迅速な災害管理のための即時的,持続可能，かつ拡張的なエッジコンピューティングの研究

Establishing Swift, Sustainable and Scalable Edge Computing for Agile Disaster Management

Jianwen Xu

Department of Sciences and Informatics
Muroran Institute of Technology

This dissertation is submitted for the degree of
Doctor of Philosophy of Engineering

March 2020

Declaration

I hereby declare that this thesis is my own work and effort and that it has not been submitted anywhere for any award. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions.

Jianwen Xu
March 2020

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Mianxiong Dong for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. I have learned so many things from you, including the research process, writing papers, giving talks, and many more.

Besides my supervisor, I would like to thank the rest of my thesis committee: Prof. Jay Kishigami, and Prof. Yukinori Suzuki, for their insightful comments and encouragement, but also for the hard questions which encouraged me to widen my research from various perspectives.

I want to thank for Prof. Kaoru Ota and Prof. Li. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

I also want to thank Prof. Wu for recommending me to Prof. Dong's ENeS Lab, so that I had the chance to start my Ph.D. study in Muroran IT. I will never forget your encouraging words when I felt hesitant.

I would like to acknowledge the financial, academic and technical support of the Muroran Institute of Technology, and the research fund from NEC C&C Foundation which has brought me many opportunities to attend the international conferences and to publish our works in journals.

I owe a debt of gratitude to Chaofeng, Wuyunzhaola, Liangzhi, Gaolei, Xi, Luyao, Axita and all my friends and colleagues in ENeS Lab at Muroran IT. Thanks for everything you have done.

Last but not the least, I would like to thank my family, my parents for supporting me spiritually throughout writing this thesis and my life in general.

Jianwen Xu
March 2020

Abstract

Natural disasters have been threatening our life all the time. Nowadays, with the development and advancement of ICTs, we are urgently demanding the timeliness of disaster management. To achieve the goal of agile disaster management, I focus on applying edge computing in providing emergency services for affected areas. However, the existed edge computing architecture is not yet able to meet the needs of disaster scenarios. As a result, I come up with the idea of combining related technologies to enhance edge computing. First, in case that the original network falls, I propose the solution of fast networking based on Information-Centric Networking (ICN) which can swiftly connect users together. Second, after rebuilding the network topology, to extend its lifetime as long as possible, I design the energy-efficient caching strategies to optimize network resource allocation. Third, to enlarge the coverage and increase the scalability of the edge network, I choose UAVs as a solution to play the role of network nodes in-the-air and discover survivors.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background

Last year, a 6.6-magnitude earthquake occurred in Eastern Iburi Subprefecture, Hokkaido, Japan. In Atsuma, a small town near the epicenter, Hokkaido Electric Power Company's coal-fired power plants were heavily damaged by the fires right after the main shock. The following blackout spilled out multiple industries and public facilities in Eastern Hokkaido [82]. Many areas even experienced completely network connectivity interruption and communication service outage for more than 1 day. Under this situation, people in affected area were even not able to receive the rescue information or send out safe-check messages.

Facing the above situation, what we usually do is to seek help from reserved power. However, this limited electricity can only meet the demands at concentrated spots such as shelters. As a result, rescue work will be extremely difficult before power being restored. And for the rescue missions in the affected areas, if we do not know where survivors are, time and manpower can be largely consumed, which may in turn further aggravate the power shortage.

In this background, one of the research problems that I found is how to quickly restore the network connections under the extreme situation of limited power supply. The main challenge here is the speed, even during the blackout, if we are able to rebuild the communication among survivors, most of them can be saved on-time.

To achieve this agile disaster management without considering the original network architecture, in the thesis I focus on applying mobile edge computing (MEC) in providing emergency services for affected areas. However, the existed MEC architecture is not yet able to meet the needs of disaster scenarios. Thus, I come up with the idea of combining related emerging technologies to design a next generation disaster response platform.

Initially as an extension of cloud computing at the edge of the network, edge computing enables cloud computing capabilities close to users which can save energy and time cost on the backhaul transmission up to cloud servers. And mobile edge computing, in which we pay attention to the case of mobile services, even our mobile phones can play the role of edge server while providing computing resource to other devices nearby. And ever since first raised by European Telecommunications Standards Institute (ETSI) in 2014 [63], we have come to realize that MEC, as a new paradigm, fits well with the urgent needs of large-scale, non-centralized distributed computing in the current Internet of Things (IoT) era. Moreover, MEC is recognized by the European 5G Infrastructure Public Private Partnership (5G PPP) research body as one of the key emerging technologies for 5G networks, with Network Functions Virtualization (NFV) and Software-Defined Networking (SDN) [61].

And in the research field of disaster management, MEC is able to maximize its advantage of decentralized structure. Especially when there exist no enough network resource, we choose to focus on how to satisfy the user requests by local resources. That is, not only the extra energy consumption on backhaul transmissions, but also the burden on backbone network is shared. Moreover, once the the uplink connection is interrupted, edge devices can still work on their own. For the case of user search, if one of users can communicate with outside world, all the others within this sub-network can be rescued.

1.2 System Outline and Challenges

To build this disaster response platform, I divide all work into three parts. As shown in Fig. 1.1, first one is emergency networking. Before restoring services, the prime problem to solve is how to rebuild the network connections.

1.2.1 Emergency Networking

When the original architecture is down led by natural disasters such as earthquakes, all our devices relying on cellular network will be out of service just like we are in deep mountain or on isolated island.

To retrieve the connections, first we need to find positions for placing access points (APs). In consideration of the restricted available resources in post-disaster scenario, the number of APs also exists a upper limit. That is, the first target in this part is to use as few APs as possible to cover all users in the affected area. Then after the necessary number of APs being placed, the next target is how to complete fast route discovery without former routing information. The challenges in the first part are listed as follow:

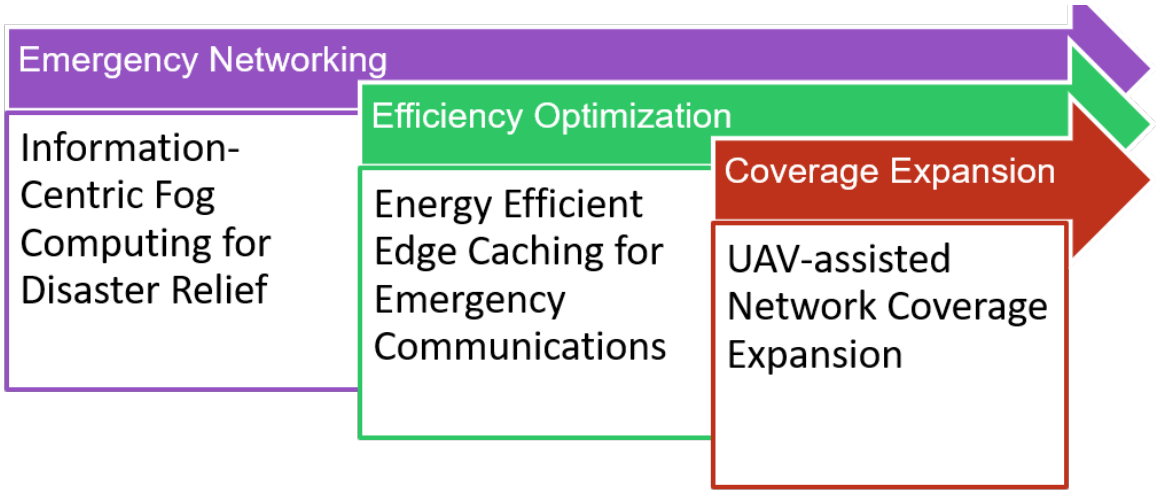


Fig. 1.1 The outline of agile disaster management.

- Networking approaches based on traditional architecture may be limited due to failures in base stations, etc.
- In case of communication resource shortage, there exist more demanding requirements for transmission efficiency.
- Considering the users themselves, how to provide services to as many as possible in different network topology is also a challenge.

1.2.2 Efficiency Optimization

After rebuilding a temporary network, to extend its life time as long as possible, energy efficient strategy is needed for properly allocating the limited network resources. I come up with the idea of edge caching which focuses on reusing data or contents being transmitted in the network.

Edge devices are usually not selected for caching due to the limited performance. However, in the situation of disaster scenario, not only because the number of available devices is restricted, but in the temporary emergency communication network, even secondary disasters are taken into account, the burden on the backbone network needs to be shared urgently. As a result, I take advantage of the decentralization in edge caching to optimize the resource allocation of the fast-built network architecture and expand the life time considering of energy cost.

For the caching strategy, the main two points are cache replacement policy and time-to-live (TTL). The former one refers to the algorithm deciding which cached items to be

discarded when capacity is full and new ones are coming. TTL here stands for the standard to determine the timing of cache replacement. The challenges in the second part are listed as follow:

- How to distribute files of different sizes to edge devices considering the maximum utilization.
- For different Time-to-Live (TTL) designs, how to choose the most suitable cache replacement policies.
- How to solve the tradeoff between existed files and new-coming ones in cache.

1.2.3 Coverage Expansion

The first two parts already complete the basic procedure of emergency networking. The last part will be the work of expanding coverage of this temporary network. In fact, when natural disasters happen, people may react and find places to take refuge. Up to now, my research still stay at the stage of normal situation. That is, placing APs to cover users according to their current positions. Once some of them move and come out of the signal range, the connections may not continue any more. Moreover, there also may be urgent cases that people are trapped in somewhere and can not approach positions within coverage of rebuilt network.

As a result, in the third part, I put emphasis on how to continue user searching and expand the existed network coverage. The challenges in the third part are listed as follow:

- How to find users without the help of network connection.
- Natural disasters may destroy the roads which can bring challenges in approaching to the target area.
- The cost of manpower as well as available resources.

In summary, in this thesis, I will work from these three parts one by one to solve each challenge and complete the construction of the entire disaster response platform.

Chapter 2

Information-Centric Fog Computing for Disaster Relief

This chapter introduces the first phase of agile disaster management and its solutions. Right after natural disaster happens, first we have to rebuild the network architecture before considering the following phases.

2.1 Motivation

2.1.1 The Difference between Fog Computing and Edge Computing

Ever since being created by *Cisco* in 2012 [9], fog computing now has become a research hotspot in both academia and industry. Just like fog's actual appearance in nature, different from cloud floating in the sky, fog is relatively near-earth and local. In fog computing, we focus on the non central part of the network which can directly response to end users in low delay and energy consumption. Fog computing can play the role of extending centralized cloud services to end users in geographical distribution. We regard fog as a necessary part of cloud, rather than an alternative.

In fact, except the definition from different organizations, there is no essential differences between fog computing and edge computing. However, we still treat fog and edge differently in researching. That is, for edge we usually refer to utilizing the end devices themselves to help each other in providing computing services. And for the case of fog, we put emphasis on introducing more available computing resources near edge, which may include routers and other local area network (LAN) hardware, etc. Sometimes, we even may design both edge and fog in the same framework to compose a cloud-fog-edge three-tier structure [87].

And in this thesis, in summary, I use edge computing to express that my target is to let the user devices at the edge of the network play the leading role in agile disaster management. And for the first part in the system outline, fog is better to express my purpose in rebuilding a temporary emergency network.

2.1.2 Fog Computing and ICN

Fog can provide a solution when we lose the connection to central servers. As one of the future Internet models, Information-Centric Networking (ICN) aims to build up a novel data-centric architecture providing receiver-driven data retrieval services. ICN is not constrained by the traditional network structures and can be used for fast networking. As a result, we believe that the Information-Centric Fog Computing (ICFC) fits the requirements of emergency communication in disaster relief.

2.2 Related Work

In this section, I present related work about ICN, fog computing, disaster management, and six degrees of separation theory.

As the current research hotspots, both fog computing and ICN are attracting extensive attention from academia and industry. Li *et al.* focus on the concept of edge-centric computing (ECC) to explore new possibilities in next generation wireless communication [42, 43]. Wu *et al.* combine fog computing and ICN, and propose a security service architecture based on content-aware filtering [84] [85]. Li *et al.* apply fog computing and deep learning in manufacture inspection to increase work efficiency [45]. Gai *et al.* present the framework of privacy-preserving communication in Internet of Things for system security improvement [26–28].

In the field of disaster management, there also have been many eye-catching research results with the emerging technologies. Erdelj *et al.* put forward a blueprint of unmanned aerial vehicles (UAV) assisted disaster management [20]. Han *et al.* pay attention to the localization in wireless sensor networks (WSNs) without geo-location information [32]. Li *et al.* focus on the distributed scene understanding by learning the depth images taken in disaster area [48]. Ernst *et al.* discuss the collaborative properties in crowdsourcing for emergency management [21]. Chung *et al.* put forward Peer-to-Peer cloud network services to provide disaster information from distributed IoT devices [15].

With this theory, we can expect to connect between any two people by a six-segmented (or less) path of ”a friend of a friend”. Besides the original application in social science, six

Table 2.1 Notations in name-based routing using SDST for disaster relief.

Symbol	Meaning
F, f	set of fog nodes in Fog Tier and one in it
U, u	set of user nodes in User Tier and one in it
k	number of edges built between neighbors in Watts-Strogatz model
β	probability of edge rebuilt in Watts-Strogatz model
CL, cl	content library and one file in it
n_u, n_f, n_{cl}	number of users, fog nodes and files in CL
r	popularity rank of cl in CL
s	an exponent to characterize the Zipf distribution
n_r^{hop}	number of forwarding hops to get the rth file in CL

degrees of separation theory (SDST) has already inspired many cross-disciplinary researches from mathematics to psychological analysis. Muldoon *et al.* discuss the small-world network properties for weighted brain networks [55]. Modern SNSs including Facebook and LinkedIn also apply SDST in designing some platform applications. In the field of computer networks, related work on efficient message delivery through user relationship networks has been around for a long time. Vespignani summarizes the development of network science in the past 20 years since Watts and Strogatz first proposed the small world model [81] in 1998 [77].

2.3 Problem Formulation

In this section, I design the ICFC network model and formulate the problem to solve. Symbols used in this thesis are listed in Table 2.1.

2.3.1 Access Points Placement Using ICN and BFS

I consider a two-dimension fixed $x \times x$ square range as a post disaster scenario. There exist three types of nodes, each one using its name as unique identifier. User nodes U_n and server nodes S_n have fixed locations within scope. Access point nodes A_n own fixed signal coverages, and only the U_n staying in the radius of a nearest access point node are able to send out packets. A_n can be set anywhere, even overlapping some U_n as a trade-off choice. S_n can be visited through A_n nearby, which means other remote A_n may have to rely on forwarding work by neighbors. Here my *first target* is how to reduce the number of A_n and make efficient usage of each one while ensuring that each U_n has at least one A_n nearby to keep connected. I use an example to further explain the details.

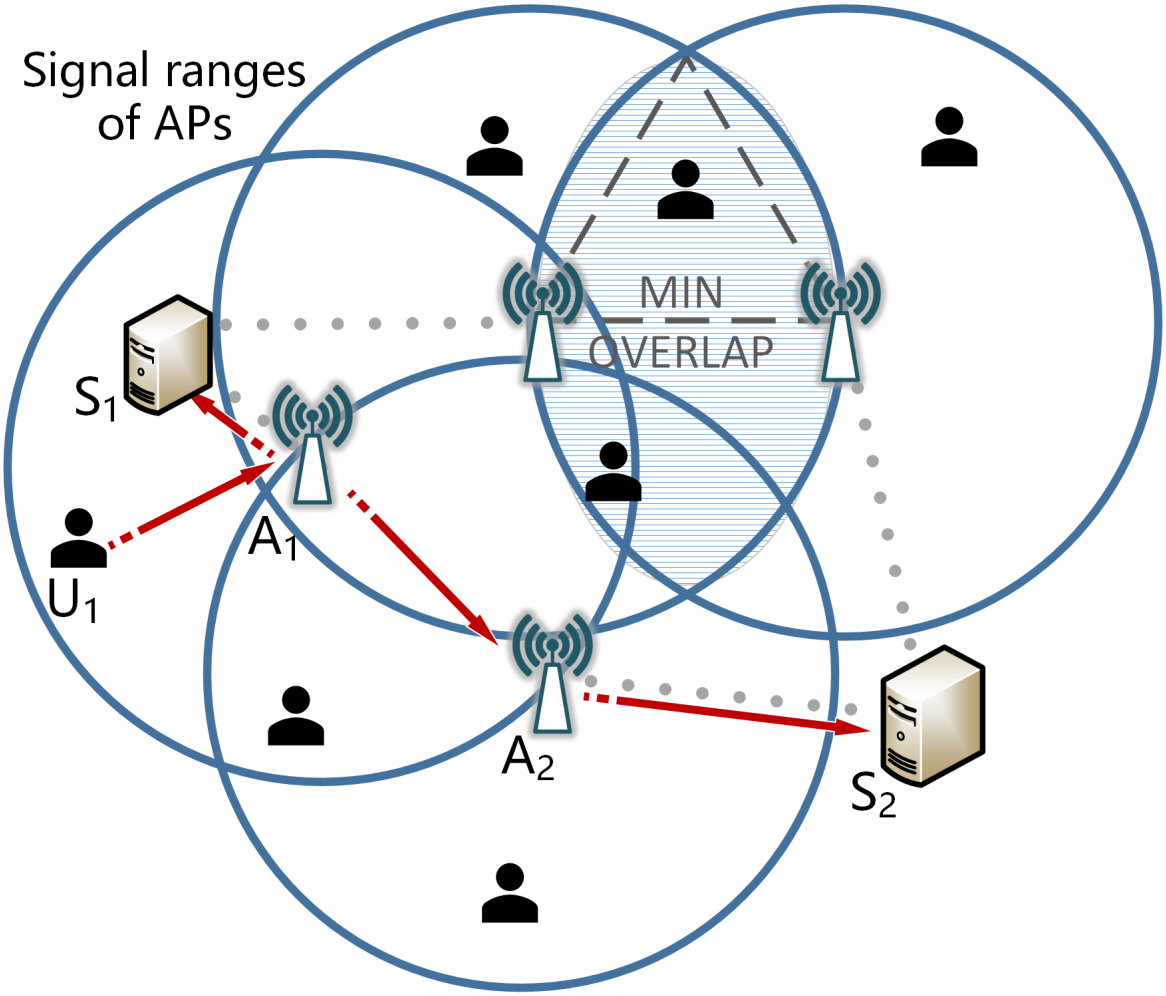


Fig. 2.1 The ICN based network structure in disaster recovery scenario

As shown in Fig. 2.1, there are nine blue U_n relying on four yellow A_n to contact with two green S_n. Yellow circles stand for signal range of A_n, and gray dotted lines are direct relations between S_n and A_n. The *MIN OVERLAP* area represents the shortest distance between A_n to ensure interconnection. For instance, if U_1 wants to visit S_1, it only takes two hops via A_1. Then if it wants to send a *Interest Packet* to S_2, the full path is $U_1 \rightarrow A_1 \rightarrow A_2 \rightarrow S_2$. Every A_n has to guarantee that a neighbor one can do a favor when it does not find any other interfaces to forward in *PIT*.

In order to use less A_n to cover more U_n, I prefer as dispersed as possible for placement work. Besides the number of needed access points, I also pay attention to workload allocation of each A_n, which means each A_n may share a similar workload on managing U_n. Therefore I keep two metrics, the new covered area and new covered U_n number when deciding the location to place the next A_n.

At first, each A_n has to make sure that at least one neighbor A_n stays in its coverage radius, i.e. new covered area is the differential between areas enclosed by all circles of placed A_n before and after placement.

$$\begin{aligned} SN_{\odot_n} = S_{\odot_n} &- \underbrace{\left(SO_{\odot_{1n}} + SO_{\odot_{2n}} ... SO_{\odot_{(n-1)n}}\right)}_{\mathrm{C}_{n-1}^{1}} \\ &+ \underbrace{\left(SO_{\odot_{12n}} + SO_{\odot_{13n}} ... SO_{\odot_{(n-2)(n-1)n}}\right)}_{\mathrm{C}_{n-1}^{2}} \\ &\vdots \\ &+ (-1)^{n-1} \underbrace{\left(SO_{\odot_{12\cdots(n-2)n}} ... SO_{\odot_{23\cdots(n-1)n}}\right)}_{\mathrm{C}_{n-1}^{n-2}} \\ &+ (-1)^{n} SO_{\odot_{123\cdots(n-2)(n-1)n}} \end{aligned} \tag{2.1}$$

New covered area ($SN_{\odot_n}$) can be calculated from Equation (2.1), in which $S_{\odot}$ and $SO_{\odot}$ respectively stands for circle area and overlap area of multiple circles. My problem turns into a variant of the classical *combined area of overlapping circles* problem. There exist many mature computing methods in both mathematics and engineering with different precisions like polygon cutting, *Simpson's rule*, *Voronoi diagram* and *Monte Carlo algorithm*. Since what I prefer is a method focusing on circles of equal radii and new covered area individually, inspired by the idea and rules of graphic union coverage, an improved equal circle coverage method is designed to give solution.

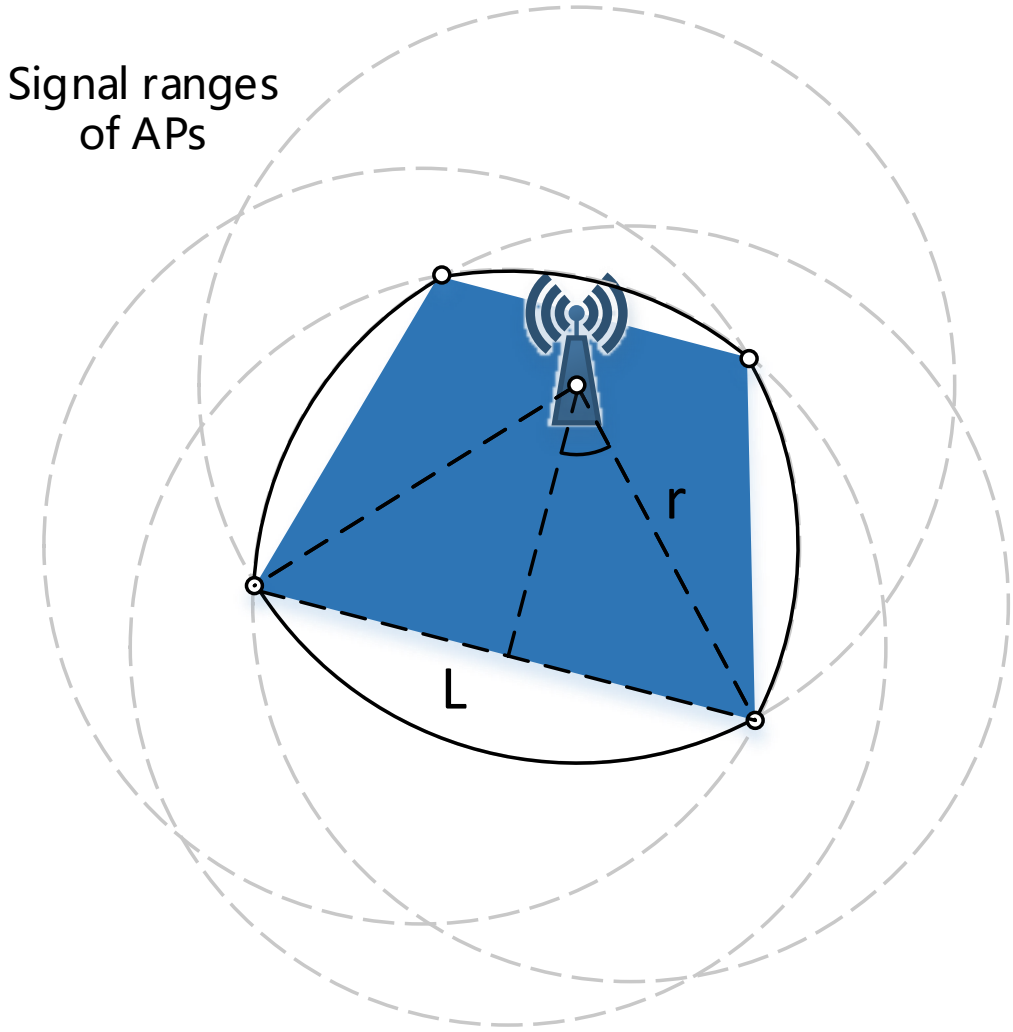


Fig. 2.2 The quasi-polygon made up of circular segments from signal range circles of APs

Secondly, I will compute all the addends and sum them up. The overlapped parts are actually specific quasi-polygons surrounded by circular segments other than irregular patterns.

