik	A binary variable indicating the location of replica for MU i's Avatar.
t_{jk}	Average E2E delay between BS j and cloudlet k.
y_{ijk}	A variable indicating the location MU i's Avatar.
ΔT	A fix time period (e.g., one day).
τ_i	Average E2E delay between MU i and its Avatar during ΔT.
p_{ij}	Occurrence probability of MU i in BS j's coverage area during ΔT.
κ	Total number of replicas for each MU's Avatar.

3.1.1 System model

Let $\boldsymbol{\mathcal{I}}$, $\boldsymbol{\mathcal{J}}$ and $\boldsymbol{\mathcal{K}}$ be the set of MUs, BSs and cloudlets, respectively. Denote x_{ik} as a binary variable indicating one replica of MU i's Avatar ($i \in \boldsymbol{\mathcal{I}}$) is located in cloudlet k (i.e., $x_{ik} = 1$, where $k \in \boldsymbol{\mathcal{K}}$) or not (i.e., $x_{ik} = 0$). Meanwhile, let t_{jk} be the average E2E delay between BS j and cloudlet k. The value of t_{jk} ($j \neq k$) can be measured and recorded by the SDN controller [59,60]. Note that if $j == k$, we say that cloudlet k is BS j's attached cloudlet. Moreover, denote y_{ijk} as a binary variable indicating

MU i's Avatar is located in cloudlet k (i.e., $y_{ijk} = 1$) or not (i.e., $y_{ijk} = 0$) when MU i is in BS j's coverage area. Let τ_{ij} be the average E2E delay between MU i and its Avatar when MU i is in the BS j's coverage area, then we have:

$$\tau_{ij} = \sum_{k\in\boldsymbol{\mathcal{K}}} t_{jk} y_{ijk}. \tag{3.1}$$

Denote τ_i as the average E2E delay between MU i and its Avatar during the period ΔT (e.g., one day); meanwhile, let p_{ij} be the predicted occurrence probability of MU i in BS j's coverage area during the period ΔT; then, we have:

$$\tau_i = \sum_{j\in\boldsymbol{\mathcal{J}}} p_{ij}\tau_{ij} = \sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}} p_{ij} t_{jk} y_{ijk}. \tag{3.2}$$

The optimal Avatar replica placement for each MU i ($i \in \boldsymbol{\mathcal{I}}$) is to minimize its average E2E τ_i during the period ΔT. Thus, we formulate the problem as follows:

$$\boldsymbol{P0}: \qquad \arg\min_{x_{ik},\ y_{ijk}} \sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}} p_{ij} t_{jk} y_{ijk} \tag{3.3}$$

$$s.t. \sum_{k\in\boldsymbol{\mathcal{K}}} x_{ik} = \kappa, \tag{3.4}$$

$$\forall j \in \boldsymbol{\mathcal{J}},\ \sum_{k\in\boldsymbol{\mathcal{K}}} y_{ijk} = 1, \tag{3.5}$$

$$\forall j \in \boldsymbol{\mathcal{J}}\ \forall k \in \boldsymbol{\mathcal{K}},\ y_{ijk} \le x_{ik}, \tag{3.6}$$

$$\forall k \in \boldsymbol{\mathcal{K}},\ x_{ik} \in \{0,1\}, \tag{3.7}$$

$$\forall j \in \boldsymbol{\mathcal{J}}\ \forall k \in \boldsymbol{\mathcal{K}},\ y_{ijk} \in \{0,1\}, \tag{3.8}$$

where κ is the total number of replicas that can be deployed among the cloudlets for each MU's Avatar. Constraint (3.4) requires that exactly κ replicas are placed for MU i. Constraint (3.5) indicates that MU i' Avatar should be located in exactly one cloudlet when MU i is in BS j's coverage area. Constraint (3.6) implies that MU i'

Avatar can only be located in a cloudlet if and only if the cloudlet contains one of its replicas (i.e., in order to satisfy Constraint (3.6), y_{ijk} could equal to 1 iff $x_{ik} = 1$; otherwise, y_{ijk} should be 0 if $x_{ik} = 0$). Constraints (3.7) and (3.8) implies x_{ik} and y_{ijk} ($j \in \boldsymbol{\mathcal{J}}$ and $k \in \boldsymbol{\mathcal{K}}$) are binary variables.

Lemma 1. *The Avatar replica placement problem (i.e., $\boldsymbol{P0}$) is NP-hard when $\kappa > 1$.*

Proof. The formulation of the Avatar replica placement problem is equivalent to the p-median problem [61] where $\kappa = p > 1$, and the p-median problem has been proved to be NP-hard on a general network topology (note that it has been demonstrated that the p-median problem can be solved in polynomial time $O(n^2p^2)$ only if the network is a tree [62]). Therefore, we need to demonstrate the topology of the proposed cloudlet network is not a tree.

Based on the cloudlet network architecture, each BS can communicate with all the cloudlets over the SDN based cellular core. Thus, the cloudlet network can be considered as a complete graph in which every vertex represents the BS-cloudlet pair. Every pair of distinct vertices is connected by a unique edge, which represents a communications link with a dedicated cost in terms of the E2E delay. Therefore, the Avatar replica placement problem is NP-hard. □

3.1.2 LatEncy Aware Replica placemeNt (LEARN)

Inspired by the Lagrangian relaxation algorithm for solving the p-median problem [63], we design the LatEncy Aware Replica placemeNt (LEARN) algorithm to optimally place the replicas among cloudlets for each MU. The basic idea of LEARN is to iteratively obtain the lower bound (LB) and upper bound (UB) of the Avatar replica placement problem through Lagrangian procedure until the differece between the LB and UB is less than a predefined value ψ.

Specifically, we relax Constraint (3.5) in $\boldsymbol{P0}$ to obtain the following Lagrangian problem:

$$\begin{aligned}\boldsymbol{P1}: \quad \max_{\lambda_j} \min_{x_{ik} y_{ijk}} \mathcal{L} &= \sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}} p_{ij}t_{jk}y_{ijk} + \sum_{j\in\boldsymbol{\mathcal{J}}}\lambda_j\left(1-\sum_{k\in\boldsymbol{\mathcal{K}}} y_{ijk}\right)\\ &= \sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}}\left(p_{ij}t_{jk}-\lambda_j\right)y_{ijk} + \sum_{j\in\boldsymbol{\mathcal{J}}}\lambda_j, \qquad (3.9)\\ &s.t.\ \ Constraints\ (3.4),(3.6),(3.7),(3.8),\end{aligned}$$

where λ_j ($\forall j \in \boldsymbol{\mathcal{J}}, \lambda_j \geq 0$) are the Lagrangian multipliers. For fixed values of the Lagrange multipliers λ_j, the above relaxed problem (i.e., $\boldsymbol{P1}$) will yield an optimal objective value that is an LB on any feasible solution of the original Avatar replica placement problem (i.e., $\boldsymbol{P0}$).

Lemma 2. *Define vector $\boldsymbol{\Delta}_i = \{\Delta_{ik}|k \in \boldsymbol{\mathcal{K}}\}$, where $\Delta_{ik} = \sum_{j\in\boldsymbol{\mathcal{J}}} \min(0, p_{ij}t_{jk} - \lambda_j)$; define the cloudlet set $\boldsymbol{\mathcal{K}}_i^{'}$ ($\boldsymbol{\mathcal{K}}_i^{'} \subset \boldsymbol{\mathcal{K}}$), where $\left|\boldsymbol{\mathcal{K}}_i^{'}\right| = \kappa$ and $\left\{\Delta_{ik}|k \in \boldsymbol{\mathcal{K}}_i^{'}\right\}$ are the κ number of the smallest values in vector $\boldsymbol{\Delta}_i$. Then, for any given set of multipliers $\boldsymbol{\lambda} = \{\lambda_j|j \in \boldsymbol{\mathcal{J}}\}$, the optimal solution of the Lagrangian problem, denoted as $\boldsymbol{\mathcal{X}}_i^* = \{x_{ik}^*|k \in \boldsymbol{\mathcal{K}}\}$ and $\boldsymbol{\mathcal{Y}}_i^* = \left\{y_{ijk}^*|j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\right\}$, can be expressed as follows:*

$$\forall k \in \boldsymbol{\mathcal{K}}, \quad x_{ik}^* = \begin{cases} 1, & k \in \boldsymbol{\mathcal{K}}_i^{'}. \\ 0, & otherwise. \end{cases} \qquad (3.10)$$

$$\forall j \in \boldsymbol{\mathcal{J}}, \forall k \in \boldsymbol{\mathcal{K}}, y_{ijk}^* = \begin{cases} 1, & p_{ij}t_{jk} - \lambda_j < 0 \ \&\ x_{ik}^* = 1. \\ 0, & otherwise. \end{cases} \qquad (3.11)$$

Proof. Obviously, in order to minimize the objective function of the Lagrangian problem (i.e., $\boldsymbol{P1}$), y_{ijk} should be chosen its maximum value if $p_{ij}t_{jk} - \lambda_j \leq 0$ ($j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}$) for any given set of Lagrangian multipliers $\boldsymbol{\lambda} = \{\lambda_j|j \in \boldsymbol{\mathcal{J}}\}$, otherwise, $y_{ijk} = 0$. Thus, by considering Constraint (3.6), the optimal solution of y^*_{ijk} is given by: $y^*_{ijk} = x_{ik}$, if $p_{ij}t_{jk} - \lambda_j \leq 0$; $y^*_{ijk} = 0$, otherwise.

By substituting the optimal solution of y^*_{ijk} into $\mathcal{L}$ (Equation (3.9)), the Lagrangian problem is transformed into:

$$\min_{x_{ik}} \mathcal{L} = \sum_{k \in \boldsymbol{\mathcal{K}}} \Delta_{ik} x_{ik} + \sum_{j \in \boldsymbol{\mathcal{J}}} \lambda_j \tag{3.12}$$
$$s.t. \ \ Constraints \ (4.6), (4.8),$$

where $\Delta_{ik} = \sum_{j \in \boldsymbol{\mathcal{J}}} \min\left(0, p_{ij}t_{jk} - \lambda_j\right)$. For any given set of Lagrangian multipliers $\boldsymbol{\lambda}$, the above problem is trivial to solve, i.e., $x^*_{ik} = 1$, if $k \in \boldsymbol{\mathcal{K}}^{'}_i$ (where $\boldsymbol{\mathcal{K}}^{'}_i$ is the set of κ number of the cloudlets, which have the smallest values of Δ_{ik} in $\boldsymbol{\Delta}_i = \{\Delta_{ik} | k \in \boldsymbol{\mathcal{K}}\}$); $x^*_{ik} = 0$, otherwise. Thus, Equations (3.10) and (3.11) have been proved. □

Note that solving the relaxed problem can provide the LB of the original Avatar replica problem, i.e.,

$$LB = \sum_{k \in \boldsymbol{\mathcal{K}}} \Delta_{ik} x^*_{ik} + \sum_{j \in \boldsymbol{\mathcal{J}}} \lambda_j. \tag{3.13}$$

However, the solution of the Lagrangian relaxation problem may not be the feasible solution with respect to the original Avatar replica problem ($\boldsymbol{P0}$), i.e., Constraint (3.5) may not be satisfied for the solution $\boldsymbol{\mathcal{Y}}^*_i = \left\{y^*_{ijk} | j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\right\}$. In order to obtain a feasible solution of the original problem, denoted as $\overline{\boldsymbol{\mathcal{Y}}}_i = \left\{\overline{y}_{ijk} | j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\right\}$, we can simply allocate the Avatar of MU i to the cloudlet, which has the lowest E2E delay among the cloudlets containing one replica of the Avatar, when MU i is in BS j, i.e., for each $j \in \boldsymbol{\mathcal{J}}$, we have:

$$\forall k \in \boldsymbol{\mathcal{K}}, \ \overline{y}_{ijk} = \begin{cases} 1, & t_{jk} = \min\left\{t_{jk} | k \in \boldsymbol{\mathcal{K}}^{''}_i\right\} \\ 0, & otherwise. \end{cases} \tag{3.14}$$

where $\boldsymbol{\mathcal{K}}_i^{''}$ is the set of available cloudlets (which contain one replica of MU i's Avatar) of MU i's Avatar, i.e., $\boldsymbol{\mathcal{K}}_i^{''} = \{k | x_{ik}^* = 1, k \in \boldsymbol{\mathcal{K}}\}$.

Substituting the feasible solution (i.e., $\overline{\boldsymbol{\mathcal{Y}}}_i = \left\{\overline{y}_{ijk} | j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\right\}$) into the objective function of the original problem (i.e., Equation (3.3)), we have the UB of the original problem:

$$UB = \sum_{j \in \boldsymbol{\mathcal{J}}} \sum_{k \in \boldsymbol{\mathcal{K}}} p_{ij} t_{jk} \overline{y}_{ijk}. \tag{3.15}$$

Note that the original problem always chooses its UB as its objective value because the UB can guarantee the existence of the feasible solution. However, selecting different values of Lagrange multiplier vector (i.e., $\boldsymbol{\lambda}$) may generate different values of the UB. Thus, by applying the subgradient method [64], we adjust the values of Lagrange multipliers in each iteration in order to obtain the smaller value of UB. The iteration terminates until $UB^{opt} - LB \leq \psi$, where UB^{opt} indicates the best (i.e., smallest) value of UB that has been found in the previous iterations.

In the n^{th} iteration ($n > 1$), the values of the Lagrangian multipliers λ_j^n ($j \in \boldsymbol{\mathcal{J}}$) are calculated based on the following expression:

$$\forall j \in \boldsymbol{\mathcal{J}}, \lambda_j^n = \max\left\{0, \lambda_j^{n-1} - \theta^n \left\{\sum_{k \in \boldsymbol{\mathcal{K}}} {y_{ijk}^*}^{n-1} - 1\right\}\right\}, \tag{3.16}$$

where λ_j^{n-1} are the Lagrangian multipliers generated in the previous iteration; ${y_{ijk}^*}^{n-1}$ ($j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}$) are the optimal solution of $\boldsymbol{P1}$ (i.e., the relaxed problem) in the previous iteration, which can be calculated based on Equations (3.11) and θ^n is the step length adopted in the n^{th} iteration, which can be calculate based on the following expression [65]:

$$\theta^n = \frac{\alpha \left(UB^{opt} - LB^{n-1}\right)}{\sum\limits_{j \in \boldsymbol{\mathcal{J}}} \left(\sum\limits_{k \in \boldsymbol{\mathcal{K}}} {y_{ijk}^*}^{n-1} - 1\right)^2}, \tag{3.17}$$

where α $(0 < \alpha < 2)$ is a decreasing adaptation parameter and LB^{n-1} is the value of LB in the previous iteration (i.e., $LB^{n-1} = \sum\limits_{k\in\boldsymbol{\mathcal{K}}} \Delta_{ik}{x_{ik}^{*}}^{n-1} + \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \lambda_j^{n-1})$. The detail of the LEARN algorithm is shown in Algorithm 1.

3.1.3 One example to illustrate the LEARN algorithm

Suppose there are three BSs in the network and each BS is attached to one cloudlet. Assume the average E2E delay vector is $\boldsymbol{\mathcal{T}} = \begin{bmatrix} 0 & 20 & 15 \\ 20 & 0 & 10 \\ 15 & 10 & 0 \end{bmatrix}$. There is an MU in the network and the occurrence probability of the MU in the respective BSs during the day is $\boldsymbol{\mathcal{P}}= [0.5, 0.3, 0.2]$. If we need to place two replicas (i.e., $\kappa = 2$) for its Avatar's virtual disk, then LEARN shall apply the following procedure to obtain the optimal replica placement for the MU:

- <u>Steps 1-2 in Algorithm 1</u>: randomly select the initial values of Lagrangian multipliers, e.g., $\boldsymbol{\lambda}= [5, 5, 5]$; initialize $LB = 0$ and $UB^{opt} = +\infty$;

- <u>Steps 4-5 in Algorithm 1</u>: given the value of $\boldsymbol{\lambda}$, calculate the values of $\boldsymbol{\mathcal{X}}^*$ and $\boldsymbol{\mathcal{Y}}^*$ for the MU based on Lemma 2; in this example, $\boldsymbol{\mathcal{X}}^*= [1, 1, 0]$ and $\boldsymbol{\mathcal{Y}}^* = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$. Then, update the value of LB based on Equation (3.13); in this example, $LB = 0$;

- <u>Steps 6-7 in Algorithm 1</u>: calculate the value of $\overline{\boldsymbol{\mathcal{Y}}}$ based on Equation (3.14). In this example, $\overline{\boldsymbol{\mathcal{Y}}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$. Then, update the value of UB based on Equation (3.15); in this example, $UB = 3.5$;

- <u>Steps 8-11 in Algorithm 1</u>: compare UB with UB^{opt}; in this example, $UB < UB^{opt}$, and thus $\boldsymbol{\mathcal{X}}^{opt} = \boldsymbol{\mathcal{X}}^* = [1, 1, 0]$ and $UB^{opt} = UB = 3.5$;

- <u>Steps 12-13 in Algorithm 1</u>: update the value of Lagrangian multipliers, i.e., $\boldsymbol{\lambda}$, based on Equation (3.16), and goes back to Steps 4-5 until $UB^{opt} - LB \leq \psi$.

The LEARN algorithm is executed offline, i.e., for a fixed period ΔT, LEARN will update the replica placement for different MUs during the off peak hours. Also, the replica placement updating period ΔT can also vary among different MUs. For instance, if MUs have similar behaviors during the workdays, LEARN only needs to update the replica placement of the MUs during the weekends; otherwise, it is preferred to update the replica placement daily. It is worth to note that a centralized controller (or a control function running in the SDN controller) is used to predict the occurrence probability for each MU, obtain the average E2E delay vector among different cloudlets and BSs from the SDN controller, and generate the replica placement vector for each MU by executing the proposed LEARN algorithm.

3.2 Adaptive Avatar Handoff

After the replicas of each MU's Avatar being deployed among cloudlets, the Avatar can be handed off among its available cloudlets based on its MU's location. Optimally, the Avatar will be handed off to the available cloudlet, which incurs the lowest E2E among its available cloudlets, when the MU roams into a new location. However, each cloudlet has its CPU and memory capacity, and so the Avatar may not be handed off to the optimal cloudlet because the optimal cloudlet may not have enough residual capacity to host the Avatar. Therefore, it is necessary to design an adaptive Avatar handoff strategy to determine the location of each MU's Avatar in each time slot in order to minimize the average E2E delay between all the MUs and their Avatars during the time slot by jointly considering the capacity limitation of each cloudlet.

Note that different from the Avatar replica placement problem (which tries to generate the replica placement solution for each MU's Avatar based on the statistics for a long time period (e.g., one day)), the adaptive Avatar handoff problem tries to obtain the location of each MU's Avatar (rather the Avatar's replicas) based on real time information (e.g., the current locations of all the MUs) and the problem should be solved in real time.

Algorithm 1 LEARN algorithm

Input: 1. The occurrence probability vector for MU i among BSs, i.e., $\boldsymbol{\mathcal{P}}_i = \{p_{ij} | j \in \boldsymbol{\mathcal{J}}\}$. 2. The average E2E delay vector $\boldsymbol{\mathcal{T}} = \{t_{jk} | j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\}$.

Output: The replica placement vector for MU i, i.e., $\boldsymbol{\mathcal{X}}_i^{opt} = \left\{x_{ik}^{opt} | k \in \boldsymbol{\mathcal{K}}\right\}$.

```
 1: Initialize the set of Lagrangian multipliers $\boldsymbol{\lambda} = \{\lambda_j | j \in \boldsymbol{\mathcal{J}}\}$.
 2: Initialize $LB = 0$ and $UB^{opt} = +\infty$.
 3: while $UB^{opt} - LB > \psi$ do
 4:     Calculate $\boldsymbol{\mathcal{X}}_i^*$ and $\boldsymbol{\mathcal{Y}}_i^*$ based on Lemma 2;
 5:     Update the value of $LB$ based on Equation (3.13);
 6:     Calculate $\overline{\boldsymbol{\mathcal{Y}}}_i$ based on Equation (3.14);
 7:     Calculate the value of $UB$ based on Equation (3.15);
 8:     if $UB < UB^{opt}$ then
 9:         $\boldsymbol{\mathcal{X}}_i^{opt} = \boldsymbol{\mathcal{X}}_i^*$;
10:         $UB^{opt} = UB$;
11:     end if
12:     Update step length $\theta^n$ based on Equation (3.17);
13:     Update Lagrangian multipliers $\boldsymbol{\lambda}$ based on Equation (3.16);
14: end while
15: return $\boldsymbol{\mathcal{X}}_i^{opt}$.
```

3.2.1 Problem formulation

Let l_{ij} be a binary indicator to identify MU i in BS j's coverage area (i.e., $l_{i,j} = 1$) or not ($l_{i,j} = 0$) in the current time slot. Meanwhile, let z_{ik} be a binary variable to indicate whether MU i's Avatar is in cloudlet k ($z_{ik} = 1$) or not ($z_{ik} = 0$) in the current time slot. $\boldsymbol{\mathcal{X}}_i^{opt} = \left\{x_{ik}^{opt} | k \in \boldsymbol{\mathcal{K}}\right\}$, which is generated by the LEARN algorithm, is the optimal replica placement vector for MU i. In order to avoid the virtual disk migration, MU i's Avatar can only be allocated to its available cloudlet k (i.e., z_{ik} could equal to 1 iff $k \in \boldsymbol{\mathcal{K}}_i^{''}$, where $\boldsymbol{\mathcal{K}}_i^{''} = \left\{k | x_{ik}^{opt} = 1, k \in \boldsymbol{\mathcal{K}}\right\}$). Thus, the average E2E delay between MU i and its Avatar in the current time slot, i.e., τ_i, can be expressed as follows:

$$\tau_i = \sum_{j \in \boldsymbol{\mathcal{J}}} \sum_{k \in \boldsymbol{\mathcal{K}}_i^{''}} l_{ij} t_{jk} z_{ik}. \tag{3.18}$$

Suppose all the MUs' Avatars are homogeneous, i.e., all the Avatars have the same CPU, memory and bandwidth configurations. Denote q_k as the capacity of cloudlet k, i.e., the total number of Avatars can be hosted by cloudlet k. Thus, we formulate the Avatar handoff problem as follows:

$$\boldsymbol{P2}: \qquad \arg\min_{z_{ik}} \sum_{i \in \boldsymbol{\mathcal{I}}} \sum_{j \in \boldsymbol{\mathcal{J}}} \sum_{k \in \boldsymbol{\mathcal{K}}_i^{''}} l_{ij} t_{jk} z_{ik} \tag{3.19}$$

$$s.t. \;\; \forall i \in \boldsymbol{\mathcal{I}}, \quad \sum_{k \in \boldsymbol{\mathcal{K}}_i^{''}} z_{ik} = 1, \tag{3.20}$$

$$\forall k \in \boldsymbol{\mathcal{K}}, \quad \sum_{i \in \boldsymbol{\mathcal{I}}} z_{ik} \le q_k, \tag{3.21}$$

$$\forall i \in \boldsymbol{\mathcal{I}} \; \forall k \in \boldsymbol{\mathcal{K}}, \quad z_{ik} \in \{0, 1\}, \tag{3.22}$$

where the objective is to minimize the average E2E delay between all the MUs and their Avatars in the current time slot. Constraint (3.20) indicates each Avatar should be hosted by one cloudlet, which contains one replica of the MU's Avatar;

Constraint (3.21) implies that each cloudlet has its capacity limitation; Constraint (3.22) indicates z_{ik} is a binary variable.