As shown in Fig. 2.2, each quasi-polygon I deal with can be divided into two parts, several circular segments (S_{cir_seg}) and a arbitrary convex polygon (S_{pol}). Circular segments are all minor parts of circles, whose area can be computed from the lengths of corresponding chords in Fig. 2.2 and Equation (2.2).

$$S_{cir\_seg} = \sum_{i=1}^{n} \left[r^2 \cdot arcsin(\frac{L_i}{2r}) - \frac{rL_i}{2} \right] \tag{2.2}$$

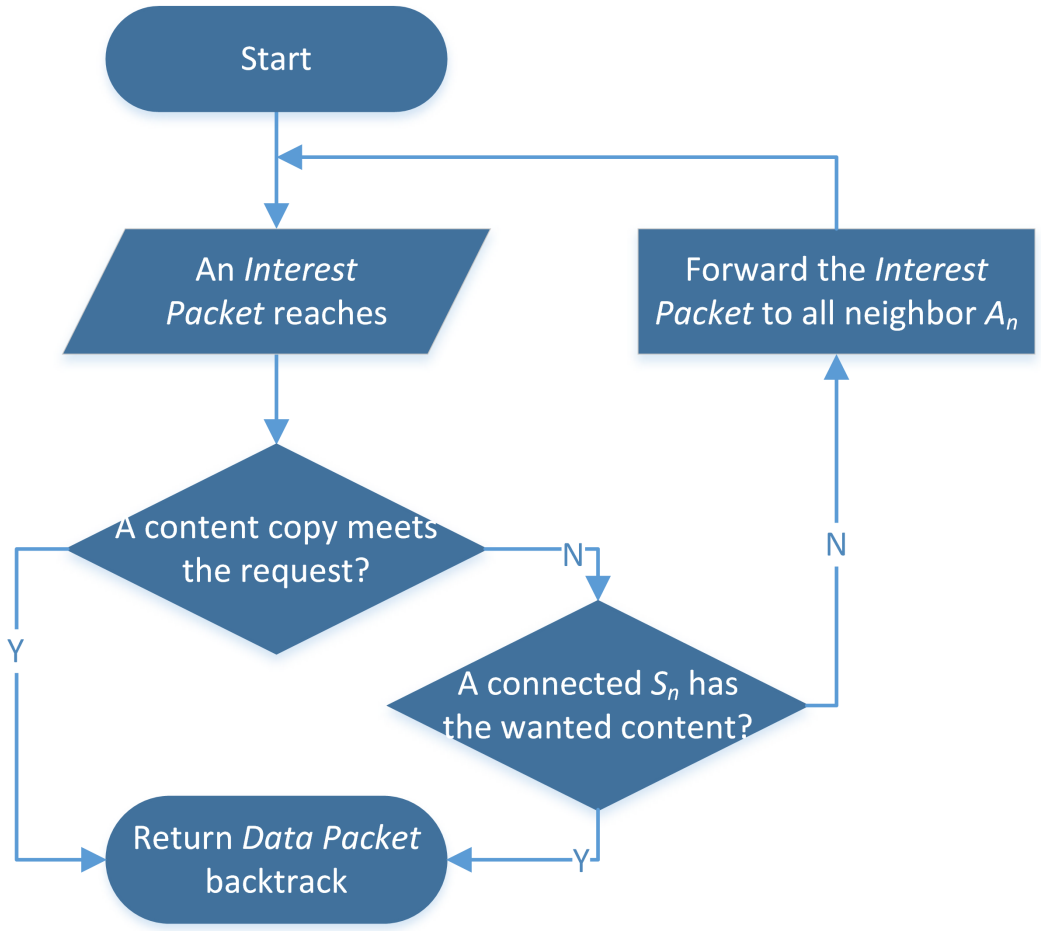


Fig. 2.3 ICN based routing & forwarding strategy

The arbitrary convex polygon also can be easily calculated by *Shoelace Formula* as (2.3). Here (x_i, y_i), $i = 1, 2, \cdots, n$ are vertices of the polygon.

$$S_{pol} = \frac{1}{2} |(x_1 y_2 + x_2 y_3 \cdots x_n y_1) - (y_1 x_2 + y_2 x_3 \cdots y_n x_1)| \tag{2.3}$$

After determining the locations of all A_n, the next step is to establish connections among three kinds of nodes. Each U_n needs to find the upper A_n within scope and save their names in its *Forwarding Information Base (FIB)*. Correspondingly, A_n record the names of U_n under management in their *Pending Interest Table (PIT)*. In the same way, A_n also exchange names with S_n.

As the *second target*, I design a routing strategy to distribute two types of packets in the fast organized Content-Centric Networking instance. *Interest Packet*s have the names of wanted contents, once any A_n or S_n can satisfy the requests, *Data Packet*s will be sent back in the reverse path. I use a flow chart to describe the routing strategy adopted by A_n.

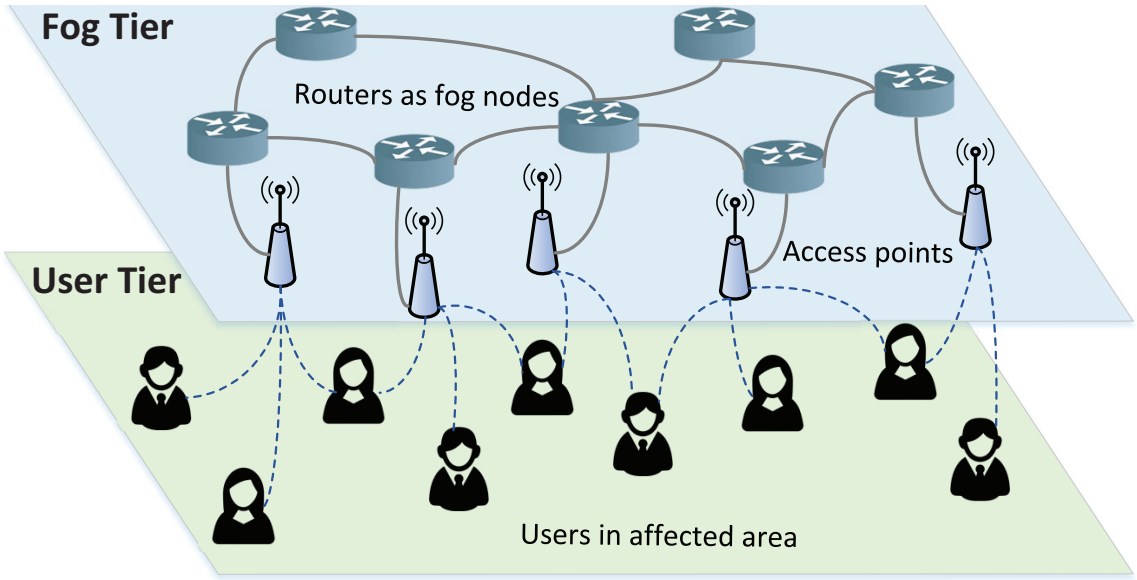


Fig. 2.4 A 2-tier information-centric fog network architecture.

At the beginning, U_n generates an *Interest Packet* with the name of wanted content and sends out to all U_n that can hear it (similar to broadcast). As shown in Fig. 2.3, when the *Interest Packet* reaches *PIT* of a A_n, firstly it needs to check if there exists a copy cached in *CS*. Once no copy matches, secondly A_n will send the packet upstream to all connected S_n and search for the wanted content by name. Either of the two cases is fulfilled, a *Data Packet* will be returned to U_n in reverse path. Otherwise, A_n has to forward it to neighbors to start a new branch judgment procedure. In this way, *Interest Packet* can get to the right place as soon as possible. Without worrying about address resolution work done by Domain Name System (DNS) server, actually I can apply the idea of *Breadth-First Search (BFS)* from graph theory into algorithm design.

2.3.2 Route Discovery Based on ICN and SDST

A 2-tier information-centric fog network architecture is shown in Fig. 2.4. Fog Tier is made up of massive ICN nodes $F = \{f_1, f_2, ..., f_n\}$ providing limited services to users in User Tier ($U = \{u_1, u_2, ..., u_n\}$). Here I use the Watts-Strogatz model [81] from SDST to describe the properties of node distribution and relationship among them. In an undirected graph, a group n nodes first form a regular ring lattice one by one. Second, each node builds edges to its k (should be an even integer) neighbors, $k/2$ by left and right sides in a ring lattice. Third, each edge created by the current node can be rebuilt with a probability of β. As a result, for any two nodes n_i and n_j, there exists an edge if and only if

$$0 < |i-j| mod(n-\frac{k}{2}-1) \leq \frac{k}{2} \tag{2.4}$$

A content library $CL = \{cl_1, cl_2, ..., cl_n\}$ includes all the files for requesting which are stored in Fog Tier. The popularity of files in CL obeys Zipf distribution. Zipf distribution or Zipf's law is an empirical law that states the relation between frequency of occurrence of an event and its rank by this frequency with all events [68]. The idea of Zipf's law has long affected the design in the Internet such as content delivery networks and peer-to-peer networks [1].

$$f_{zipf}(r;s,n_{cl}) = \frac{1/r^s}{\sum_{i=1}^{n_{cl}}(1/i^s)} \tag{2.5}$$

Equation (2.5) shows the normalized frequency of cl_r in a CL with n_{cl} files in total. Here r stands for the popularity rank of cl_r in CL. s is the value of the exponent which characterizes the distribution. When any file is requested from User Tier for the first time, it has to be downloaded from the original node. After any file is being forwarded back, a replica can be left at any node it passes. Then from the second time, any ICN node has replicas also can answer the request.

As a result, my target is to design name-based routing strategy to answer the requests from users by finding out the suitable nodes having the wanted files or replicas with high work efficiency. The first metric I choose is the number of forwarding hops in total. That is, to cope with the same amount of user requests with as few forwarding hops as possible.

$$\begin{aligned} minimum \quad & \sum_{r=1}^{n_{cl}} n_r^{hop} f_{zipf}(r;s,n_{cl}) \\ & = \sum_{r=1}^{n_{cl}} \frac{n_r^{hop}/r^s}{\sum_{i=1}^{n_{cl}}(1/i^s)} \\ subject\ to \quad & s \geq 0, r \in \{1,2,...,n_{cl}\} \end{aligned} \tag{2.6}$$

Besides the efficiency on transmission, I also consider the number of update messages [34]. As discuss in the introduction section, existed routing protocols have to rely on frequent information exchange to ensure real-time mastery of network topology. Excessive messages for updating may not only occupy the limited network resources in disaster relief, but also bring the risk of losing effective information when encountering node changes. Thus, my second target is to integrate the inter-node relationships with SDST and reduce the number of update messages.

2.4 Algorithm Design

In this section, I will set about solving the problems raised earlier in fast network organizing, access point placement and package delivery. Two algorithms are designed in building up the Content-Centric Network in disaster recovery scenario.

2.4.1 Algorithm Design on Access Point Placement

Algorithm 1 Access Point Placement Method

$U_n \leftarrow$**U**ser nodes within range
$Q_U \leftarrow$**Q**ueue of User nodes
$A_n \leftarrow$**A**ccess point nodes
$r_A \leftarrow$cover **r**adius of **A**ccess points
$C_{this} \leftarrow$**C**enter of current circle
$C_{last} \leftarrow$**C**enter of **last** circle
bubble sort all U_n by dist to square center and push into Q_U
while $Q_U == \emptyset$ **do**
  if triangle circumradius of first three $U_n <= r_A$ **then**
    C_{this} = triangle circumcenter
  else if half of dist between first two $U_n <= r_A$ **then**
    C_{this} = midpoint
  else
    C_{this} = first U_n
  end if
  if $dist(C_{this}, C_{last}) > r_A$ **then**
    truncate the dist to r_A
  end if
  set an A_n at C_{this}
  bubble sort all U_n in Q_U by dist to C_{this}
  pop out all U_n that dist to $C_{this} <= r_A$
  $C_{last} = C_{this}$
end while

As shown in Algorithm 1, Q_U stands for a queue saving all names of uncovered U_n, r_R is the signal coverage radius, C_{this} and C_{last} are locations of A_n just placed and last one. Algorithm 1 provides a method for placing the minimum required number of A_n to cover all U_n while each A_n should at least stay in signal radius of another A_n. Inspired by a proven geometry theorem saying *All regular simple polygons, all isosceles trapezoids, all triangles and all rectangles are cyclic* [39], which means three is the maximum number of nodes that

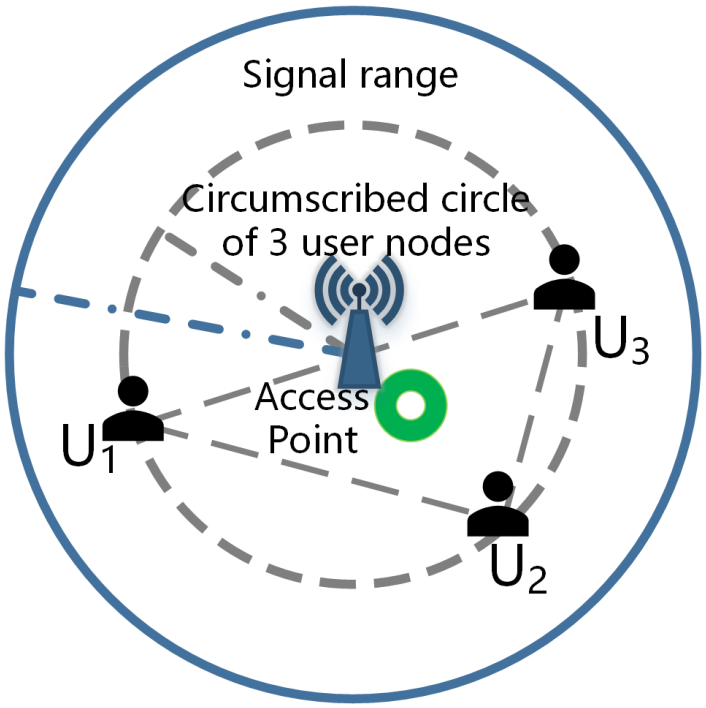


Fig. 2.5 The situation that AP can be placed when signal range is larger than circumscribed circle

must exist a circumscribed circle passing through all vertices. In a word, I am going to place a A_n at three cases in sequence.

- *Case 1:* Place at the circumcenter of the triangle composed of the first three U_n in Q_U (*line 9-10* in Algorithm 1);
- *Case 2:* Place at the midpoint of the first 2 U_n in Q_U (*line 11-12*);
- *Case 3:* Place at the first U_n in Q_U (*line 13-14*).

Once the previous case mismatches the condition, next one will be considered, I use a figure to describe *Case 1*, it is the same with the other two.

As shown in Fig. 2.5 and 2.6, since we can soon draw the sole circumscribed circle of an arbitrary triangle, the second-to-last step before I determine the location of a new A_n is to judge the length comparison between two circle radii. In fact, when distribution of U_n in a fixed range is dense enough, *Case 1* can be easily satisfied, so does *Case 2*. And setting a A_n at the location of a U_n (*Case 3*) is never a good choice, I only treat it as a supplement to maintain algorithm integrity.

Lastly, to satisfy the specific requirement of network organizing and routing, we need each A_n capable of seeking help from neighbors when there is no connection to the right S_n.

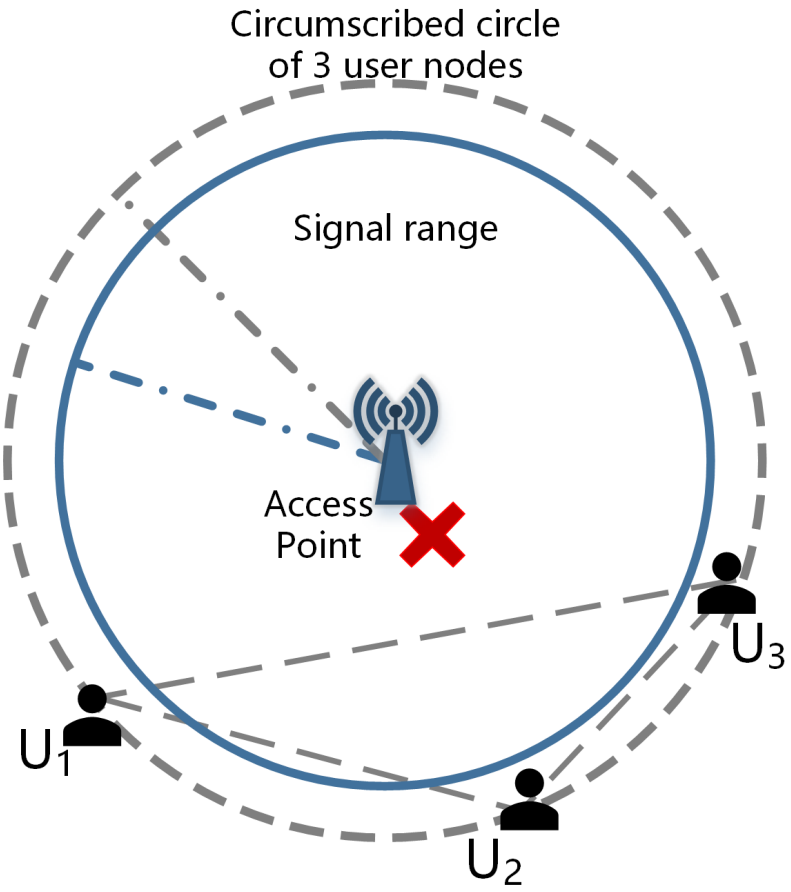


Fig. 2.6 The situation that AP can not be placed when signal range is smaller than circumscribed circle

As a result, I truncate the distance between current A_n (C_{this}) and last one (C_{last}) by finally placing it at the arc of last circle.

In Fig. 2.7, A_1 is the last access point placed before, next I am going to choose location for A_2. The *if* statement in *line 9-15* of Algorithm 1 first considers the gray one, then after a distance truncation finally moves left. In the situation, connection status of U_n is changed partly: U_2 involves in, U_5 steps out, U_3 and U_4 stay the same and U_1 now has two A_n to request. In experiment simulation, according to different U_n densities in setup, U_n-*out* are more than U_n-*in* to a various extent. Nevertheless, we still have to pay more on covering relatively isolated U_n.

2.4.2 Name-based Routing Using SDST

In this section, I design a routing strategy using SDST for providing information-centric fog services to users in affected area.

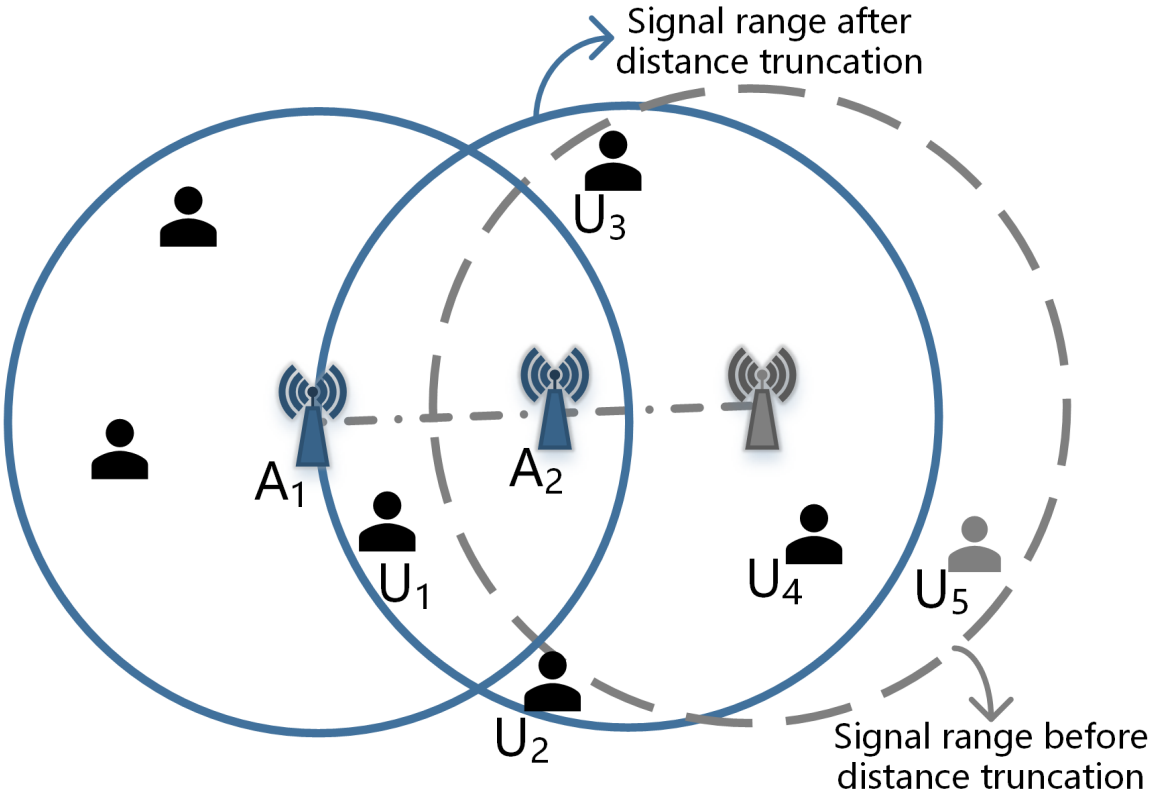


Fig. 2.7 An example of distance truncation

Algorithm 2 LRER: Limited Relationship Expansion Routing

$f\_next(f_i, cl_j)$ ←the name of next node to forward for f_i to find cl_j, $\emptyset$ means cl_i is already cached in CS of f_i
$n_{re}(cl_j)$ ←left number of replicas for each cl_j
hop_count ←record of forwarding hops in transmission

1: a user u_{this} send out a request to f_i ask for cl_j
2: $f_{this} \leftarrow f_i$, $hop\_count \leftarrow hop\_count + 2$
3: **while** $f\_next(f_{this}, cl_j) \neq \emptyset$ **do**
4: **if** $find(f_{this}.CS = cl_j)$ **then**
5: $hop\_count \leftarrow hop\_count + 2$
6: **break**
7: **else**
8: $f_{this} \leftarrow f\_next(f_{this}, cl_j)$
9: **end if**
10: **if** $n_{re}(cl_j) > 0$ **then**
11: **if** $!find(f_{this}.CS = cl_j)$ **then**
12: push cl_j into $f_{this}.CS$
13: **end if**
14: **end if**
15: **end while**

As shown in Algorithm 2, to achieve name-based routing in ICFC for disaster relief, I minimize the conditions to be as close as possible to emergency communications in post-disaster environment. When traditional facilities such as cellular networks are no longer available, trapped users are in urgent need of getting in touch with the outside world, even if it is one-way communication. To cope with this situation, I first cover all areas where survivors may be present with multiple fog nodes. Instead of storing and managing a variety of information in FIB from neighbor reports, I only need the name of next node to search for file ($f\_next(f_i, cl_j)$).

As a result, the only thing a fog node needs to do when receiving request is to forward it to the next one. Update messages are being exchanged only when new node comes in. Even when any file is cached in the CS of passed node (line 12 in Algorithm 2), I do not need to update instantly. That is, I check the CS of each node if it has the wanted file (line 4) to reduce extra forwarding. Some periodic overall updates are also helpful in adapting to the topology changes of the network. When regarding the nodes as our own, to help deliver the request to its destination, in the case of limited knowledge of outside information, I always choose anyone most likely to be close to the target. Time complexity of Algorithm 2 is $O(n_f)$.

2.5 Performance Evaluation

In this section, I evaluate the proposed routing strategy for disaster relief through simulation experiments.