Note that it is not easy to solve $\boldsymbol{P2}$ since the available cloudlet set $\boldsymbol{\mathcal{K}}_i^{''}$ varies among different MUs' Avatars that makes the summation index k in the object function of $\boldsymbol{P2}$ to vary among different MUs' Avatars. The following lemma facilitates a dual problem of $\boldsymbol{P2}$, which can be readily solved because the summation index k in the objective function of the new problem does not vary among different MUs' Avatars.

Lemma 3. *Let $\tau_i^{'}$ be*

$$\tau_i^{'} = \sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}} \frac{l_{ij}t_{jk}}{x_{ik}^{opt}+\varepsilon} z_{ik}, \tag{3.23}$$

where ε is a very small positive value close to zero. Then, $\boldsymbol{P2}$ can be equivalently transformed into:

$$\begin{aligned}
\boldsymbol{P3}: \qquad & \arg\min_{z_{ik}} \sum_{i\in\boldsymbol{\mathcal{I}}}\sum_{j\in\boldsymbol{\mathcal{J}}}\sum_{k\in\boldsymbol{\mathcal{K}}} \frac{l_{ij}t_{jk}}{x_{ik}^{opt}+\varepsilon} z_{ik},\\
s.t.\ \ & \forall i\in\boldsymbol{\mathcal{I}},\ \sum_{k\in\boldsymbol{\mathcal{K}}} z_{ik} = 1,\\
& \forall k\in\boldsymbol{\mathcal{K}},\ \sum_{i\in\boldsymbol{\mathcal{I}}} z_{ik} \le q_k,\\
& \forall i\in\boldsymbol{\mathcal{I}}\ \forall k\in\boldsymbol{\mathcal{K}},\quad z_{ik}\in\{0,1\}.
\end{aligned}$$

Proof. For each $i\in\boldsymbol{\mathcal{I}}$, if $x_{ik}^{opt}=0$ (i.e., $k\in\boldsymbol{\mathcal{K}}\backslash\boldsymbol{\mathcal{K}}_i^{''}$), $\frac{l_{ij}t_{jk}}{x_{ik}^{opt}+\varepsilon}\to+\infty$, and thus z_{ik} should be set to zero in order to minimize the value of $\tau_i^{'}$; if $x_{ik}^{opt}=1$ (i.e., $k\in\boldsymbol{\mathcal{K}}_i^{''}$), the expression of $\tau_i^{'}$ is approximately equal to τ_i. Thus, $\boldsymbol{P2}$ and $\boldsymbol{P3}$ are equivalent. $\square$

By applying Lemma 3, the problem (i.e., $\boldsymbol{P2}$) can be transformed into:

$$\arg\min_{z_{ik}} \sum_{i\in\boldsymbol{\mathcal{I}}}\sum_{k\in\boldsymbol{\mathcal{K}}} c_{ik}z_{ik} \tag{3.24}$$

$$s.t.\ \ \textit{Constraints}\ (3.20),(3.21),(3.22),$$

where $c_{ik} = \frac{\sum\limits_{j \in \boldsymbol{\mathcal{J}}} l_{ij} t_{jk}}{x_{ik}^{opt} + \varepsilon}$ is the weighted E2E delay between MU i and its Avatar located in cloudlet k.

The proposed Avatar handoff problem is considered as a special case of the generalized assignment problem (in which Constraint (3.21) is depicted as $\forall k \in \boldsymbol{\mathcal{K}}, \sum\limits_{i \in \boldsymbol{\mathcal{I}}} a_{ik} z_{ik} \leq q_k$, where $a_{ik} > 0$), which is proven to be NP-hard. Recently, many heuristics have been designed to find the suboptimal solution of the generalized assignment problem and each of them has its tradeoff between the complexity and the performance. In the proposed network, we need to allocate tens of thousands of MUs into tens of thousands of cloudlets in each time slot. Thus, we design a novel LatEncy aware Avatar hanDoff (LEAD) algorithm to efficiently solve the proposed Avatar handoff problem.

3.2.2 LatEncy aware Avatar hanDoff (LEAD)

First, we build a relaxed Avatar handoff problem (i.e., $\boldsymbol{P4}$) by relaxing Constraint (3.21), i.e.,

$$\begin{aligned} \boldsymbol{P4}: \quad & \arg\min_{z_{ik}} \sum_{i \in \boldsymbol{\mathcal{I}}} \sum_{k \in \boldsymbol{\mathcal{K}}} c_{ik} z_{ik} \\ s.t.\ \ & \forall i \in \boldsymbol{\mathcal{I}}, \ \sum_{k \in \boldsymbol{\mathcal{K}}} z_{ik} = 1, \\ & \forall i \in \boldsymbol{\mathcal{I}}\ \forall k \in \boldsymbol{\mathcal{K}}, \quad z_{ik} \in \{0,1\}. \end{aligned}$$

The optimal solution of $\boldsymbol{P4}$, denoted as $\boldsymbol{\mathcal{Z}}_i^{'} = \left\{ z_{ik}^{'} | k \in \boldsymbol{\mathcal{K}} \right\}$, can be easily derived, i.e.,

$$\forall i \in \boldsymbol{\mathcal{I}}, \quad z_{ik}^{'} = \begin{cases} 1, & k = \arg\min\limits_{\bar{k}} \left\{ c_{i\bar{k}} | \bar{k} \in \boldsymbol{\mathcal{K}} \right\}. \\ 0, & otherwise. \end{cases} \tag{3.25}$$

The optimal solution of $\boldsymbol{P4}$ generates the optimal cloudlet (which incurs the minimum weighted E2E delay among the cloudlets) for each Avatar. However, it may not be the feasible solution of the original Avatar handoff problem, i.e., the total number of Avatars that are hosted by some cloudlets may exceed their capacity. Denote these set of cloudlets as $\boldsymbol{\mathcal{K}}_1$, i.e., $\boldsymbol{\mathcal{K}}_1 = \left\{ k | \sum\limits_{i \in \boldsymbol{\mathcal{I}}} z_{ik}^{'} > q_k, k \in \boldsymbol{\mathcal{K}} \right\}$; denote the set of cloudlets, which have enough resources to host at least one Avatar, as $\boldsymbol{\mathcal{K}}_2$, i.e., $\boldsymbol{\mathcal{K}}_2 = \left\{ k | \sum\limits_{i \in \boldsymbol{\mathcal{I}}} z_{ik}^{'} < q_k, k \in \boldsymbol{\mathcal{K}} \right\}$; denote the set of MUs, whose Avatars are hosted by cloudlet $k \in \boldsymbol{\mathcal{K}}_1$, as $\boldsymbol{\mathcal{I}}_1$, i.e., $\boldsymbol{\mathcal{I}}_1 = \left\{ i | z_{ik}^{'} = 1, k \in \boldsymbol{\mathcal{K}}_1, i \in \boldsymbol{\mathcal{I}} \right\}$. The basic idea of the LEAD algorithm is to choose a suitable MU i's Avatar, whose optimal cloudlet has violated its capacity limitation (i.e., $i \in \boldsymbol{\mathcal{I}}_1$), and reallocate the Avatar into its suboptimal cloudlet, which has enough space for hosting at least one Avatar, for each iteration. The suboptimal cloudlet, denoted as $k^{'}$, of MU i is defined as the cloudlet that incurs the minimum weighted E2E delay among the cloudlets, which have enough resources to host at least one Avatar, i.e., $k^{'} = \arg\min\limits_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_2 \right\}$.

Denote Δc_i ($i \in \boldsymbol{\mathcal{I}}_1$) as the weighted E2E delay degradation by reallocating MU i's Avatar from its optimal cloudlet k into its suboptimal cloudlet $k^{'}$, i.e.,

$$\forall i \in \boldsymbol{\mathcal{I}}_1, \quad \Delta c_i = c_{ik} - c_{ik^{'}}, \tag{3.26}$$

where $k = \arg\min\limits_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_1 \right\}$ and $k^{'} = \arg\min\limits_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_2 \right\}$. Thus, the LEAD algorithm will select to reallocate a suitable MU i's Avatar, where $i = \arg\min\limits_{\overline{i}} \left\{ \Delta c_{\overline{i}} | \overline{i} \in \boldsymbol{\mathcal{I}}_1 \right\}$, in each iteration in order to minimize the weighted E2E delay degradation. The iteration continues until $\boldsymbol{\mathcal{K}}_1 = \emptyset$. The detail of the LEAD algorithm is shown in Algorithm 2.

The complexity of each iteration in LEAD (i.e., from Step-5 to Step-9 in Algorithm 2) is $O(|\boldsymbol{\mathcal{I}}| \, |\boldsymbol{\mathcal{K}}|) + O(|\boldsymbol{\mathcal{I}}|) + 3 \times O(1) = O(|\boldsymbol{\mathcal{I}}| \, |\boldsymbol{\mathcal{K}}|)$, where $O(|\boldsymbol{\mathcal{I}}| \, |\boldsymbol{\mathcal{K}}|)$ is the complexity of Step-5, $O(|\boldsymbol{\mathcal{I}}|)$ is the complexity of Step-6, and $3 \times O(1)$ is the complexity for executing Step-7 to Step-9. In the worst case scenario, the total number of iterations of LEAD is $|\boldsymbol{\mathcal{I}}|$. Thus, the complexity of LEAD is $O(|\boldsymbol{\mathcal{I}}|^2 \, |\boldsymbol{\mathcal{K}}|)$.

Note that the LEAD algorithm cannot guarantee that all the Avatars can be placed in one of its available cloudlets, i.e., it might happen that some Avatars, all of whose available cloudlets are full, cannot be hosted by one of its available cloudlets. Then, these Avatars will be placed in the central cloud (by default, the central cloud contains at least one replica of each MU's Avatar).

Lemma 4. *The LEAD algorithm terminates after a finite number of iterations, producing an feasible solution to the original Avatar handoff problem.*

Proof. Let $\xi = \sum_{k \in \boldsymbol{\mathcal{K}}_1} \left(\sum_{i \in \boldsymbol{\mathcal{I}}} z_{ik}^{'} - q_k \right)$. Assume $\boldsymbol{\mathcal{K}}_1 \neq \emptyset$ initially, and so $\xi > 0$. Then, in each iteration, the value of ξ is decreased by one because LEAD will reallocate MU i's Avatar from cloudlet k (i.e., set $z_{ik}^{'} = 0$), where $k = \arg\min_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_1 \right\}$, into cloudlet $k^{'}$, where $k^{'} = \arg\min_{\overline{k}} \left\{ c_{i\overline{k}} | \overline{k} \in \boldsymbol{\mathcal{K}}_2 \right\}$. Thus, ξ will be reduced to zero in a finite number of steps, and hence $\boldsymbol{\mathcal{K}}_1 = \emptyset$. $\square$

Similar to LEARN, the LEAD algorithm is also executed in a centralized manner to determine the locations of Avatars for their MUs in each time slot.

3.3 Simulation Results

In order to evaluate our proposed replica placement algorithm and Avatar handoff algorithm, we have obtained data traces of more than 13,000 MUs and extracted their mobility in one day in Heilongjiang province in China[5]. The whole area contains 5,962 BSs and each MU's location is monitored during one day period. Specifically, each packet that is transmitted to/from a MU is monitored, and the packet analyzer extracts the BS information (i.e., the BS's ID and location) from each packet and considers the BS's location to be the current location of this MU (for instance, if a packet from the MU contains the information of BS-A, then we say the MU is currently associated with BS-A and the current location of the MU is BS-A's location).

[5]The authors acknowledge the Center for Data Science of Beijing University of Posts and Telecommunications for providing these invaluable data traces.

Algorithm 2 LEAD algorithm

Input: 1. The replica placement vector for each MU, i.e., $\boldsymbol{\mathcal{X}}_i^{opt} = \left\{x_{ik}^{opt} | k \in \boldsymbol{\mathcal{K}}\right\}, \forall i \in \boldsymbol{\mathcal{I}}$. 2. The average E2E delay vector, i.e., $\boldsymbol{\mathcal{T}} = \{t_{jk} | j \in \boldsymbol{\mathcal{J}}, k \in \boldsymbol{\mathcal{K}}\}$. 3. The location indicator vector for all MUs in the current time slot, i.e., $\boldsymbol{\mathcal{L}} = \{l_{ij} | i \in \boldsymbol{\mathcal{I}}, j \in \boldsymbol{\mathcal{J}}\}$.

Output: Avatar location indicator vector for all MUs, i.e., $\boldsymbol{\mathcal{Z}}^{'} = \left\{z_{ik}^{'} | i \in \boldsymbol{\mathcal{I}}, k \in \boldsymbol{\mathcal{K}}\right\}$.

```
1: Initialize $z_{ik}^{'}$ by solving the relaxed problem (i.e., $\boldsymbol{P4}$) based on Equation (3.25).
2: Initialize the cloudlet sets $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$ based on their definition.
3: Initialize the MU set $\boldsymbol{\mathcal{I}}_1$ based on its definition.
4: while $\boldsymbol{\mathcal{K}}_1 \neq \emptyset$ do
5:     $\forall i \in \boldsymbol{\mathcal{I}}_1$, calculate $\Delta c_i$ based on Equation (3.26);
6:     Find MU $i$, where $i = \arg\min_{\bar{i}} \left\{\Delta c_{\bar{i}} | \bar{i} \in \boldsymbol{\mathcal{I}}_1\right\}$;
7:     Reallocate MU $i$'s Avatar from its optimal cloudlet $k$ (i.e., set $z_{ik}^{'} = 0$) into its suboptimal cloudlet $k^{'}$, (i.e., set $z_{ik^{'}}^{'} = 1$);
8:     Update the cloudlet sets $\boldsymbol{\mathcal{K}}_1$ and $\boldsymbol{\mathcal{K}}_2$;
9:     Update the MU set $\boldsymbol{\mathcal{I}}_1$;
10: end while
11: return $\boldsymbol{\mathcal{Z}}^{'}$.
```

We apply these MU mobility traces to obtain the occurrence probability vector for each MU among BSs during the day (i.e., the values of $\boldsymbol{\mathcal{P}}_i = \{p_{ij} | j \in \boldsymbol{\mathcal{J}}\}$, where $p_{ij} = \frac{\textit{the total amount of time MU i is associated with BS j}}{\textit{one day period}}$). Meanwhile, we assume that each BS is attached to a cloudlet and the average E2E delay between a BS and a cloudlet (i.e., the value of t_{jk}, where $j \neq k$) is a function of the geographic distance between the BS and the cloudlet [66], i.e., $t_{jk} = 0.016d_{jk} + 22.3$, where d_{jk} is the distance between BS j and cloudlet k in unit km and the unit of t_{jk} is in ms. Each BS's geographic location (i.e., the longitude and the latitude of the BS) is given by the MU mobility traces and each cloudlet is attached to a BS, and thus the value of d_{jk} is known; consequently, the value of t_{jk} can be calculated for all $j \in \boldsymbol{\mathcal{J}}$ and $k \in \boldsymbol{\mathcal{K}}$ based on the E2E delay model.

3.3.1 Performance of LEARN

First, we simulate the proposed LEARN algorithm based on the mentioned MU mobility trace. Specifically, we first calculate the value of p_{ij} based on the mentioned MU mobility trace. Then, by taking $\boldsymbol{\mathcal{P}}_i$ and $\boldsymbol{\mathcal{T}}$ as input parameters, we further obtain the replica placement vector for each MU by applying the LEARN algorithm. Consequently, the average E2E delay for all the MUs during the day is derived given the replica placement vector for each MU. For comparisons, we considered the scenario that all the MUs' Avatars are located in the central data center (i.e., MUs' Avatars cannot migrate when MUs are roamed among BSs), which is placed in the southeast point of the area. In this scenario, we also calculate the average E2E delay between MUs and the data center during the day.

As shown in Figure 3.6, given the number of replicas (i.e., the value of κ), the average E2E delay achieved by LEARN is significantly reduced as compared to that of the traditional big data network (in which the MUs access their Avatars in the remote cloud/data center via the Internet). Moreover, as the number of replicas

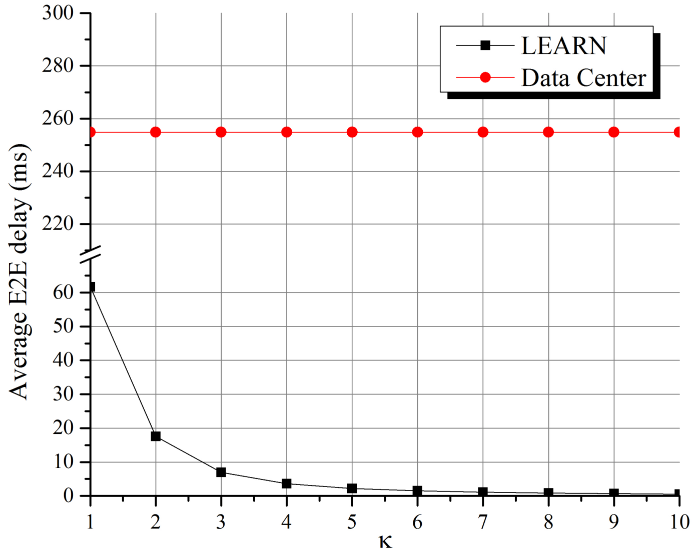

Figure 3.6 Average E2E delay of the cloudlet network.

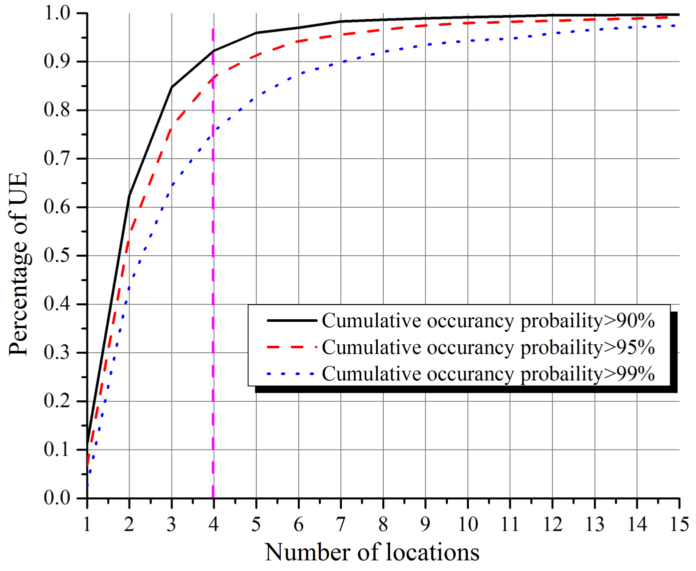

Figure 3.7 Statistical results of the MU mobility trace.

increases, the average E2E delay is reduced accordingly. Specifically, as compared to the traditional data center network, the average E2E delay achieved by LEARN is improved by 75.79%, 93.12%, 97.27%, and 98.59% when the value of is selected to be 1, 2, 3, and 4, respectively. Note that $\kappa = 1$ (i.e., there is one replica for each MU) indicates the location of each Avatar is fixed (i.e., the Avatar is placed in the location where its MU most visit) and the Avatar cannot handoff among the cloudlets because

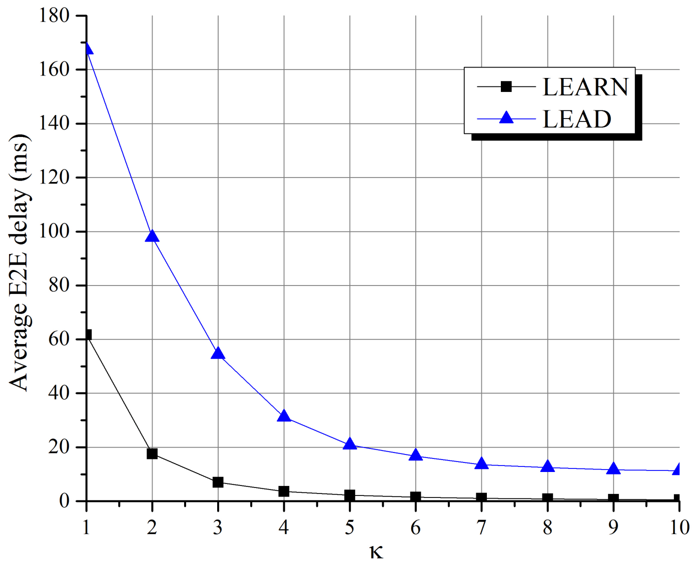

Figure 3.8 Average E2E delay of LEAD and LEARN.

no extra replicas are placed in other cloudlets. Also, when the value of κ is greater than 4, the decrement of the E2E delay by increasing the value of κ is not significant. We further analyze the mobility trace by calculating at least how many locations, i.e., the BSs' coverage area, that each MU will stay over 90%, 95% and 99% of the time during the day, respectively. As shown in Figure 3.7, 92.22%, 86.93% and 75.65% of the MUs spend 90%, 95% and 99% of the time during the day (in terms of 21.6, 22.8 and 23.76 hours) to stay at only four locations, respectively. Thus, placing five replicas for the MUs may not significantly benefit the average E2E delay during the day as compared to placing four replicas. Note that placing more replicas for each Avatar may increase the CAPEX to the cloudlet network provider by deploying more storage resources. Also, allocating more replicas for each Avatar may generate more synchronous traffic, and thus increase the traffic load of the cloudlet network.

3.3.2 Performance of LEAD

We further evaluate the performance of our proposed LEAD algorithm. Each Avatar's replicas have already been placed to the corresponding cloudlets, which are calculated by the LEARN algorithm. Still, MUs' mobilities and the locations of the BSs follow the mobility traces that we have sampled from the real world. By applying the mobility traces, the location indicator vector for all MUs (i.e., the values of $\boldsymbol{\mathcal{L}}$) in different time slots during the day can be obtained. Initially, we setup the capacity of all the cloudlet to be the same, i.e., $\forall k \in \boldsymbol{\mathcal{K}}, q_k = 10$ (each cloudlet can host at most 10 Avatars in each time slot). We first test the average E2E delay between MUs and their Avatars during the day. As shown in Figure 3.8, as compared to the results of the LEARN algorithm[6] (which generates the optimal average E2E delay between MUs and their Avatars without considering the capacity constraints), the average E2E delay generated by the LEAD algorithm is increased because some Avatars cannot handoff to their optimal cloudlets (due to the cloudlet capacity limitation) when the MUs roam away. Consequently, these Avatars need to handoff to their suboptimal cloudlets or to the remote data center. Note that as the number of replicas increases, the average E2E delay decreases accordingly. This is because as the number of replicas increases, the Avatars, whose optimal cloudlets exceed their capacity limitation, have higher probability to handoff to their suboptimal cloudlets with lower E2E delay. For instance, as shown in Figure 3.9, assume there are two Avatar's replicas that have been optimally placed in cloudlet A and cloudlet B. Suppose the optimal location of the Avatar is cloudlet A at the current time slot but the cloudlet A is full, and so the Avatar needs to handoff to the suboptimal cloudlet, which is cloudlet B; this may increase the E2E delay between the Avatar and its MU, and we denote the E2E delay increment as t_{A-B}. Then, if there are one more Avatar's replicas that have been placed in the cloudlet C, and so the Avatar can be handed off to the suboptimal cloudlet, which can be cloudlet B or cloudlet C. Thus, the E2E delay increment by handing off the Avatar to the suboptimal cloudlet is $\min\{t_{A-B}, t_{A-C}\}$, where t_{A-C} is the E2E delay increment by handing off the Avatar to cloudlet C. Obviously,

[6]We consider the average E2E delay of the LEARN algorithm as the lower bound in terms of the best case scenario of the LEAD algorithm.