Table 2.2 Experimental settings of name based routing

Parameter	Value
Number of files in CL	100
Number of users/fog nodes	200/50
β in Watts-Strogatz model	0.15/0.5
k in Watts-Strogatz model	2$\sim$20
s in Zipf distribution	1
Number of available replicas	0$\sim$4

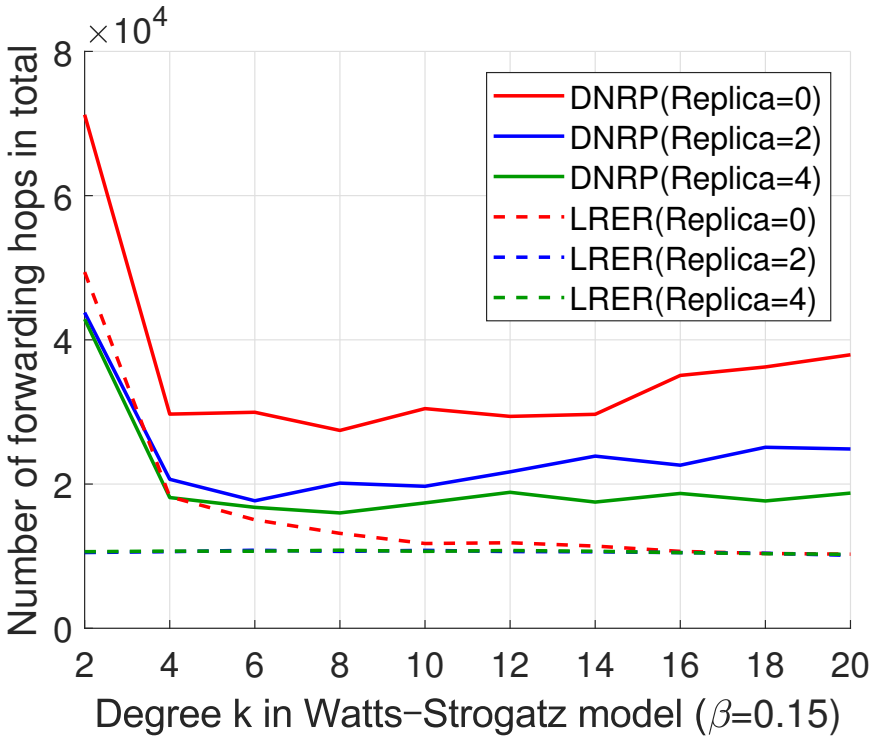


Fig. 2.8 Results of name-based routing: Number of forwarding hops

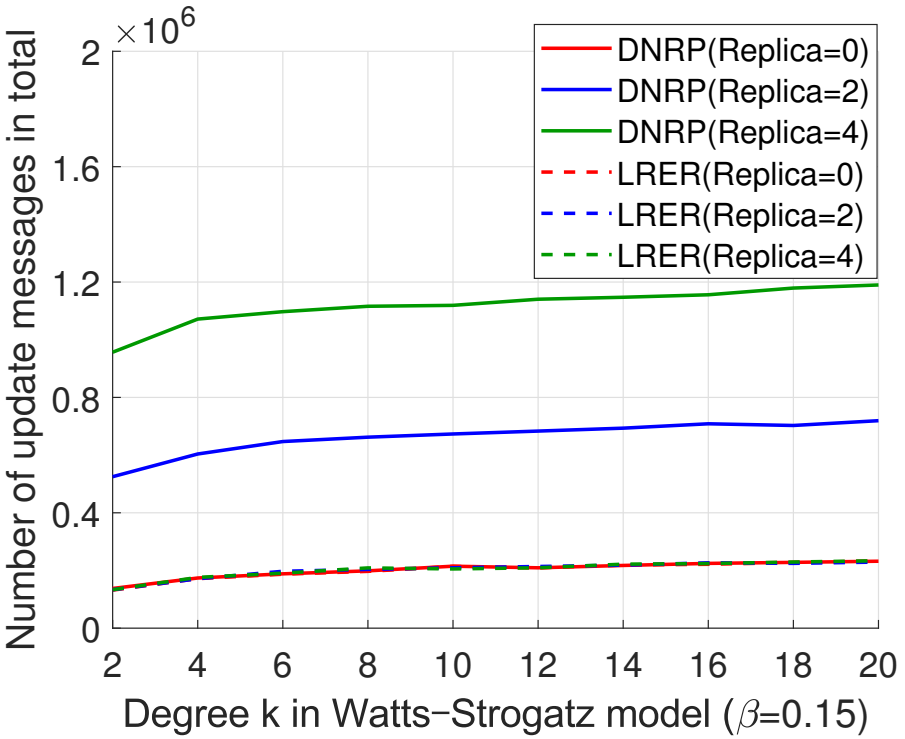


Fig. 2.9 Results of name-based routing: Number of update message

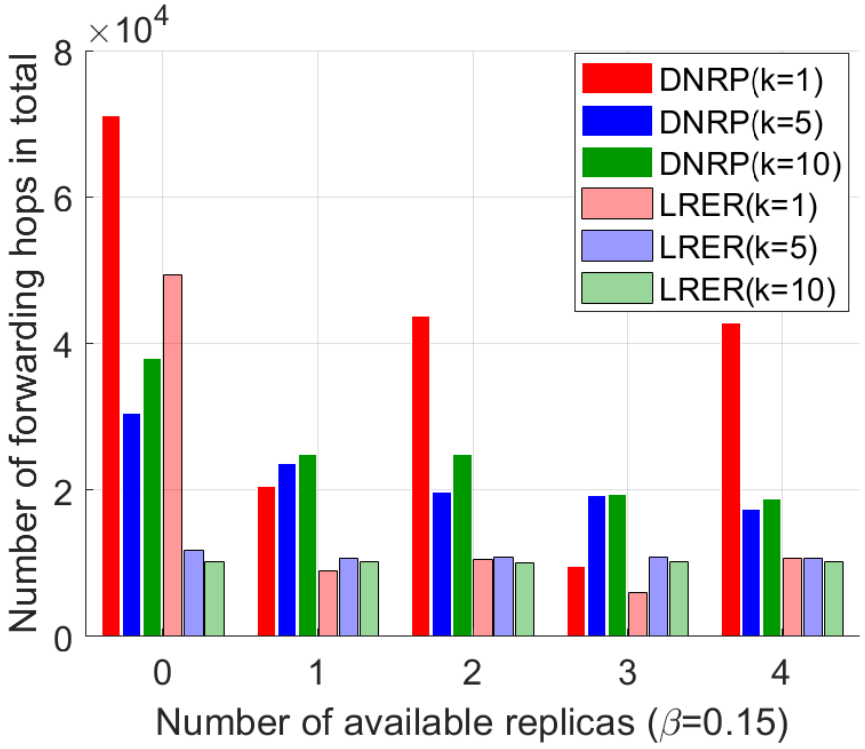


Fig. 2.10 Results of name-based routing: Forwarding hops by replicas

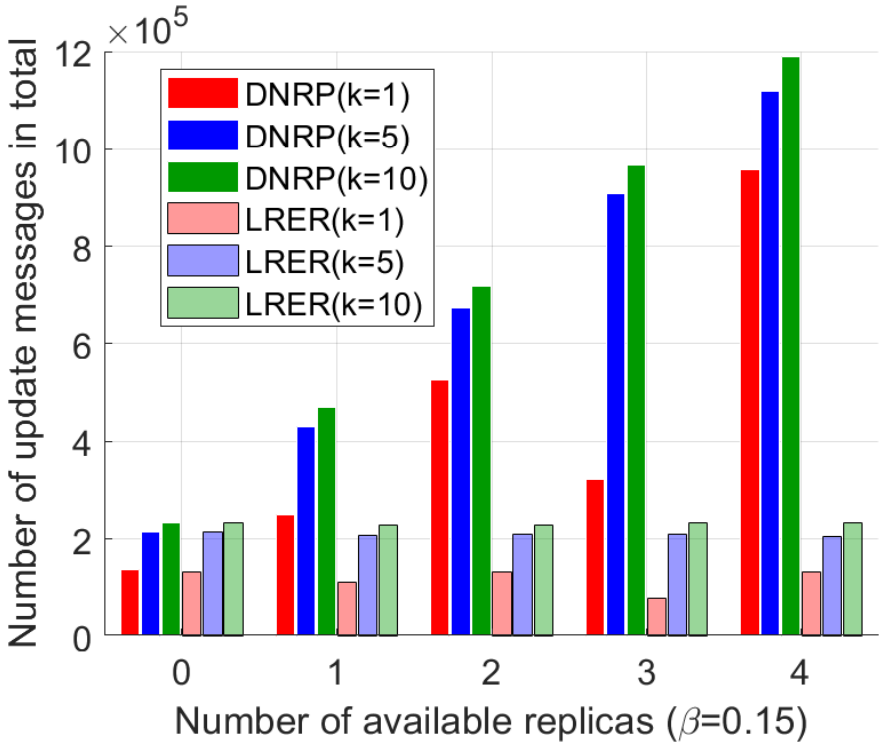


Fig. 2.11 Results of name-based routing: Update messages by replicas

As shown in Table 2.2, in a open area after disaster, there are 200 users and 50 fog nodes. Users are sending requests for one of the 100 files in *CL*. All the files are originally saved in one of the fog nodes. When any file is delivered back to user, a replica can be left at the passed nodes. I limit the number of available replicas for each file within $[0 \sim 4]$. I repeat the process from requesting to getting satisfied 2000 times for each set with different β. I compare the metrics of forwarding hops and update messages with DNRP under the same experimental setups.

DNRP is proposed in 2018 [35] by the research team led by J.J. Garcia-Luna-Aceves. As one of the state-of-the-art name-based protocols, DNRP looks for current best path selection by recording and comparing the link cost at each step of forwarding. That is to say, neighbors know everything a node is doing under the premise of timely update. However, for emergency communication in post-disaster scenario, stable connections are often not guaranteed. As a result, how to make use of limited resources to complete minimal messaging is what LRER aims to do.

Fig. 2.8 to 2.11 show the results of name-based routing for disaster relief when β is 0.15. Here I use colors to indicate the number of different replicas allowed. Red is no replica, blue is 2 replicas and green is 4. When the rebuilt probability is low, fog nodes here own the similar degrees. Or I may look forward to a more regular network topology in which users are evenly distributed everywhere waiting for communication and rescue. First in Fig. 2.8, I calculate the number of forwarding hops for answering requests from 200 users. In the case of the same number of replicas, LRER is always less frequently forwarded than DNRP. LRER with 2 or 4 replicas can even stay almost the same when k is changing. Second in Fig. 2.9, LRER cuts most the overhead on update messages. DNRP with no replica nearly coincides with the results of LRER which can be explained that no frequent need to exchange information between neighbors. Fig. 2.10 and Fig. 2.11 display the results by different numbers of available replicas. Compared with DNRP, nearly all results in LRER are lower in numerical values, which is consistent with the previous inferences.

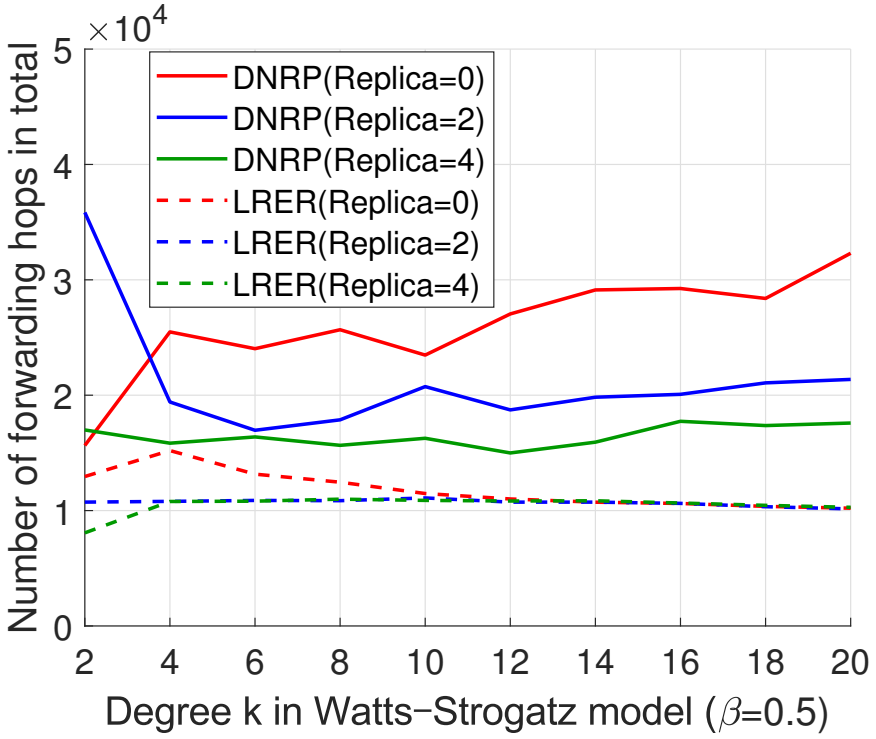


Fig. 2.12 Results of name-based routing: Forwarding hops by different k

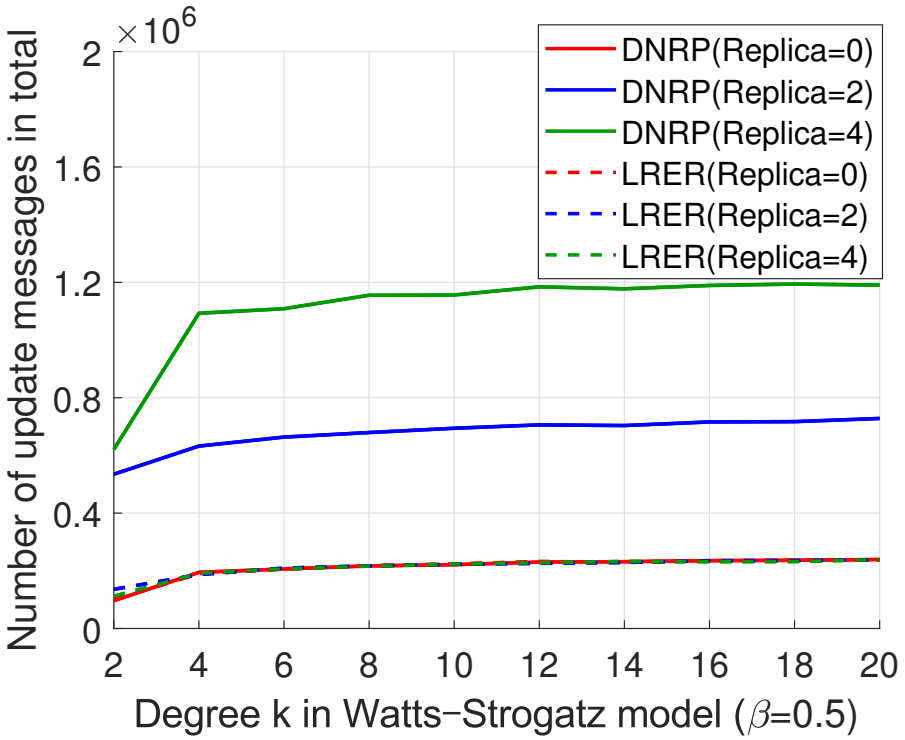


Fig. 2.13 Results of name-based routing: Update messages by different k

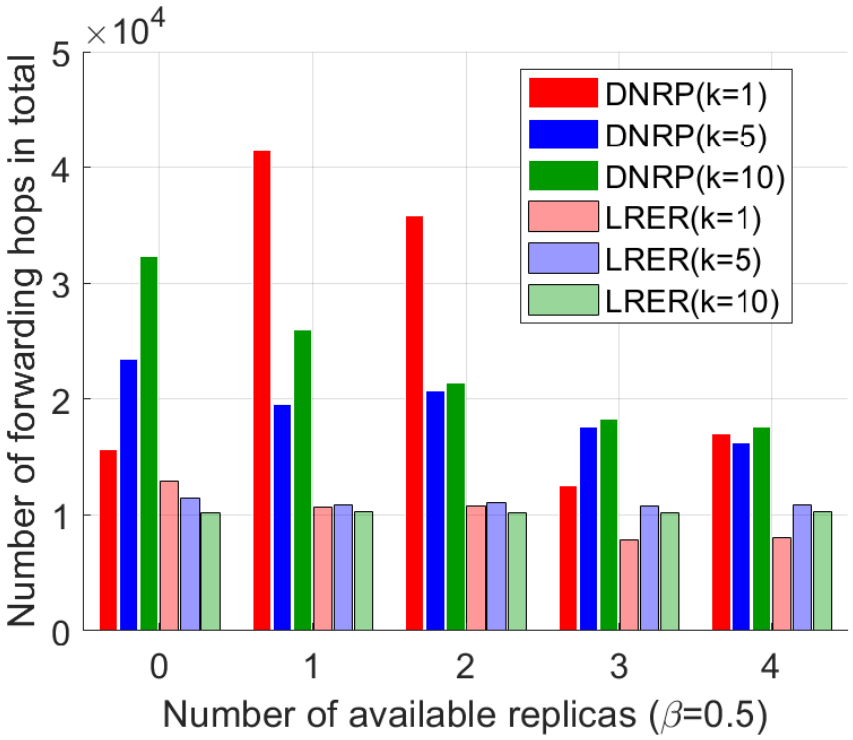


Fig. 2.14 Results of name-based routing: Forwarding hops by replicas

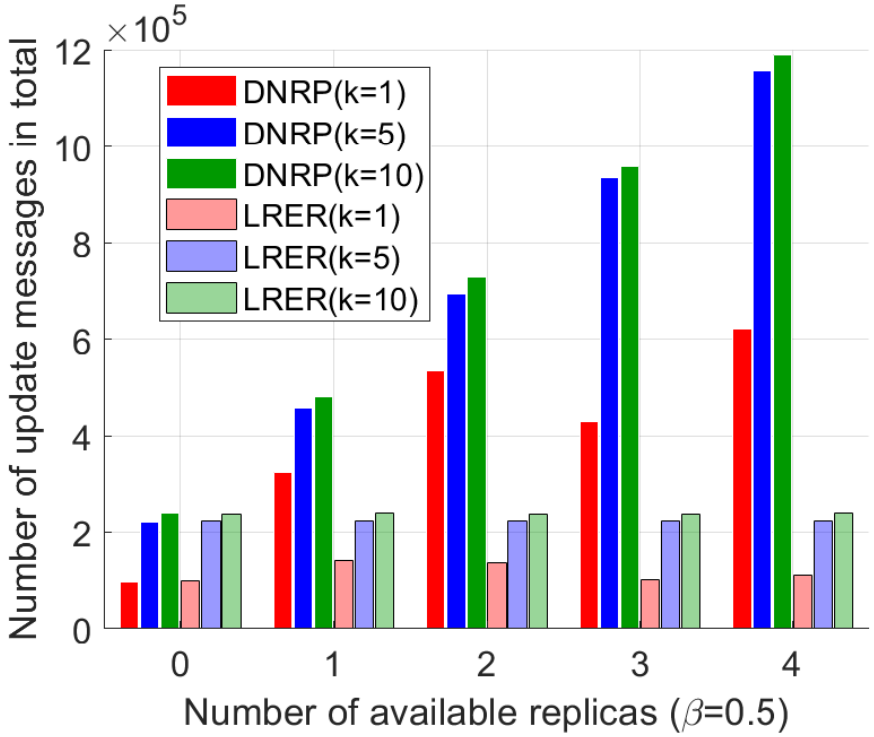


Fig. 2.15 Results of name-based routing: Update messages by replicas

In order to further simulate the adaptability of the proposed routing strategy in the Watts-Strogatz model, I add another set of results with $\beta = 0.5$. Compared with Fig. 2.8∼2.11, Fig. 2.12∼2.15 has higher randomness in network connectivity and may be closer to the disordered state in the scenario of disaster relief. The trend of the polylines also confirms this point. When each node only has a few connection options, not only will the nodes be differentiated from each other by degree, but they together may form some single long paths. Results of forwarding hops in Fig. 2.12 and Fig. 2.14 show more fluctuations. Especially when k is small, DNRP with no replica even has an increasing overall trend. LRER maintains high stability and does not receive too much influence from β. Finally in Fig. 2.13 and Fig. 2.15, I do not observe any significant changes. Or I can say that under the current experimental settings, the randomness of network connections is not a major factor affecting the number of update messages in both strategies.

In summary, I compare the performance of the proposed LRER with the existing DNRP in numbers of forwarding hops and update messages through two sets of experiments. The results show that my strategy can achieve higher work efficiency and may be more suitable as a technology for network reconstruction in disaster relief.

Chapter 3

Energy Efficient Edge Caching for Emergency Communications

This chapter introduces second phase after the network topology being rebuilt, in which my target is to extend its life time as long as possible. I design the energy saving and caching strategies to optimize network resource allocation.

3.1 Motivation and Related Work

In this section, I present some related works about edge computing, edge caching, and then introduce some researches on in-memory storage and processing.

3.1.1 Edge Computing and Edge Caching

Cloud services have long remained a part of people's lives, ever since cloud computing became known in 2005 and was quickly utilized in a wide variety of fields. Together with the current IoT boom, in the come-at-able 2020, total amount of data created by any device will reach 600 zettabytes (ZB) per year while annual global data center IP traffic will only be 15.3 ZB at the end of this decade [59]. As a result, in the near future, we are no longer able to put all computing tasks on the cloud and pin our hopes on continuously updating hardware levels, increasing the number of end equipments. I need edge devices to share the workload and solve the bottleneck in data transmission and processing [73].

Edge computing, before attracting wide attention and extensively applied among research institutions, is already studied by a number of technology companies, such as the key players including *Cisco Systems Inc.* and *HP*, etc. Early in 2012, Bonomi *et al.* from *Cisco* start from making clear the position of edge computing in IoT era and prove that fog owns the

characteristics of serving as platforms for IoT services from connected vehicle, smart grid to smart city. They define the fundamental characteristics as low latency & location awareness, widespread geographical distribution, mobility, large numbers of devices, predominant role of wireless connection, streaming and real-time applications and heterogeneity [9]. Vaquero*et al.* from *HP* offer a comprehensive definition *the fog* to include cloud, sensor network, peer-to-peer network, etc. They also combine Network Function Virtualization (NFV) and Software-Defined Networking (SDN) together to achieve a new *Softwareisation* network management [76]. Zhao *et al.* discuss the combination of Internet of Vehicles (IoV) in designing cognitive routing method [93]. Li *et al.* come up with the idea of designing an intelligent edge network function virtualization framework [46].

Many works on edge computing in recent years focus on interdisciplinary researches and try to find the relation with other fields to help promote common development. Liu *et al.* design a device-to-device video recovery system based on heterogeneous network for picocell edge users. In the paper they discuss the possibilities of achieving the video on demand (VoD) application to improve the current performance [51]. As a branch discipline, mobile edge computing pays more attention on wireless communication among smartphones, tablets and other hand-held devices. Sardellitti *et al.* consider a multiple-input and multiple-output (MIMO) multicell system and design a whole set of joint optimization algorithms for mobile edge computing [69]. Research group from the European Telecommunications Standards Institute (ETSI) regards mobile edge computing to an independent field of study and combine with the fifth generation (5G) mobile networks. They also analyze the market drivers and business value of mobile edge computing services [60].

In order to further utilize edge devices to balance the workload in the expanding network, caching on edge can make a contribution on reducing bandwidth usage, server load and so on. Researches on edge caching also have many different directions. Early in 2005 before cloud computing entering the public consciousness and widely applied in production and living, Ramaswamy *et al.* propose the idea of building cache clouds to deal with documents in edge networks. In the paper they design a dynamic hashing scheme to improve document placement in cache clouds [67]. Gabry *et al.* put forward a maximum-distance separable (MDS) encoded caching scheme to achieve energy-efficient edge computing in heterogeneous network [25]. In recent years, with fast development of wireless communication, caching on mobile/wireless edge becomes a research hotspot. Liu *et al.* summarize the design aspects and challenges of wireless edge caching. They focus on two key features of content delivery traffic and compare the performance of caching at base stations and users [49].

3.1.2 In-Memory Storage and Processing

Compared by disk storage like HDD, flash memory and faster Solid-State Drive (SSD), in-memory or main memory mainly refers to volatile RAM could spend the same amount of time while reading/writing data regardless of physical location. Even though still limited by fault-tolerance, consistent power supply and high manufacturing cost, from all kinds of electronic equipments, personal computers, to large professional servers, I still can not rely on main memory to store the data for long time. However, the last decade has witnessed rapidly decreasing cost of main memory and growing demand of high-speed computing which makes it possible for turning in-memory into the new disk.

Related works on in-memory storage and processing involve different levels from application domain analysis, technical breakthrough to business development prospect. Zhang *et al.* introduce the recent years' development in in-memory big data management and processing. In the paper they classify and summarize all existing commercial and academic management systems for in-memory operations [91]. Beneventi *et al.* apply in-memory processing tools to help do machine learning in High-Performance Computing (HPC) infrastructure models [6].

3.1.3 Hybrid Edge Caching in 5G Era

Since the fourth generation (4G) represented by Long-Term Evolution (LTE) entered the stage of commercial deployment, both industry and academic community have been scrambling to focus on the upcoming fifth generation (5G). And from the user's point of view, everyone is constantly thinking about what innovations the next generation of wireless communication systems will bring to us. Besides faster connection speed, wider bandwidth range and larger throughput, we also consider 5G's performance in terms of user experience and environmental sustainability. The concept of Tactile Internet has emerged as the times require.

Tactile Internet is first defined by the International Telecommunication Union (ITU) as an network architecture that combines ultra low latency with extremely high availability, reliability and security. As the name suggests, Tactile Internet aims to provide a reliable, easy-to-use, low-power, real-time interactive network system just like how human tactile experience be sensed.

As shown in Fig. 3.1, I compare the reaction time/latency of human sensing and different generations of wireless communications. First I choose three traditionally recognized senses, auditory (hearing), visual (seeing) and tactile (touching) as references of how long it takes to sense the outside world through human organs. Red, green and blue bars respectively stand for average reaction time of human sensing [74], ping latency of WiFi and wireless systems from 2G to 4G. From the figure, 4G LTE has reached the comparative latency level

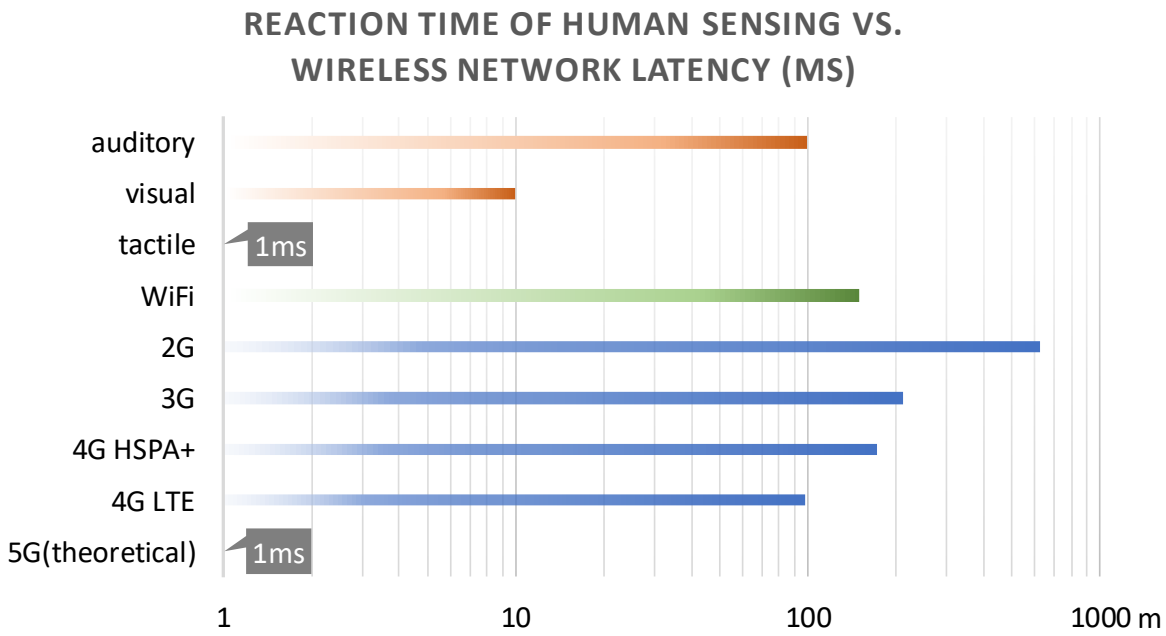


Fig. 3.1 Reaction time/latency of human sensing and different generations of wireless communications

of auditory about 100 ms which means we have already achieved the so-called auditory Internet. And for 5G, to reduce the latency of 4G by two orders of magnitude, we skip the stage of visual Internet (10 ms) and set the final goal in Tactile Internet (1 ms).