$t_{A-B} \geq \min\{t_{A-B}, t_{A-C}\}$. Therefore, the average E2E delay between MUs and their Avatars decreases when the number of the replicas increases.

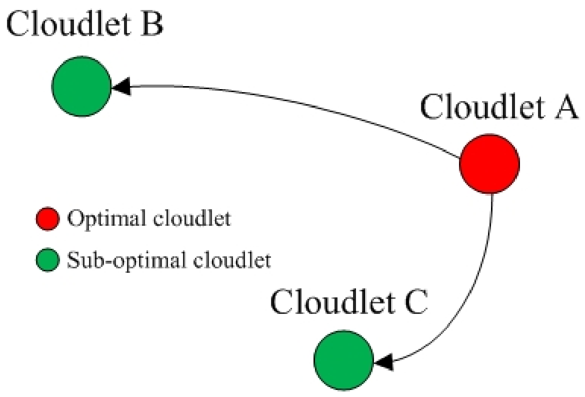

Figure 3.9 Illustration of the average E2E delay reduction.

Note that, as mentioned before, the LEAD algorithm cannot guarantee that all the Avatars can be handed off to their available cloudlets, i.e., some Avatars need to be handed off to the remote data center, which is the main factor of increasing the average E2E delay because the average E2E between the MUs and the remote data center reaches 254.8 ms, which is significantly longer than the average E2E delay produced by the LEAD algorithm. Thus, we further test the average number of the *remote Avatars*, which are defined as the Avatars that have to be handed off to the remote data center, during the day. As shown in Figure 3.10, the average number of the *remote Avatars* decreases as the number of each Avatar's replicas increases. This is because that the probability (that all the Avatar's available cloudlets are full) decreases as the number of the Avatar's replicas increases.

As mentioned previously, the average E2E delay of the LEAD algorithm is longer than that of the LEARN algorithm because LEARN does not consider the capacity limitation of the cloudlets. In other words, owing to the capacity constraints, some Avatars cannot be handed off to their optimal cloudlets, thus resulting in the average E2E delay growth in LEAD. In order to study how the cloudlet capacity impacts the performance of the LEAD algorithm. We further record the average E2E delay of

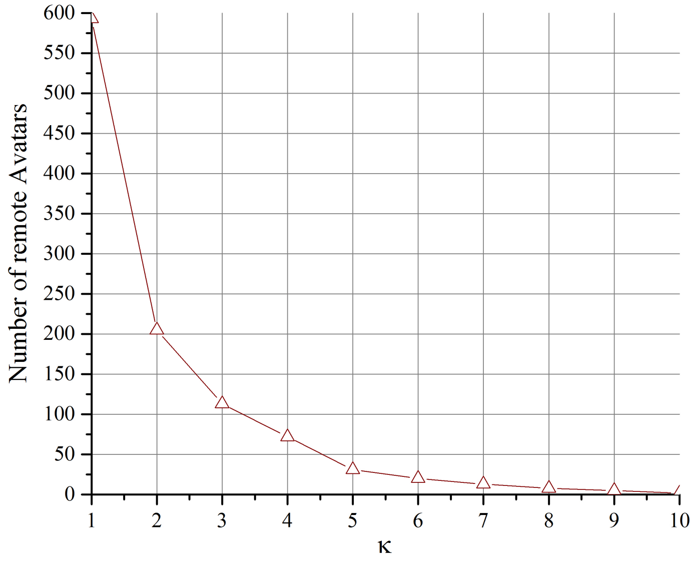

Figure 3.10 Average number of *remote Avatars*.

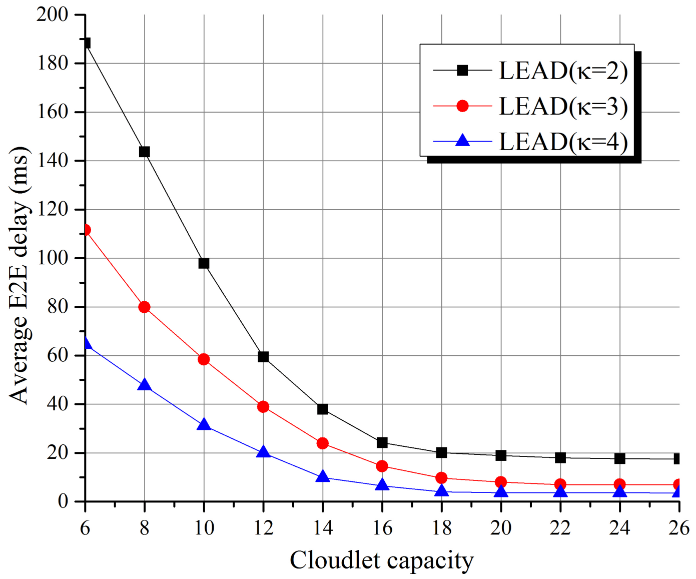

Figure 3.11 Average E2E delay over different cloudlet capacities.

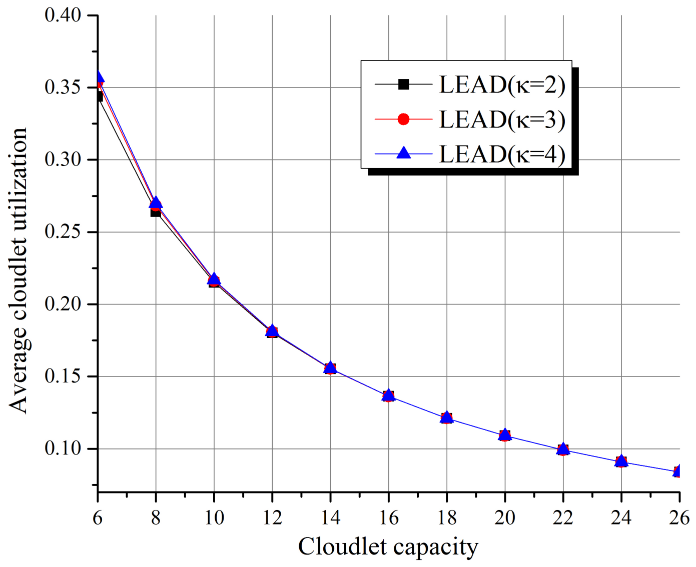

Figure 3.12 Average cloudlet utilization over different cloudlet capacities.

the LEAD algorithm by changing the capacity of each cloudlet. As shown in Figure 3.11, when the cloudlet capacity increases (from 6 to 16), the average E2E delay of the LEAD algorithm improves tremendously. As the cloudlet capacity reaches 16, the average E2E delay of the LEAD algorithm does not improve significantly. This is because most of the Avatars can be handed off to their optimal cloudlets as their MUs roam away. Although increasing the cloudlet capacity can improve the average E2E delay, the average cloudlet utilization[7] is reduced accordingly. As shown in Figure 3.12, the average cloudlet utilization drops from 35% to 10.9% as the cloudlet capacity increases from 6 to 26.

Therefore, there is a tradeoff between the average E2E delay and the cloudlet utilization. In order to optimize the tradeoff, the capacity of different cloudlets should be varied. Specifically, the cloudlets, whose connected BSs have higher MU density (such as shopping malls and public transportations), should have larger capacity, and vice versa. Thus, in the future, we will try to design a cloudlet deployment strategy

[7] $cloudlet\ utilization = \frac{total\ number\ of\ Avatars\ hosted\ by\ the\ cloudlet}{the\ cloudlet\ capacity}$

to determine the capacity of each cloudlet such that the average cloudlet utilization can be maximized and the average E2E delay is guaranteed.

CHAPTER 4

CLOUDLET NETWORK EMPOWERED INTERNET OF THINGS

Today, a tremendous number of smart devices and objects are embedded with sensors, enabling them to sense real-time information from the environment. This phenomenon has culminated to the intriguing concept of the Internet of Things (IoT) in which all the smart things, such as smart cars, wearable devices, laptops, sensors, and industrial and utility components, are connected via a network of networks and empowered with data analytics that are forever changing the way we work, live and play. In the past few years, many startups are embracing and actualizing the concept of IoT in areas such as smart homes/buildings, smart cities, intelligent healthy cares, smart traffic, smart environments, etc. Essentially, IoT technology enables sensors to transmit their sensed data and actuators to be controlled so as to facilitate users to understand and change the physical world. Basically, as shown in Figure 4.1, the IoT architecture comprises three layers, i.e., the perception layer, the network layer, and the application layer [67, 68]. Specifically, the perception layer represents the physical IoT devices, which perform different functionalities that are directly related to the hardware, e.g., temperature sensors capture the current surrounding temperature values and the switch automatically turns off the light. The perception layer digitizes and transmits the data to/from the network layer, whose aim is to provide the connectivity among different IoT devices by applying different communications technologies. The application layer provides various functionalities (such as resource discovery, data management, and communications management) and interfaces to access different hardware resources and provision high-quality smart services to customers.

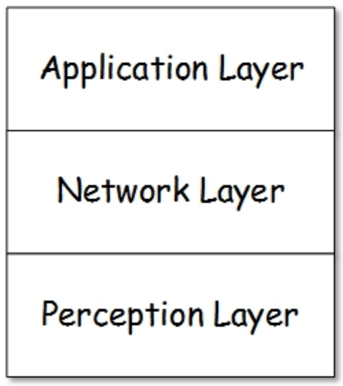

Figure 4.1 Basic IoT architecture.

IoT devices are mostly battery-constrained [69], and it is thus not energy efficient to enable an IoT device (e.g., a temperature sensor) to send its IoT content (e.g., the current temperature value) to a large number of clients, who try to retrieve the contents generated by IoT devices. Caching IoT contents in the network layer is proposed to solve the energy inefficiency problem and speed up the content delivery process. Specifically, network nodes (e.g., routers and gateways) would cache the received IoT contents based on their caching strategies [70–74]. Then, if a network node receives a content retrieval request (from a client) and it has the requested content, the network node would reply to this request without forwarding it to the original IoT device. The feasibility of using in-network caching technologies for IoT has been discussed in the Information-Centric Networking Research Group (ICNRG) under the Internet Research Task Force (IRTF) [75, 76]. However, the traditional in-network caching strategies applied in Content Delivery Networks (CDNs) may not suitable for the IoT system owing to the unique features of IoT [70,74]. First, most of the IoT devices are resource constrained, and so the main objective of content caching placement in IoT is to minimize the energy consumption rather than to minimize the delay for delivering contents to users in CDNs. Second, the contents generated by IoT servers exhibit transient feature [70, 77–79]; this feature is quite different from the contents cached in CDNs, whose popularity remains stable over long timescale.

Typical examples include popular news with short videos, which are updated every 2-3 hours; new movies, which change their popularity every week; new music videos, which change their popularity about every month [80]. Third, the sizes of IoT contents may be smaller than the sizes of contents in CDNs, but the number of IoT contents may be larger than the number of contents in CDNs.

Owing to these unique features of the IoT system, many in-network content caching (i.e., caching IoT contents in the network layer) strategies in the context of IoT has been proposed. Vural *et al.* [70, 71] proposed an in-networking caching method to facilitate IoT content caching. Specifically, IoT contents are cached in the edge routers and they argued that clients' obtaining the contents from the edge routers may lose freshness (i.e., the obtained data may not be up-to-date) but reduce the network traffic as compared to the clients' obtaining the contents from the original servers. Thus, they dynamically modified the edge routers' content caching probabilities in order to optimize the tradeoff between content freshness and network traffic. Similarly, Hail *et al.* [72] proposed a network layer IoT content caching strategy in the multi-hop wireless network scenario, in which IoT devices are equipped to cache the forwarding contents. They designed a novel distributed probabilistic caching strategy, which is based on the freshness of the content as well as the energy level and the storage capability of the device, in order to improve the energy efficiency of the IoT devices and reduce the content delivery delay. Niyato *et al.* [81] considered the case that the contents generated by the IoT devices should be cached in the local wireless access point and clients should always retrieve the corresponding contents from the wireless access point. They designed an optimal caching update period for each IoT device (in updating its content in the wireless access point) in order to maximize the hit rate in terms of the probability that the clients can successfully obtain the corresponding contents. Li *et al.* [82] proposed a novel distributed access control method to enable network nodes to secure their cached contents, i.e., only authorized clients can access the contents cached in network nodes. In order to solve the cache inconsistency problem, Ha and Kim [83] proposed to enable each IoT device to select and maintain a set of network nodes in caching its content. Since the IoT device is aware of which network nodes have cached its content, the IoT device is able to update the cached content in these network nodes to guarantee the consistency. This approach, however, incurs heavy burdens on the IoT device, i.e., the IoT device needs to periodically obtain the information from all the network nodes to determine which network nodes are suitable to cache its content; meanwhile, the IoT device needs to send the up-to-date content to all the network nodes (which have cached the content) by itself. In order to guarantee a client in receiving the up-to-date content, Quevedo *et al.* [84] proposed to enable each client to specify the freshness requirement of the content (i.e., the maximum delay of the content from being generated to being requested) in the content retrieval request. Thus, if a network node (which has cached the related content) receives the related content retrieval request, it would respond with the cached content if the freshness of the cached content satisfies the requirement; otherwise, it would forward the content retrieval request to the next hop. Yet, it is difficult for a client to determine the freshness requirement of its requested content; a lower freshness requirement may result in the content being retrieved from the IoT device more frequently, and a higher freshness requirement may incur the out-of-date content being received by the client.

Caching IoT contents in the network layer can potentially solve the energy inefficiency problem and speed up the content delivery process. However, this would incur the cache inconsistency problem [75, 85, 86], i.e., the cached content may not accurately reflect the current state of the corresponding IoT device. For example, the content generated by a parking spot sensor indicates the state of the parking spot (i.e., empty or occupied). Suppose that the parking spot is initially empty and this content is cached by a network node. Afterwards, the parking spot becomes occupied and a client, whose content retrieval request is responded by the network node (which previously cached the content of the parking spot being empty), may not obtain the correct state of the parking spot. The reason for the cache inconsistency problem is the transparent nature of IoT contents (i.e., the value of an IoT content is quickly diminished, and thus this IoT content needs to be replaced by a fresh one) and the local caching decision made by network nodes (i.e., an IoT device is unaware of its content having been cached by the network nodes. The IoT device is thus unable to update the caches in those network nodes).

In order to solve the cache inconsistency as well as the energy inefficiency problem during the IoT content delivery process, we propose to cache IoT resources at the mobile edge [77, 87]. Specifically, an IoT resource (which is different from an IoT content) is defined as a specific physical phenomenon captured by a specific IoT device. Yet, an IoT content represents the state of a physical phenomenon at a specific moment. For instance, a temperature sensor is to sense the temperature value of Bob's smart home and the current temperature value is 30^oC. Then, "the temperature value of Bob's smart home" is an IoT resource (which is hosted by the temperature sensor) and "30^oC" is an IoT content. Caching an IoT resource at the mobile edge indicates that the IoT resource is cached in a broker (which is an IoT middleware with computing, communications, and storage capabilities) placed in the nearby cloudlet. Thus, the IoT device will send the up-to-date content to the broker and clients can retrieve the content of the IoT resource from the broker (rather than the IoT device) via mobile networks.

Note that IoT resource caching in brokers is different from IoT content caching in the IoT network layer. First, caching IoT resources, rather than IoT contents, is implemented in brokers, where an IoT resource is considered as an identifier to specify a set of related IoT contents, which are generated by the same device and describe the same physical phenomenon under different time slots. Thus, IoT resources do not have the freshness feature. Thus, clients can retrieve up-to-date contents by accessing the corresponding IoT resources. Second, IoT resources are cached in a broker (i.e., a middleware) rather than network nodes. The broker is considered as a cache entity, which is discoverable by clients and IoT devices. Third, IoT devices make caching decisions by themselves, and thus are aware of where their cached IoT resources are located. Hence, it is feasible for IoT devices to update the contents and add access control policies of the IoT resources cached in the broker. Therefore, IoT resource caching in the application layer can basically resolve the cache inconsistency and security problems.

In the next sections, we will explain how to achieve IoT resources caching in brokers based on the existing IoT application layer protocols and how to leverage the cloudlet network in conducting IoT resources caching.

4.1 Implementation of Caching IoT Resources in Brokers

In this section, we illustrate the implementation of caching IoT resources in brokers by applying the current application layer communications protocols.

4.1.1 Constraint Application Protocol (CoAP)

The Constrained RESTful Environments (CoRE) working group within the Internet engineering task force (IETF) has focused on the development of application layer protocols by considering the resource constrained nature of IoT devices. Constraint Application Protocol (CoAP) [88], a Web application transfer protocol intended to provide RESTful services in constrained nodes and networks, is the main contribution of such working group.

In CoAP, there are two logical entities: server and client. A server is a resource host, which generates contents of the resource. A client is a resource requester, which tries to retrieve contents of the resource. A resource is an object reflecting a specific physical phenomenon; normally, a resource is identified by a Uniform Resource Identifier (URI) [89]. For instance, if Bob's mobile phone tries to obtain the current temperature value provided by the temperature sensor, which is equipped in Bob's smart home, the temperature sensor is a server, Bob's mobile phone is a client, and "the temperature value in Bob's smart home" is a resource hosted by the server; meanwhile, this resource can be identified by a URI (i.e., a unique address that can be searched on the Internet), such as "coap://bob_home/temp". The interactions between a server and a client follow a request/response model. For instance, as shown in Figure 4.2, the client (e.g., Bob's mobile phone) sends a resource retrieval request, which includes the URI that points to a specific resource in the server (e.g.,"the temperature value in Bob's smart home" resource hosted by the temperature sensor). Consequently, the server responds to the resource retrieval request by sending the content of the resource to the client. Note that servers and clients are not restricted to be physical IoT devices, i.e., virtual devices/objects can also be servers and clients. The concept of a virtual device/object refers to a digital counterpart of a real physical entity in IoT [90]. For instance, a service/application provided by a mobile phone can be a virtual device/object.

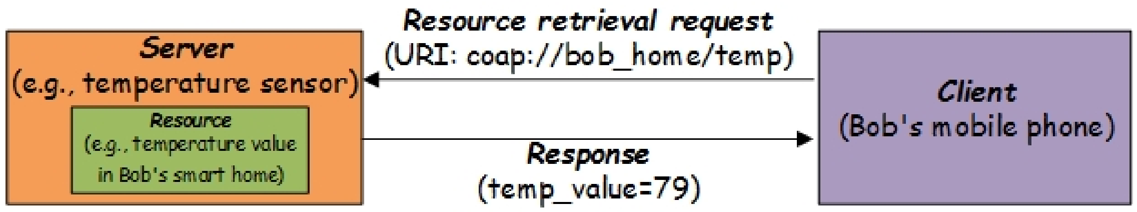
Server
(e.g., temperature sensor)
Resource
(e.g., temperature value in Bob's smart home)
Resource retrieval request
(URI: coap://bob_home/temp)
Response
(temp_value=79)
Client
(Bob's mobile phone)

Figure 4.2 CoAP request/response interaction model.

A client can continuously observe the resource by sending a resource observe request to the server, which hosts the resource. The resource observe request should contain the URI of the resource as well as the observe conditions, which indicate the criteria for the server to transmit the contents of the resource to the client. For instance, as shown in Figure 4.3, Client-1 (e.g., bob's mobile phone) sends a resource observe request to the server (e.g., temperature sensor), which indicates that Client-1 tries to observe the resource (which is identified by the resource URI) and the server should send the up-to-date content of the resource every 5 *mins*.

We can divide the clients into two classes: *read clients* and *observe clients*. The read clients send resource retrieval requests to obtain the contents of resources in the corresponding servers. Normally, the servers would not store any information (e.g., URIs) of these clients. The observe clients send resource observe requests to continuously monitor the status of resources. The servers should maintain the URIs of the servers and their corresponding observe conditions by storing them in the local binding table. As shown in Figure 4.3, two clients (e.g., bob's mobile phone and the air conditioner) send the resource observe requests to the server (e.g., temperature sensor). The server should create a binding table [91], which maintains the URIs of the two clients and their corresponding observe conditions[1]. Consequently, the server can send the contents of the resource to the corresponding clients once their observe conditions are satisfied.

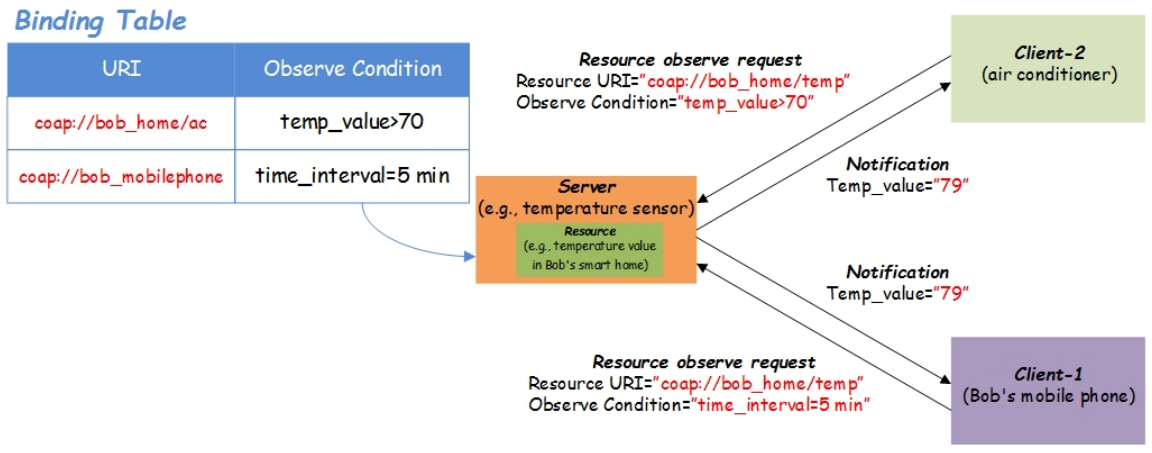

Figure 4.3 Clients observe the resource.

[1]The thesis uses different terminologies, from those applied in the corresponding IETF RFCs and drafts.

4.1.2 Resource Directory (RD)

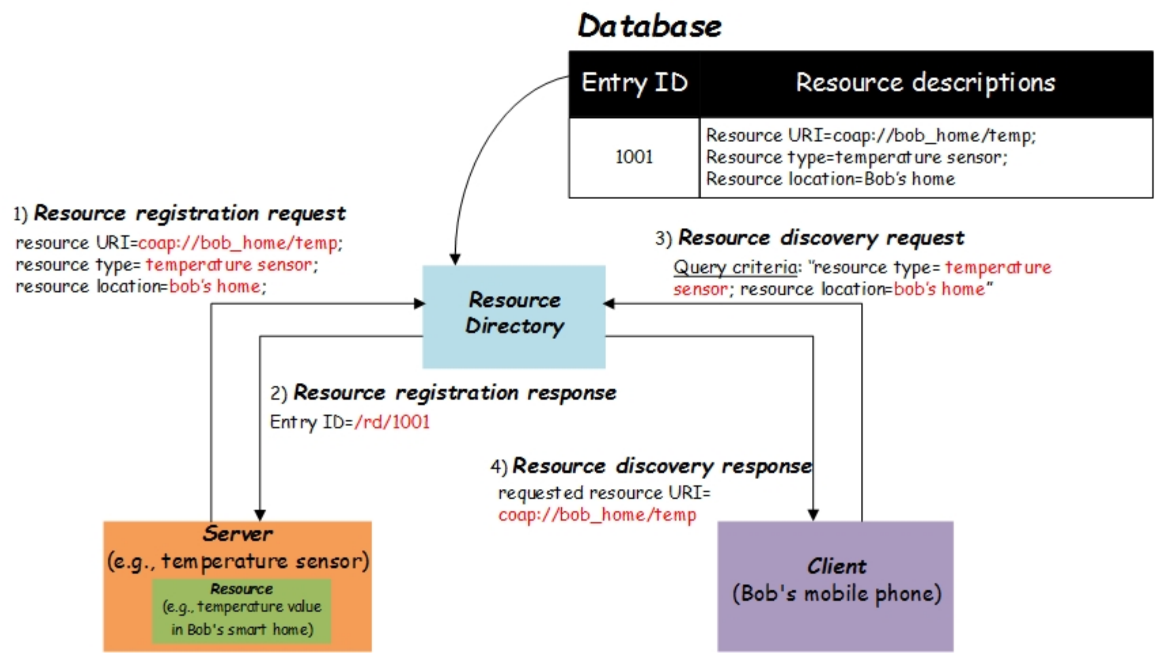
Database
Entry ID
Resource descriptions
1001
Resource URI=coap://bob_home/temp;
Resource type=temperature sensor;
Resource location=Bob's home
1) Resource registration request
resource URI=coap://bob_home/temp;
resource type= temperature sensor;
resource location=bob's home;
3) Resource discovery request
Query criteria: "resource type= temperature sensor; resource location=bob's home"
Resource Directory
2) Resource registration response
Entry ID=/rd/1001
4) Resource discovery response
requested resource URI=
coap://bob_home/temp
Server
(e.g., temperature sensor)
Resource
(e.g., temperature value in Bob's smart home)
Client
(Bob's mobile phone)

Figure 4.4 Interactions among server, client, and RD.

Clients need to know the URIs of the resources before they can send the resource retrieval/observe requests to obtain the contents of the resources. In order to enable clients to discover the URIs of the interested resources, another entity, i.e., Resource Directory (RD), has been proposed [92]. An RD hosts the descriptions of resources and provides the resource lookup functionality to clients. Note that the descriptions of a resource include the URI and the context information (e.g., the resource type and the resource location) of the resource. Figure 4.4 illustrates the interactions among a server, a client, and an RD. First, a server registers its hosting resource to an RD by sending a resource registration request, which includes the descriptions of the resource, to the RD[2]. Second, the RD stores the descriptions of the resource into its database and returns the database entry ID of the resource[3] (e.g., /rd/1001) to

[2]Note that the URI of the RD is well-known or the server can discover the RD by broadcasting/multicasting an RD discover message [92].

[3]The database entry ID of the resource identifies the location of the resource in the RD's database. The server should know this information such that it can update or delete the descriptions of the resource in the RD later on.

the server. Third, a client may send a resource discovery request to discover the URI of a specific resource. Note that the resource discovery request may include a set of query criteria (e.g., resource type='temperature sensor' and location='bob's home') describing the resource that the client wants to discover. As a result, the RD would return the URI(s) of the resource(s), which matches the query criteria.

4.1.3 Caching IoT resources in brokers

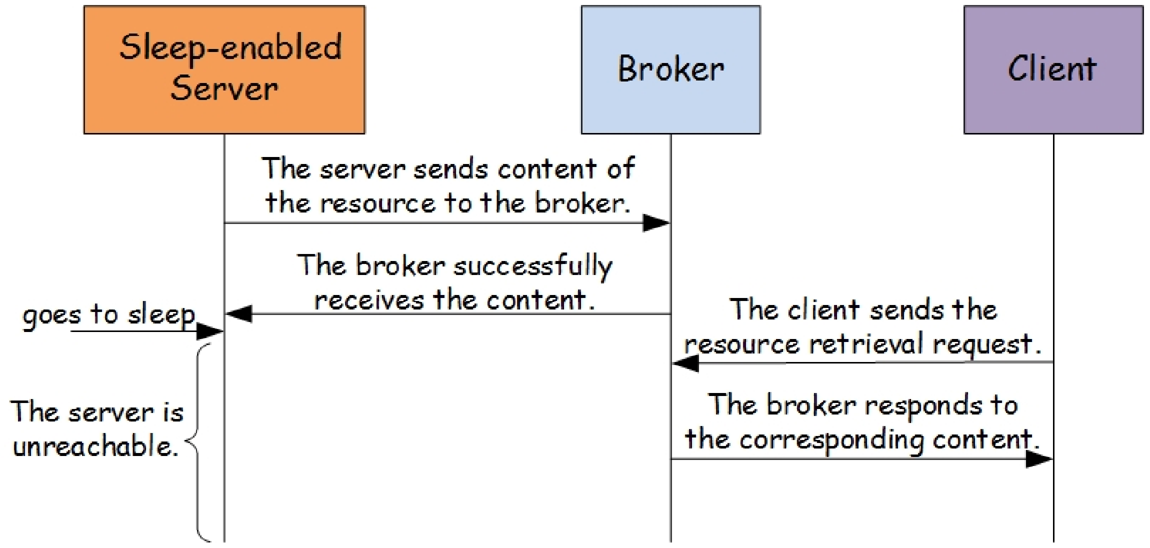
Sleep-enabled Server
Broker
Client
The server sends content of the resource to the broker.
The broker successfully receives the content.
goes to sleep
The client sends the resource retrieval request.
The server is unreachable.
The broker responds to the corresponding content.

Figure 4.5 Interactions among server, client, RD, and broker.

The interactions among servers, clients, and RDs enable the clients to find their interested resources' URIs and obtain the contents of these resources. However, it is not efficient when many clients try to obtain the same resource during a time period, i.e., the server may transmit a huge amount of data to these clients. This situation happens when some popular events take place. For example, when a football game is held in a stadium, tens of thousands of smart cars would look for empty parking spots near the stadium. Thus, each street parking meter may need to transmit a huge amount of data to these smart cars in order to report its parking status (i.e., if the parking spot is empty and when it will be empty if it is currently occupied). Obviously, it is not efficient to enable the street parking meters to transmit the huge amount of data, which may exhaust the network resources and the energy supplies of the street parking meters (which may be powered by batteries). Note that exhausting the network resources of the street parking meters results in increased delay for the street parking meters in transmitting the contents to the clients, and exhausting the energy supplies of the street parking meters disables the street parking meters from sensing and transmitting data.

In order to efficiently deliver the contents of resources, a new entity, i.e., broker, is introduced by the CoAP Publish/Subscribe protocol [93]. A broker is an application layer middleware node equipped with powerful hardwares and sufficient energy supplies. One of the functionalities of the broker is to cache IoT resources. Specifically, a broker can cache a resource hosted by a server, and thus the server would periodically transmit the up-to-date content of the resource to the broker, which consequently helps the server forward the content of the resource to the clients upon requests. Therefore, enabling the broker to deliver the content of the resource has the potential to reduce the traffic loads of the server, which may finally reduce the energy consumption of the server. The communications for the server in enabling the broker to deliver the content of the resource is illustrated in Figure 4.5. 1) The server would first send a resource creation request to the broker to cache a resource. The resource creation request should contain the resource URI (indicating which resource is requested to be cached) as well as the binding table information associated with this resource. 2) The broker would send the URI of corresponding resource cached in the broker back to the server if the broker determines to cache the resource. Note that there are currently two URIs related to this resource: the URI of the resource hosted by the server and the one cached in the broker. 3) After receiving the response from the broker, the server would periodically send the up-to-date content of the resource to the broker. 4) Meanwhile, the server should update the URI of the resource in the RD's database such that the clients can find the URI of the resource cached in the broker. 5) The read clients, who are interested in the resource, can find the URI of the resource via the RD. 6) Consequently, these clients can obtain the up-to-date content of the resource by sending the resource retrieval/observe requests to the broker. 7) The broker sends the content of the resource to the observe clients based on the information in the corresponding binding table.

4.2 Broker Deployment

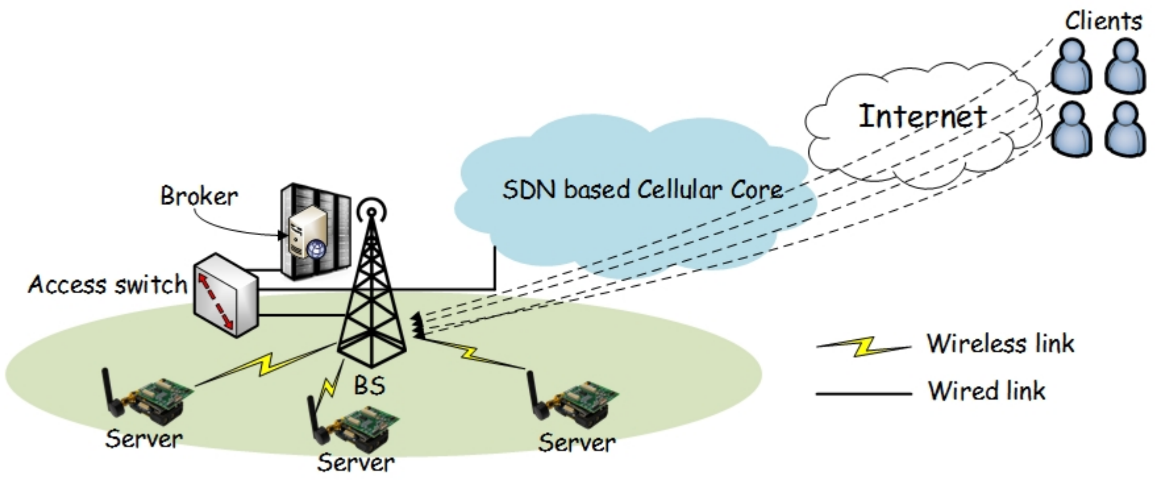

Figure 4.6 Broker deployment in the cloudlet network.

A broker can be a logical entity (i.e., a function embedded in a router or switch) or a physical entity. A good broker deployment enables each server to discover and communicate with at least one specific broker such that this server's hosting resources can be cached in the corresponding broker. Cloudlets in the proposed cloudlet network architecture the suitable entity to host brokers. Specifically, as shown in Figure 4.6, each Base Station (BS), which has already been deployed in the mobile network and provides high radio coverage, is equipped with multiple wireless interfaces (such as Zigbee, bluetooth low energy, and low power area network) such that different servers can communicate with the corresponding BSs. Thus, each BS is considered as a smart gateway to provide various communications interfaces to its local servers.

Meanwhile, each BS is connected to a cloudlet, which can host a broker to conduct IoT resource caching functionality. Thus, each server can actually communicate with the broker via the corresponding BS. In addition, each broker can communicate with the Internet as well as other BSs/brokers via the SDN based cellular core. As mentioned previously, the all the OpenFlow switches are controlled by the SDN controller via the OpenFlow protocol [38] in the SDN based cellular core. The OpenFlow controller manages the forwarding plane of BSs and OpenFlow switches, monitors the traffic at the data plane, and establishes user sessions. The heterogeneous feature of IoT devices (i.e., requiring different QoS, adopting various communications protocols, applying different semantics to represent their data, etc.) hinders the communications among them. The SDN based cellular core can essentially facilitate the communications among IoT devices (i.e., servers, clients, and brokers). Specifically: 1) The SDN controller can dynamically set up suitable forwarding rules in OpenFlow switches, thus provisioning QoS routing to meet the various requirements (e.g., different delay and jitter requirements) of IoT devices in delivering their contents [94–96]. 2) The SDN controller can dynamically add protocol conversion functions [97] in BSs (or in access OpenFlow switches). Thus, IoT devices can communicate with each other even if they use different communications protocols. For instance, if a client, which applies the CoAP protocol, tries to retrieve the content of a resource hosted by a server, which applies the MQTT protocol [98], a function of converting between CoAP and MQTT will be added in the BS (or the access OpenFlow switch), which serves the client/server. 3) The SDN controller can dynamically add semantic capabilities and related ontologies in BSs/OpenFlow switches. Hence, the raw data generated by IoT devices can be converted into semantic data (e.g., RDF[4]), thus improving the interoperability among IoT devices [100].

[4]Resource Description Framework (RDF) is a type of semantic data model that has been widely used in IoT. Applying RDF to annotate IoT data allows different IoT devices to understand the meaning of data. Details of RDF can be found in [99].

4.3 Popular Resource Caching at the IoT Application Layer

In this section, we first illustrate that caching IoT resources may not always reduce the energy consumption of their servers. In order to minimize the energy consumption of servers, we propose to cache IoT resources only if they are popular. We provide a method to measure the popularity of an IoT resource. Table 4.1 summarizes the main notations applied in this chapter.

Table 4.1 List of Important Notations in Resource Caching

Notation	**Definition**
$\mathcal{I}$	Set of all the IoT resources in the network.
$\mathcal{J}$	Set of all the popular IoT resources in the network.
l_i	Average content size of resource i.
λ_i	Average resource retrieval request arrival rate of resource i.
ψ_i	Content update rate of resource i.
η	Predefined threshold to measure the popularity of IoT resources.
x_{jk}	Binary cache indicator between popular resource j and broker k.
λ_k	Average resource retrieval request arrival rate of broker k.
u^b	Average service rate of broker k.
u^s_j	Average service rate of the server, which hosts popular resource j.
t^b	Average delay of the broker in transmitting a content.
t^s_j	Average delay of the server in transmitting a content of resource j.
ΔT	Duration of a time slot.
ϵ_i	Energy coefficient of the server, which hosts resource i.

Essentially, "a resource is cached in the broker" implies that CoAP Pub/Sub is applied to enable the broker to deliver the content of the resource to the clients; otherwise, CoAP is applied to enable the server (which hosts the resource) to deliver *the content of the resource to the clients.* We next provide the smart parking use case to illustrate the pros and cons for caching resources in the broker.

4.3.1 Use case

In smart cities, there are many street parking spots located near a stadium. Each street parking spot is equipped with a smart parking meter, which generates the parking spot status indicating whether the parking spot is empty and when it will become available if it is currently occupied. Assume there is a broker located near the stadium and available for all the street parking meters. If there is a football game hosted in the stadium, as shown in Figure 4.7, tens of thousands of smart cars would look for the available parking spots near the stadium before the game starts. Consequently, each parking meter would receive a large number of resource retrieval requests from smart cars and need to respond to them accordingly. This would tremendously increase the energy consumption of the parking meter, which may be powered by its battery.

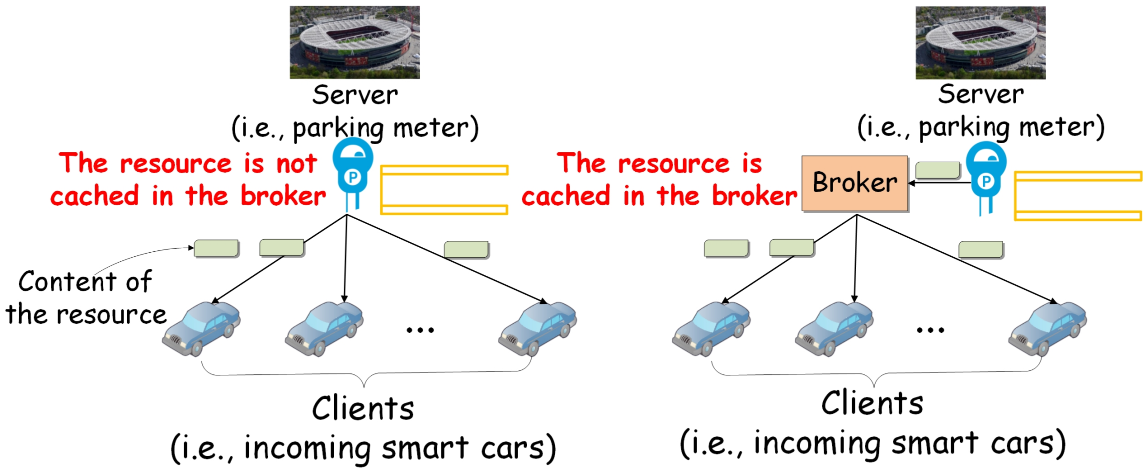
Server
(i.e., parking meter)
The resource is not cached in the broker
Content of the resource
Clients
(i.e., incoming smart cars)
Server
(i.e., parking meter)
The resource is cached in the broker
Broker
Clients
(i.e., incoming smart cars)

Figure 4.7 Use case of a street parking meter nearby a stadium before a football game.

Now, if each parking meter caches its resource in a broker by periodically sending the up-to-date content of the resource to the broker and let the broker respond to

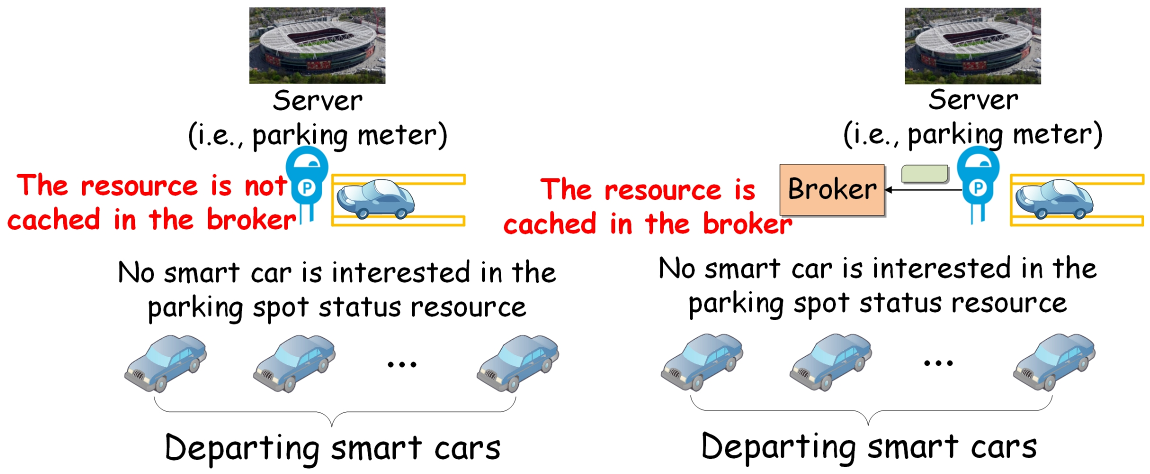
Server
(i.e., parking meter)
The resource is not cached in the broker
No smart car is interested in the parking spot status resource
Departing smart cars
Server
(i.e., parking meter)
The resource is cached in the broker
Broker
No smart car is interested in the parking spot status resource
Departing smart cars

Figure 4.8 Use case of a street parking meter nearby a stadium after a football game.

the resource retrieval requests from the smart cars, the parking meter can save a huge amount of energy by sending less amount of data. The broker normally has abundant power supply and powerful hardware. In other words, the transmission rate of the broker should be much higher than that of a parking meter, and thus caching the parking spot status resources in the broker can reduce the average delay for delivering contents of the resources to the clients.

Caching the resource in the broker may not always be the optimal solution. Consider the case that the broker caches many resources and needs to handle a huge number of resource retrieval requests from the clients; consequently, the average delay for the broker in delivering contents of the resources may be unbearable. Consider another case when the football game finishes, as shown in Figure 4.8; smart cars are no longer interested in the parking spot status resources and some parking spots are still occupied. If these parking meters do not cache their parking spot status resources in the broker, they do not need to transmit any packet since nobody is interested in these resources. Yet, if the parking meters cache their parking spot status resources in the broker, these parking meters need to transmit their up-to-date statuses to the broker. Therefore, without caching the resource in the broker may consume less energy in this scenario. Therefore, caching the resource in the broker by applying

CoAP Pub/Sub may not always benefit the servers. It is necessary to determine whether to cache the resources in the broker or not based on different scenarios.

4.3.2 Definition of popular resources

Caching a resource in a broker cannot always benefit the server (which hosts the resource) because once the resource is cached in the broker, the server needs to periodically send the up-to-date content of the resource to the broker (in order to keep the content of the resource in the broker fresh) even if no client is interested in the resource. On the other hand, if the resource is not cached in the broker (i.e., the server would respond to the resource retrial requests by itself), the server does not need to transmit any data when no client is interested in the resource. Therefore, caching the resource in the broker may not always reduce the traffic load of the server. Note that a heavier traffic load of the server incurs a higher energy consumption of the server. Here, the traffic loads mean the total amount of data needs to be transmitted in response to the resource retrial requests during a time slot. Note that the resource retrieval requests for each resource comprises two parts: 1) the resource retrieval requests from the read clients; 2) the current content satisfies some observe conditions in the binding table, and thus the server/broker would automatically generate the resource retrieval requests in order to deliver current content to the corresponding observe clients.