Proactive in-network caching, which has been receiving great attention in related researches on information centric networking (ICN), is assumed as one of the promising candidate technologies for latency reduction in next generation wireless communication systems [62] [72]. There are mainly four types of caching schemes, local caching, device-to-device (D2D) caching, small base station (SBS) caching and macro base station (MBS) caching.

Fig. 3.2 gives the four different in-network caching schemes. Local caching refers to caching at the user devices themselves while being requested the same content from the second time [8]. D2D caching bases on D2D communications within small cells [12]. SBS caching and MBS caching are available in respective base stations [31]. Each scheme has its own technical details and almost no overlap of four schemes exists in the actual application scenarios. As a result, in this part, I try to propose a hybrid edge caching scheme to reduce latency and optimize the overall energy efficiency of edge caching. Our target is to integrate the four existing schemes taking effect in different parts of the network for performance improvement and pave the way for Tactile Internet in the near future.

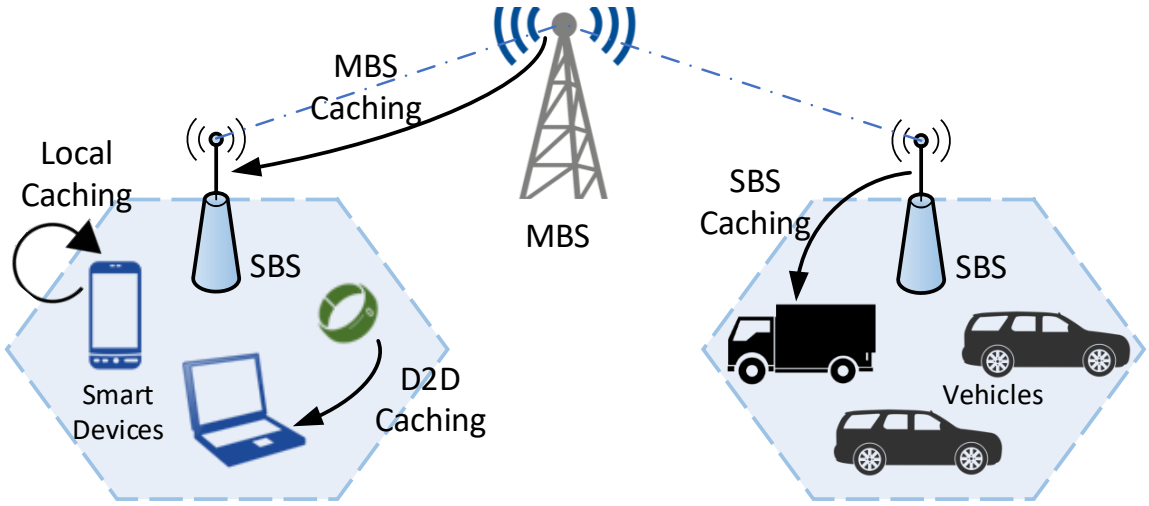


Fig. 3.2 Different types of proactive in-network caching schemes

3.2 Problem Formulation

In this section, I design a 3-tier heterogeneous network structure as the experimental scenario for simulate in-memory edge caching for big data. As shown in Fig. 3.3, from top to bottom a 3-tier network model can be described as follows.

- *Server Tier:* in this tier multiple servers play the tole of cloud data centers, each file is only stored in one of the servers;
- *Edge Tier:* in this tier routers are used both as forwarders and edge devices which can cache files in packets passed by;
- *User Tier:* a tier made of user nodes that keep moving randomly and requesting files originally stored in servers or cached in routers.

3.2.1 System Outline

In my network structure design, user nodes are moving in the *RWP* model which is one of the most popular mobility models applied in mobile ad hoc network (MANET) [7]. In Fig. 3.3, user nodes (u_1, u_2, ..., u_n) in *User Tier* send packets to request files at random time intervals and move to next positions under normal distribution before next sending. Each file, in an unfixed size, is randomly saved in one of the servers (s_1, s_2, ..., s_n) in *Server Tier*. Then in *Edge Tier*, routers (r_1, r_2, ..., r_n) as edge devices will check if the needed files are already cached in storage before forwarding to neighbor r or upward to s. Since information or contents are easy to be outdated, once the original files in *Server Tier* being

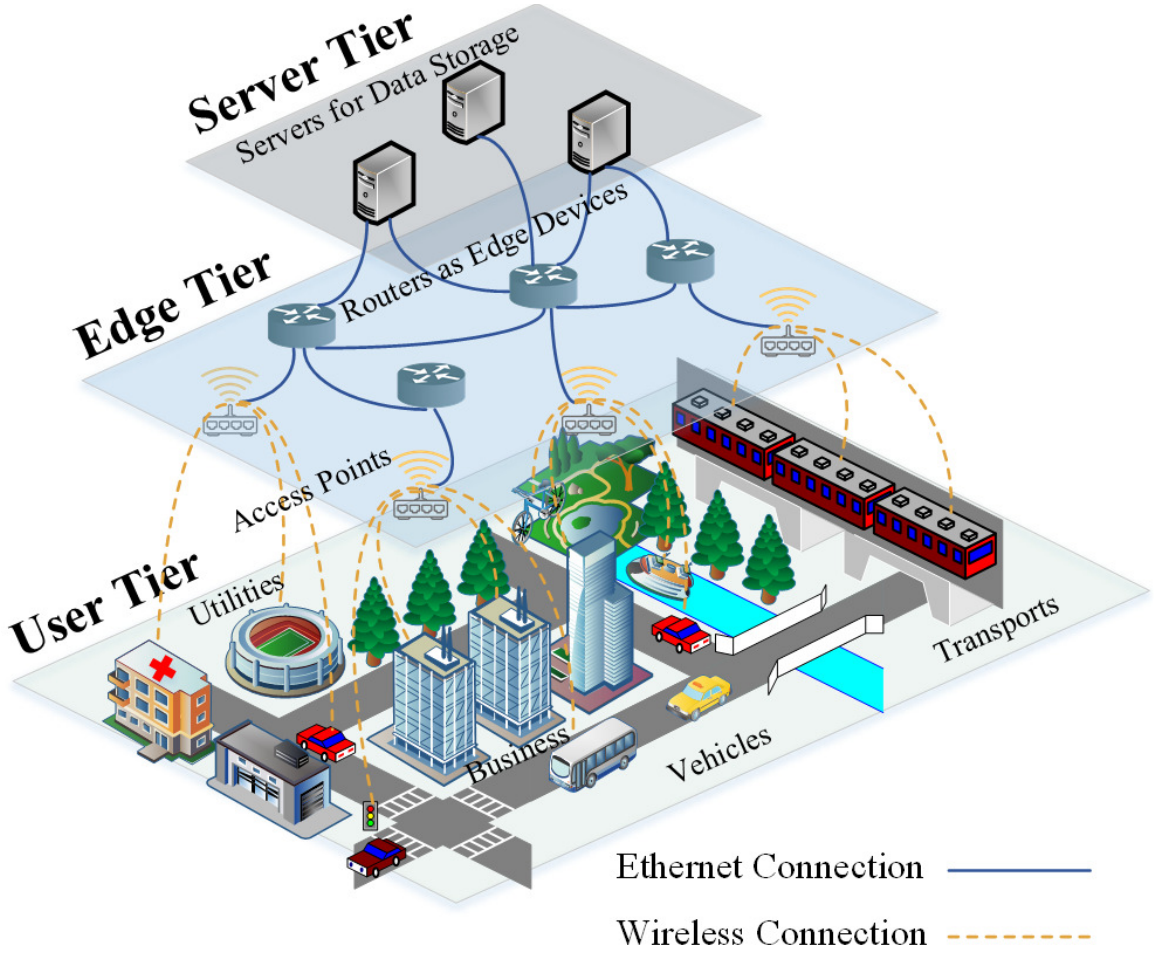


Fig. 3.3 A 3-tier heterogeneous network structure

modified or deleted, all cached copies become useless. That is to say, edge caching need to consider Time to Live (TTL) to make sure that most requests be satisfied with unexpired files [95]. Transmissions between *User Tier* & *Edge Tier* are wireless broadcasting, those inside *Edge Tier* are wired broadcasting and those between *Edge Tier* & *Server Tier* are wired point-to-point.

Normally RAM is associated with volatile types of memory whose data storage would be lost when power is off. That is to say, in-memory may not support long time storage which can be suitable for scenarios of edge caching. As a result, I consider both TTL for cached data and consistency of in-memory storage in designing caching methods.

3.2.2 Performance Metrics

In face of mass date generated by billions of IoT devices, I always prefer less energy consumption on data transmission and processing. For this reason, edge caching aims at reducing repeated data transmission from original servers which means unnecessary energy consumption on packet delivery between *Edge Tier* and *Server Tier* can be saved. Moreover, caching itself also consumes extra energy while keeping RAM or disk memory running. To determine and sum up the overall cost of the entire simulation on the 3-tier network architecture, I consider 3 parts. The part to maintain all devices in 3 tiers is not expressed in the equation since it is a fixed cost and can only be reduced by shutting down some devices [25]. In summary, I use two equations to present the calculation of total energy consumption.

$$E_{total} = E_{cache}(t) + E_{trans}$$
$$E_{cache}(t) = \omega \sum_{i=1}^{n} (s_i^{cache} t_i)$$

in which $E_{cache}(t)$ and E_{trans} respectively stands for caching energy cost and transmission energy cost. $E_{cache}(t)$ is running time correlated and can be calculated by energy consumption per byte ω. n represents the times when the current caching size of all r in *Edge Tier* is changed. Thus I sum up the n products of variational caching sizes s_i^{cache} and their duration t_i.

$$\begin{aligned} E_{trans} =& E_{send} + E_{recv} \\ =& \sum_{j=1}^{x} (m_{send} s_i^{trans} + b_{send} + m_{recv} s_i^{trans} + b_{recv}) \end{aligned}$$

Comparatively, E_{trans} has no relation to time and only depends on size of data being transmitted s_i^{trans}. Here I separately calculate the energy consumption while devices in three tiers sending and receiving packets. Moreover, as a heterogeneous network model, communications between *User Tier* & *Edge Tier* and among *r* inside *Edge Tier* are regarded as wireless while those from *Edge Tier* to *Server Tier* and backward are Ethernet transmissions. *m* and *b* are linear coefficients obtained from experimental results [23].

To figure out the total energy cost as close as possible to the practical situation, I take some more details into consideration. First, different bit rates of wireless and Ethernet transmissions. Second, the wave propagation speed, I respectively choose speed of light and thick coax as the communication media for wireless and wired.

Besides energy consumption, I also pay attention to how edge caching help reduce workload on end servers. I add a backhaul rate as another metric to test what is percentage may in-memory edge caching takes in completing the task of fetching files from servers across tiers.

3.2.3 In-network Content Caching

To deploy hybrid edge caching in a Tactile Internet network architecture, and achieve the target of reducing end-to-end latency and improving energy efficiency, first I design a 3-tier network model.

As shown in Fig. 3.4, from top to bottom there are Server Tier, MBS Tier and SBS Tier. Server Tier is made up of central cloud servers and storage units. As the top tier, Server Tier has the absolute advantage in processing and storage capabilities. Besides, Server Tier owns a complete content library including all original files is the only tier that can handle user requests without regard to in-network caching. MBS Tier consists of macrocells served by cell sites with high signal transmission power. I use tower symbols to represent macro base stations. The light red oval area is used only to identify the current tier which means that the relative positions of MBSs in the figure does not equal to the actual distances. For SBS Tier, I use multiple hexagons as small cells. Compared with MBSs, SBSs provide low-powered cellular radio access. As mobile users (smart devices, vehicles), when I move across the signal scope of small cells, our requests will then be answered by the current SBSs. Both MBS Tier and SBS Tier can provide edge computing service to mobile users.

In this model, $M = \{m_1, m_2, ...\}$ and $S = \{s_1, s_2, ...\}$, *m* and *s* respectively stands for single MBS node and SBS node. Mobile users are $U = \{u_1, u_2, ...\}$ who send out requests for files within content library. Let content library and files in it be denoted by $CL = \{cl_1, cl_2, ..., cl_{n_{CL}}\}$ which are sorted by popularity. n_{CL} is the number of files in *CL*. As a result, the popularity of one of the n_{CL} files can be expressed as [64]

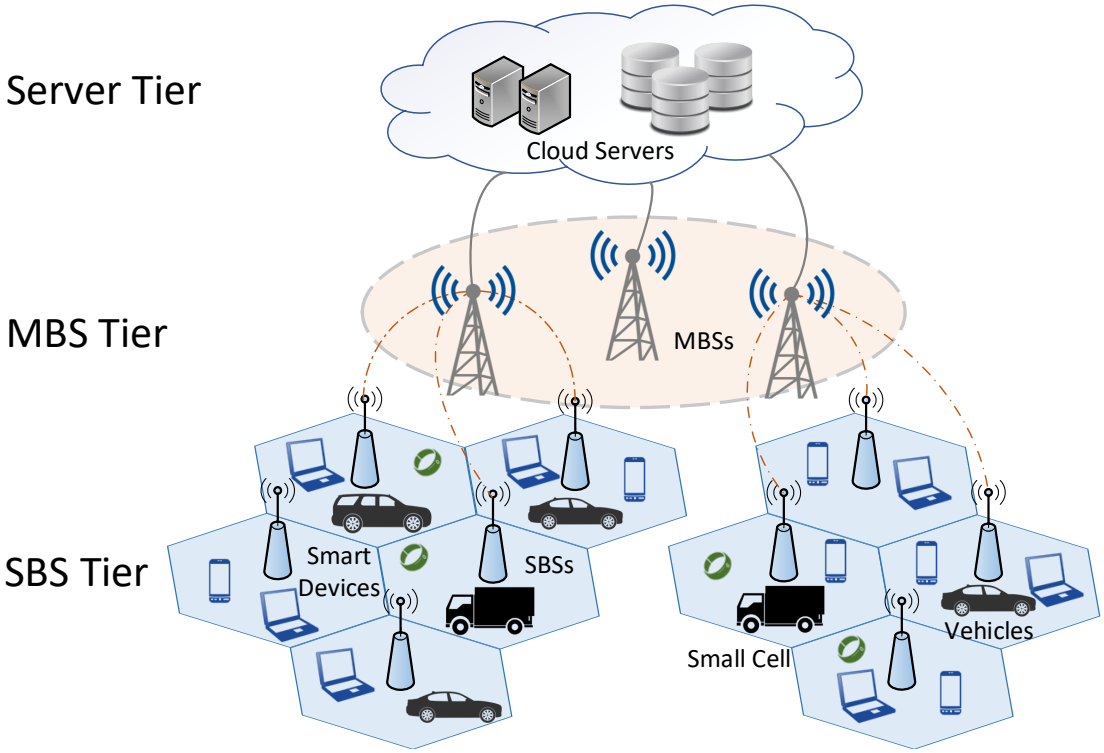


Fig. 3.4 A 3-tier heterogeneous network model for hybrid edge caching

$$p_{cl_k}(k,x,n_{CL}) = \frac{1/k^x}{\sum_{i=1}^{n_{CL}}(1/i^x)} \tag{3.1}$$

where $k \in \{1,2,...,n_{CL}\}$ stands for the rank or we can say it is the kth popular file in CL. x is value of the exponent which characterizes the Zipf distribution. Here I use it in describing the popularity of files in content library which can help arrange the limited storage capacity of each tier when designing the caching method. That is, we can obtain the popularity (frequency of occurrence) of a given file in CL only from its rank (k) by Equation (3.1).

When u_i sends a request for cl_k, firstly u_i will check if cl_k already be cached locally. Then within the current small cell, other users may also check if they have it. In the case that both local caching and D2D caching can not come in handy, the current SBSs and MBS will take turns to search cl_k in respective storage units. For local caching and MBS caching, I only consider one edge node, the current user device itself and macro base station. D2D caching needs to traverse all neighbor devices as edge nodes. And for the case of SBS caching, we may get help from all the SBSs within the same MBS.

To calculate the probability that any u_i can get cl_k from cache, I need to consider the four caching methods separately. Here I use $EN = \{en_1, en_2, ...\}$ to represent edge nodes in one of the four methods. That is, the EN can be one of the {local, D2D, SBS, MBS} which

means edge nodes here include user device itself (local), user devices nearby (D2D), SBSs and MBSs. A_{EN} is the total coverage area of EN and C_{EN} is the total storage capacity for caching (or how many files can be cached at most), respectively. Assume the number of occurrences that cl_k is not cached in EN within A_{EN} follows a homogeneous Poisson point process [4] [29], I have the cumulative distribution function (CDF) that cl_k is not cached in any en

$$F_{cl_k}^{EN}[no\_cache] = e^{-\lambda_k A_{EN}} \tag{3.2}$$

where λ_k is the intensity or arrival rate in homogeneous Poisson point process. Here λ_k stands for the expected number of edge nodes that have the cl_k per unit area which means $\lambda_k = \rho_{EN} \cdot P_k$. P_k is the probability that cl_k is cached in EN, and ρ_{EN} is the spatial density of EN in A_{EN}. As a result, the probability that cl_k is cached in at least one en should be

$$F_{cl_k}^{EN} = 1 - F_{cl_k}^{EN}[no\_cache] = 1 - e^{-\rho_{EN} P_k A_{EN}} \tag{3.3}$$

With Equation (3.1), the total probability that u_i can be satisfied by edge caching is

$$\begin{aligned} F^{EN} &= \sum_{k=1}^{n_{CL}} p_{cl_k}(k,x,n) \cdot F_{cl_k}^{EN} \\ &= \sum_{k=1}^{n_{CL}} \frac{k^{-x}(1 - e^{-\rho_{EN} P_k A_{EN}})}{\sum_{i=1}^{n} i^{-x}} \end{aligned} \tag{3.4}$$

n_{CL} is number of files in CL. Thus, the probabilities of four different caching methods are

$$F^{local} = \sum_{k=1}^{n_{CL}} \frac{k^{-x}(1 - e^{-P_k^U})}{\sum_{i=1}^{n} i^{-x}} \tag{3.5}$$

where P_k^U is the probability that cl_k is cached in U. Since I can use $\rho_{EN} \cdot A_{EN}$ to express the number of en within the coverage area, in the case of local caching, the value of $\rho_{EN} \cdot A_{EN}$ becomes 1.

$$F^{D2D} = \sum_{k=1}^{n_{CL}} \frac{k^{-x}(1 - e^{-\rho_U P_k^U A_{D2D}})}{\sum_{i=1}^{n} i^{-x}} = \sum_{k=1}^{n_{CL}} \frac{k^{-x}(1 - e^{-\rho_U P_k^U \pi r_U^2})}{\sum_{i=1}^{n} i^{-x}} \tag{3.6}$$

where ρ_U is the spatial density of U, A_{D2D} is the signal coverage area of D2D communications and r_U is the coverage radius of a single user device, respectively.

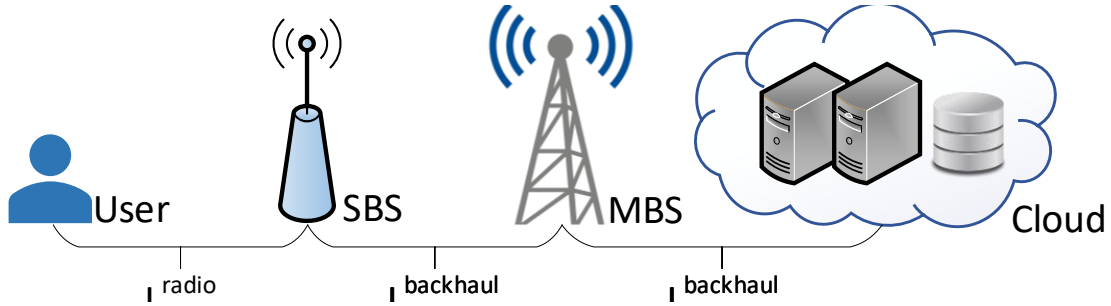


Fig. 3.5 End-to-end latency

$$
\begin{aligned}
F^{SBS} &= \sum_{k=1}^{n_{CL}} \frac{k^{-x}(1-e^{-\rho_{SBS}P_k^{SBS}A_{MBS}})}{\sum_{i=1}^{n} i^{-x}} \\
&= \sum_{k=1}^{n_{CL}} \frac{k^{-x}(1-e^{-\rho_{SBS}P_k^{SBS}\pi r_{MBS}^2})}{\sum_{i=1}^{n} i^{-x}}
\end{aligned}
\tag{3.7}
$$

where ρ_{SBS} is spatial density of S. P_k^{SBS} is probability that cl_k is cached in S. As shown in Fig. 3.4, each user can have one MBS and multiple SBSs for service. After an SBS receiving the original request from user, it will forward to other SBSs within the same MBS before sending upwards. That is, A_{EN} here equals to the coverage area of multiple SBSs under the same MBS. Or we may say that a user can communicate with all the SBSs under one MBS, but some of them undertake more forwarding hops.

$$
F^{MBS} = \sum_{k=1}^{n_{CL}} \frac{k^{-x}(1-e^{P_k^{MBS}})}{\sum_{i=1}^{n_{CL}} i^{-x}} \tag{3.8}
$$

where P_k^{MBS} is the probability that cl_k is cached in M. Similar with local caching, since $A_{EN} \cdot \rho_{EN}$ can express the number of *en* within the coverage area, for MBS caching, there is only one MBS to provide service, then the value of $\rho_{MBS} \cdot A_{MBS}$ becomes 1.

The two main metrics I choose for performance evaluation are end-to-end latency and energy efficiency. End-to-end latency includes the time cost on two-way transmission from user to cloud server. And once the user request can be satisfied by one of the four caching methods, the corresponding part can be saved. Energy efficiency is calculated by the total size of the requested files divided by energy consumption.

$$
L^{e2e} = 2 \times (L^{radio} + L_{2,3}^{backhaul} + L_{1,2}^{backhaul}) \tag{3.9}
$$

L^{e2e} stands for the end-to-end latency. When no cache is found for user request under the current network conditions, the results are shown in Fig. 3.5 and Equation (3.9). L^{radio}

and $L^{backhaul}$ respectively stands for transmission time between users/edge nodes (SBSs and MBSs), and edge nodes/central cloud servers. For L^{radio}, the communications is wireless broadcast. The case of $L_{2,3}^{backhaul}$ between SBSs and MBSs is wireless point-to-point. The case of $L_{1,2}^{backhaul}$ is wired point-to-point. With edge caching, actually we can save part of the latency on backhaul or routing. For example, the latency of MBS caching L_{e2e}^{MBS} is $2 \times (L^{radio} + L_{2,3}^{backhaul})$.

As a result, the problem of reducing total latency in the 3-tier network model is

$$\begin{aligned} &minimum && \mathscr{E}[F^{EN'} L^{EN'}] \\ &subject\ to && \sum_{k=1}^{n_{CL}} b_k^{EN} |cl_k| \leq C_j^{EN} \\ & && \forall j \in \{1,2,...,n_{EN}\} \\ & && b_k^{EN} \in \{0,1\} \end{aligned} \tag{3.10}$$

where $F^{EN'}$ and $L^{EN'}$ are probability and end-to-end latency in answering one user request. The apostrophe after EN means that the case of requests not being answered by cache is also considered in the calculation of latency. $\mathscr{E}$ stands for the expected value. Here I need the results of end-to-end latency on the repetitions of user requests being satisfied by any caching method. To pursue the minimum, we have to make sure that the total size of cached files not exceeds the capacity C_j^{EN} in any edge node. The absolute value notation here means the size of cl_k in content library. I add a boolean variable b_k^{EN} to denote that cl_k is cached in en_j, or not.

Energy cost on edge caching is another aspect I consider in performance evaluation. The longer a user request travels, the more it may spend. Furthermore, the parts of forwarding hops in the routing procedures need extra consumption. As a result, energy consumed in end-to-end transmission during L^{e2e} can be expressed as

$$E^{e2e} = L^{e2e}(p^{tran}/\eta + p^{stat}) \tag{3.11}$$

E^{e2e} is the energy consumption of end-to-end transmission. p^{tran} and p^{stat} respectively represents transmission power and static power of all other circuit blocks in edge nodes as transmitter and receiver [89] [11]. η is the efficiency of transmit power amplifier.

Then I can calculate total energy efficiency EE [88] as

$$EE = \frac{R \cdot L_{total}^{e2e}}{\sum_{r=1}^{n_t} F^{EN'} E_r^{e2e}} = \frac{R \cdot \sum_{r=1}^{n_t} L_r^{e2e}}{\sum_{r=1}^{n_t} F^{EN'} L_r^{e2e} (p_r^{tran}/\eta + p^{stat})} \tag{3.12}$$

where R stands for the data rate of the overall network. n_t stands for the number of user requests. Thus, the product of R and L_{total}^{e2e} is the total size of data generated in edge caching.