In order to reduce the energy consumption of servers, only popular resources should be cached in brokers. A resource is considered popular if caching this resource by a broker will result in energy consumption reduction of its hosting server by η during ΔT time period, where $\eta \geq 0$ is the predefined traffic load threshold. Denote $\boldsymbol{\mathcal{I}}$ as the set of the resources in the network, i as the index of the resource, l_i as the average content size of resource i, λ_i as the average arrival rate of resource retrieval request (i.e., the average number of resource retrieval requests per second during a

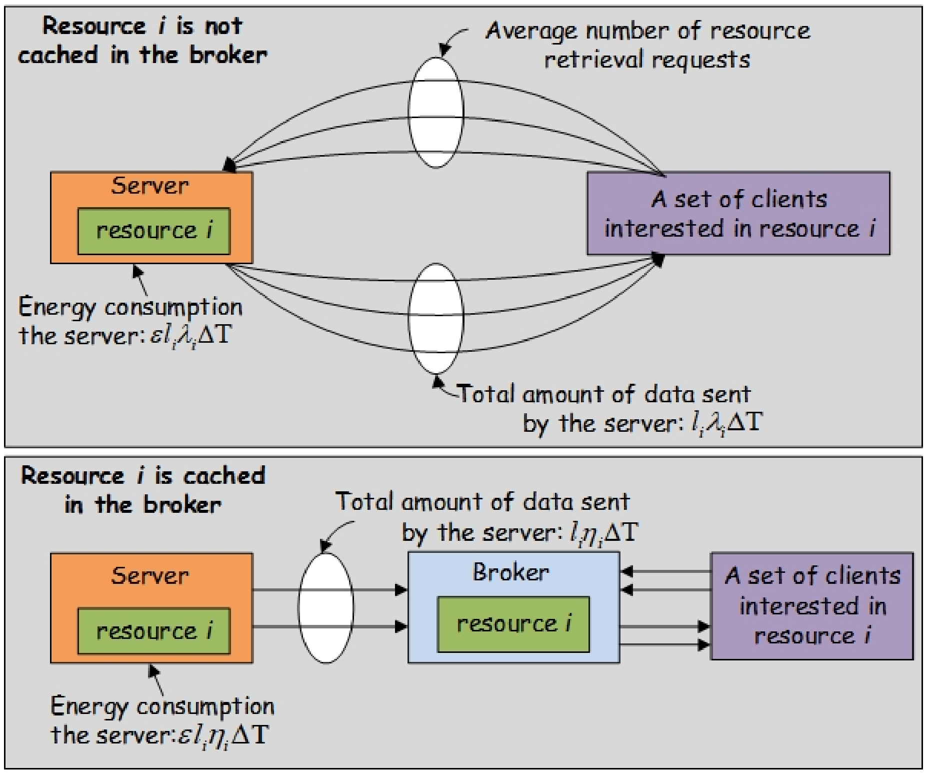
Resource *i* is not cached in the broker
Average number of resource retrieval requests
Server
resource *i*
A set of clients interested in resource *i*
Energy consumption the server: $\varepsilon l_i \lambda_i \Delta T$
Total amount of data sent by the server: $l_i \lambda_i \Delta T$
Resource *i* is cached in the broker
Total amount of data sent by the server: $l_i \eta_i \Delta T$
Server
resource *i*
Broker
resource *i*
A set of clients interested in resource *i*
Energy consumption the server: $\varepsilon l_i \eta_i \Delta T$

Figure 4.9 Amount of data sent by a server by adopting two different methods.

time slot) for resource i, η_i as the average content delivery rate of resource i (i.e., the number of times that the server delivers the up-to-date content of resource i to the broker per second during a time slot) when the resource is cached in the broker, and ΔT as the duration of a time slot.

As shown in Figure 4.9, if resource i is not cached in the broker, the server needs to transmit $l_i \lambda_i \Delta T$ amount of data in order to publish the contents of resource i to the clients; accordingly, the energy consumption of the server is $\epsilon_i l_i \lambda_i \Delta T$, where ϵ_i is the energy coefficient of the server (which hosts resource i), which maps transmitting one bit of data into energy consumption. On the other hand, if resource i is cached in the broker, the server needs to send $l_i \eta_i \Delta T$ amount of data to the broker and let the broker publish the content of resource i to the clients; accordingly, the energy consumption of the server is $\epsilon_i l_i \eta_i \Delta T$. Apparently, if Eq. 4.1 is satisfied, then the server can reduce its energy consumption by η during ΔT time period if resource i hosted by the server is cached in the broker.

$$\epsilon_i l_i \Delta T \left(\lambda_i - \eta_i\right) > \theta. \tag{4.1}$$

Thus, we define resources which satisfy Eq. 4.1 as popular resources (i.e., $\boldsymbol{\mathcal{J}} = \{i | \epsilon_i l_i \Delta T \left(\lambda_i - \eta_i\right) > \theta, i \in \boldsymbol{\mathcal{I}}\}$, where $\boldsymbol{\mathcal{J}}$ denotes as the set of popular resources in the network) and define resources, which do not satisfy Eq. 4.1, as unpopular resources. Note that a server would send a resource caching request to the broker if its resource becomes popular.

4.3.3 Resource caching strategy in brokers

A broker only caches popular resources. If a resource is no longer popular, the broker would notify the server (which hosts the resource) that the resource is no longer suitable to be cached in the broker. Accordingly, the server would deliver the contents of the resource by itself. On the other hand, a broker may be congested if it caches all the popular resources such that the broker needs to transmit a huge volume of data for delivering the cached resources' contents. Consequently, the average delay for enabling the broker to deliver a content of a resource may be longer than the average delay for enabling a server itself to deliver a content of its hosting resource. Therefore, it is beneficial to design a dynamic resource caching mechanism for the broker to determine which popular resource should be cached in the broker in order to maximize the total energy savings from servers, while guaranteeing the average delay.

Average delay for the broker to deliver the content of the popular resource

The broker should estimate the average delay for publishing a popular resource's content to respond to the corresponding resource retrieval request. Denote j as the index of these popular resources. Let x_j to be the binary variable indicating whether popular resource j should be cached in broker (i.e., $x_j = 1$) or not (i.e., $x_j = 0$); thus, $\boldsymbol{\mathcal{X}} = \{x_j | j \in \boldsymbol{\mathcal{J}}\}$ denotes the popular resource caching strategy adopted by the broker. Meanwhile, we assume that the content size of the popular resources, i.e., $\{l_j | j \in \boldsymbol{\mathcal{J}}\}$, follows a Poisson distribution and $\bar{l}$ is the average content size among all the popular resources. Consequently, the service rate of the broker (the average number of resource retrieval requests handled by the broker per second) also follows a Poisson distribution and $\frac{\bar{l}}{u^b}$ is the average service rate of the broker, where u^b is the average transmission rate of the broker. In addition, assume that the resource retrieval request arrivals for each resource exhibits a Poisson distribution and λ_j is the average arrival rate of resource retrieval request for resource j. Then, we can model the broker's publishing contents of the cached popular resources in response to the resource retrieval requests from the clients as an M/M/1 queueing model, and so the average delay (i.e., the average queueing delay plus the average transmission delay) for the broker's publishing a popular resource's content in response to the resource retrieval request can be expressed[5]:

$$t^b = \frac{1}{\frac{u_b}{\bar{l}} - \sum\limits_{j \in \boldsymbol{\mathcal{J}}} \lambda_j x_j}. \tag{4.2}$$

Average delay for the server to deliver the content of the resource The broker estimates the average delay if the content of the popular resource is delivered by the server. If the average delay to deliver the content of the resource by the server is lower than that by the broker, this popular resource is not suitable to be cached in the broker.

[5]The average delay of a broker is the average queueing delay of a resource's content waiting in the broker's network queue plus the average transmission delay of the broker in sending the content out of its network interface. The propagation delay for transmitting the content to the client over the network is not considered.

Assume each server hosts one resource and denote u_j^s as the average transmission rate of the server, which hosts popular resource j. Since the content size of popular resource j, i.e., the value of l_j, is normally fixed over time, the average service rate for this server in handling the resource retrieval requests is deterministic, which is $\frac{u_j^s}{l_j}$. Meanwhile, since the resource retrieval request arrival rate for popular resource j exhibits a Poisson distribution with the average arrival rate of λ_j, we model the server in delivering the content of resource j to the clients as an M/D/1 queueing model, and so the average delay for the server in delivering resource j's content to a client, denoted as t_j^s, can be expressed:

$$t_j^s = \frac{l_j}{2u_j^s} \times \frac{\lambda_j}{\frac{u_j^s}{l_j} - \lambda_j}. \tag{4.3}$$

Problem formulation We formulate the dynamic resource caching problem (i.e., $\boldsymbol{P5}$) in the broker as follows:

$$\boldsymbol{P5}: \qquad \arg\max_{x_j} \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i l_i \Delta T \left(\lambda_i - \eta_i\right) x_j, \tag{4.4}$$

$$s.t. \quad \forall j \in \boldsymbol{\mathcal{J}},\ \frac{x_j}{\frac{u_b}{\bar{l}} - \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \lambda_j x_j} \le t_j^s, \tag{4.5}$$

$$\sum_{j\in\boldsymbol{\mathcal{J}}} \lambda_j x_j < \frac{u_b}{\bar{l}}, \tag{4.6}$$

$$\forall j \in \boldsymbol{\mathcal{J}},\ x_j \in \{0,1\}. \tag{4.7}$$

$\boldsymbol{P5}$ is to maximize the energy savings from servers. Constraint (4.5) is to guarantee $t^b \le t_j^s$ for all the resources cached in the broker. Constraint (4.6) is to ensure the system is stable, i.e., the average arrival rate should be less than the average service rate in the broker to assure the queue does not overflow. Constraint (4.7) imposes x_j to be a binary variable.

Theorem 1. $\boldsymbol{P5}$ *is NP-hard.*

Proof. By combining Constraints (4.5) and (4.6), $\boldsymbol{P5}$ can be transformed into:

$$\begin{aligned}
\boldsymbol{P6}: \qquad \boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}}) = &\arg\max_{x_j} \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i \Delta T l_i \left(\lambda_i - \eta_i\right) x_j, \\
&s.t. \;\; \forall j \in \boldsymbol{\mathcal{J}}, \; \sum_{j\in\boldsymbol{\mathcal{J}}} \lambda_j x_j + \frac{1}{t_j^s} x_j \le \frac{u_b}{\bar{l}}, \qquad (4.8) \\
&\forall j \in \boldsymbol{\mathcal{J}}, \; x_j \in \{0,1\}.
\end{aligned}$$

Consider the case that Constraint (4.8) is equivalent to $\sum\limits_{j\in\boldsymbol{\mathcal{J}}} \left(\sum\limits_{j\in\boldsymbol{\mathcal{J}}} \lambda_j x_j + \frac{1}{t_j^s} x_j \right) \le \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \frac{u_b}{\bar{l}}$, which is transformed into:

$$\sum_{j\in\boldsymbol{\mathcal{J}}} \left(|\boldsymbol{\mathcal{J}}| \lambda_j + \frac{1}{t_j^s} \right) x_j \le \frac{|\boldsymbol{\mathcal{J}}| u^b}{\bar{l}}. \qquad (4.9)$$

$\boldsymbol{P6}$ can be transformed into:

$$\begin{aligned}
\boldsymbol{P7}: \qquad &\arg\max_{x_j} \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i l_i \Delta T \left(\lambda_i - \eta_i\right) x_j, \\
&s.t. \quad \textit{Constraints } (4.7) \textit{ and } (4.9). \qquad (4.10)
\end{aligned}$$

Obviously, $\boldsymbol{P7}$ is the 0-1 knapsack problem, where $\epsilon_i l_i \Delta T \left(\lambda_i - \eta_i\right)$ is the value of item j, $|\boldsymbol{\mathcal{J}}| \lambda_j + \frac{1}{t_j^s}$ is the weight of item j, and $\frac{|\boldsymbol{\mathcal{J}}| u^b}{\bar{l}}$ is the weight capacity of the knapsack. Note that the 0-1 knapsack problem is a well known NP-hard problem. Therefore, we conclude that the 0-1 knapsack problem is reducible to the original dynamic resource caching problem (i.e., $\boldsymbol{P5}$), and so $\boldsymbol{P5}$ is NP-hard. □

Energy Aware and latency guaranteed dynamic reSourcE caching (EASE)

We propose EASE to solve $\boldsymbol{P6}$[6]. Specifically, we relax Constraint (4.8) to construct

[6]As mentioned previously, $\boldsymbol{P6}$ is equivalent to $\boldsymbol{P5}$. Thus, we will try to solve $\boldsymbol{P6}$.

the following Lagrangian problem of $\boldsymbol{P6}$:

$$\begin{aligned}\boldsymbol{\mathcal{L}(\mathcal{W})} = \max_{x_j} \sum_{j\in\boldsymbol{\mathcal{J}}} \left(\epsilon_i l_i \Delta T\left(\lambda_i - \eta_i\right) - \lambda_j \sum_{j\in\boldsymbol{\mathcal{J}}} \omega_j - \frac{\omega_j}{t_j^s} \right) x_j \\ + \left(\frac{u_b}{\overline{l}}\right) \sum_{j\in\boldsymbol{\mathcal{J}}} \omega_j, \end{aligned} \tag{4.11}$$

$$s.t. \quad Constraint\ (4.7),$$

where $\boldsymbol{\mathcal{W}} = \{\omega_j \geq 0 | j \in \boldsymbol{\mathcal{J}}\}$ are the Lagrangian multipliers. Note that the above Lagrangian problem will have the optimal solution $\boldsymbol{\mathcal{X}}^* = \left\{x_j^* | j \in \boldsymbol{\mathcal{J}}\right\}$, where:

$$x_j^* = \begin{cases} 1, & \epsilon_i l_i \Delta T\left(\lambda_i - \eta_i\right) - \lambda_j \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \omega_j - \frac{\omega_j}{t_j^s} > 0. \\ 0, & \epsilon_i l_i \Delta T\left(\lambda_i - \eta_i\right) - \lambda_j \sum\limits_{j\in\boldsymbol{\mathcal{J}}} \omega_j - \frac{\omega_j}{t_j^s} \leq 0. \end{cases} \tag{4.12}$$

Lemma 5. $\boldsymbol{\mathcal{L}(\mathcal{W})}$ *provides an upper bound on* $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}})$.

Proof. Assume $\hat{\boldsymbol{\mathcal{X}}} = \{\hat{x}_j | j \in \boldsymbol{\mathcal{J}}\}$ is a feasible solution of $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}})$. Thus, $\forall j \in \boldsymbol{\mathcal{J}}$, $\sum\limits_{j\in\boldsymbol{\mathcal{J}}} (\lambda_j \hat{x}_j) + \frac{1}{t_j^s}\hat{x}_j - \frac{u_b}{\overline{l}} \leq 0$. Since $\forall j \in \boldsymbol{\mathcal{J}}, \omega_j \geq 0$, we can derive:

$$\sum_{j\in\boldsymbol{\mathcal{J}}} \left(\omega_j \left(\sum_{j\in\boldsymbol{\mathcal{J}}} (\lambda_j \hat{x}_j) + \frac{1}{t_j^s}\hat{x}_j - \frac{u_b}{\overline{l}} \right) \right) \leq 0,$$

i.e.,

$$\begin{aligned} &\sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i l_i \Delta T\left(\lambda_i - \eta_i\right)\hat{x}_j - \sum_{j\in\boldsymbol{\mathcal{J}}} \left(\omega_j \left(\sum_{j\in\boldsymbol{\mathcal{J}}} (\lambda_j \hat{x}_j) + \frac{1}{t_j^s}\hat{x}_j - \frac{u_b}{\overline{l}} \right) \right) \\ &\geq \sum_{j\in\boldsymbol{\mathcal{J}}} \epsilon_i l_i \Delta T\left(\lambda_i - \eta_i\right)\hat{x}_j, \end{aligned}$$

which implies that $\boldsymbol{\mathcal{L}(\mathcal{W})} \geq \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}})$. □

Since $\boldsymbol{\mathcal{L}(\mathcal{W})}$ is an upper bound on $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}})$, the next step is to select suitable values of $\boldsymbol{\mathcal{W}}$ such that the gap between $\boldsymbol{\mathcal{L}(\mathcal{W})}$ and $\boldsymbol{\mathcal{F}}(\boldsymbol{\mathcal{X}})$ is as small as possible. We apply the subgradient method to iteratively select the values of $\boldsymbol{\mathcal{W}}$ in order to find the minimum gap. Specifically, we first select the initial values of $\boldsymbol{\mathcal{W}}$, denoted as $\boldsymbol{\mathcal{W}}^{(0)} = \left\{\omega_j^{(0)} \geq 0 | j \in \boldsymbol{\mathcal{J}}\right\}$. Then, $\boldsymbol{\mathcal{W}}$ are updated in each iteration based on the following equation:

$$\begin{aligned}\forall j \in \boldsymbol{\mathcal{J}}, \omega_j^{(k+1)} &= \omega_j^{(k)} - \alpha^{(k)} \frac{\partial \boldsymbol{\mathcal{L}}(\boldsymbol{\mathcal{W}}^{(k)})}{\partial \omega_j^{(k)}} \\ &= \omega_j^{(k)} + \alpha^{(k)} \left(\sum_{j \in \boldsymbol{\mathcal{J}}} \left(\lambda_j x_j^{*(k)} \right) + \frac{1}{t_j^s} x_j^{*(k)} - \frac{u_b}{l} \right),\end{aligned} \tag{4.13}$$

where $\omega_j^{(k)}$ is the value of ω_j in the k^{th} iteration; $x_j^{*(k)}$, which is calculated based on Equation 4.12, is the optimal solution of the Lagrangian problem in the k^{th} iteration (note that $x_j^{*(k)}$ may not be the feasible solution of $\boldsymbol{P6}$ since $x_j^{*(k)}$ may not satisfy Constraint(4.8); thus, we have to map the optimal solution of the Lagrangian problem into the feasible solution of $\boldsymbol{P6}$, i.e., $\hat{x}_j^{(k)} = \boldsymbol{\mathcal{M}}(x_j^{*(k)})$, where $\hat{x}_j^{(k)}$ is the feasible solution of $\boldsymbol{P6}$ in the k^{th} iteration and $\boldsymbol{\mathcal{M}}(\cdot)$ specifies the mapping); $\alpha^{(k)}$ is the step size adopted in the k^{th} iteration [101]:

$$\alpha^{(k)} = \beta \frac{\boldsymbol{\mathcal{L}}\left(\boldsymbol{\mathcal{W}}^{(k)}\right) - \boldsymbol{\mathcal{F}}^{max}}{\sum\limits_{j \in \boldsymbol{\mathcal{J}}} \left(\sum\limits_{j \in \boldsymbol{\mathcal{J}}} \left(\lambda_j \hat{x}_j^{(k)} \right) + \frac{1}{t_j^s} \hat{x}_j^{(k)} - \frac{u_b}{l} \right)^2}, \tag{4.14}$$

where β is a decreasing adaption parameter with $0 < \beta < 2$ and $\boldsymbol{\mathcal{F}}^{max}$ is the maximum objective value for $\boldsymbol{P6}$ found so far, i.e., $\boldsymbol{\mathcal{F}}^{max} = \max\left\{\boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}^{(1)}), \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}^{(2)}), ..., \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}^{(k)})\right\}$ (where $\hat{\boldsymbol{\mathcal{X}}}^{(k)} = \left\{\hat{x}_j^{(k)} | j \in \boldsymbol{\mathcal{J}}\right\}$ is the feasible solution calculated in the k^{th} iteration and $\boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}^{(k)})$ is the corresponding objective value for $\boldsymbol{P6}$).

The values of $\boldsymbol{\mathcal{W}}$ continue to be updated until the gap between the Lagrangian problem and the maximum objective value for $\boldsymbol{P6}$, i.e., $\boldsymbol{\mathcal{L}}\left(\boldsymbol{\mathcal{W}}^{(k)}\right) - \boldsymbol{\mathcal{F}}^{max}$, does not change over iterations or the number of iterations is equal to or larger than a predefined threshold. EASE is summarized in Algorithm 3.

Algorithm 3 The EASE algorithm

Input:

1. The content size vector of popular resources, i.e., $\boldsymbol{\mathcal{L}} = \{l_j | j \in \boldsymbol{\mathcal{J}}\}$.
2. The average resource retrieval request arrival rate vector for popular resources, i.e., $\boldsymbol{\lambda} = \{\lambda_j | j \in \boldsymbol{\mathcal{J}}\}$.
3. The average transmission rate of the broker, i.e., u^b, and the average transmission rate of the servers (which hosts the popular resources), i.e., $\boldsymbol{\mathcal{U}}^s = \left\{u_j^s | j \in \boldsymbol{\mathcal{J}}\right\}$.

Output: The resource caching vector in the broker, i.e., $\boldsymbol{\mathcal{X}} = \{x_j | j \in \boldsymbol{\mathcal{J}}\}$.

```
1:  Calculate $\boldsymbol{\mathcal{T}}^s = \left\{t_j^s | j \in \boldsymbol{\mathcal{J}}\right\}$ based on Equation 4.3.
2:  Initialize $\boldsymbol{\mathcal{W}} = \mathbf{0}$.
3:  Calculate $\boldsymbol{\mathcal{X}}^* = \left\{x_j^* | j \in \boldsymbol{\mathcal{J}}\right\}$ based on Equation 4.12.
4:  Calculate $\hat{\boldsymbol{\mathcal{X}}} = \{\hat{x}_j | j \in \boldsymbol{\mathcal{J}}\}$, where $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{M}}\left(\boldsymbol{\mathcal{X}}^*\right)$.
5:  Initialize $\boldsymbol{\mathcal{F}}^{max} = \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}})$ and $\boldsymbol{\mathcal{X}} = \hat{\boldsymbol{\mathcal{X}}}$.
6:  Calculate the value of $\boldsymbol{\mathcal{L}}\left(\boldsymbol{\mathcal{W}}\right)$ based on Equation 4.11.
7:  while $\boldsymbol{\mathcal{L}}\left(\boldsymbol{\mathcal{W}}\right) - \boldsymbol{\mathcal{F}}^{max}$ changes over the iterations && The number of iterations
    is less than the threshold do
8:      Update the step size $\alpha$ based on Equation 4.14;
9:      Update the values of $\boldsymbol{\mathcal{W}}$ based on Equation 4.13;
10:     Update the values of $\boldsymbol{\mathcal{X}}^*$ based on Equation 4.12;
11:     Calculate the values of $\hat{\boldsymbol{\mathcal{X}}}$, where $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{M}}\left(\boldsymbol{\mathcal{X}}^*\right)$.
12:     if $\boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}}) > \boldsymbol{\mathcal{F}}^{max}$ then
13:         $\boldsymbol{\mathcal{F}}^{max} = \boldsymbol{\mathcal{F}}(\hat{\boldsymbol{\mathcal{X}}})$ and $\boldsymbol{\mathcal{X}} = \hat{\boldsymbol{\mathcal{X}}}$.
14:     end if
15: end while
16: return $\boldsymbol{\mathcal{X}}$.
```

As mentioned previously, the mapping function $\boldsymbol{\mathcal{M}}(\cdot)$ involved in Algorithm 3 is to convert the optimal solution of the Lagrangian problem (i.e., $\boldsymbol{\mathcal{X}}^* = \left\{x_j^* | j \in \boldsymbol{\mathcal{J}}\right\}$) into the feasible solution of the primal problem (i.e., $\boldsymbol{P1}$) such that Constraint 4.8 is satisfied by all $j \in \boldsymbol{\mathcal{J}}$. The basic idea of the mapping function is that the broker iteratively drops a suitable popular resource (note that the broker dropping a popular resource means the popular resource is not selected to be cached by the broker) among all the popular resources that are currently cached by the broker (which are calculated by Equation 4.12) until Constraint 4.8 is satisfied by all $j \in \boldsymbol{\mathcal{J}}$. The suitable popular resource, denoted as $j^{'}$, is defined as the popular resource that incurs the minimum average delay (for enabling its server to deliver the content of the resource) among all the currently cached popular resources, i.e.,

$$j^{'} = \arg\min_{j} \left\{ t_j^s | x_j^* = 1, j \in \boldsymbol{\mathcal{J}} \right\}. \tag{4.15}$$

The mapping function is summarized in Algorithm 4.

Algorithm 4 The mapping function $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{M}}(\boldsymbol{\mathcal{X}}^*)$.

```
1: Obtain the popular resource set $\boldsymbol{\mathcal{J}}^{'} = \left\{ j | x_j^* = 1, j \in \boldsymbol{\mathcal{J}} \right\}$.
2: Find the suitable resource $j^{'}$ based on Equation 4.15.
3: while $\forall j \in \boldsymbol{\mathcal{J}}, \sum_{j \in \boldsymbol{\mathcal{J}}} \lambda_j x_j^* + \frac{1}{t_j^s} x_j^* \leq \frac{u_b}{l}$ cannot be satisfied do
4:     $x_{j^{'}}^* = 0$;
5:     Remove $j^{'}$ from resource set $\boldsymbol{\mathcal{J}}^{'}$;
6:     Find the suitable resource $j^{'}$ based on Equation 4.15;
7: end while
8: return $\hat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{X}}^*$.
```

4.4 Evaluations

In this section, we will present simulation results to demonstrate the performance of EASE. Consider the scenario with $N = 100$ servers deployed in an area covered by a

Base Station (BS), which is attached to a broker. Each server can communicate with the broker via the BS based on different kinds of communications technologies (e.g., NarrowBand IoT, Bluetooth Low Energy, Zigbee, etc.) and clients are able to retrieve the contents of resources (from the broker or the server) via the BS. Meanwhile, each server hosts one IoT resource and the content size of each resource is generated from a Poisson distribution with mean $\bar{l} = 500\ Kb$. In addition, the average transmission rate of each server is obtained from a Poisson distribution with the average value of 8 $Mbps$. If the resource is cached in the broker, the server should deliver the up-to-date resource content to the broker and we assume that the average content delivery rate is the same among all the resources during a time slot, i.e., $\eta_i = 0.01$ *update/second*[7]. Moreover, the average transmission rate of the broker is $u^b = 350\ Mbps$. The energy coefficients of all the servers in the network are the same, i.e., $\epsilon_i = 1$ *unit/bit*, where $i \in \boldsymbol{\mathcal{I}}$. Clients from the Internet send the resource retrieval requests to retrieve the contents of the resources. The average arrival rate of resource retrieval requests for a resource is randomly selected between 0 and 0.1 *request/second* during a time slot, i.e., $\lambda_i = U(0, 0.1)$. The energy threshold θ is set to be 0 and the duration of a time slot is 10 $mins$.

We compare the performance of EASE with other two baseline strategies, i.e., Caching Preferred (CP) and Caching Non-Preferred (CNP). The basic idea of CP is that each popular resource j ($j \in \boldsymbol{\mathcal{J}}$) is preferred to be cached in the broker until the broker is overflowed[8]. Meanwhile, the intuition of CNP is that each popular resource j is not preferred to be cached in the broker if its server can still handle the corresponding resource retrieval requests (i.e., $\frac{l_j}{u_j^s} > \lambda_j$). Note that if the server

[7]The physical meaning of $\eta_i = 0.01$ *update/second* implies that the server, which hosts resource i, would generate and deliver one updated content within every interval of 100 seconds.

[8]The broker is overflowed means the average utilization of the broker's network card is no less than 1. That is, the average arrival rate of resource retrieval request of the broker is no less than the average service rate of the broker, i.e., $\sum\limits_{i \in \boldsymbol{\mathcal{I}}} \lambda_i x_i \geq \frac{u_b}{\bar{l}}$.

cannot handle the resource retrieval requests (i.e., $\frac{l_j}{u_j^s} \leq \lambda_j$), then this resource has to be cached in the broker in CNP. Note that each server would send the resource caching request to the broker as long as its resource becomes popular in each time slot and the broker would determine whether to cache these popular resources by running the three caching strategies in each time slot. The total simulation period is 100 time slots and the Monte Carlo results are generated to measure the performance of the three caching strategies.

4.4.1 Overall performance

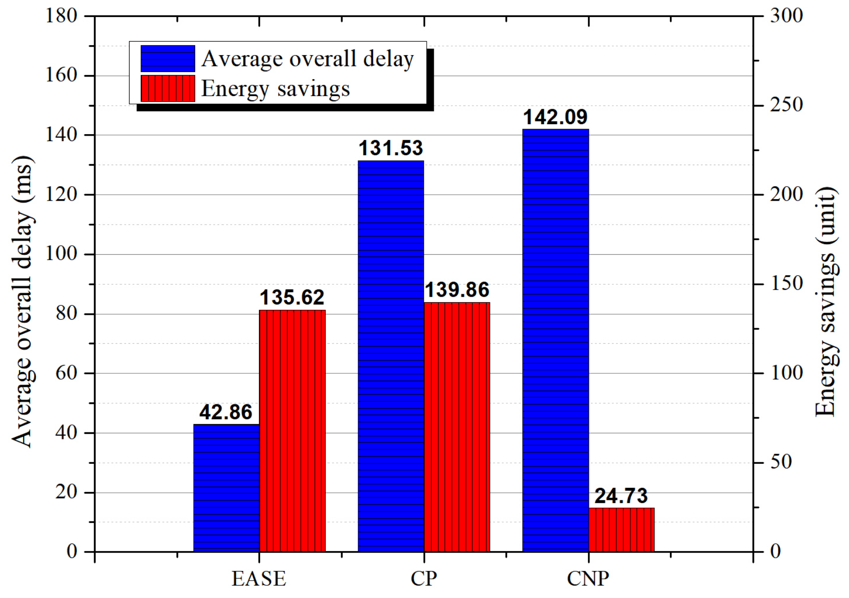

Figure 4.10 Average overall delay and the amount of energy savings.

As shown in Figure 4.10, EASE and CP save similar amount of energy from servers, which is much more than the one saved by CNP. This is because CNP does not prefer to cache popular resources in the broker, and so servers need to transmit the contents of their hosting resources by themselves, which cannot saves energy of the servers. Figure 4.11 shows the number of popular resources that are cached in the broker by applying different strategies. CNP has the fewest number of resources

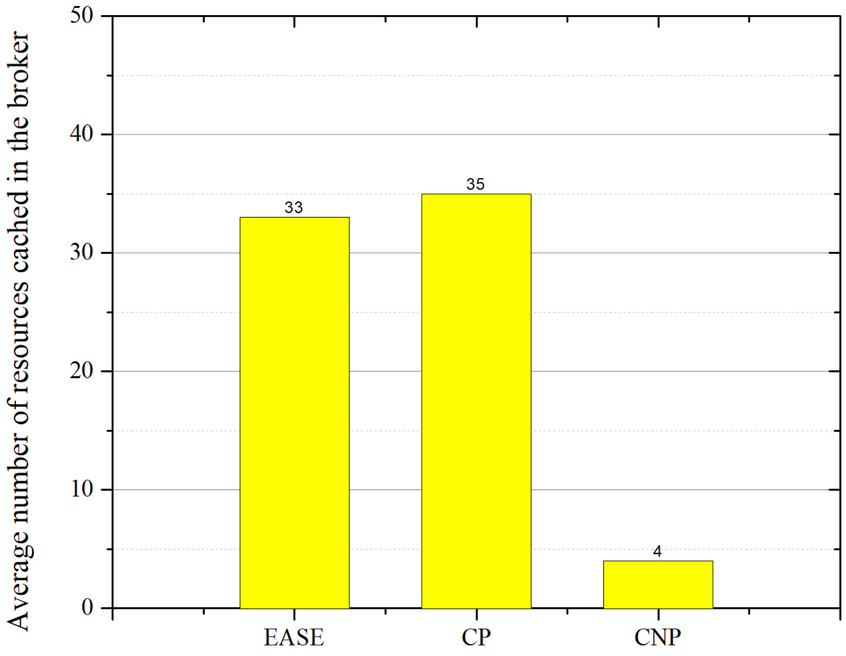

Figure 4.11 Average number of resources cached in the broker.

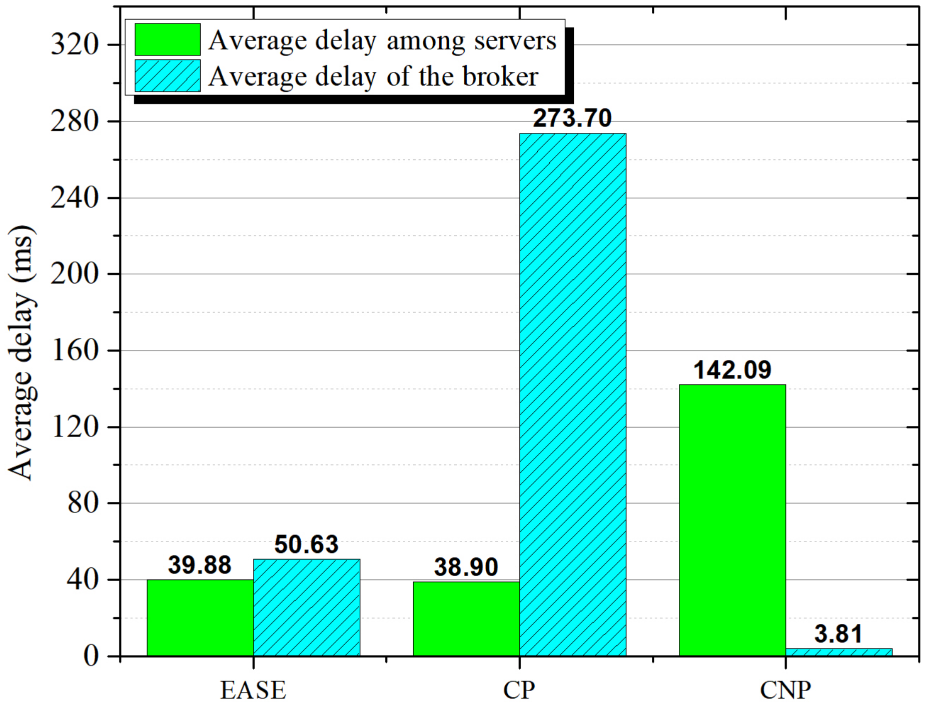

Figure 4.12 Average delay among servers and Average delay of the broker.

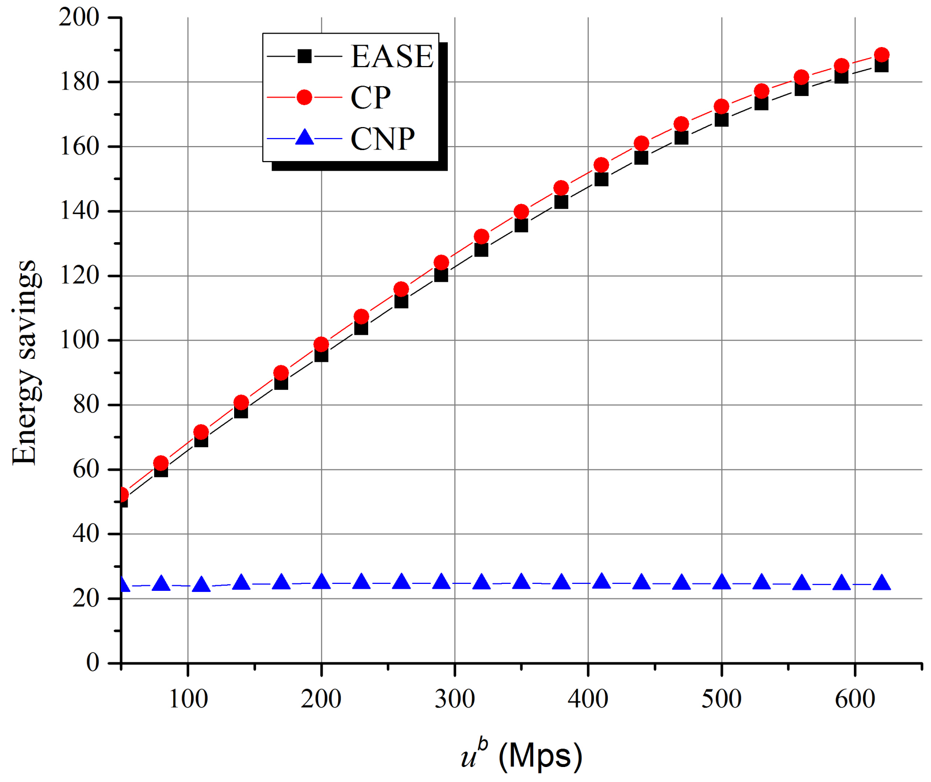

Figure 4.13 Amount of energy savings.

cached in the broker than CP and EASE; this is why CNP saves less energy from servers than CP and EASE.

As shown in Figure 4.10, although EASE and CP generate similar energy savings, EASE incurs much lower *average overall delay*[9] for publishing a content of a resource as compared to CP. To explain this, we analyze the *average delay among the*

[9]Average overall delay is the mean of the average delay of all the resources, i.e., *average overall delay* $= \sum_{i \in \mathcal{I}} \left(t^b x_i + t_i^s (1 - x_i) \right) / |\mathcal{I}|$, where $t^b x_i + t_i^s (1 - x_i)$ refers to the average delay for delivering a content of resource i and $|\mathcal{I}|$ is the total number of the resources.

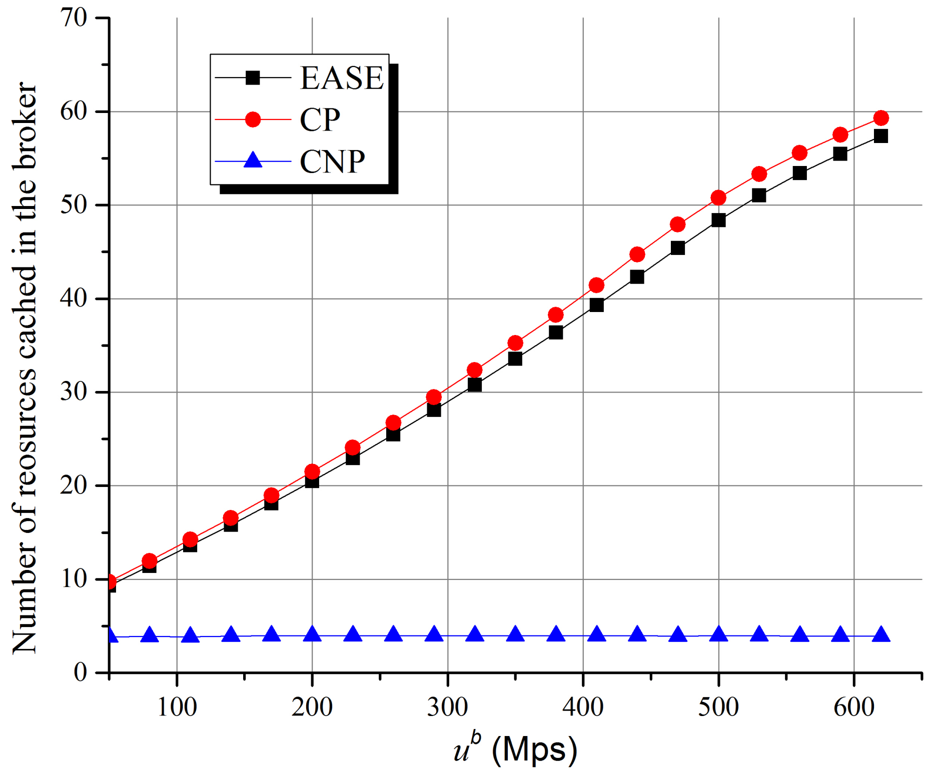

Figure 4.14 Number of cached resources.

servers[10] and the *average delay of the broker*[11] with respect to the three strategies. As shown in Figure 4.12, CP incurs much higher average delay of the broker than EASE and CNP because CP only tries to maximize the energy savings by enabling the brokers to cache as many popular IoT reosources as possible. This would result in the broker being congested, thus incurring high average delay of the broker in

[10]Average delay among the servers indicates the mean of the average delay of all the resources, whose contents are delivered by their servers (i.e., the resources that are not cached by the broker), i.e., *average delay among servers* $= \sum\limits_{i\in\mathcal{I}} t_i^s(1-x_i)/\sum\limits_{i\in\mathcal{I}}(1-x_i)$, where $t_i^s(1-x_i)$ refers to the average delay for the server in delivering a content of resource i, which is not cached by the broker, and $\sum\limits_{i\in\mathcal{I}}(1-x_i)$ is the total number of the resources not being cached by the broker.

[11]The average delay of the broker is the mean of the average delay of all the resources, whose contents are delivered by the broker (i.e., the resources that are cached by the broker), i.e., *average delay of the broker* $= \sum\limits_{i\in\mathcal{I}} t^b x_i/\sum\limits_{i\in\mathcal{I}} x_i$, where $t^b x_i$ refers to the average delay for the broker in delivering a content of resource i, which is cached by the broker, and $\sum\limits_{i\in\mathcal{I}} x_i$ is the total number of the resources being cached by the broker.

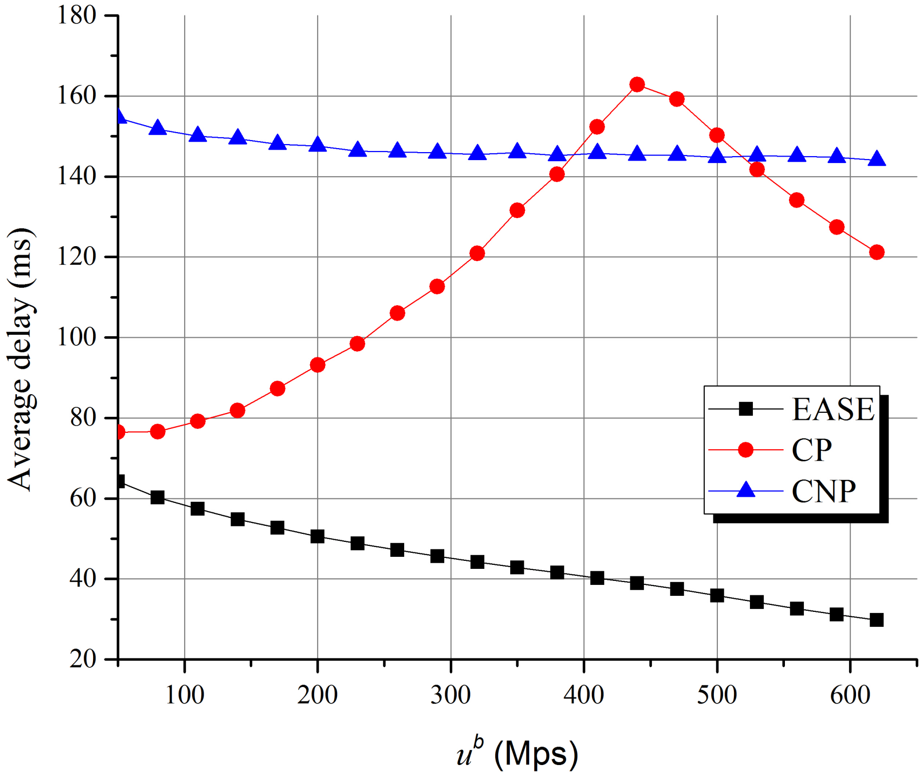

Figure 4.15 Average overall delay.

delivering the content of a resource. In other words, the broker needs to handle too many resource retrieval requests such that the average delay for the broker in delivering a content of a resource increases significantly. Accordingly, high average delay of the broker causes high average overall delay for CP. On the other hand, although CNP incurs the lowest average delay of the broker (since a few number of popular resources are cached in the broker, and thus the broker is lightly loaded), it generates the highest average delay among servers as compared to EASE and CP. This is because some servers may be over-loaded to handle many resource retrieval requests, thus suffering from high delay. As a result, high average delay among servers causes high average overall delay for CNP. Therefore, we conclude that EASE can determine suitable resource to be cached in the broker in order to save almost the same amount of energy from servers as compared to CP while ensuring the lowest average overall delay.

4.4.2 Performance comparison by varying the average transmission rate of the broker

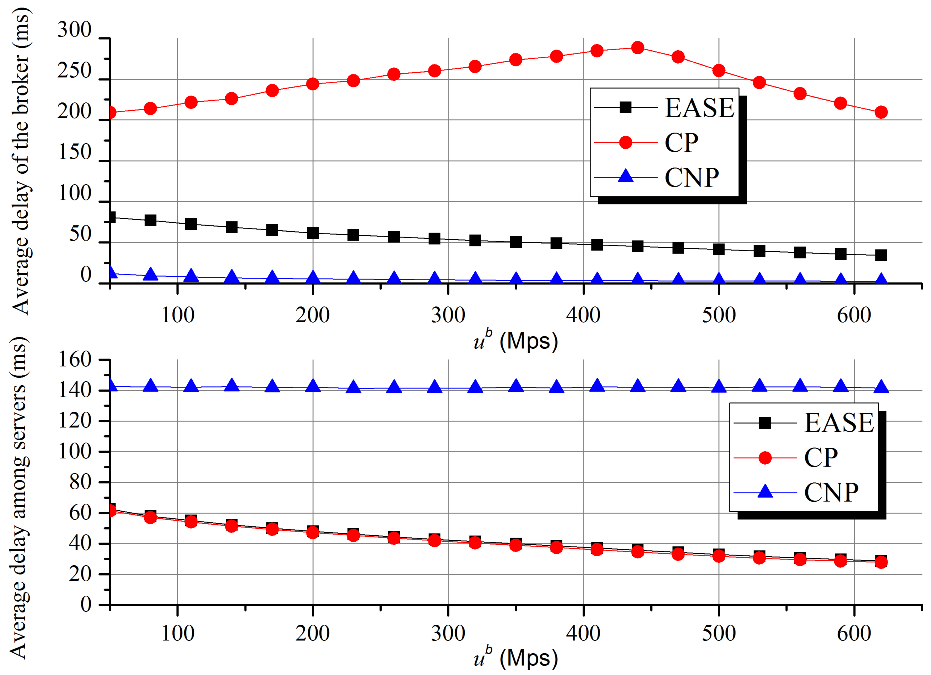

Figure 4.16 Average delay among servers and average delay of the broker.

We further compare the performance of the three strategies by changing the average transmission rate of the broker, i.e., the value of u^b. As shown in Figure 4.13, EASE and CP always incur the similar amount of energy savings; meanwhile, the amount of energy savings increases as u^b increases. This is because when u^b increases, the broker caches more popular resources by applying EASE and CP, thus potentially reducing the energy consumption of the servers. Figure 4.14 demonstrates that the number of resources cached in the broker increases as u^b increases when EASE and CP are applied. However, as shown in Figure 4.14, the amount of energy savings incurred by CNP does not change as u^b varies because CNP does not prefer to cache the resources in the broker even if the broker has a larger capacity to cache more popular resources.