As a result, the second problem of improving energy efficiency in the 3-tier network model is

$$\begin{aligned} &maximum \quad EE \\ &subject\ to \quad \sum_{k=1}^{n_{CL}} b_k^{EN} |cl_k| \leq C_j^{EN} \\ &\forall j \in \{1,2,...,n_{EN}\} \\ &b_k^{EN} \in \{0,1\} \end{aligned} \tag{3.13}$$

where EE is obtained by Equation (3.12). Limitation factors are the same with Equation (3.10).

Besides end-to-end latency and energy efficiency, I also take into consideration cache hit ratio. cache hit ratio here plays the role of showing how caching schemes can maximize the number of cache hits and minimize the number of misses.

$$Cache\ hit\ ratio = \frac{Cache\ hits}{Cache\ hits + Cache\ misses} \tag{3.14}$$

As shown in Equation (3.14), to calculate cache hit ratio, I need to respectively count the times that cache hits and misses within a given period of time. High cache hit ratio can show the probability that the user request is satisfied by cache.

3.3 Algorithm Design

In this section, I propose two caching methods based on different TTL designs, TTL of requested times (TRT) and caching time (TCT).

TRT is counting how many times a cached file being requested by *u* in *User Tier*. When the needed file being found at any *s* in *Server Tier* and sent back, each *r* in the full path may check if it already has the copy of the file. If not, *r* will cache the file in its in-memory or disk. Later once a *r* receives request and finds the needed file in cached data, it may send back the copy and count the requested times of the cached file. If number of requested times reaches the maximum value being set, the cached file will be dropped. Accordingly in the calculation equation of $E_{cache}(t)$ of last section, s_i^{cache} is changed and a s_{i+1}^{cache} is needed.

Rather than counting requested times, the other caching method TCT keeps a timer for each cached file. Caching and routing follow the same rules of the first method, similarly when any timer reaches the maximum value, the cached file will be dropped.

In addition, both methods consider the volume of in-memory/disk storage, that is, if the next file to be cached exceeds the memory volume of r, before caching the newcome data I have to free up some space by popping out cached data. As result, to decide which one to drop when the volume of disk or in-memory is full, I apply four common cache replacement policies based on different ideas on how to improve the performance of edge caching. Another point to note is, the drop behaviors in cache replacement policies have no relation to TTL designs since both are needed to guarantee the usability of cached data, that is, still alive and within the capacity of the current r.

First, First In First Out (FIFO) policy regards volume disk/in-memory as a FIFO queue and drops the head of queue that gets pushed in the earliest when queue is full. Second, Least Frequently Used (LFU) policy does not consider the order of cached files and chooses the cached file with the least requested times currently. Third, similar to LFU, Least Recently Used (LRU) also focus on the cached files which are not so popular but chooses the least recently used one to drop. Last, the Random Replacement (RR) policy serves as a contrast to compare the performance of TTL designs with the other policies.

3.3.1 Hybrid Edge Caching Algorithm

To reduce the total latency and improve energy efficiency of edge caching, I integrate four existing schemes into a hybrid one which considers all the computing resources available within the network. Moreover, since the capacities of edge devices are limited, we have to discard parts of the cached data or files when memories are full. I need a suitable cache replacement policy to help each cached file do its best in saving as much time as possible on backhaul transmissions. My target is to ensure that files with high popularity can extend TTL by migrating between cache segements instead of coming out. As a result, when we reasonably occupy the cache capacity of each edge device, and improve the utilization of files during their cache TTL as a whole, time cost required for upward transmission will be greatly reduced. At the same time, energy efficiency given by Equation (3.12) and (3.13), which is calculated as the total size of the requested files devided by total energy consumption. And energy consumption (Equation (3.11)) is also calculed from time cost. Therefore, when I cut down the latency, energy efficiency can be improved correspondingly.

In the design of caching scheme, cache replacement policies refers to the algorithms choosing to keep or discard the cached items when cache capacity is full. As a result, each item in cache may own a time-to-live (TTL) which indicates the length of time it can stay in the cache memory. Actually, different cache replacement policies use different rules to define and calculate TTL. As one of the most common policies, first in first out (FIFO) maintains a FIFO queue and evicts the items entering first when any newcomer has no position. Least

recently used (LRU) monitors all the cached items and evicts the one used least recently. Random replacement (RR) chooses item to evict without considering any information about history or forecast. RR is often used as a benchmark for performance comparisons, but it can also have unexpected results in some cases. That is, no cache replacement policy is always the best. And what I want is to find the most suitable one for my caching scheme.

In the scenario of solving the two problems in the proposed 3-tier network model, to minimize end-to-end latency and maximize energy efficiency I need a cache replacement policy to distinguish the popularity of the files in *CL* and distribute limited capacities of edge devices to the most popular ones that may be requested frequently.

As shown in Algorithm 3, *req* is the requested file from user and *evict* is the file to be evicted from cache. Q_{seg1} and Q_{seg2} respectively stands for the queues of cached files in two segments with different priorities. $left$ means the rest of the cache capacity. *size* is the size of request file. $pop()$ and $push()$ are the pop and push functions. *head* is the cached file at the queue head. $any()$ is the function telling whether a file is cached in the queue. Here I propose a double segmented LRU cache replacement (S2LRU) policy for hybrid edge caching for Tactile Internet. The motivation of Algorithm 3 is to design a optimized cache replacement approach suitable for hybrid edge caching. Key idea of Algorithm 3 is that with more than one cache segments, files of different popularity are treating separately. Popular files will not be easily discarded, and relatively unpopular files will be replaced soon. I divide the total cache capacity into two prioritized parts. The Q_{seg1} with low priority is checked first, once the cache hits, the cached file will leave Q_{seg1} and move to higher Q_{seg2}. Then I check the Q_{seg2}, once the cache hits, the cached file come to the queue's tail. And for files cached in Q_{seg2}, once the left capacity ($C_{seg2}.left$) is not enough to accommodate any newcomer, files at the queue's head ($Q_{seg2}.head$) may have to walk out of Q_{seg2} and come back to Q_{seg1}. When the same situation occurs in Q_{seg1}, files will be evicted from the current *en*.

The proposed S2LRU takes advantage of several common policies. First, as shown in the title, I extend the TTL of cached files that are recently used and correspondingly make the files rarely used easy to expire. Second, least-frequently used (LFU) policy also plays a role in S2LRU since only files being requested more than one time may enter Q_{seg2}. Third, I choose double segmented LRU rather than triple (S3LRU) or higher because of the limited capacity of edge devices. Multiple segments may lead to fragmentation of the cache space, making the efficiency drop dramatically when processing large files. Time complexity of Algorithm 3 is $O(n_{CL})$ which stands for the worst case that all the files in *CL* are cached in this *en* and the requested one is found lastly.

Then I design a hybrid edge caching scheme based on the four types of methods shown in Fig. 3.2. My target is to coordinate all the computing resources available in network

Algorithm 3 Segmented LRU for Hybrid Edge Caching

$req, evict \leftarrow$requested file from user and evicted file from cache
$Q_{seg1}, Q_{seg2} \leftarrow$queues of two segmented cache capacity

if $any(Q_{seg1} = req)$ **then**
  $C_{seg1}.left \leftarrow C_{seg1}.left + req.size$
  $Q_{seg1}.pop(req)$
  while $C_{seg2}.left < req.size$ **do**
    $evict_{seg2} \leftarrow Q_{seg2}.head$
    $C_{seg2}.left \leftarrow C_{seg2}.left + evict_{seg2}.size$
    $Q_{seg2}.pop()$
    while $C_{seg1}.left < evict_{seg2}.size$ **do**
      $C_{seg1}.left \leftarrow C_{seg1}.left + Q_{seg1}.head.size$
      $Q_{seg1}.pop()$
    end while
    if $C_{seg1}.left \geq evict_{seg2}.size$ && $!any(Q_{seg1} = evict_{seg2})$ **then**
      $Q_{seg1}.push(evict_{seg2})$
      $C_{seg1}.left \leftarrow C_{seg1}.left - evict_{seg2}.size$
    end if
  end while
  if $C_{seg2}.left \geq req.size$ && $!any(Q_{seg2} = req)$ **then**
    $Q_{seg2}.push(req)$
    $C_{seg2}.left \leftarrow C_{seg2}.left - req.size$
  end if
else if $any(Q_{seg2} = req)$ **then**
  $Q_{seg2}.pop(req)$
  $Q_{seg2}.push(req)$
else
  while $C_{seg1}.left < req.size$ **do**
    $Q_{seg1}.pop()$
  end while
  $Q_{seg1}.push(req)$
end if

model in Fig. 3.4, and reasonably arrange the caching and discarding of files with different popularities under the Zipf distribution.

Algorithm 4 HECT: Hybrid Edge Caching for Tactile Internet

$u, u.in_sbs, u.in_sbs.in_mbs \leftarrow$current user, the SBS and MBS beyond current user
$n_{d2d} \leftarrow$number of other users within the signal scope of u
$n_{sbs} \leftarrow$number of SBSs within the same MBS

1: **if** $find(u.cache = req)$ **then**
2: calculate L and EE by local caching
3: **else**
4: **for** $i \leftarrow 1$ to n_{d2d} **do**
5: **if** $find(u_i^{d2d}.cache = req)$ **then**
6: calculate L and EE by D2D caching
7: cache req at u
8: **end if**
9: **end for**
10: **if** $find(u.in\_sbs.cache = req)$ **then**
11: calculate L and EE by SBS caching
12: cache req at u
13: **else**
14: **for** $j \leftarrow 1$ to n_{sbs} // routing among all SBSs under the same MBS **do**
15: **if** $find(sbs_j^{u.in\_sbs.in\_mbs}.cache = req)$ **then**
16: calculate L and EE by SBS caching
17: cache req at u and all sbs in the forwarding route
18: **end if**
19: **end for**
20: **if** $find(u.in\_sbs.in\_mbs.cache = req)$ **then**
21: calculate L and EE by MBS caching
22: cache req at u and $u.in\_sbs$
23: **else**
24: calculate L and EE by no caching
25: cache req at u, $u.in\_sbs$ and $u.in\_sbs.in\_mbs$
26: **end if**
27: **end if**
28: **end if**

As shown in Algorithm 4, u is the current user sending out request. $u.in\_sbs$ and $u.in\_sbs.in\_mbs$ are the SBS first receiving the user request and the MBS. n_{d2d} is the number of other users that can communicate with the current user in D2D mode. n_{sbs} is the number of SBSs within the signal of the same MBS. $find()$ is the function telling if the requested file can be found in the cache, TRUE for found and FALSE for not. The motivation of Algorithm

4 is to set rules for caching on any of the four parts based on the replacement policy in Algorithm 3. The key idea of Algorithm 4 is that from a global perspective, making full use of each device in the network architecture that can provide cache storage.

After a user u sending out a request *req*, I may prepare for the time-saving and energy efficient caching service from the perspective of the overall network provider. The most basic priority principle is the relative physical distance from users. First, check if the wanted file is already in local storage. Once it can be satisfied, I regard this case as no transmission latency. Second, *req* will be forwarded to nearby users within the scope of D2D communication. Third, when no cache can be found in any user, we may seek help from SBS Tier. Line 14 in Algorithm 4 includes the procedure of routing among SBSs under the signal range of the same MBS. In this way I can limit the routing times and avoid meaningless multi-hop forwarding. Fourth, once SBS caching is also failed, we need to upload the request to MBS Tier. Finally, if user request can not be satisfied by cache, we have to seek help from cloud server. Together with cache replacement policy shown in Algorithm 3, I may leave a copy at the each node of the route while downloading the requested file. Time complexity of Algorithm 4 is $O(n_{d2d} + n_{sbs})$.

3.4 Simulation and Analysis

3.4.1 In-memory Edge Caching

In the subsection, I carry out experimental simulations to compare the performance of edge caching in in-memory and conventional disk memory under two TTL designs.

The simulation scenario is a 10 km^2 square open area, and I set 2,000 user nodes in *User Tier* with random initial positions. Each user node randomly moves to next position under normal distribution after sending request packet to fetch one of the 200 files originally stored in one of the five servers in *Server Tier*. I use 100 routers in *Edge Tier* as edge devices to cache the files locally. Four cache replacement policies are applied in designing edge caching methods.

As shown in Table 3.1, the setups of experiment include bit rate & wave propagation speed of packet transmission, *Edge Tier*'s device settings and energy consumption coefficients of both transmission and caching. As a result, to calculate the total energy consumption E_{total}, I have to count the number of packets and sum up their sizes, then compute the time cost from transmission and reading/writing from disk/in-memory. I carry out 10 rounds of simulations in two designed edge caching methods & four cache replacement policies. I

Table 3.1 Experimental settings of in-memory edge caching

Bit Rate of Transmission	
Wireless (802.11ad)	6.8 *Gbps*
Ethernet	10 *Gbps*
Wave Propagation Speed of Transmission	
Wireless (air)	c (speed of light)
Ethernet (thick coax)	0.77 c
Device Settings of Edge Tier	
MTR of Disk (SSD)	2500 *MB/s*
MTR of In-Memory (DDR3)	6400 *MB/s*
Disk Volume	256 *MB*
In-Memory Volume	25 *MB*
Energy Consumption Coefficients	
	10^{-8} *J/MB* 10^{-6} *J*
Broadcast Send	2.1 $\times$ size + 272
Point-to-Point Send	0.48 $\times$ size + 431
Broadcast Receive	0.26 $\times$ size + 50
Point-to-Point Receive	0.12 $\times$ size + 316
Power Consumption of Caching	8 $\times 10^{-3}$ *W/MB*

repeat the part of each method & policy 10 times and get the average results. The simulation environment is *MATLAB R2017b*.

Fig. 3.6 shows the simulation result of total energy consumption of TRT method in four cache replacement policies. Blue, red, green and yellow represent the policies of FIFO, LFU, LRU and RR. The solid lines and dotted lines respectively stand for caching in in-memory storage and disk storage. From the patterns of eight broken lines, I can know that the overall energy consumption of traditional disk caching is larger than in-memory caching. In the 10 rounds of simulation, the trends of disk and in-memory methods are also different. Energy consumption of disk in TRT varies like a checkmark symbol which firstly falls down a little and then after a smooth transition period increases rapidly to a high value. My in-memory method in TRT shows a similar trend at the first half from round 1 to 5 but become steady

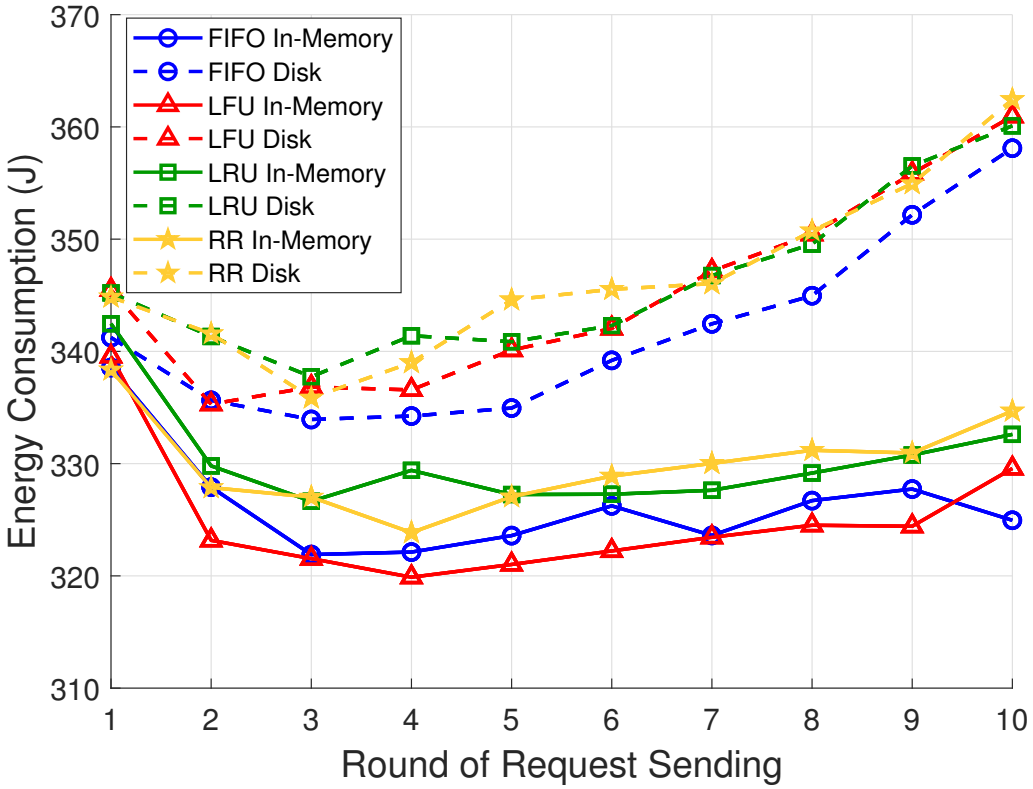


Fig. 3.6 Total energy consumption of edge caching in TTL of Requested Times (TRT)

after then which means energy consumption on edge caching may finally turns into a stable state. The differences of the trends are in line with my expectation since disk which own larger storage volumes and lower MTR call for more unit energy consumption and longer operation time. The positive correlation of the two factors lead to the continuing growth of total consumption.

Different cache replacement policies also have respective functions on the results of total energy cost. From the four lines of disk caching in the top half of the figure I can see three of them, LFU, LRU and RR fluctuate until coinciding in the end. Only blue line of FIFO policy shows a slight advantage in total energy consumption which means a simpler rule maybe more suitable for disk caching by TRT. Then from the other four colorful lines of in-memory caching, although still some changes in order of energy cost values, I may get the overall order of four policies as RR, LRU, FIFO and LFU. RR policy does suffer from random choice of dropping cached files which costs more energy in total. The special case of round 4 when LRU exceeds RR may be explained by some occasional error when RR just drops the right files which are not requested much later. LFU policy seems to own the optimal performance in in-memory caching by TRT because of the most suitable dropping choice which focuses on reserving popular files probably from the server directly connected to the current router with only one hop and dropping a most unpopular one which may come from a remote server after several hops of forwarding. Green line applying LRU also pays attention to the popularity of cached files but directly considers the caching time and drops the one not being requested of the longest interval. However, compared to LFU, the practice of LRU may face exceptions like some popular files being dropped due to the timing that the other files are just being cached or requested.

The experimental result of the other TTL design is shown in Fig. 3.7. In comparison with TRT, the trend of disk's four colorful lines of different cache replacement policies is relatively smoother, gaps between each two of the policies are visible. However, the order of lines in different policies us some kind of abnormal. Green line applying LRU shows the highest total energy consumption and RR policy even achieve high performance in value. In my point of view, since disk have far larger storage volume than in-memory, although more files are cached live for enough time, not many of them finally waited for a request or even get a remote request which may save no time than fetching the original file from servers. As a result, replacement policies trying to delay the time for dropping popular files may in turn cause more energy cost.

The patterns of in-memory by TCT are similar with TRT. RR policy fluctuates more violent. LRU seems to improve its performance compared to TRT and even shows a certain degree of convergence with FIFO. In my point of view, replacement policy here share the

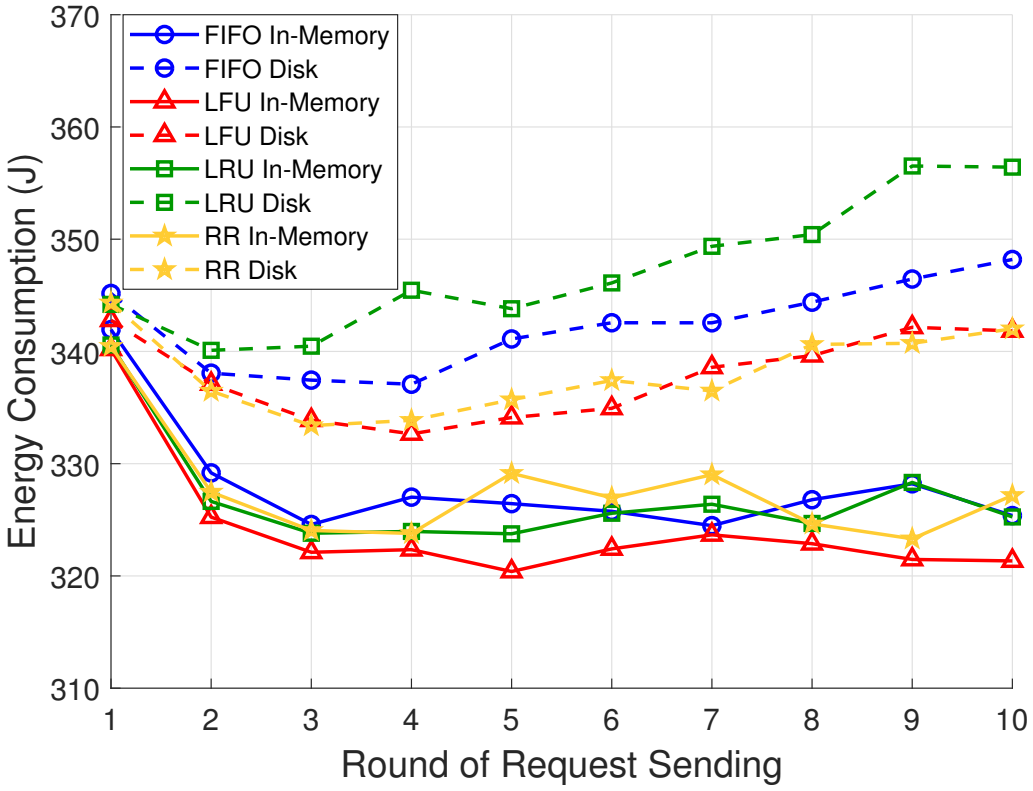


Fig. 3.7 Total energy consumption of edge caching in TTL of Caching Time (TCT)

same method with TCT by dropping cached files by keeping check the their timestamps. The collective effect may lead to some promotion in efficiency.

To make clear how TRT and TCT methods may help reallocate the whole workload and achieve energy efficient caching, I add the experimental results of backhaul rates. As the values of results in different cache replacement policies are similar to each other, here I choose FIFO as an example. I change the set maximum requested times & caching time of each single file being cached in routers as edge devices and get two 3-D surface graphs (values of maximum caching time are corresponding values in simulation).

Under the same coordinate axis, Fig. 3.8 and 3.9 show the variation of backhaul rates by rounds of packet sending and maximum requested times/caching time. First in Fig. 3.8, when number of requested times is small, regardless of which round of packet sending, about more 70 percent of requests still have to reach *Server Tier* to get needed files. The other side, numerical range of the same maximum times in 10 rounds changes not so great which means although stay high in value with small requested times number, backhaul rate of TRT method can rapidly reach steady state as number increases. Finally more than 70 percent of requests can be satisfied by in-memory edge caching.

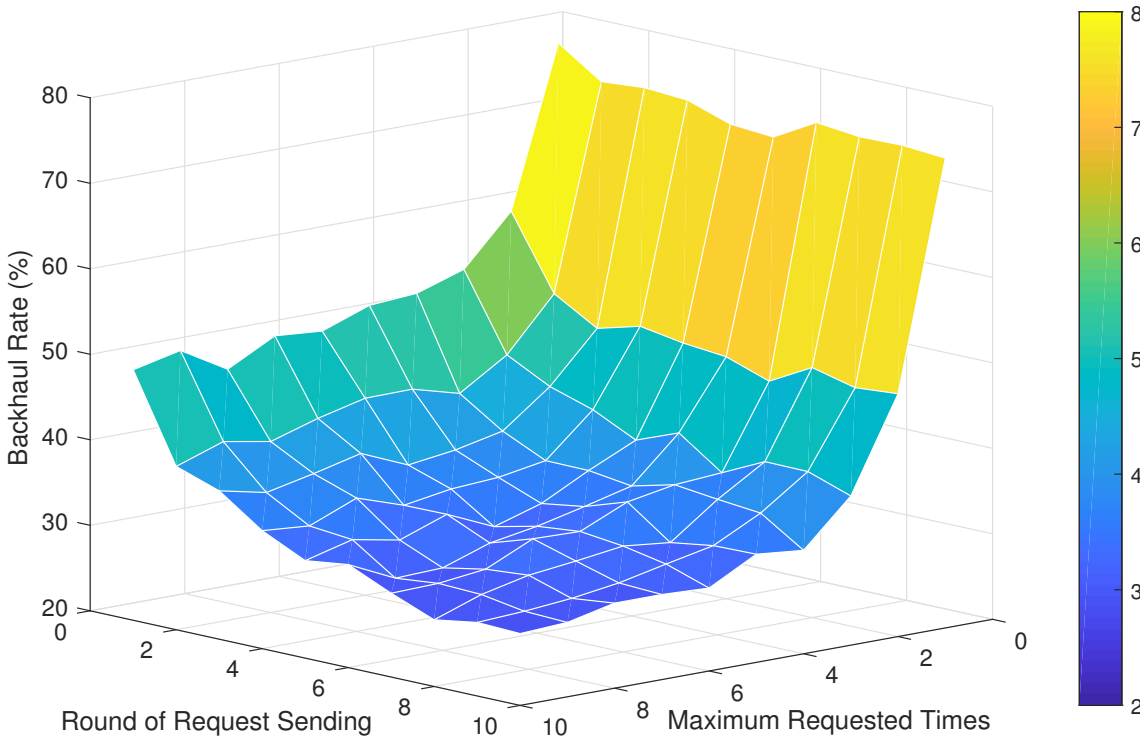


Fig. 3.8 TTL of Requested Times (TRT)

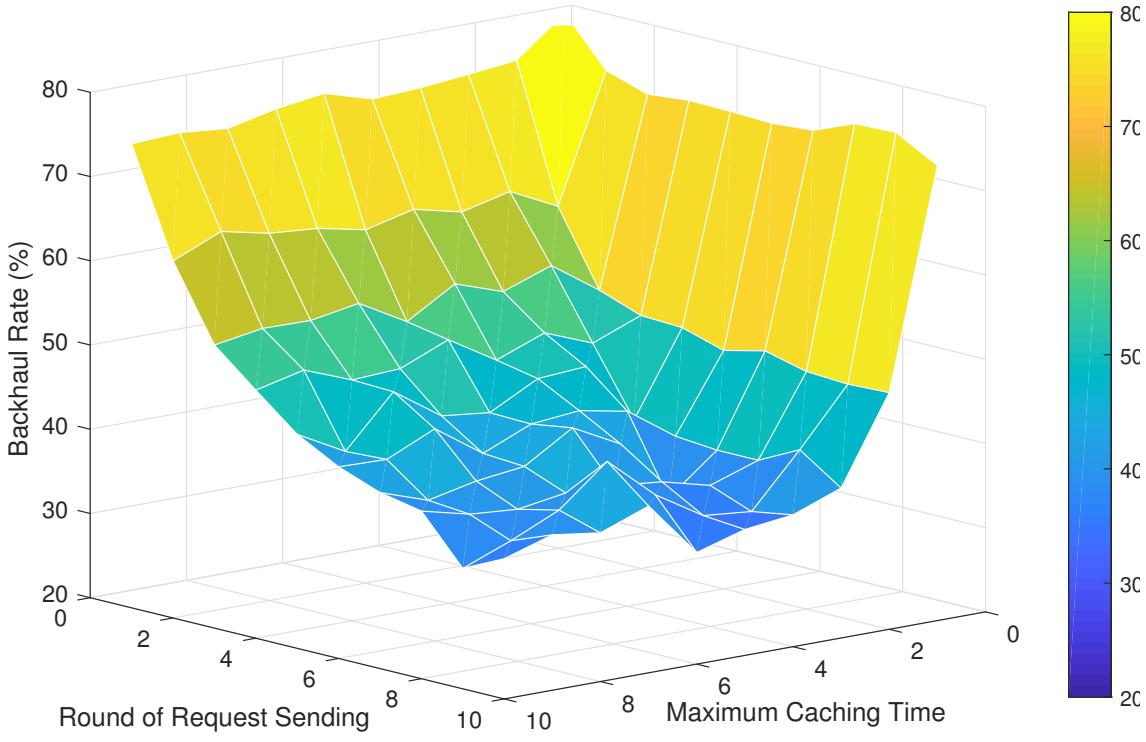


Fig. 3.9 TTL of Caching Time (TCT)

Table 3.2 Experimental settings of hybrid edge caching

Parameter	Value
Simulation area	$950\sqrt{3} \times 1300\ m^2$
Number of files in *CL*	200
Number of users/SBSs/MBSs	$1000/100/4$
Cache capacity of users/SBSs/MBSs	$2/10/50\ MB$
Signal scope radius of D2D/SBS	$20/100\ m$
Date rate in 5G	$10\ Gbps$
Wave propagation speed	**c** (speed of light)
User power in D2D communication	$100\ mW$
SBS/MBS power	$2/100\ W$
Distance between adjacent SBSs	$100\sqrt{3}\ m$

Then in Fig. 3.8, the main difference of the pattern is the variation when I continue to send packets. That is, no matter how long the caching time is set, *Server Tier* still has to take charge of most workload at the first rounds. However, with more and more files being fetched originally from end servers and then cached in in-memory of passed routers, TCT method also can reach a comparatively ideal low backhaul rate, though higher in value than TRT as well as some fluctuations.

3.4.2 Hybrid Edge Caching

In this subsection, I carry out experimental simulations to compare and analyze the performance of my proposed cache replacement policy and hybrid edge caching scheme with existing methods.

As shown in TABLE 3.2, there are 1000 users, 100 small cells and 4 macro cells in a rectangular open area. In *CL* there are 200 files. The cache capacities of users, SBSs and MBSs are 2, 10, 50 *MB*. Sizes of the files in *CL* are uniformly distributed random numbers in the interval (0,1) *MB*. Signal range radius of D2D communication between users is 20 m, and radius of small cell is 100 m. According to the IMT-2020 5G specifications [71] which gives the peak data rate of 20 Gbps and user experienced data rate of 1Gbps in enhanced Mobile Broadband (eMBB) scenario, I set the 5G data rate in the simulation as 10Gbps. User power in D2D communication is 100 mW, which equals to 20 dBm. For SBS and MBS, 2 W and 100 W repectively equals to power level of 33 dBm and 50 dBm.

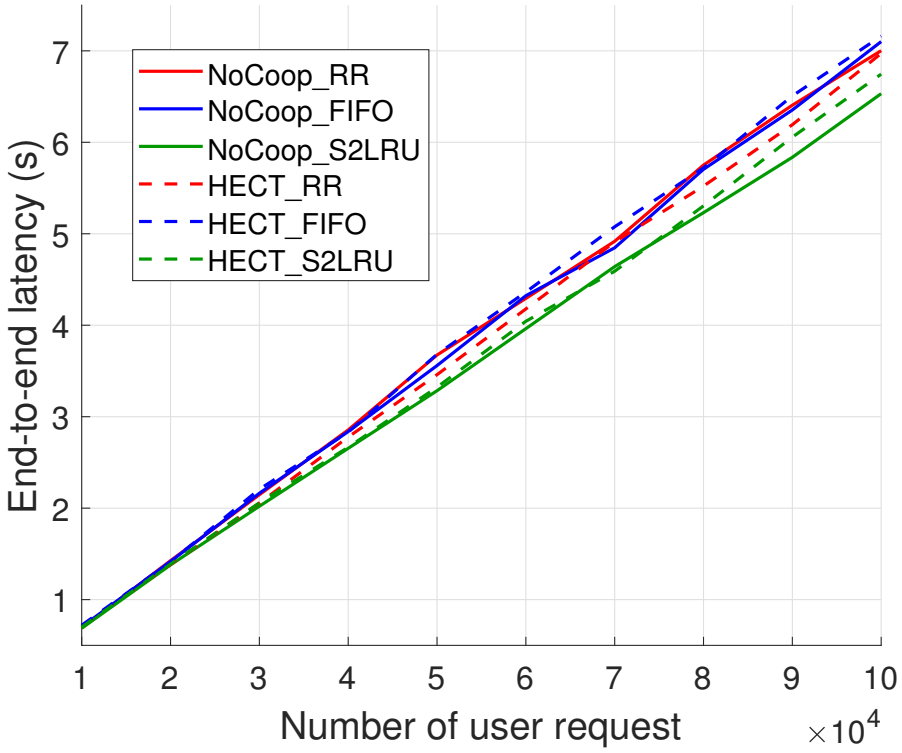


Fig. 3.10 Simulation results of end-to-end latency: Different numbers of user requests

As comparison, I choose FIFO, Random Replacement (RR) and a NoHybd cache scheme. FIFO plays the role of a low-overhead replacement policy, which can be easily achieved by setting FIFO queues. RR is considered as a simple one which discards randomly and requires no access history. NoHybd stands for the traditional edge caching scheme. Regardless of the combination of the basic methods, NoHybd only caches during the vertical forwarding procedure. That is, no packet delivery happens among devices with the same network location including users and SBSs. I hope to use some extreme settings to evaluate the performance of cache replacement policies and cache schemes through multiple metrics, and provide valuable references for the realization of Tactile Internet in the near future.

3.4.3 Latency and Energy Consumption

Fig. 3.10 and 3.11 give the simulation results of end-to-end latency. In Fig. 3.10, I add together end-to-end latency of the ten groups in different user request numbers. First from the overall polyline trends of matches between replacement policy and cache scheme, the gaps are not large. Especially when number of requests is small, we can hardly judge the pros and cons of different methods through current experimental conditions. And as the number of user requests increases, there are gradually subtle changes. Caching schemes in S2LRU

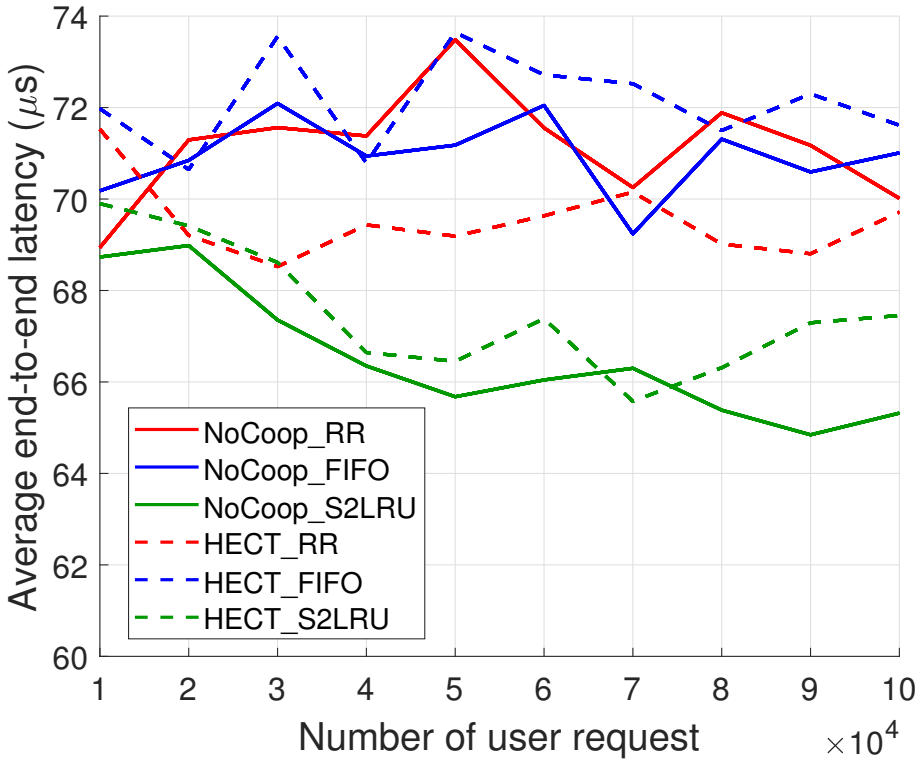


Fig. 3.11 Simulation results of end-to-end latency: Average results of single request

policy have certain advantages in numerical values. Actually my designed hybrid scheme is at a slight disadvantage in terms of latency which may be explained by the extra attempts looking for cached copies. That is, there exists no consumption on D2D communication and routing among SBSs under the same MBS.

Some further evaluation can be made from the Fig. 3.11. I average the results of Fig. 3.10 by single request. As shown in this subfigure, polylines of NoHybd_RR, NoHybd_FIFO and HECT_FIFO are in a state of fluctuation as well as high value. HECT_RR is relatively stable, and lower in the overall value. Both schemes in S2LRU first decline and then stabilize as the number of user requests increases. As a result, I may judge that S2LRU can achieve better performance in reducing latency than regular replacement policies, especially when facing large request number.

Fig. 3.12 and 3.13 show the results of energy consumption. Compared with Fig. 3.10 and 3.11, energy cost on processing requests from 1000 users show different patterns. HECT with three replacement policies saves more energy than NoHybd, respectively. HECT_FIFO still performs poorly, which can be explained that for fast-accumulated copies, always discarding the ones first coming in is not only unable to take care of the popular files, but may also be counterproductive. In particular, as edge devices do not have much capacity for caching,

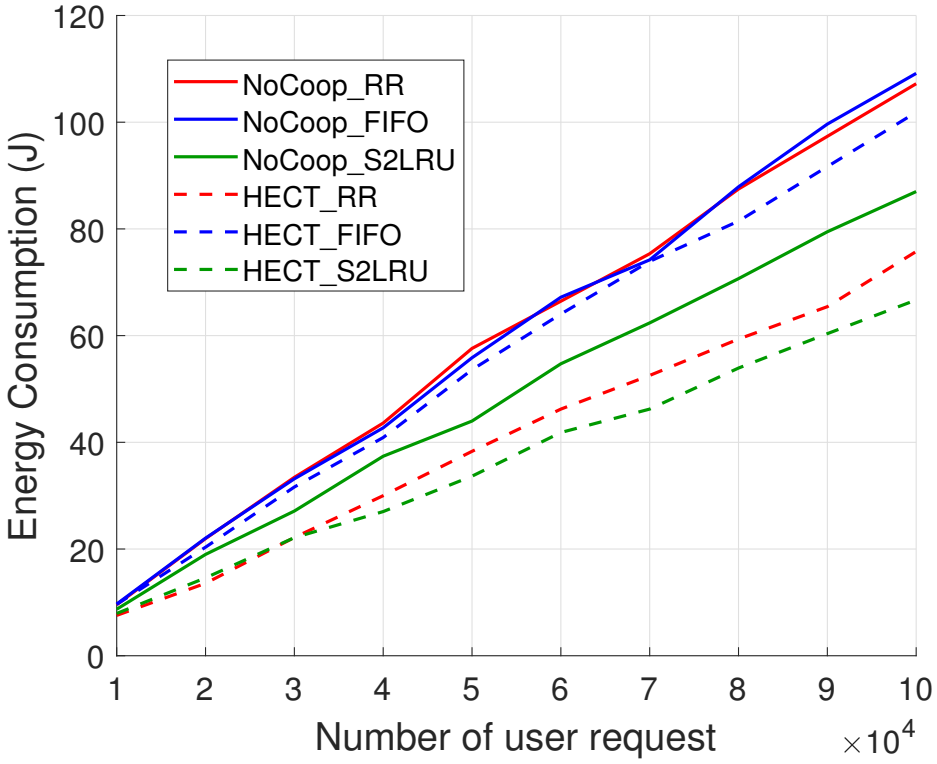


Fig. 3.12 Simulation results of energy consumption: Different numbers of user requests

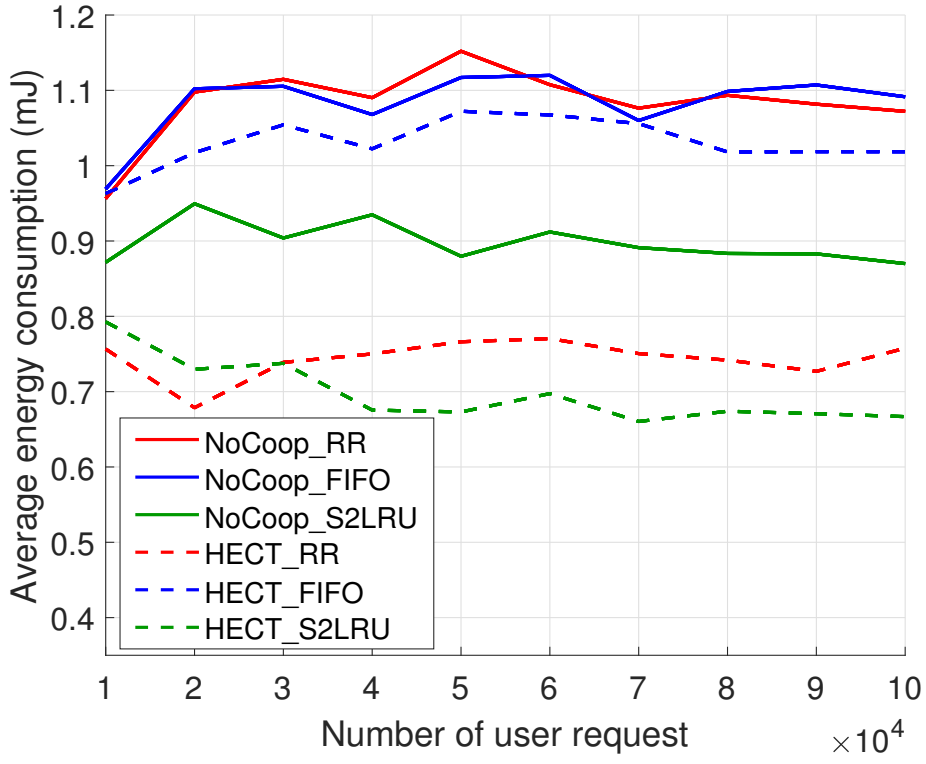


Fig. 3.13 Simulation results of energy consumption: Average results of single request

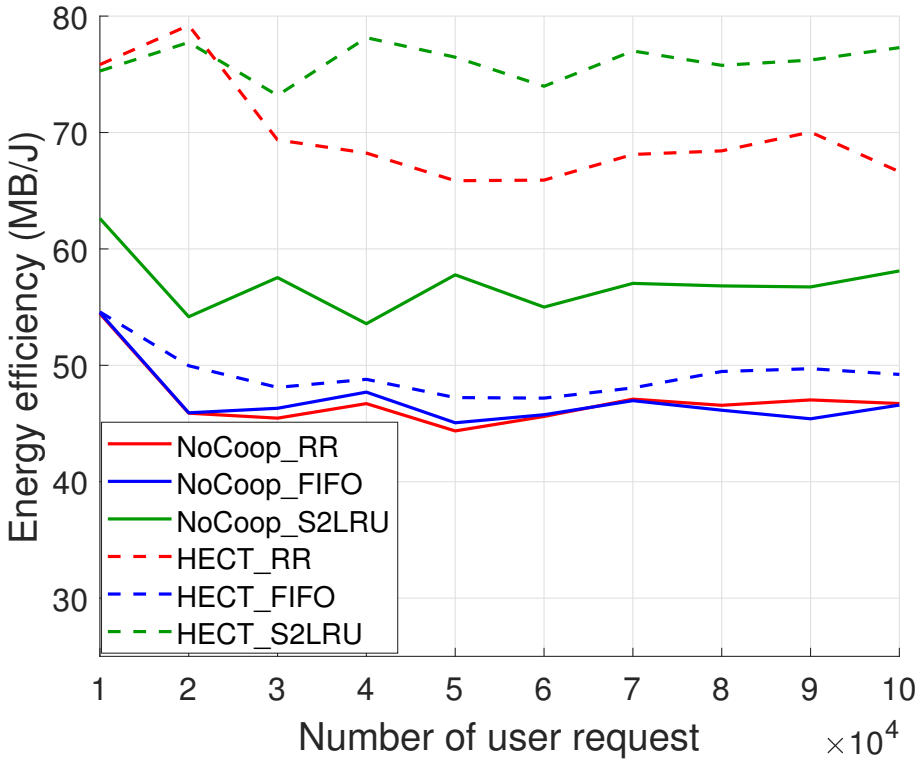


Fig. 3.14 Simulation results of energy efficiency

files prioritized in Zipf distribution may be thrown away before giving preferential treatment. HECT_RR shows unexpected fair performance. Random eviction here gives the same level of TTLs to files with different popularities from the aspect of cache replacement.

3.4.4 Energy Efficiency and Cache Hit Ratio

According to Equation (3.12) and (3.14), Fig. 3.14 and 3.15 show the results of energy efficiency (EE) and cache hit ratio.

In pursuit of doing the most with the least amount of energy, we need to make reasonable scheduling of limited resources. For the problem of edge caching, I leave the copies on the file delivery route back to users and make efforts on finding needed copies for each user as near as possible. In Fig. 3.14, the maximal gap exists between HECT_S2LRU and NoHybd_RR/NoHybd_FIFO. In numerical value, for each Joule we may expect delivering 30 more MB with HECT_S2LRU than NoHybd_RR/NoHybd_FIFO. Since EE is in relation to latency and energy consumption, I can obtain some similar analysis results. FIFO shows small difference in two edge caching schemes.

I add cache hit ratio to show the percentage that requests are satisfied by cache. The order of combinations of two cache schemes and three replacement policies is about the same with

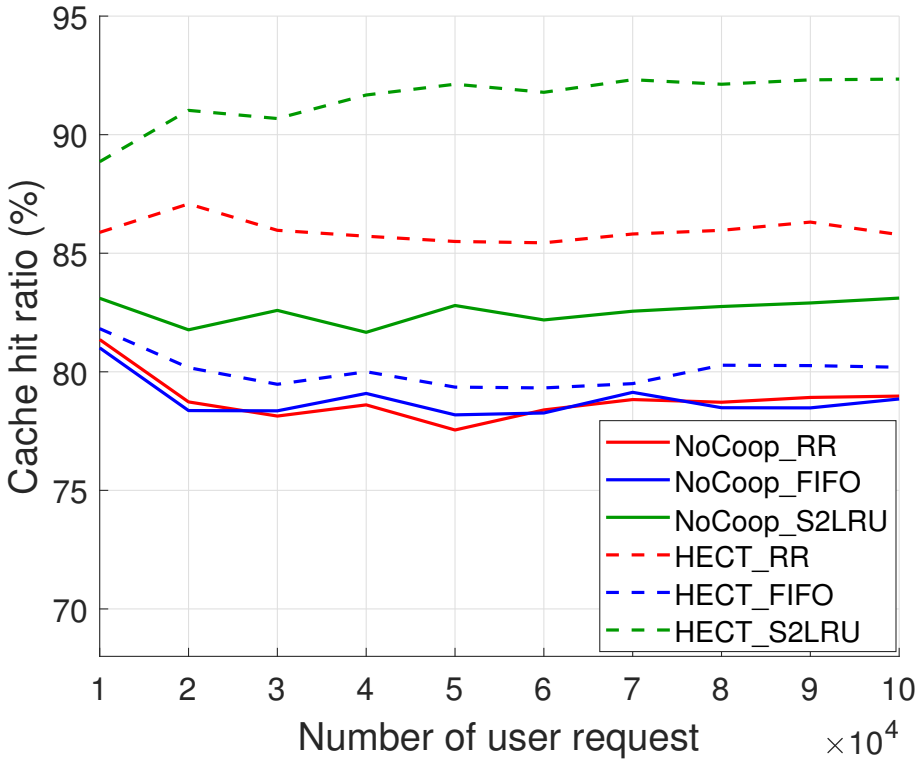


Fig. 3.15 Simulation results of cache hit ratio

EE. The exception happens when request number is no more than 30,000, HECT_RR and HECT_S2LRU are numerically close to each other in EE. This may be explained by the greater uncertainty of RR with small number of repeated trials. As a result, for HECT_S2LRU under the current experimental settings, more than 90% of user requests can be satisfied by cache.

3.4.5 Energy Efficiency of Each File in Zipf Distribution

After the discussion of latency, energy consumption, efficiency and cache hit ratio in the face of different numbers of user requests, lastly I focus on the EE of each file in *CL* that obeys Zipf distribution.

Fig. 3.16 gives the total number of times each file requested in simulation. We set x in Equation (3.1) to 1 which is the classic version. And to display the values more intuitively, I change the y-axis to an exponential coordinate with a base of 10. Files in the top of Zipf ranking account for most of the total number (550,000) of requests. And files in bottom of ranking are close to each other.

Then I calculate the energy efficiency results different matches of replacement policies and edge cache schemes. Fig. 3.17 gives the EE of all 200 files in Zipf ranking. Two

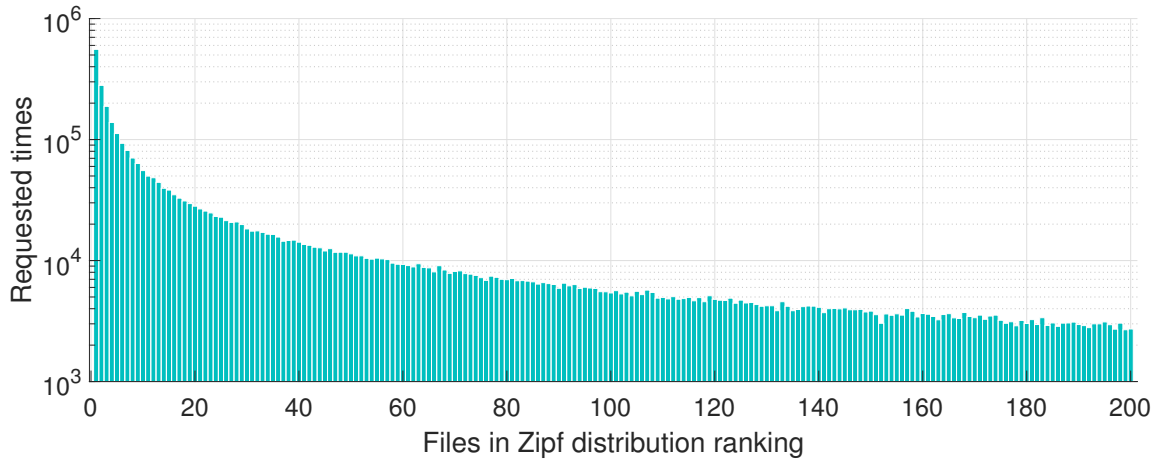


Fig. 3.16 Requested times of files in Zipf distribution ranking

Fig. 3.17 Results of non-hybrid edge caching

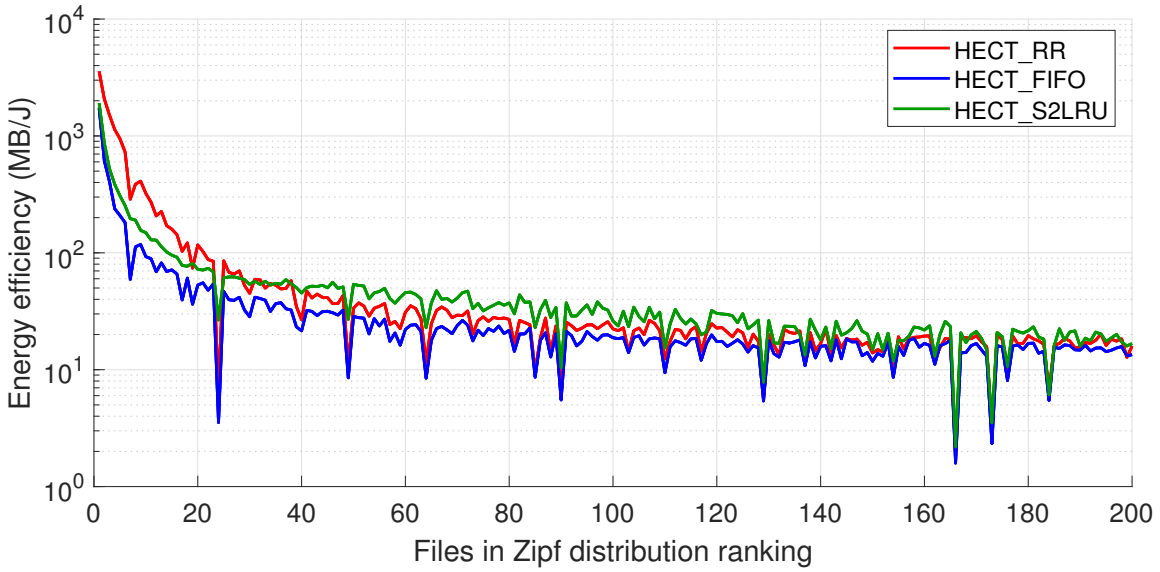


Fig. 3.18 Average energy efficiency of files in Zipf distribution ranking

polylines of RR and FIFO are almost completely coincident. S2LRU in green achieve higher EE for the first 50 files in the front which may explain why the green solid line representing NoHybd_S2LRU can be ahead of the other three in Fig. 3.14. Lastly Fig. 3.18 displays the results of HECT. The order of three replacement policies is S2LRU, RR and FIFO, however, the first 30 files in the front obtains higher EE in RR. For each cached file, I win the opportunity to extend the TTL in segmented LRU. And this may partly lead to some uniformization of the treatment of files in the Zipf distribution. That is, the copies of a file in the top 30 may live no much longer than a top 31$\sim$60 (here I discuss the TTL of any single file copy). As a result, together with the hybrid edge caching scheme, S2LRU is able to give more attention to the files not in the front.