Figure 4.15 shows the average overall delay by applying the three strategies. The average overall delay of EASE decreases as u^b increases and EASE always outperforms

CP and CNP. Note that the average overall delay of CP is monotonically increasing when $u^b < 440$ $Mbps$ and monotonically decreasing when $u^b \geq 440$ $Mbps$ because as u^b increases, CP would cache more popular resources in the broker, and so the broker needs to handle more resource retrieval requests from the clients. Thus, as shown in Figure 4.16, the average delay of the broker for applying CP is increasing as u^b increases; this increases the average overall delay of CP accordingly. When $u^b \geq 440$ $Mbps$, all the popular resources have already been cached in the broker, i.e., the average arrival rate of resource requests of the broker would not increase as u^b increases. Consequently, as shown in Figure 4.16, the average delay of the broker for applying CP decreases, thus resulting in the decrease of the average overall delay of CP.

CHAPTER 5

FUTURE WORK

The cloudlet network has been proposed in Chapter 2 to provision mobile edge computing. In order to maintain low E2E delay between MUs and their Avatars, the adaptive Avatar handoff process has been proposed in Chapter 3 to hand off Avatars among cloudlets based on the movements of their MUs. Caching popular IoT resources in the context of the cloudlet network has been proposed in Chapter 4 to reduce the energy consumption of IoT devices and speed up the content delivery process. In this chapter, we will briefly discuss how to further enhance the IoT content delivery process in the context of the cloudlet network and how to increase the throughput of the mobile access network.

5.1 Traffic Load Balancing among Brokers in the Cloudlet Network

As mentioned in Chapter 4, brokers are hosted by cloudlets, which are distributed at the mobile edge; meanwhile, caching popular IoT resources in the nearby brokers can substantially reduce the energy consumption of servers. However, caching popular IoT resources in the nearby brokers is not the optimal solution, i.e., the traffic loads among brokers are not balanced (some brokers are heavily loaded, but others are lightly loaded), and thus the delay of delivering the contents of IoT resources is not minimized [87]. For example, as shown in Figure 5.1, broker-1 caches two popular resources, i.e., resource-1 and resource-2, and broker-2 caches one popular resource, i.e., resource-3. Thus, broker-1 needs to respond to $2n$ and $3n$ resource retrieval requests related to resource-1 and resource-2 during a time slot, respectively. Meanwhile, broker-2 needs to respond to n resource retrieval requests related to resource-3 during a time slot. If the sizes of these resources' contents are the same, broker-1 would transmit more data to the clients than broker-2 would, i.e., traffic loads are unbalanced between the two brokers. The unbalanced traffic loads may significantly increase the average delay of delivering the contents of popular resources to clients.

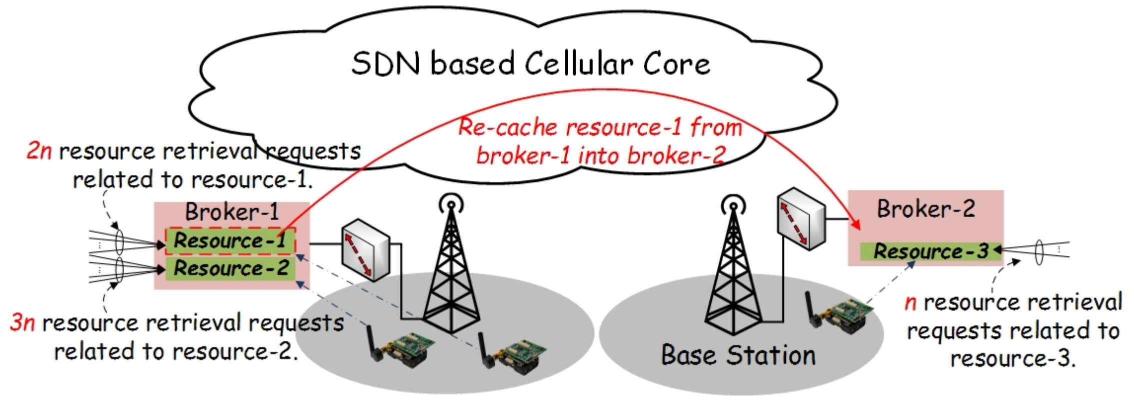
SDN based Cellular Core
Re-cache resource-1 from broker-1 into broker-2
2n resource retrieval requests related to resource-1.
Broker-1
Resource-1
Resource-2
3n resource retrieval requests related to resource-2.
Broker-2
Resource-3
n resource retrieval requests related to resource-3.
Base Station

Figure 5.1 Illustration of unbalanced traffic loads among brokers and the resource re-caching process.

In order to reduce the average delay, traffic loads can be offloaded from heavily loaded brokers to lightly loaded brokers. For example, as shown in Figure 5.1, broker-1 can offload its traffic loads to broker-2 by enabling broker-2 to cache resource-2 (such that broker-2 should take the responsibility to respond to the resource retrieval requests related to resource-2). Here, we define the process of a popular resource, which is originally cached by one broker, to be cached by another broker as resource re-caching. Essentially, balancing the traffic loads among brokers is implemented by resource re-caching.

Although re-caching resources from heavily loaded brokers to lightly loaded brokers can reduce the average delay to distribute the contents, re-caching resources may generate extra communications overheads, which may potentially increase the energy consumption of the server as well as the traffic load of the SDN based cellular core. Figure 5.2 shows the procedure of a popular resource being re-cached from Broker-1 into Broker-2:

- Step-1: once the popular resource is determined to be cached by Broker-2, Broker-1 should send a resource creation request to Broker-2. The resource creation request includes the URI of the popular resource hosted by the server as well as the binding table of the popular resource. Note that transmitting the binding table to Broker-2 is to inform Broker-2 about the observe clients (which are defined in Section III.A) of the popular resource such that Broker-2 can continue to transmit the up-to-date contents of the popular resource to those observe clients.

- Step-2: once Broker-2 receives the resource creation request, it would store the binding table, create a new URI, which identifies the popular resource cached in Broker-2, and return this new URI to Broker-1.

- Step-3: Broker-1 needs to inform the server about the new URI (which points to the popular resource in Broker-2) by sending a notification message to the server such that the server can send the up-to-date contents to Broker-2 (rather than Broker-1).

- Step-4: after receiving the notification message, the server needs to respond with a confirmation message to Broker-1.

- Step-5: after receiving the new URI, the server should update the URI of the popular resource in the RD by sending a resource update request to the RD such that the read clients can find the new URI, which points to the popular resource in Broker-2.

- Step-6: after receiving the resource update request, the RD needs to send to the server with a resource update response in order to confirm that the URI has been updated.

Extra messages are generated by the two brokers in Step-1, Step-2, and Step-3. Denote ζ as the amount of communications overheads incurred from Step-2 to Step-3. Note that the value of ζ is the same among different popular resources, i.e., no matter which popular resource is re-cached, the total amount of communications overheads incurred from Step-2 to Step-3 is the same. The value of ζ depends on the size of the resource creation response message (in Step-2) and the notification message (in Step-3). Denote ω_j as the amount of data incurred by Step-1. The value of ω_j depends on the size of the binding table of resource j, which is further determined by the number of observe clients of resource j, i.e., more observe clients of resource

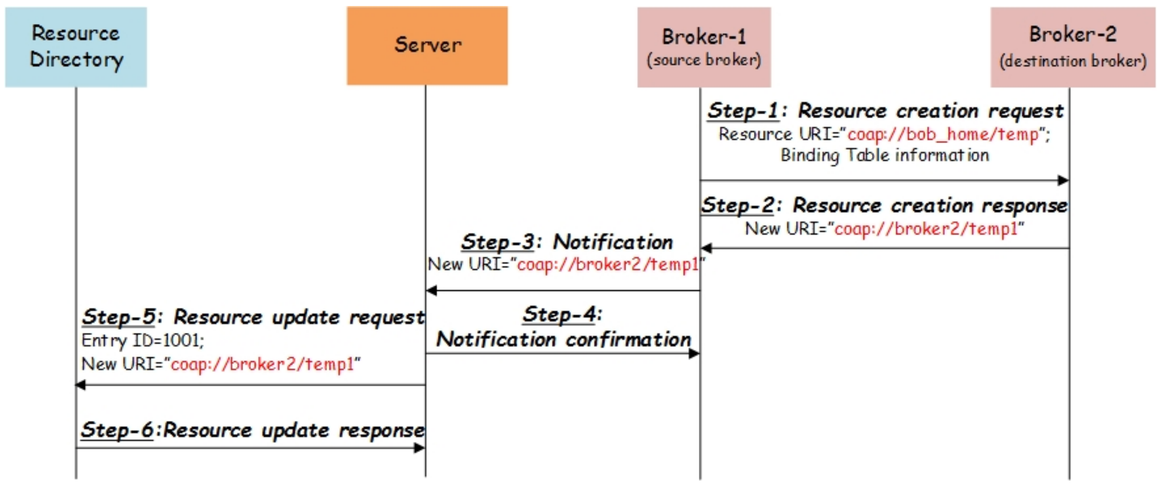
Resource Directory
Server
Broker-1 (source broker)
Broker-2 (destination broker)
Step-1: Resource creation request
Resource URI="coap://bob_home/temp";
Binding Table information
Step-2: Resource creation response
New URI="coap://broker2/temp1"
Step-3: Notification
New URI="coap://broker2/temp1"
Step-4: Notification confirmation
Step-5: Resource update request
Entry ID=1001;
New URI="coap://broker2/temp1"
Step-6:Resource update response

Figure 5.2 Illustration of communications overheads incurred by popular resource re-caching.

j result in a larger size of the binding table of resource j, thus increasing the value of ω_j. Hence, we have $\omega_j = a + bn_j$, where n_j is the number of the observe clients of resource j, b is the coefficient that maps the number of the observe clients into communications overheads, and a is the offset of communications overheads. Note that the values of a and b can be obtained by measuring the size of the resource creation request (in Step-1) under different numbers of the observe clients. Thus, we can derive the total amount of communications overheads by re-caching popular resource j as follows:

$$o_j = a + bn_j + \zeta. \tag{5.1}$$

On the other hand, extra messages are generated by the server (which originally hosts the resource that is re-cached by a different broker) in Step-4 and Step-5. Thus, the total amount of communications overheads incurred by the server is determined by the size of the notification confirmation message (in Step-4) and resource update request (in Step-5).

All in all, re-caching popular IoT resources from heavily loaded brokers to lightly loaded brokers can reduce the average delay to distribute the contents; however, re-caching a popular IoT resource from one broker into another broker may incur non-negligible communications overheads, which may increase the energy consumption of the server as well as the traffic load of the SDN based cellular core. Therefore, it is necessary to design an efficient IoT resource re-caching method to determine the location of each IoT resource (i.e., each IoT resource is cached by which broker in the cloudlet network) such that both the energy consumption of servers and the average delay of delivering the contents of IoT resources are minimized.

5.2 Drone Aided Mobile Access Networks

Global mobile data traffic is increasing dramatically over the years. According to Cisco Visual Networking Index [102], global mobile data traffic reached 7.2 exabytes per month at the end of 2016. Monthly global mobile data traffic will reach 49 exabytes by 2021, i.e., the amount of data traffic in 2021 will be almost seven times that of in 2016. On the other hand, the average mobile network connection speeds cannot catch up with the growth of mobile traffic. Table 5.1 lists the average mobile network connection speeds among different areas, where in North America, the mobile network speed in 2021 (25.2 Mbps) is less than two times that of in 2016 (13.7 Mbps). Thus, we can conclude that mobile access networks could be the bottleneck of the entire system. This may degrade the performance of mobile edge computing.

In order to increase the throughput of mobile access networks, the current solution is to deploy a massive number of small cells (e.g., femto cells and pico cells) in the network. It is reported that hundreds of thousands of small cells are expected to be deployed across the U.S. by 2020 to support the exponential growth of the traffic demands. The number of small cell sites could surpass traditional wireless towers (macro base stations) by 2019, possibly numbering approximately 455,000 by 2020, assuming speedy siting procedures [103]. Deploying a massive number of small cells is a way to increase the throughput of mobile access networks, but not an efficient one.

Table 5.1 Projected Average Mobile Network Connection Speeds (Mbps)

	2016	2017	2018	2019	2020	2021
AsiaPacific	9.8	10.6	12.9	16	18.8	20.4
LatinAmerica	3.8	4.9	6.4	7.9	10	12.4
North America	13.7	16.3	17.6	19.8	22.8	25.2
WesternEurope	11.4	16	18.6	21.6	25.7	28.5
CentralandEastern Europe	6.3	10.1	12.3	13.6	16.2	18.4
Middle EastandAfrica	3.8	4.4	5.3	6.8	8.5	10.8

Source: [102].

This is because 1) deploying a small cell, which costs from $15,000 to $20,000 [104], is not cheap; 2) small cells are statically placed to cover some possible hotspots. However, some hotspots appear unpredictably and occasionally, thus hampering these small cells from covering these hotspots efficiently. A new hotspot might arise, for example, after an accident owing to an auto accident, when MUs begin to stress the access point by downloading and watching related news content. A stadium would become a hotspot when a football game or a concert is underway; 3) small cells are normally deployed near the ground, and thus the links between MUs and small cells are likely to be non-light-of-sight (NLOS), which may reduce the mobile access network speed.

To address this problem, Drone-mounted Base Stations (DBSs) have been developed to help accelerate traffic delivery to MUs [105–107]. A drone is a low-cost unmanned aerial vehicle designed to be flown under remote control or autonomously using embedded software and sensors (e.g., GPS) [108]. Drones have been deployed for many applications, such as public safety [109], rescue missions [110], and reconnaissance over disaster recovery [111]. A DBS is essentially a movable small cell that can be automatically and flexibly deployed in any hotspot to assist the access node (e.g., the macro BS) to deliver traffic to MUs in the hotspot. Specifically, as shown in Figure 5.3, a DBS hovering over a hotspot can download data from the access node via the wireless backhaul link and relay them to the MUs in the hotspot at a higher data rate than that achievable when the MUs are downloading data directly from the access node [106].

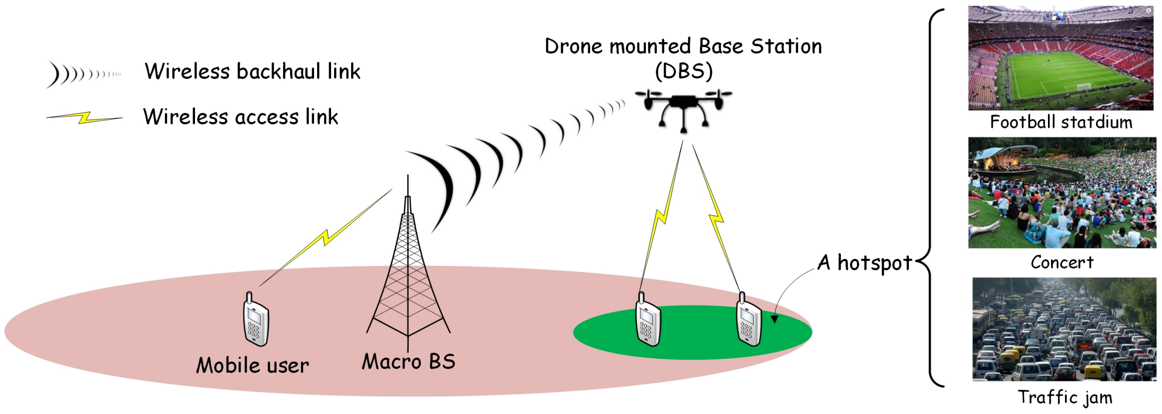

Figure 5.3 DBS aided mobile access network architecture.

Different from traditional small cells (whose locations are normally fixed), DBSs have been identified with their unique advantages, i.e., faster and cheaper to deploy, more flexibly reconfigured, and likely to have better communications channels owing to the presence of short-range Line-of-Sight (LoS) links [112–114]. These features enable DBSs to be quickly deployed to the random hotspots in the network to increase the data rate of the MUs in the hotspot areas.

Although deploying DBS to assist mobile access networks can speed up the macro base station in delivering traffic to MUs and provide flexibility to mobile network providers, some challenges are still existed.

1. **Optimal DBS placement**
   Essentially, the optimal DBS placement is to determine the longitude, latitude, and altitude of a DBS such that the overall throughput of the mobile access network is maximized, which depends on the throughput of both wireless access link as well as wireless backhaul link. The optimal DBS placement problem is difficult to be solved because the DBS placement may affect the throughput of both wireless access link and wireless backhaul link.

DBS placement impacts on the throughput of the mobile access link: traffic demands in the network exhibit temporal and spatial dynamics. Deploying a DBS over an area with high traffic demands (i.e., a hotspot) may significantly speed up traffic delivery from an access node to the MUs in the area. However, many hotspots may coexist in an access node's coverage area. Some can be served by the access node, but some cannot. Thus, it is critical to optimize the latitude and longitude of the DBS in order to maximize the throughput of the mobile access network. In addition, the altitude of the DBS also affects the throughput of the mobile access link. Specifically, the DBS at lower altitude reduces the distance between the DBS and the MUs, thus potentially reduce the path loss[1] from the DBS to the MUs; on the other hand, since the wireless access channel between the DBS and the MUs is modeled by a probabilistic LoS channel [115–118], the DBS with lower altitude increases the probability of None LoS (NLoS) to the MUs, thus potentially increasing the path loss from the DBS to the MUs.

DBS placement impacts on the throughput of the wireless backhaul link: the data rate of the wireless backhaul link (between the macro base station and the DBS) is a function of the distance between the DBS and the macro base station, i.e., the DBS placed closer to the macro base station would incur higher throughput of the wireless backhaul link, and vice versa.

2. **Limited battery life**

   A DBS is normally powered by its portable battery which limits the aloft time of the DBS. The DJI PHANTOM 4, for example, can only fly approximate 28 minutes [119] and the Yuneec Q500 has the maximum flight time of 25 minutes [120]. The short battery life of a drone cannot keep a DBS aloft continuously to assist the access node in delivering flash crowd traffic. Simply increasing the battery capacity of a drone cannot significantly increase the flight time because the weight of the drone also increases accordingly [121]. Therefore, the drone has to be charged when its battery is drained.

   Harvesting energy from the sun is an option for charging a drone without landing [122–125]. However, the solar energy harvesting efficiency heavily depends on the sky condition [126], i.e., a drone cannot prolong its flight time by harvesting energy from solar if the sky is covered by clouds. Rossi *et al.* [125] concluded that harvesting energy from solar can increase the flight time of a drone up

[1]Note that lower path loss between the DBS to the MUs results in higher throughput.

to 20% in the best case scenario. Meanwhile, a drone can also be charged wirelessly, which is more convenient than ground charging [121,127,128]. That is, the charging station on the ground may transfer energy to the drone through radio waves or microwaves while the drone is hovering in the air. For example, Gmez-Tornero *et al.* [129] proposed to use a frequency agile coaxial magnetron as the power source (with 70 kW peak power) to transfer power carried at Ku-band (in a 200 MHz range from 16.15 GHz to 16.35 GHz), with only 21.3 W of power reaching a drone at a 25 *meter* distance.

3. **Logistic challenges**
   Federal Aviation Administration (FAA) has already made rules of how to operate drones in the real world [130]. For example, a drone should be no more than 55 pounds; a drone must stay with operators line of sight; a drone cannot fly over 400 ft, etc. These rules actually hamper the usage of DBSs to be applied in mobile access networks. Definitely, regulations are needed when flying a drone. But, too much is too much. We could image a drone could be much safer than any vehicle in the future and some rules are not necessary when flying a drone.

CHAPTER 6

CONCLUSION

The MCC technology is used to offload the workloads from mobile devices to the cloud, and so mobile devices only need to conduct some simple tasks; on the other hand, the remote cloud would help the mobile devices execute offloaded tasks and return related results to the mobile devices. MCC can not only reduce energy consumption of mobile devices but also accelerate the execution time of the applications. However, owing to the long E2E delay between mobile devices and the cloud, offloading the workloads of many interactive mobile applications (e.g., augmented reality, virtual reality, online gaming, etc.) to the cloud may not be suitable. In order to reduce the E2E delay, a novel cloudlet network architecture has been proposed to bring the computing and storage resources from the remote cloud to the mobile edge. In the proposed cloudlet network, each BS is attached to a cloudlet, which comprises a number interconnected physical machines providing computing and storage resources to mobile devices. Each MU is associated with a specific Avatar in the nearby cloudlet. Thus, MUs can offloaded their workloads to their Avatars with low E2E delay. Note that the proposed cloudlet network architecture not only facilitates MUs in offloading their workloads to their Avatars but also benefits other applications, such as mobile crowd sensing and mobile big data analysis. In particular, each Avatar can process mobile data streams from its MU, extract the high-level knowledge information hidden behind the mobile data streams, and send the knowledge information to the corresponding application VM, which provides related services to MUs. Three use cases, i.e., Terrorist Detection, ParkNet, and FaceDate, have been given to demonstrate how the cloudlet network architecture facilitates mobile crowd sensing and mobile big data analysis.

MUs may roam among BSs in the mobile network and so the E2E delay between MUs and their Avatars may become worse if the Avatars remain in their original cloudlets. Thus, the idea of handing off a MU's Avatar when the mobile user roams away has been proposed to reduce the E2E ddelay between the MU and its Avatar. In order to reduce the average Avatar handoff time, placing each Avatar's replicas into suitable cloudlets has been proposed. The LEARN algorithm has been designed to determine the locations of each Avatar's replicas in order to minimize the average E2E between an MU and its Avatar during a time period (i.e., one day). After each Avatar's replicas have been deployed into their suitable cloudlets, the LatEncy aware Avatar handDoff (LEAD) algorithm has been designed to determine the location of each MU's Avatar in each time slot (e.g., 30 minutes) in order to minimize the average E2E delay among all the MUs and their Avatars. The performance of LEAD has been demonstrated via extensive simulations.

The cloudlet network architecture not only facilitates mobile users in offloading their computational tasks but also empowers Internet of Things (IoT). The idea of caching popular IoT resources in brokers has been proposed to reduce the energy consumption of the servers, which host these popular IoT resources. Here, the brokers are considered as application layer middleware nodes hosted by cloudlets in the cloudlet network, and servers can always communicate with nearby brokers via their associated BSs. In order to minimize both the total energy consumption of servers and the average delay of delivering contents of IoT resources, the EASE algorithm has been designed to enable each broker to cache suitable popular IoT resources. The performance of EASE has been demonstrated via extensive simulations.

Two future works have been presented. First, caching popular IoT resources in the nearby brokers has been identified to result in unbalanced traffic load among the brokers in the cloudlet network. The unbalanced traffic load may significantly increase the average delay of delivering contents of IoT resources to clients. How to balance the traffic load among brokers has been investigated to accelerate the IoT content delivery process. Second, mobile access networks may be the bottleneck of the whole system, and thus degrade the performance of mobile edge computing. In order to increase the throughput of the mobile access networks, the drone aided mobile access network architecture has been proposed. In addition, three challenges and the potential solutions in the context of the drone aided mobile access networks have been discussed.