In summary, in this section, I prove through simulation results that the proposed replacement policy and cache scheme can improve the network performance in latency, energy consumption and energy efficiency from multiple aspects.

Chapter 4

UAV-assisted Network Coverage Expansion

This chapter introduces the third phase in which I plan to enlarge the coverage and increase the scalability of the edge network. I choose UAVs as solution to play the role of network nodes in-the-air and discover survivors.

4.1 Motivation and Related Work

In this section, I introduce some related work about UAV-aided disaster management and deep learning for robotics.

Nowadays, with the development of civilian UAV manufacturing which led by DJI, researchers in various fields have regarded UAV as a flexible and practical technology for many different application scenarios. In the field of disaster management, UAVs mainly play the role of carriers or relay nodes for wireless connection. As an emerging topic, existed researches mainly are about using UAVs as supporting roles in emergency networking or rescue jobs. In fact, today's UAVs already have some onboard processing capacity with little attention. In my opinion, the reason why it is often neglected is that UAVs have restricted performance even as edge devices under the harsh power supply and load conditions. And in this thesis, I decide to take advantage of this unique onboard processing capacity to realize the autonomous control of UAVs based on image recognition using deep learning. Mozaffari *et al.* focus on the D2D (device-to-device) communications with UAV. In their work, UAVs are regarded as base stations on the air, and with their help, D2D can enjoy the convenience brought by the wireless access from above. They also carry out studies on UAV's performance issues when assisting in wireless networking including hovering time as well as antenna

array gain [54]. Motlagh *et al.* present a MEC & UAV-based platform for providing Internet of Things (IoT) services. They demonstrate the potential of UAV technology in wireless communication networks and solve the problem of computation offloading in video stream processing to achieve face recognition based crowdf surveillance [53]. Dong *et al.* present a protocol for clone detection in cyber-physical systems (CPS) [19]. Sekander *et al.* evaluate the practicability of applying multi-tier UAV cellular networking in 5G and beyond 5G (B5G) and obtain the results of the spectral efficiency of transmission in the designed terrestrial network model through simulation [70]. Al-Hourani *et al.* focus on the problems of path-loss terrestrial and aerial covering in a real-world suburban area. They collect the reference signal received power (RSRP) from Long-Term Evolution (LTE) signal as the metric to test the performance in different propagation distances [2]. Hu *et al.* settle the problem of limited bandwidth assignment in UAV-assisted cellular networks [38].

In all aspects of disaster management (prevention, preparedness, relief and recovery [83]), UAVs play an essential role in taking work that is difficult for manpower to complete. Ota *et al.* propose a game theory model in crowd sensing applications to make sure that all participants can be satisfied [58]. Naqvi *et al.* try to implement a UAV-aided resilient wireless network infrastructure with a cross-layer resource allocation strategy for multiple application scenarios including IoT service and disaster relief [56]. Erdelj *et al.* combine traditional wireless sensor network (WSN) and UAV-assisted disaster management into a mixed development scenario [20]. Dong *et al.* propose a data gathering approach using UAV as platform [18]. Hayajneh *et al.* come up with the idea of post-disaster recovery cellular networks in which a statistical framework using stochastic geometry is designed to improve the performance of UAV enabled communications [33]. Sánchez-García *et al.* pay attention to the problems of generating optimal UAV trajectories to help survivors in disaster scenarios [75].

Ever since AlexNet designed by Alex Krizhevsky won the ImageNet Large Scale Visual Recognition Challenge in 2012 [40], deep learning has ushered in an explosive growth period that continues to this day. Together with ever-improving hardware performance, many ideas used to be impracticable in the early history of deep learning are becoming reality. And with UAVs, there are many existed studies about how to apply deep learning in decision making. Giusti *et al.* use a Deep Neural Network (DNN) in designing supervised image classifier for robots and drones [30]. Ota *et al.* survey the studies and application on mobile multimedia [57]. Chen *et al.* apply UAV and image recolonization in smart agriculture. They draw support from deep learning in counting the fruits by photos taken by gimbal cameras [14]. Li *et al.* come up with the idea of realizing the learning at the edge of the network which paves the way for lightweight deep learning implementation [43, 44].

Zhang *et al.* discuss the solution of high-resolution UAV image classification by stacked auto-encoder (SAE) and stacked denoising auto-encoder (SDAE) networks [92]. Wang *et al.* design a tensor-based computation and optimization model for data processing in Cyber-Physical System (CPS) [86, 80, 79] which can be applied on UAVs. Li *et al.* work on the three-dimension robotic perception based on a view-invariant Convolutional Neural Network (CNN) model designed for disaster scenarios [48, 47]. Zhao *et al.* design an emergency network in which UAVs are transceivers in realizing multi-hop device-to-device (D2D) communication [94]. Existed studies on deep learning and robotics are mainly about the ground robots. That is, in consideration of the demand on both real-time and computing power, there is little room for UAVs to undertake the tasks in deep learning. In this thesis, I make use of the onboard processing modules and offloading devices as auxiliary equipment to realize the light-weighting of UAV operations.

4.1.1 UAV-assisted Edge Computing

Existed studies in post-disaster networking and emergency communication mainly focus on how to increase the robustness of network structures and expand the scope of wireless signals, etc. However, similar practices may be costly on implementation, and strategies, as well as schemes, also have to consider a variety of actual situations including network topology, user density, and device configuration. As a result, my target is to propose some approaches with high applicability and can be quickly deployed. Moreover, we also need countermeasures to cope with the failure of the original network infrastructure.

First, in the face of rapid deployment and high applicability, I choose UAVs as carriers to provide mobile edge services to users in affected areas. Unmanned aerial vehicle (UAV), or we used to call it a drone, refers to an aircraft without a pilot. UAV technology has been a new research hotspot in a number of areas and disciplines after rapid development in the past decade. The relatively low cost and flexible mobility instantly activate the internal potential of civilian UAVs. Especially for wireless communication and other fields, it offers many possibilities and scalability of turning the ideas that used to exist only in the concept stage into reality. The leader of UAV manufacturers, DJI [16], has provided a complete set of solutions for multiple disciplines and fields such as agriculture, energy, public safety, construction, and infrastructure. And mobile edge computing (MEC), created by the European Telecommunications Standards Institute (ETSI) [63] with computing capacities and services implemented at the edge of the networks. MEC's decentralization attribute may enrich the structure of network topology that edge nodes can be deployed according to actual needs in disaster scenarios. As a result, with the help of UAV-mounted MEC, we are able to

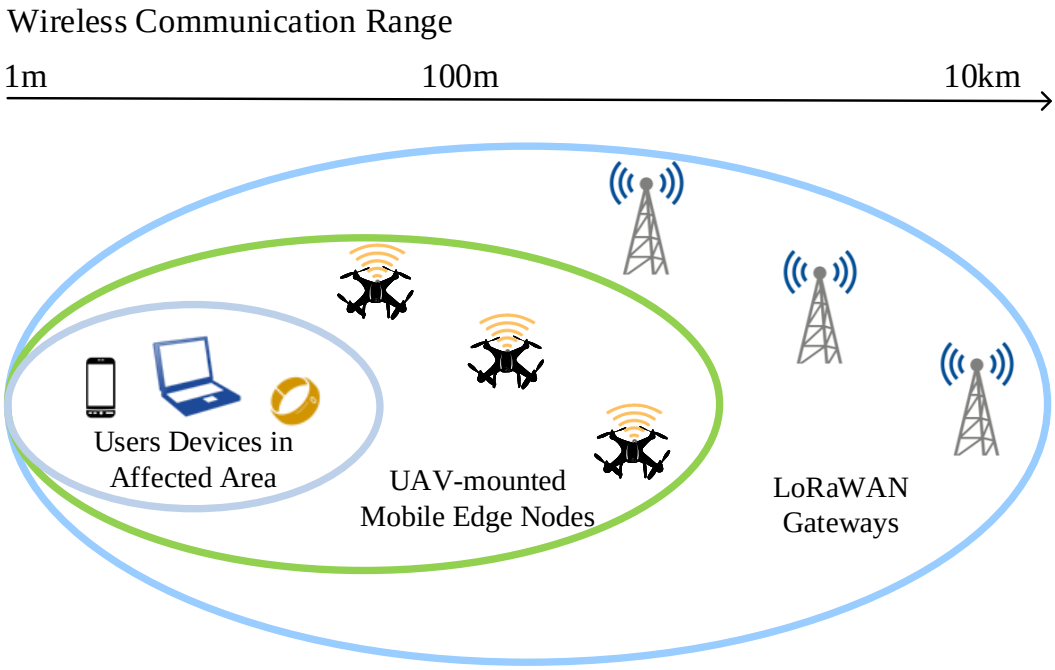


Fig. 4.1 A schematic of UAV-mounted MEC using LoRaWAN.

achieve fast networking regardless of the topographic factors such as mountains or damaged roads.

Second, after finding the suitable service and its carrier, there still remains the issue of how to establish connections in the disaster environment. That is, for emergency communication, I prefer low-cost long-range transmission methods in the case of resource shortage including electrical energy. To settle this issue, I choose LoRaWAN (Long Range Wide Area Networking) as my solution [65]. Working on the license-free radio frequency from 433 MHz (Europe), 915 MHz (North America) to 923 MHz (Asia), LoRa enables long-distance wireless connection up to 10 *km* with ultra-low power consumption [52]. As the IoT-oriented lightweight communication architecture other than the QoS awareness next-generation communication system, LoRa can just undertake the connection job between UAV-mounted mobile edge nodes and the remote control center.

A schematic of UAV-mounted MEC using LoRaWAN is shown in Fig. 4.1. Here I use a coordinate scale in the front to indicate the wireless communication range among user devices in the affected area, mobile edge nodes, and LoRa gateways. Here I make full use of the high mobility and unconstrained features of the UAVs and apply LoRa's long-distance low-power transmission in sending control information to achieve fast networking and emergency communication. In my vision, the UAV-mounted MEC using LoRaWAN (UML) system can realize the big data on the fly, that is, collecting and processing data in the

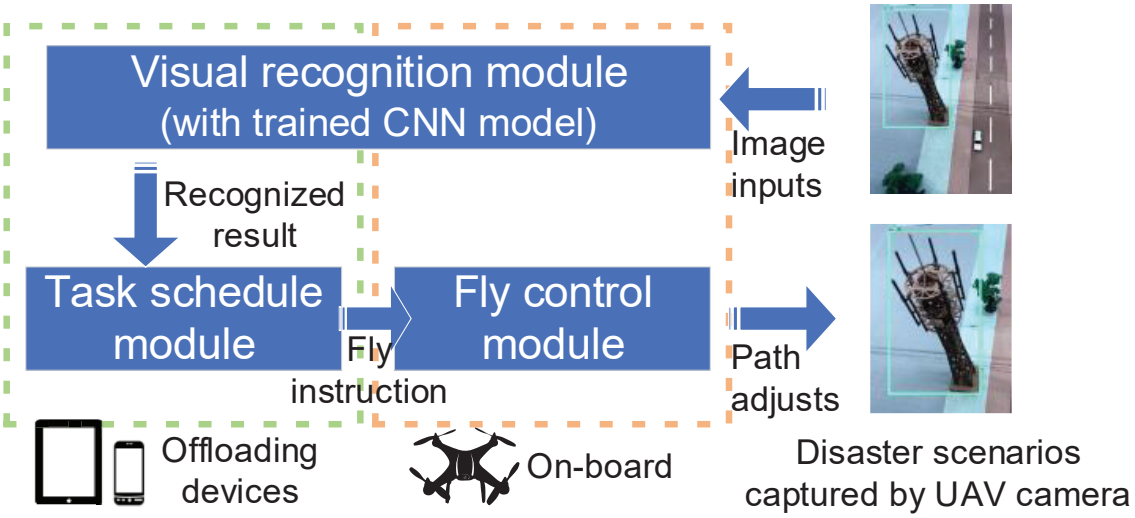


Fig. 4.2 A lightweight UAV navigation system based on airborne vision for disaster management

air by means of aerial vehicles and long-distance transmission, so as to offer new technical support for disaster management.

4.2 Problem Formulation: Airborne Vision Based UAV Navigation

In this section, I design the system model of lightweight UAV navigation based on the airborne vision for disaster management, and formulate the problems to solve.

4.2.1 System Model

The navigation system is made up of two parts, UAV and offloading devices. And there are three modules, visual recognition module, task schedule module and fly control module working cooperatively.

As shown in Fig. 4.2, a complete process from image sampling to flight path adjustment is as follow. Firstly, the image captured through the UAV camera is sent to the visual recognition module. This module is made up of both on-board and offloading devices. The trained CNN model is embedded at the offloading device to help find out the targeted object in the image. Then after task schedule module receiving the recognized result, it makes the decision on instructions of flying and sends to control module. The recognized result here refers to the array of structures which stores the targeted object in percentage within the

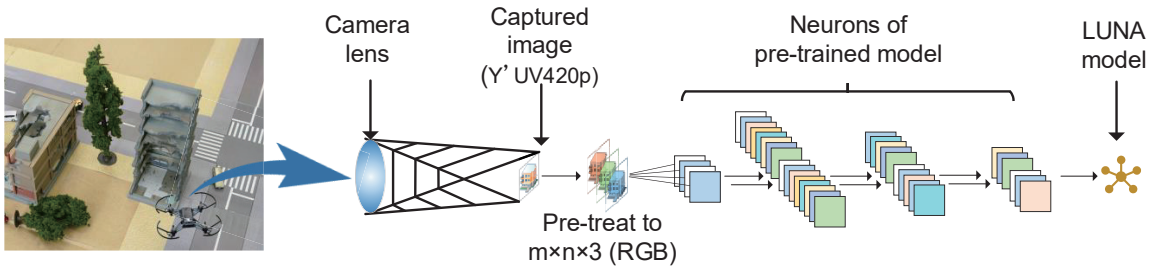


Fig. 4.3 Network architecture of LUNA

image. Lastly, knowing the position of the targeted object, fly control module may adjust the direction or move close while the camera shot changes.

Image captured by gimbal camera is in Y'UV420p format which luma Y' stands for non-linear perceptual brightness after gamma compression and U, V are chrominance components. Here I need to transfer it to RGB for next steps.

$$(R,G,B) = Conv(Y',U,V) \tag{4.1}$$

where conversion function needs to split and reorganize three values first, then perform recoding in pixels. As a result, the data structure size I used for model training is $m \times n \times 3$ in which m, n are the size of captured image.

The network architecture is shown in Fig. 4.3. I choose MobileNet [37] as the neural network platform to work on. MobileNet applies depth-wise separable convolution instead of traditional ones to support small but still efficient CNN models. In consideration of both supply from UAV on-board and demand on disaster environment, MobileNet is the architecture that helps us most right now.

I also add the design of transfer learning during the model training process. That is, there exists an urgent need in an emergency situation and the actual difficulty of collecting training data in complex disaster scenarios. And it can be costly for coping with incomplete or damaged objects, such as collapsed houses, submerged vehicles, etc. In my system model, I use a pre-trained model to continue training with collected image data from disaster scenarios. This practice may greatly save time on UAV moving or patrolling, as well as processing power, which all being supplied by the airborne battery.

4.2.2 Markov Chain Modeling

Here I use a discrete-time Markov chain to model the process of path-finding based on the airborne vision in task schedule module and fly control module. Suppose a UAV is already

Table 4.1 Table of fly instructions

Fly instructions	**Value**
None	000
Forward/Backward (001)	001
Left/Right (010)	010
Forward/Backward (001)+Left/Right (010)	011
Ascend/Descend (100)	100
Forward/Backward (001)+Ascend/Descend (100)	101
Left/Right (010)+Ascend/Descend (100)	110
All	111

near the position of a task to be performed. Since my work is based on visual recognition, the long-distance movement before the UAV camera can capture the target still requires positioning methods such as GPS. In the following part, I consider the discussion about First I denote time-slot as τ. Within a time-slot, a recognized result is sent to the task schedule module. And I have the state of $r(\tau) \in \{0,1\}$. 1 or 0 stands for whether the target is found or not.

Then for the fly instructions, I have three basic ones according to the operation commands of UAV manual control as move forward/backward, change direction, ascend/descend. Here I use positive and negative values to distinguish between the two opposite cases of each command. For example, 30 stands for turning left 30 degrees while -30 for right 30 degrees. I denote fly instruction by fi. A fly instruction being sent to the control module can include any of the basic commands. I give the different binary values to the three commands: move forward/backward (001), change direction (010), and ascend/descend (100). Each one appear once at most, then I have the table of fly instructions as

Table 4.1 gives the range of fly instruction value. There exist seven possible values which can be calculated in only one way. In other ways, I have the scope of fly instruction of $S_{fi} = \{000, 001, ..., 111\}$ and fly instruction in the current time-slot $fi(\tau) \in S_{fi}$.

Then I define the work state of UAV as ws, 0 means the state of hovering and waiting for the next instruction. While I have the work state of UAV in the current $ws(\tau)$, the work state in the next time-slot will be

$$ws(\tau+1) = [1 - r(\tau)]ws(\tau) + r(\tau) \cdot fi(\tau) \tag{4.2}$$

As shown in Equation (4.2), if no wanted object be found in the image, fly control module then continues its former work state. And when $r(\tau) = 1$ that a recognized result with the targeted object being sent to schedule module, a new fly instruction enters control module. At this time, to ensure that the current flight path is based on the newest airborne vision, UAV then turns to work state based on the newest vision of its current position.

Besides recognized result and fly instruction, I also consider the situation that the former task is completed and UAV then restarts a new one. I denote this situation as $tc(\tau) \in \{0,1,2,...\}$ that when $tc(\tau) > 0$, $ws(\tau+1) = 1000$ and $tc(\tau+1) = tc(\tau) - 1$. That is, the numerical value here refers to the number of time-slots before UAV reaching the next spot and restart visual navigation. 1000 means an exceptional work state for the movement until new spot. Instead, I have the expression of $ws(\tau+1)$ with $tc(\tau)$

$$ws(\tau+1) = \begin{cases} [1-r(\tau)]ws(\tau) + r(\tau) \cdot fi(\tau), \ if \ tc(\tau) = 0 \\ 1000, \ if \ tc(\tau) => 0, \ tc(\tau+1) = tc(\tau) - 1 \end{cases} \tag{4.3}$$

As a result, I denote the state space of the UAV navigation system is by S_s that

$$S_s(\tau) = \{r(\tau), ws(\tau), tc(\tau)\} \tag{4.4}$$

As shown in Equation (4.4), within a given area, the time cost on long distance movement is finite. Suppose the maximum numerical value of tc is M_{tc}, this discrete state space can be a three-dimensional matrix with 2 by $sizeof(S_{fi}) + 1$ by M_{tc}.

In order to realize the light-weighting of UAV navigation, the target is to reduce the necessary time cost while accomplishing the same quantity of tasks. First I denote the available UAVs by $U = \{u_1, u_2, ..., u_{n_u}\}$ in which n_u stands for the number in total. Then for one u_i, in time-slot τ, I have the three parts of time cost

$$\begin{aligned} t_i^{F/B}(\tau) &= Pr_{\{ws(\tau)\&001=001, tc(\tau)=0\}}(\tau) \cdot l_\tau \\ t_i^{L/R}(\tau) &= Pr_{\{ws(\tau)\&010=010, tc(\tau)=0\}}(\tau) \cdot l_\tau \\ t_i^{A/D}(\tau) &= Pr_{\{ws(\tau)\&100=100, tc(\tau)=0\}}(\tau) \cdot l_\tau \\ t_i^{tc}(\tau) &= Pr_{\{tc(\tau)>0\}}(\tau) \cdot l_\tau \end{aligned} \tag{4.5}$$

where $t_i^{F/B}(\tau)$, $t_i^{L/R}(\tau)$, and $t_i^{A/D}(\tau)$ are time cost on forward/backward, change direction, and ascend/descend, respectively. Pr stands for probability while the conditions in braces are satisfied. For example, to calculate $t_i^{F/B}(\tau)$, through a bitwise AND operation I can figure out whether forward/backward fly instruction is included in the work state $ws(\tau)$. $\_\tau$ is the length of time-slot. As an exception, $t_i^{tc}(\tau)$ refers to the situation of the period that the former task is completed and UAV is on the way to restart a new one. By minimizing the time cost

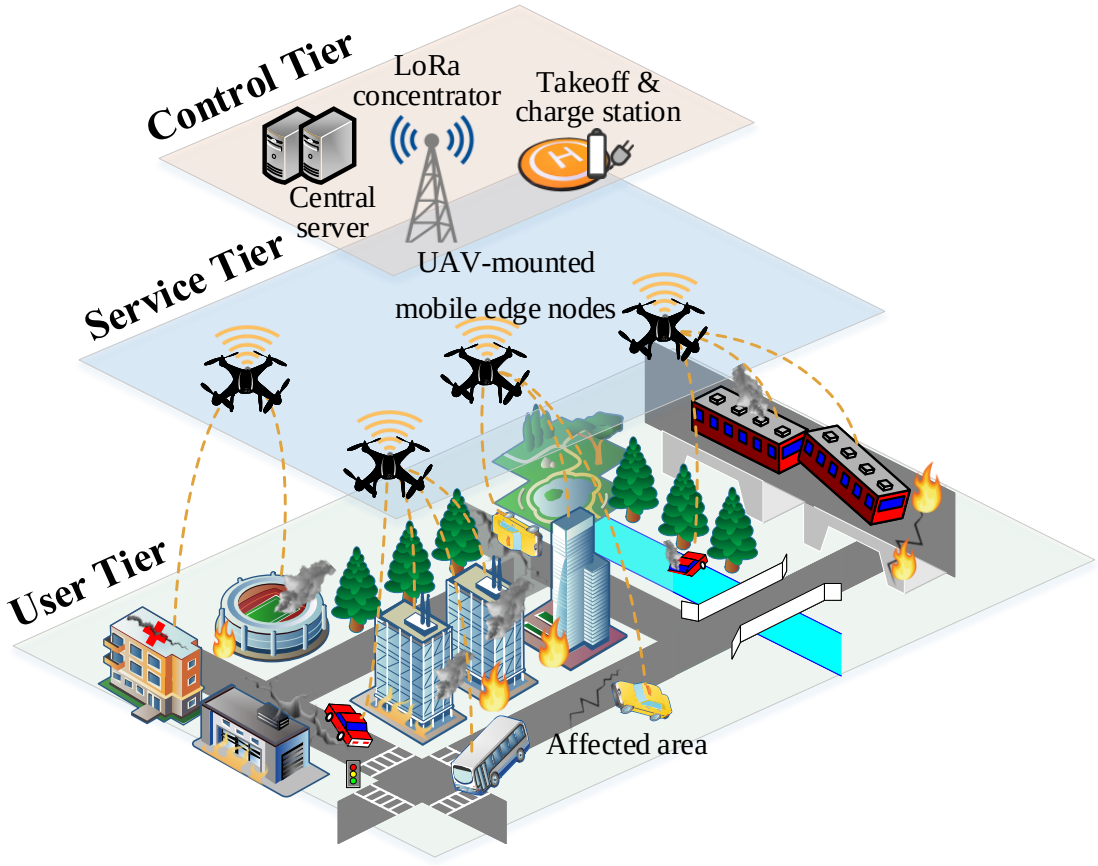


Fig. 4.4 UAV-mounted mobile edge computing network model using LoRaWAN

on three basic commands in UAV flying, I are able to cut down any unnecessary energy consumption. That is, a lightweight navigation strategy here can make use of results from visual recognition in providing a set of agile and effective UAV path-finding methods for post-disaster scenarios.