BIBLIOGRAPHY

[1] P. Jonsson, S. Carson, J. S. Sethi, M. Arvedson, R. Svenningsson, P. Lindberg, K. hman, and P. Hedlund, "Ericsson mobility report november 2017." [Online]. Available: https://www.ericsson.com/assets/local/mobility-report/documents/2017/ericsson-mobility-report-june-2017.pdf (accessed on April 12, 2018).

[2] comScore, "The 2017 U.S. mobile app report." [Online]. Available: https://www.comscore.com/Insights/Presentations-and-Whitepapers/2017/The-2017-US-Mobile-App-Report (accessed on April 12, 2018).

[3] International Data Corporation, "Consumerscape 360: Mobile purchase drivers purchase." [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=US41731716 (accessed on April 12, 2018).

[4] X. Sun and N. Ansari, "Energy-optimized bandwidth allocation strategy for mobile cloud computing in LTE networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, March 2015, pp. 2120–2125.

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[6] X. Sun, N. Ansari, and R. Wang, "Optimizing resource utilization of a data center," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2822–2846, Fourthquarter 2016.

[7] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in *the 8th ACM International Conference on Mobile Systems, Applications, and Services*, San Francisco, CA, June 2010, pp. 49–62.

[8] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *the 6th ACM conference on Computer Systems*, Salzburg, Austria, April 2011, pp. 301–314.

[9] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *the 31st IEEE International Conference on Computer Communications (INFOCOM)*, Orlando, FL, March 2012, pp. 945–953.

[10] Y. Li and W. Wang, “Can mobile cloudlets support mobile applications?” in *the 33rd IEEE Conference on Computer Communications (INFOCOM)*, Toronto, ON, April 2014, pp. 1060–1068.

[11] I. Farris, R. Girau, L. Militano, M. Nitti, L. Atzori, A. Iera, and G. Morabito, “Social virtual objects in the edge cloud,” *IEEE Cloud Computing*, vol. 2, no. 6, pp. 20–28, Nov 2015.

[12] M. Bguena, G. Samaras, A. Pamboris, M. L. Sichitiu, P. Pietzuch, and P. Manzoni, “Towards enabling hyper-responsive mobile apps through network edge assistance,” in *the 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Las Vegas, NV, Jan 2016, pp. 399–404.

[13] L. Yang, J. Cao, G. Liang, and X. Han, “Cost aware service placement and load dispatching in mobile cloud systems,” *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1440–1452, May 2016.

[14] R. Yao, T. Heath, A. Davies, T. Forsyth, M. Mitchell, and P. Hoberman, “Oculus VR best practices guide v 0.008.” [Online]. Available: http://static.oculusvr.com/sdk-downloads/documents/OculusBestPractices.pdf (accessed on April 12, 2018).

[15] Huawei, “Huawei whitepaper on vr-ar-open file.” [Online]. Available: http://www.huawei.com/minisite/hwmbbf16/insights/HUAWEI-WHITEPAPER-VR-AR-Final.pdf (accessed on April 12, 2018).

[16] M. Dick, O. Wellnitz, and L. Wolf, “Analysis of factors affecting players’ performance and perception in multiplayer games,” in *the 4th ACM Special Interest Group on Data Communication (SIGCOMM) Workshop on Network and System Support for Games*, Hawthorne, NY, October 2005, pp. 1–7.

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

[18] W. Hu, B. Amos, Z. Chen, K. Ha, W. Richter, P. Pillai, B. Gilbert, J. Harkes, and M. Satyanarayanan, “The case for offload shaping,” in *the 16th International Workshop on Mobile Computing Systems and Applications*, Santa Fe, NM, February 2015, pp. 51–56.

[19] Z. Chen, L. Jiang, W. Hu, K. Ha, B. Amos, P. Pillai, A. Hauptmann, and M. Satyanarayanan, “Early implementation experience with wearable cognitive assistance applications,” in *the 13th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys) Workshop on Wearable Systems and Applications*, Florence, Italy, May 2015, pp. 33–38.

[20] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen, P. Pillai, and M. Satyanarayanan, "Quantifying the impact of edge computing on mobile applications," in *the 7th ACM Special Interest Group on Operating Systems (SIGOPS) Asia-Pacific Workshop on Systems*, Hong Kong, Hong Kong, August 2016, pp. 5:1–5:8.

[21] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky, D. Siewiorek, and M. Satyanarayanan, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *the 2nd ACM/IEEE Symposium on Edge Computing*, October 2017, pp. 14:1–14:14.

[22] W. Hu, Z. Feng, Z. Chen, J. Harkes, P. Pillai, and M. Satyanarayanan, "Live synthesis of vehicle-sourced data over 4g lte," in *the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems*, Miami, FL, November 2017, pp. 161–170.

[23] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: at the leading edge of mobile-cloud convergence," in *6th International Conference on Mobile Computing, Applications and Services*, Austin, TX, November 2014, pp. 1–9.

[24] Nokia, "Nokia multi-access edge computing (MEC) solutions." [Online]. Available: https://www.faa.gov/uas/media/Part_107_Summary.pdf (accessed on April 12, 2018).

[25] V. Bahl, "Emergence of micro datacenter (cloudlets/edges) for mobile computing." [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/11/Micro-Data-Centers-mDCs-for-Mobile-Computing-1.pdf (accessed on April 12, 2018).

[26] X. Sun and N. Ansari, "Cloudlet networks: Empowering mobile networks with computing capabilities," *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) Communications Frontiers*, vol. 12, no. 4, pp. 6–12, July 2017.

[27] Q. Fan, N. Ansari, and X. Sun, "Energy driven avatar migration in green cloudlet networks," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1601–1604, July 2017.

[28] C. Borcea, X. Ding, N. Gehani, R. Curtmola, M. A. Khan, and H. Debnath, "Avatar: Mobile distributed computing in the cloud," in *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, San Francisco, CA, March 2015, pp. 151–156.

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

[30] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A virtual machine-based platform for trusted computing," *ACM Special Interest Group on Operating Systems (SIGOPS) Operating Systems Review*, vol. 37, no. 5, pp. 193–206, October 2003.

[31] S. Ouellette, L. Marchand, and S. Pierre, "A potential evolution of the policy and charging control/QoS architecture for the 3GPP IETF-based evolved packet core," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 231–239, May 2011.

[32] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E. D. Schmidt, "A virtual SDN-enabled LTE EPC architecture: A case study for s-/p-gateways functions," in *IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, Italy, November 2013, pp. 1–7.

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

[34] ——, "Green cloudlet network: A sustainable platform for mobile cloud computing," *IEEE Transactions on Cloud Computing*, doi: 10.1109/TCC.2017.2764463, early access.

[35] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and flexible cellular core network architecture," in *the 9th ACM conference on Emerging Networking Experiments and Technologies*, Santa Barbara, CA, December 2013, pp. 163–174.

[36] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, December 2016.

[37] ——, "PRIMAL: Profit maximization avatar placement for mobile edge computing," in *IEEE International Conference on Communications (ICC)*, Kuala Lumpur,Malaysia, May 2016, pp. 1–6.

[38] Open Networking Foundation, "Openflow switch specification v 1.5.1." [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (accessed on April 12, 2018).

[39] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "Parknet: drive-by sensing of road-side parking statistics," in *the 8th International Conference on Mobile Systems, Applications, and Services*, San Francisco, CA, June 2010, pp. 123–136.

[40] P. Neog, H. Debnath, J. Shan, N. R. Paiker, N. Gehani, R. Curtmola, X. Ding, and C. Borcea, "FaceDate: a mobile cloud computing app for people matching," in *the 11th EAI International Conference on Body Area Networks*, Turin, Italy, December 2016, pp. 184–190.

[41] W. Zhang, K. T. Lam, and C. L. Wang, "Adaptive live VM migration over a WAN: Modeling and implementation," in *the 7th IEEE International Conference on Cloud Computing (CLOUD)*, Anchorage, AK, June 2014, pp. 368–375.

[42] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the man/wan," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 901–907, October 2006.

[43] T. Wood, K. K. Ramakrishnan, P. Shenoy, J. V. der Merwe, J. Hwang, G. Liu, and L. Chaufournier, "Cloudnet: Dynamic pooling of cloud resources by live wan migration of virtual machines," *IEEE/ACM Transactions on Networking*, vol. 23, no. 5, pp. 1568–1583, October 2015.

[44] U. Deshpande and K. Keahey, "Traffic-sensitive live migration of virtual machines," in *the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, Shenzhen, China, May 2015, pp. 51–60.

[45] A. Strunk and W. Dargie, "Does live migration of virtual machines cost energy?" in *the 27th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Barcelona, Spain, March 2013, pp. 514–521.

[46] A. Strunk, "A lightweight model for estimating energy cost of live migration of virtual machines," in *the 6th IEEE International Conference on Cloud Computing*, Santa Clara, CA, June 2013, pp. 510–517.

[47] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in *the 20th ACM International Symposium on High Performance Distributed Computing*, San Jose, California, USA, June 2011, pp. 171–182.

[48] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, and M. Satyanarayanan, "Adaptive VM handoff across cloudlets," *Technical Report CMU-C S-15–113, CMU School of Computer Science*, June 2015.

[49] X. Sun and N. Ansari, "Adaptive avatar handoff in the cloudlet network," *IEEE Transactions on Cloud Computing*, doi: 10.1109/TCC.2017.2701794, early access.

[50] ——, "Energy-optimized bandwidth allocation strategy for mobile cloud computing in LTE networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, 2015, pp. 2120–2125.

[51] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, September 2013.

[52] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, June 2012.

[53] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with YCSB," in *the 1st ACM Symposium on Cloud Computing*, Indianapolis, IN, June 2010, pp. 143–154.

[54] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *the 7th ACM Symposium on Operating Systems Design and Implementation*, Seattle, WA, November 2006, pp. 307–320.

[55] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *the 26th IEEE Symposium on Mass Storage Systems and Technologies (MSST)*, Incline Villiage, NV, May 2010, pp. 1–10.

[56] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *the 17th ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, August 2011, pp. 1082–1090.

[57] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction in wireless networks," in *the 21st Century IEEE Military Communications Conference (MILCOM)*, Los Angeles, CA, October 2000, pp. 491–495.

[58] A. Nadembega, A. Hafid, and T. Taleb, "A destination and mobility path prediction scheme for mobile networks," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2577–2590, June 2015.

[59] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow software-defined networks," in *IEEE Network Operations and Management Symposium (NOMS)*, Krakow, Poland, May 2014, pp. 1–8.

[60] C. Yu, C. Lumezanu, A. Sharma, Q. Xu, G. Jiang, and H. V. Madhyastha, "Software-defined latency monitoring in data center networks," in *16th International Conference on Passive and Active Network Measurement*, New York, NY, March 2015, pp. 360–372.

[61] G. Laporte, S. Nickel, and F. S. da Gama, *Location science*. New York City, NY: Springer, 2015.

[62] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems II: The p-medians," *SIAM Journal on Applied Mathematics*, vol. 37, no. 3, pp. 539–560, December 1979.

[63] M. S. Daskin, *Network and discrete location: models, algorithms, and applications*. Hoboken, NJ: John Wiley & Sons, 2011.

[64] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical Programming*, vol. 6, no. 1, pp. 62–88, December 1974.

[65] M. Held and R. M. Karp, "The traveling-salesman problem and minimum spanning trees: Part II," *Mathematical Programming*, vol. 1, no. 1, pp. 6–25, December 1971.

[66] R. Landa, J. T. Araújo, R. G. Clegg, E. Mykoniati, D. Griffin, and M. Rio, "The large-scale geography of internet round trip times," in *IFIP Networking Conference, 2013*, Brooklyn, NY, May 2013, pp. 1–9.

[67] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the Internet of Things: perspectives and challenges," *Wireless Networks*, vol. 20, no. 8, pp. 2481–2501, November 2014.

[68] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[69] X. Guo, L. Chu, and X. Sun, "Accurate localization of multiple sources using semidefinite programming based on incomplete range matrix," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5319–5324, July 2016.

[70] S. Vural, N. Wang, P. Navaratnam, and R. Tafazolli, "Caching transient data in internet content routers," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1048–1061, April 2017.

[71] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of Internet-of-Things data," in *IEEE International Conference on Communications (ICC)*, Sydney, Australia, June 2014, pp. 3185–3190.

[72] M. A. Hail, M. Amadeo, A. Molinaro, and S. Fischer, “Caching in named data networking for the wireless Internet of Things,” in *International Conference on Recent Advances in Internet of Things (RIoT)*, Singapore, April 2015, pp. 1–6.

[73] E. Baccelli, C. Mehlis, O. Hahm, T. C. Schmidt, and M. Wählisch, “Information centric networking in the IoT: Experiments with NDN in the wild,” in *the 1st ACM Conference on Information-Centric Networking*, Paris, France, September 2014, pp. 77–86.

[74] M. Amadeo, C. Campolo, J. Quevedo, D. Corujo, A. Molinaro, A. Iera, R. L. Aguiar, and A. V. Vasilakos, “Information-centric networking for the Internet of Things: Challenges and opportunities,” *IEEE Network*, vol. 30, no. 2, pp. 92–100, March 2016.

[75] ICN Research Group, “Requirements and challenges for IoT over ICN.” [Online]. Available: https://tools.ietf.org/pdf/draft-zhang-icnrg-icniot-requirements-01.pdf (accessed on April 12, 2018).

[76] ——, “Proposed design choices for IoT over information centric networking.” [Online]. Available: https://tools.ietf.org/pdf/draft-lindgren-icnrg-designchoices-00.pdf (accessed on April 12, 2018).

[77] X. Sun and N. Ansari, “Dynamic resource caching in the IoT application layer for smart cities,” *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 606–613, April 2018.

[78] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, “Internet of Things: Vision, applications and research challenges,” *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, September 2012.

[79] H. Khedher, H. Afifi, and H. Moustafa, “Optimal placement algorithm (OPA) for IoT over ICN,” in *IEEE International Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Atlanta, GA, May 2017, pp. 372–377.

[80] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, “Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution,” *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, April 2013.

[81] D. Niyato, D. I. Kim, P. Wang, and L. Song, “A novel caching mechanism for Internet of Things (IoT) sensing service with energy harvesting,” in *IEEE International Conference on Communications (ICC)*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

[82] R. Li, H. Asaeda, and J. Li, “A distributed publisher-driven secure data sharing scheme for information-centric IoT,” *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 791–803, June 2017.

[83] M. Ha and D. Kim, “On-demand cache placement protocol for content delivery sensor networks,” in *International Conference on Computing, Networking and Communications (ICNC)*, Santa Clara, CA, January 2017, pp. 207–216.

[84] J. Quevedo, D. Corujo, and R. Aguiar, “Consumer driven information freshness approach for content centric networking,” in *IEEE International Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Toronto, ON, April 2014, pp. 482–487.

[85] A. Lindgren, F. B. Abdesslem, B. Ahlgren, O. Scheln, and A. M. Malik, “Design choices for the IoT in information-centric networks,” in *the 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Las Vegas, NV, January 2016, pp. 882–888.

[86] A. Rao, O. Schelén, and A. Lindgren, “Performance implications for iot over information centric networks,” in *the 11th ACM Workshop on Challenged Networks*, New York City, NY, October 2016, pp. 57–62.

[87] X. Sun and N. Ansari, “Traffic load balancing among brokers at the iot application layer,” *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 489–502, March 2018.

[88] Internet Engineering Task Force (IETF), “The constrained application protocol (coap).” [Online]. Available: https://tools.ietf.org/html/rfc7252 (accessed on April 12, 2018).

[89] Network Working Group , “RESTful design for Internet of Things systems.” [Online]. Available: https://tools.ietf.org/html/draft-keranen-t2trg-rest-iot-03 (accessed on April 12, 2018).

[90] M. Nitti, V. Pilloni, G. Colistra, and L. Atzori, “The virtual object as a major element of the internet of things: A survey,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1228–1240, Secondquarter 2016.

[91] CoRE Working Group, “Dynamic resource linking for constrained restful environments.” [Online]. Available: https://tools.ietf.org/html/draft-ietf-core-dynlink-01 (accessed on April 12, 2018).

[92] ——, “Core resource directory.” [Online]. Available: https://tools.ietf.org/html/draft-ietf-core-resource-directory-09 (accessed on April 12, 2018).

[93] Network Working Group, “Publish-subscribe broker for the constrained application protocol (coap).” [Online]. Available: https://tools.ietf.org/html/draft-koster-core-coap-pubsub-05 (accessed on April 12, 2018).

[94] N. Bizanis and F. A. Kuipers, "SDN and virtualization solutions for the Internet of Things: A survey," *IEEE Access*, vol. 4, pp. 5591–5606, 2016.

[95] C. Lin, K. Wang, and G. Deng, "A QoS-aware routing in SDN hybrid networks," *Procedia Computer Science*, vol. 110, pp. 242–249, July 2017.

[96] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the Internet of Things," in *IEEE Network Operations and Management Symposium (NOMS)*, Krakow, Poland, May 2014, pp. 1–9.

[97] R. Narayanan and C. S. R. Murthy, "A probabilistic framework for protocol conversions in IIoT networks with heterogeneous gateways," *IEEE Communications Letters*, vol. 21, no. 11, pp. 2456–2459, November 2017.

[98] OASIS, "Mqtt v3.1.1." [Online]. Available: http://mqtt.org (accessed on April 12, 2018).

[99] RDF Working Group, "Resource description framework (RDF)." [Online]. Available: https://www.w3.org/RDF/ (accessed on April 12, 2018).

[100] N. Ansari and X. Sun, "Mobile edge computing empowers Internet of Things," *IEICE Transactions on Communications*, vol. E101.B, no. 3, pp. 604–619, 2018.

[101] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM Journal on Control and Optimization*, vol. 30, no. 4, pp. 838–855, July 1992.

[102] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 20162021 white paper." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html (accessed on April 12, 2018).

[103] L. Beyoud, "Small cell fight carries big stakes for Sprint ATT." [Online]. Available: https://www.bna.com/small-cell-fight-n57982086205/ (accessed on April 12, 2018).

[104] The Future Of Wireless Infrastructure In The United States. [Online]. Available: http://www.digitalbridgellc.com/press/future-wireless-infrastructure-united-states (accessed on April 12, 2018).

[105] I. Bor-Yaliniz and H. Yanikomeroglu, "The new frontier in RAN heterogeneity: Multi-tier drone-cells," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 48–55, November 2016.

[106] P. Yang, X. Cao, C. Yin, Z. Xiao, X. Xi, and D. Wu, "Proactive drone-cell deployment: Overload relief for a cellular network under flash crowd traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2877–2892, October 2017.

[107] X. Sun and N. Ansari, "Latency aware drone base station placement in heterogeneous networks," in *IEEE Global Communications Conference (GLOBECOM)*, Singapore, December 2017, pp. 1–6.

[108] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "Intrusion detection and ejection framework against lethal attacks in UAV-aided networks: A bayesian game-theoretic methodology," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1143–1153, May 2017.

[109] D. He, S. Chan, and M. Guizani, "Drone-assisted public safety networks: The security aspect," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 218–223, April 2017.

[110] J. Manathara, P. B. Sujit, and R. Beard, "Multiple UAV coalitions for a search and prosecute mission," *Journal of Intelligent & Robotic Systems*, vol. 32, no. 1, pp. 125–158, June 2011.

[111] I. Maza, F. Caballero, J. Capitan, J. R. Martinez-De-Dios, and A. Ollero, "Experimental results in multi-UAV coordination for disaster management and civil security applications," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1, pp. 563–585, December 2010.

[112] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, May 2016.

[113] J. Lyu, Y. Zeng, and R. Zhang, "Cyclical multiple access in uav-aided communications: A throughput-delay tradeoff," *IEEE Wireless Communications Letters*, vol. 5, no. 6, pp. 600–603, December 2016.

[114] M. M. Azari, F. Rosas, K. C. Chen, and S. Pollin, "Optimal UAV positioning for terrestrial-aerial communication in presence of fading," in *IEEE Global Communications Conference (GLOBECOM)*, Washington, DC, December 2016, pp. 1–7.

[115] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs," *IEEE Transactions on Wireless Communications*, vol. 15, no. 6, pp. 3949–3963, June 2016.

[116] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, December 2014.

[117] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1647–1650, August 2016.

[118] E. Kalantari, M. Z. Shakir, H. Yanikomeroglu, and A. Yongacoglu, "Backhaul-aware robust 3D drone placement in 5G+ wireless networks," in *IEEE International Conference on Communications Workshops (ICC Workshops)*, Paris, France, May 2017, pp. 109–114.

[119] DJI, "Phantom 4 specs." [Online]. Available: https://www.dji.com/phantom-4/info#specs (accessed on April 12, 2018).

[120] Yuneec, "Yuneec q500 specifications." [Online]. Available: https://yuneec-forum.com/threads/yuneec-q500-specifications.7/ (accessed on April 12, 2018).

[121] M. Simic, C. Bil, and V. Vojisavljevic, "Investigation in wireless power transmission for UAV charging," *Procedia Computer Science*, vol. 60, pp. 1846–1855, 2015.

[122] Altadevices, "Solar power for small UAVs." [Online]. Available: https://www.altadevices.com/wp-content/uploads/2016/11/uav-app-brief.pdf (accessed on April 12, 2018).

[123] J.-K. Shiau, D.-M. Ma, P.-Y. Yang, G.-F. Wang, and J. H. Gong, "Design of a solar power management system for an experimental UAV," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 4, pp. 1350 – 1360, October 2009.

[124] A. Alsharoa, H. Ghazzai, A. Kadri, and A. E. Kamal, "Energy management in cellular HetNets assisted by solar powered drone small cells," in *IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, CA, March 2017, pp. 1–6.

[125] M. Rossi and D. Brunelli, "Autonomous gas detection and mapping with unmanned aerial vehicles," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 4, pp. 765–775, April 2016.

[126] N. Sharma, J. Gummeson, D. Irwin, and P. Shenoy, "Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems," in *the 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, Boston, MA, June 2010, pp. 1–9.

[127] T. M. Mostafa, A. Muharam, and R. Hattori, "Wireless battery charging system for drones via capacitive power transfer," in *IEEE PELS Workshop on Emerging Technologies: Wireless Power Transfer (WoW)*, Chongqing, China, May 2017, pp. 1–6.

[128] C. Song, H. Kim, D. H. Jung, K. Yoon, Y. Cho, S. Kong, Y. Kwack, and J. Kim, "Three-phase magnetic field design for low EMI and EMF automated resonant wireless power transfer charger for UAV," in *IEEE Wireless Power Transfer Conference (WPTC)*, Boulder, CO, May 2015, pp. 1–4.

[129] J. L. Gmez-Tornero, M. Poveda-Garca, R. Guzmn-Quirs, and J. C. Snchez-Arnause, "Design of Ku-band wireless power transfer system to empower light drones," in *IEEE Wireless Power Transfer Conference (WPTC)*, Aveiro, Portugal, May 2016, pp. 1–4.

[130] Federal Aviation Administration, "FAA news." [Online]. Available: https://www.faa.gov/uas/media/Part_107_Summary.pdf (accessed on April 12, 2018).