4.3 Problem Formulation: UAV-assisted Edge Computing

Here I design a UAV-mounted mobile edge network model using LoRaWAN and formulate the problems to solve.

In Fig. 4.4, I design a 3-tier network model using MEC and UAVs. First for User Tier, in an open area $\mathbb{R}^2$ there are trapped users which follow spatial homogeneous Poisson point process [5]. Suppose the density of users is ρ_u which stands for the user number per unit area. As a result, the probability of n_u users existing in the unit area is

$$P(n_u) = \frac{{\rho_u}^{n_u}}{n_u!} e^{-\rho_u} \tag{4.6}$$

Then for Service Tier, to provide MEC services to User Tier, I need UAVs as carriers to take edge devices including Raspberry Pi, LoRa module while patrolling above the affected area. When any UAV-mounted edge node m_j receives a request from user u_i, it will switch to fly mode and draw near the sender by GPS coordinates $\{\phi, \lambda, h\}$. In my model design, I focus on the horizontal flight of UAVs, and the h is mainly for calculation of path loss. As a result, the distance between the current UAV node and hover position suitable for rendering MEC service is

$$d_{i,j} = \sqrt{(\phi_i - \phi_j)^2 + (\lambda_i - \lambda_j)^2 + (h_i - h_j)^2} \tag{4.7}$$

where $\{\phi_i, \lambda_i, h_i\}$ and $\{\phi_i, \lambda_i, h_i\}$ are GPS coordinates of UAV node m_j and hover position for user u_i.

Lastly, the top Control Tier, as shown in Fig. 4.4, there exist central server, LoRaWAN gateway and takeoff & charge station for UAVs. The central server includes the data storage needed by disaster relief and responds to uplink messages sent back by UAV nodes according to the proposed task management strategy. Gateways in LoRaWAN are also known as concentrators, which play the role of signal relay equipment to build connections between the server and end devices as LoRaWAN nodes. The takeoff & charge station is for UAVs.

4.3.1 System Model

After introducing the three tiers of the network model, first, I model the connection between Service Tier in the air and User Tier on the ground. There exist two-path loss models in case of the line-of-sight (LoS) and non-line-of-sight (NLoS) in the air-to-ground part [13].

$$Pl^{los}(d_{i,j}) = Pl^{fs}(d_{fs}) + 10\gamma_{los} log_{10} d_{i,j} + X_g^{los} \tag{4.8}$$

$$vcxPl^{nlos}(d_{i,j}) = Pl^{fs}(d_{fs}) + 10\gamma_{nlos} log_{10} d_{i,j} + X_g^{nlos} \tag{4.9}$$

Equations (4.8) and (4.9) show the results of LoS path loss and NLoS in in Decibel (dB). γ and X_g are path loss exponent and Gaussian random variable with zero-mean. d_{fs} is reference distance of free-space model. Pl^{fs} stands for the free-space path loss which is (in dB)

$$\begin{aligned} Pl^{fs}(d) &= 20log_{10}(\frac{df4\pi}{c}) \\ &= 20log_{10}d + 20log_{10}f - 147.55 \end{aligned} \tag{4.10}$$

where f stands for the frequency (Hz) and c as the speed of light. d as the distance between the antennas of UAV node and user device. I have $d \gg c/f$ and both antennas are in the far-field of each other. Then according to the actual situation in which the occlusion may occur, I have the probability of LoS [3]

$$P^{los}(\theta) = \frac{1}{1+\alpha e^{-\beta(\theta-\alpha)}} \tag{4.11}$$

where α and β refer to the parameters of this Sigmoid function. θ is the elevation angle calculated by

$$\theta = sin^{-1}(\frac{|h_i - h_j|}{d_{i,j}}) \tag{4.12}$$

And the probability value of NLoS is

$$P^{nlos}(\theta) = 1 - P^{los}(\theta) \tag{4.13}$$

As a result, I have the path loss model of the first air-to-ground part

$$Pl^{atg}(d_{i,j},\theta) = P^{los}(\theta)Pl^{los}(d_{i,j}) + P^{nlos}(\theta)Pl^{nlos}(d_{i,j}) \tag{4.14}$$

Using the result of path loss, then we calculate the signal-to-noise ratio (SNR)

$$SNR^{atg} = \frac{p_m^{tran}}{10^{0.1Pl^{atg}(d_{i,j},\theta)} \cdot \sigma_g^2} \tag{4.15}$$

where p_m^{tran} is the transmission power of UAV node. σ_g^2 stands for the power of Gaussian noise. Lastly, I have the channel capacity between UAV node m_j and user device u_i which means the tight upper bound on the transmission rate (bits/s)

$$CC^{atg} = \frac{b_m}{n_u} log_2(1 + SNR^{atg}) \tag{4.16}$$

As shown in Equation (4.16), b_m is the bandwidth of UAV node.

Second, I model the communication between LoRaWAN concentrator in Control Tier and UAV-mounted mobile edge nodes in Service Tier. Based on the LoRaWAN technology, we are able to build connections between the remote concentrator and UAV nodes. Different

from the air-to-ground part, I choose the log-distance path loss model to figure out the path loss here [10].

$$Pl^{rta}(d_j) = Pl(d_0) + 10\gamma_{ld} log_{10}(d_j/d_0) + X_g^{ld} \tag{4.17}$$

where d_j is the distance between UAV node m_j and LoRa concentrator. $Pl(d_0)$ refers to the path loss at the reference distance d_0. γ_{ld} & X_g^{ld} respectively stands for path loss exponent and Gaussian random variable with zero-mean. The SNR of remote-to-air part is given by

$$SNR^{rta} = \frac{p_c^{tran}}{10^{0.1Pl^{rta}(d_j)} \cdot \sigma_g^2} \tag{4.18}$$

where p_c^{tran} stands for the transmission power of concentrator. And channel capacity between the concentrator and UAV node m_j is

$$CC^{rta} = \frac{b_c}{n_m} log_2(1 + SNR^{rta}) \tag{4.19}$$

where b_c is the bandwidth of LoRa concentrator, and n_m is the number of UAV nodes.

4.3.2 Performance Metrics

To achieve efficient task management in UAV-mounted mobile edge computing using LoRaWAN, I choose two main metrics, service time and energy consumption for performance evaluation.

First, to calculate service time, I consider three parts, that is, patrol time, fly time and hover/transmission time.

$$t_{i,j}^{serv} = t^{ptl} + t_{i,j}^{fly} + t^{hov} \tag{4.20}$$

As shown in Fig. 4.5 and Equation (4.20), after taking off from ground station, UAV nodes first may switch to patrol mode. I regard this mode as a preparation phase that enables UAV nodes to surround the affected area according to the specified route within the signal coverage of the LoRaWAN concentrator. Patrol time t^{ptl} can be calculated by v_p.

When a m_j receives a request from the user device, it then may temporarily suspend the patrol mode and determine the hovering position based on the GPS information being uploaded. And to get close to the target as quickly as possible, UAV nodes need to speed up.

$$t_{i,j}^{fly} = t_{ac} + t_{un} + t_{de} \tag{4.21}$$

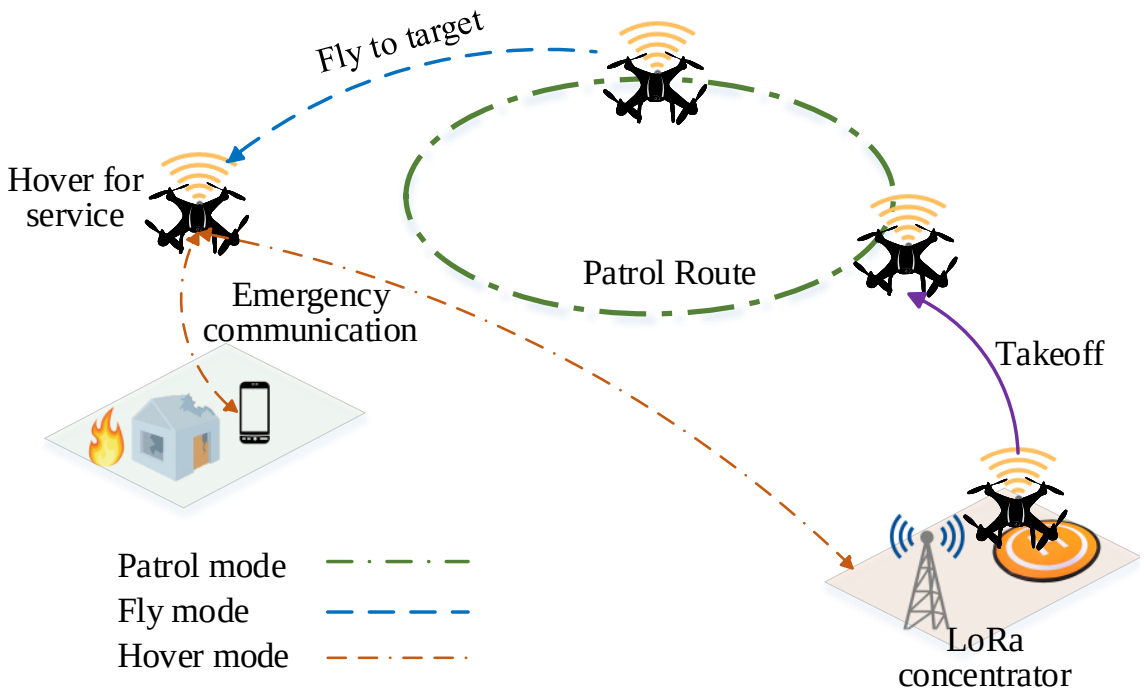


Fig. 4.5 A sketch of UAV-mounted MEC service mode

where t_{ac}, t_{un}, t_{de} respectively stands for the time cost on the process of acceleration, uniform speed and deceleration. In case the maximum speed v_{max} is reached, we have

$$\begin{aligned} d_{i,j} =& d_{ac} + d_{un} + d_{de} \\ =& v_p t_{ac} + \frac{1}{2} a_{ac} t_{ac}^2 + v_{max} t_{un} + \frac{1}{2} a_{de} t_{de}^2 \end{aligned} \tag{4.22}$$

where d_{ac}, d_{un} and d_{de} are distances traveled during acceleration, uniform speed and deceleration. v_p is the patrol speed. a_{ac} and a_{de} are acceleration and deceleration values.

Last and most important, the time cost on inter-tier emergency communications. After switching to the hover mode, UAV nodes actually play the role of signal relay equipment. That is, sharing LoRa signals transmitted over long distances to user devices.

$$\begin{aligned} t^{hov} =& t^{atg} + t^{rta} \\ =& s_{pkt} / r_{bit}^{atg} + s_{pkt} / r_{bit}^{rta} + (d_{i,j} + d_{j,s}) / c \end{aligned} \tag{4.23}$$

As shown in Equation (4.23), I calculate the transmission time cost on both two parts, air-to-ground and remote-to-air. Both parts need to consider transmission delay and propagation delay. For the former, the size of the packet to be transmitted s_{pkt} and data transmission rate r_{bit} are needed. For the latter, $d_{i,j}$ & $d_{j,s}$ (distances between u_i, m_j and central server), and c (the speed of light) are needed.

According to the demand of users, sometimes we do not need help from Control Tier, the request can be answered by UAV nodes, such as uploading the injury report. In other cases, users may want to send messages to the outside world, thus I have to consider t_{rta}.

Secondly, the energy consumption of UAV-mounted MEC using LoRaWAN in three modes. Here I focus on is the energy cost of UAV nodes, that is, the part provided by the UAV onboard batteries. Under current manufacturing technology, battery capacity is still one of the bottlenecks limiting the work performance of UAVs. As a result, in this thesis, I am committed to maximizing the utilization of battery power when processing requests from User Tier with a suitable task management strategy.

$$e^{serv} = e^{ptl}(t^{ptl}) + e^{fly}_{i,j}(t^{fly}_{i,j}) + e^{hov}(t^{hov}) \tag{4.24}$$

Similar to time cost, I consider three parts of power consumption on UAV batteries. $e^{fly}_{i,j}$ stands for the part on movement from the where m_j is to the hover position for u_i. The reason why I do not consider the part on transmission using LoRaWAN is that comparing with the cost of flying, transmission cost almost can be negligible. For example, DJI Matrice 100 as the UAV model for development can hover for 22 min with the standard 4500 mAh TB47D battery [17], and LoRa module RAK811 which can be embedded on UAV has the maximum transmit power of 20 dBm/100 mW [RAK].

$$\begin{aligned} & e^{ptl+fly}(t^{ptl} + t^{fly}_{i,j}) \\ &= \int_0^{t^{ptl}+t^{fly}_{i,j}} [\mu_1 v(t)^3 + \frac{\mu_2}{v(t)}(1 + \frac{a_c(t)^2}{g_0^2})]dt \\ &\quad + \frac{1}{2}m[v(t^{fly}_{i,j})^2 - v(0)^2] \end{aligned} \tag{4.25}$$

As shown in Fig. 4.25 [41] [90], according to some principles in helicopter aerodynamics, I use $v(t)$ to indicate the speed which can be changed in acceleration and deceleration. g_0 stands for the gravitational acceleration and m refers to the mass of UAV node. μ_1 and μ_2 are two parameters given by

$$\begin{aligned} \mu_1 &= \frac{1}{2}\rho_{air} C_{D,0} A_p \\ \mu_2 &= \frac{2(mg_0)^2}{\pi e_0 r_{as} \rho_{air} A_p} \end{aligned} \tag{4.26}$$

where ρ_{air} stands for the air density and $C_{D,0}$ is the zero-lift drag coefficient which relates to UAV's size, speed, and flying altitude. A_p is the area of the propellers on UAV, r_{as} is the aspect ratio of propellers.

Since the procedure of $e_{i,j}^{fly}$ is switched from patrol mode and ending at hover mode while fly speed drops to zero. As a result, the right half of Equation (4.25) can be

$$\frac{1}{2}m[v(t_{i,j}^{fly})^2 - v(0)^2] = \frac{1}{2}m[0^2 - v_p^2] = -\frac{1}{2}mv_p^2 \tag{4.27}$$

In addition, a_c in Equation (4.25) means the centrifugal acceleration which refers to an inertial force directed away from the axis of rotation.

$$a_c(t) = \sqrt{\mathbf{a}(t)^2 - \frac{(\mathbf{a}^T(t)\mathbf{v}(t))^2}{\mathbf{v}(t)^2}} \tag{4.28}$$

where $\mathbf{a}^T$ and $\mathbf{v}$ are acceleration and speed in vector form. Energy consumption on UAV hovering is given by [24]

$$e^{hov}(t) = e^{atg} + e^{rta} = p^{hov}(t^{atg} + t^{atg}) \tag{4.29}$$

4.3.3 Markov Chain Modeling

After modeling the UAV flying and metrics for performance evaluation, in this subsection, I present a time-homogeneous Markov chain for modeling the procedure of task management. First, I define a task queue Q_t to collect in the central server all the requests from user devices which are forwarded by UAV nodes through LoRa connections. And for the process of the task at the head of Q_t, I denote it as one time-slot. That is, in the current τ my target is to handle the task of the queue header. Then for τ, there exist two results that the current task can be finished, by UAV nodes in Service Tier or by the central server in Control Tier [50].

$$x_i^m(\tau), x_i^s(\tau) \in \{0,1\} \tag{4.30}$$

Equation (4.30) shows the results in time-slot τ. $x_i^m(\tau) = 1$ means the u_i is asking for service from UAV nodes, and $x_i^m(\tau) = 0$ means not, the same with $x_i^s(\tau)$ of central server. That is, $\{x_i^m(\tau), x_i^s(\tau)\} = \{1,1\}$ stands for the case that u_i asks for help from either of the two tiers. And $\{x_i^m(\tau), x_i^s(\tau)\} = \{0,0\}$ is for the case that u_i does out send request. I use an FIFO (First In First Out) $Q_t(\tau)$ to display the number of requests still not finished in τ. As a result, in $\tau + 1$ we have

$$Q_t(\tau+1) = \begin{cases} Q_t(\tau) - f(\tau) + \eta(t), & if\ Q_t(\tau) + \eta(\tau) \le C_Q \\ C_Q - f(\tau), & if\ Q_t(\tau) + \eta(\tau) > C_Q \end{cases} \tag{4.31}$$

where C_Q stands for the maximum capacity of the queue. In my design, UAV nodes can collect user requests and upload to LoRa concentrator no matter they are patrolling, flying or hovering for service. And $\eta(\tau)$ stands for the number of new tasks being pushed into Q_t within this τ. $f(\tau)$ is a function given by

$$f(\tau) = \sum_{i=1}^{n_u} f_i(\tau) = \frac{1}{2}\sum_{i=1}^{n_u}[x_i^m(\tau) + x_i^s(\tau) + |x_i^m(\tau) - x_i^s(\tau)|] \tag{4.32}$$

Equation (4.32) gives the results of requests sent by all n_u user devices in τ. In the case that task needs the help of both tiers $\{x^m(\tau), x^s(\tau)\} = \{1,1\}$, I expect the output to be no more than 1. Next, I determine the work state of Service Tier and Control Tier.

$$y^m(\tau), y^s(\tau) \in Z,\ 0 \le y^m(\tau) \le n_m,\ 0 \le y^s(\tau) \le C_s \tag{4.33}$$

Similar with Equation (4.30), numerical values in Equation (4.33) stand for the number of tasks occupying this tier, 0 stands for idle state. For Service Tier, each UAV node can only serve for one user device, that is $y^m(\tau) = n_m$ means all n_m UAV nodes are busy. And for Control Tier, $y^s(\tau) = C_s$ means central server is reaching the capacity of computational resource in τ.

$$y^m(\tau+1) = \begin{cases} y^m(\tau) + \sum\limits_{i=1}^{n_u} x_i^m(\tau) - \xi^m(\tau), & if\ y^m(\tau) + \sum\limits_{i=1}^{n_u} x_i^m(\tau) \le n_m \\ n_m - \xi^m(\tau), & if\ y^m(\tau) + \sum\limits_{i=1}^{n_u} x_i^m(\tau) > n_m \end{cases} \tag{4.34}$$

$$y^s(\tau+1) = \begin{cases} y^s(\tau) + \sum\limits_{i=1}^{n_u} x_i^s(\tau) - \xi^s(\tau), & if\ y^s(\tau) + \sum\limits_{i=1}^{n_u} x_i^s(\tau) \le C_s \\ C_s - \xi^s(\tau), & if\ y^m(\tau) + \sum\limits_{i=1}^{n_u} x_i^m(\tau) > C_s \end{cases} \tag{4.35}$$

As shown in Equations (4.34) and (4.35), in the next time-slot $\tau+1$, $x_i^m(\tau)$ and $x_i^s(\tau)$ newly come into the processing unit of two tiers. $\xi(\tau)$ stands for the number of tasks finishing. Here I use *SS* to denote the state space of the UML system in Markov chain.

$$SS(\tau) = \{Q_t(\tau), y^m(\tau), y^s(\tau)\} \tag{4.36}$$

As a result, the discrete state space is finite and can be shown as a three-dimensional matrix with C_Q by n_m by C_s.

4.3.4 Problem Formulation

First, according to Equations (4.20) and (4.23), the problem on service time in total is given by

$$\begin{aligned}\sum_{\tau}^{n_\tau} t_\tau^{serv} = &\sum_{\tau}^{n_\tau}\sum_{i=1}^{n_u}[P_{1,0}^{x_i}(\tau)(t^{atg,i}+t_i^{ptl}+t_i^{fly}) \\ &+(P_{0,1}^{x_i}(\tau)+P_{1,1}^{x_i}(\tau))(t^{atg,i}+t^{rta,i}+t_i^{ptl}+t_i^{fly})]\end{aligned} \tag{4.37}$$

where t_τ^{serv} denotes the service time generated in τ. $P_{0,0}^{x}(\tau)$, $P_{0,1}^{x}(\tau)$, $P_{1,0}^{x}(\tau)$ and $P_{1,1}^{x}(\tau)$ are probabilities of four cases in Equation (4.30). That is, within this time-slot, some of requests/tasks sent by user devices in front of the Q_t get answered. The ones on Control Tier enter the processing unit of central server, and the ones on Service Tier are being allocated to the UAV nodes nearby. Thus, together with Equation (4.6), my target of reducing time cost in UML system is

$$\begin{aligned}&minimum \quad P(n_u)\sum_{\tau}^{n_\tau} t_\tau^{serv} \\ &subject\ to \quad C1: P_{0,0}^{x}(\tau)+P_{0,1}^{x}(\tau)+P_{1,0}^{x}(\tau)+P_{1,1}^{x}(\tau)=1 \\ &\qquad\qquad C2: 0\le\Pi_i\le 1,\ \sum_{i\in SS}=1,\ \Pi_i=\sum_{j\in SS}\Pi_j pr_{i,j}\end{aligned} \tag{4.38}$$

$C2$ in (4.38) is the steady-state condition in time-homogeneous Markov chain [78]. Π stands for the stationary distribution and $pr_{i,j}$ is the transition probability from Π_j to Π_i.

Second, according to Equation (4.25), the problem of energy cost on UAV batteries is given by

$$\begin{aligned}\sum_{\tau}^{n_\tau} e_\tau^{ptl+fly} = &\sum_{\tau}^{n_\tau}\sum_{i=1}^{n_u}\{\int_0^{t_i^{ptl}+t_i^{fly}}[\mu_1 v(t)^3+\frac{\mu_2}{v(t)}(1+\frac{a_c(t)^2}{g_0^2})]dt \\ &+\frac{1}{2}m[v(t_i^{ptl}+t_i^{fly}+\tau l_\tau)^2-v(\tau l_\tau)^2]\}\end{aligned} \tag{4.39}$$

where l stands for the length of a single time-slot, that is, I calculate the part energy consumption from the end of the current τ to arrival at hover position.

$$\begin{aligned}\sum_{\tau}^{n_\tau} e_\tau^{hov} = &p^{hov}\sum_{\tau}^{n_\tau}\sum_{i=1}^{n_u}\{P_{1,0}^{x_i}(\tau)t^{atg,i} \\ &+[P_{0,1}^{x_i}(\tau)+P_{1,1}^{x_i}(\tau)](t^{atg,i}+t^{rta,i})\}\end{aligned} \tag{4.40}$$

Similarly, together with Equations (4.6) and (4.24), my target of reducing energy consumption in UML system is

$$\begin{aligned} &minimum \quad P(n_u)\sum_{\tau}^{n_\tau}(e_\tau^{ptl+fly}+e_\tau^{hov}) \\ &subject\ to \quad C1: P_{0,0}^x(\tau)+P_{0,1}^x(\tau)+P_{1,0}^x(\tau)+P_{1,1}^x(\tau)=1 \\ &\qquad\qquad C2: 0\le\Pi_i\le 1,\ \sum_{i\in SS}=1,\ \Pi_i=\sum_{j\in SS}\Pi_j pr_{i,j} \end{aligned} \tag{4.41}$$

In consideration of the uncertainty in calculating different parts in Equations (4.38) and (4.41), there may even exist difference in order of magnitude. As a result, in the design of task management strategies, I focus on taking care of each part from a global perspective and looking for multi-objective solutions.

4.4 Algorithm Design

4.4.1 Lightweight UAV Navigation Strategy Based on Airborne Vision

In this section, I design a lightweight UAV navigation strategy based the visual recognition results from images captured by gimbal camera.

To realize the lightweight control of UAV, firstly I need to learn the valid information from the recognition results including the position of the targeted object and its size within the image.

As shown in Fig. 4.6, here I cope with an example of recognized object in post-disaster scenario. Through the frame drawn by visual recognition module supported by both onboard computer and offloading devices. I obtain the information of where the object is and how large is it. (x_{rt}, y_{rt}) and (x_{lb}, y_{lb}) are two points at the right top and left bottom, respectively. As a result, I have the size of target as width w $(x_{rt}-x_{lb})\times$ height h $(y_{rt}-y_{lb})$. Then I can obtain the center of the target in the image as $\{\frac{1}{2}(x_{rt}+x_{lb}), \frac{1}{2}(y_{rt}+y_{lb})\}$. As a result, the center offset o_c and offset angle o_a are

$$o_c = \sqrt{\frac{1}{4}(x_{rt}+x_{lb})^2+\frac{1}{4}(y_{rt}+y_{lb})^2} \tag{4.42}$$

$$o_a = \begin{cases} arccos(\frac{x_{rt}+x_{lb}}{2o_c})(rad), & if\ y_{rt}+y_{lb}\ge 0 \\ -arccos(\frac{x_{rt}+x_{lb}}{2o_c})(rad), & if\ y_{rt}+y_{lb}<0 \end{cases} \tag{4.43}$$

Thus, fly control module can take steps in turning around and approaching. Moreover, my strategy is enabled with feedback. That is, facing complex disaster scenarios, UAVs often cannot lock the target by following the above steps once. Sometimes it is necessary to modify the path multiple times. As a result, I not only can reduce cost on manpower decision

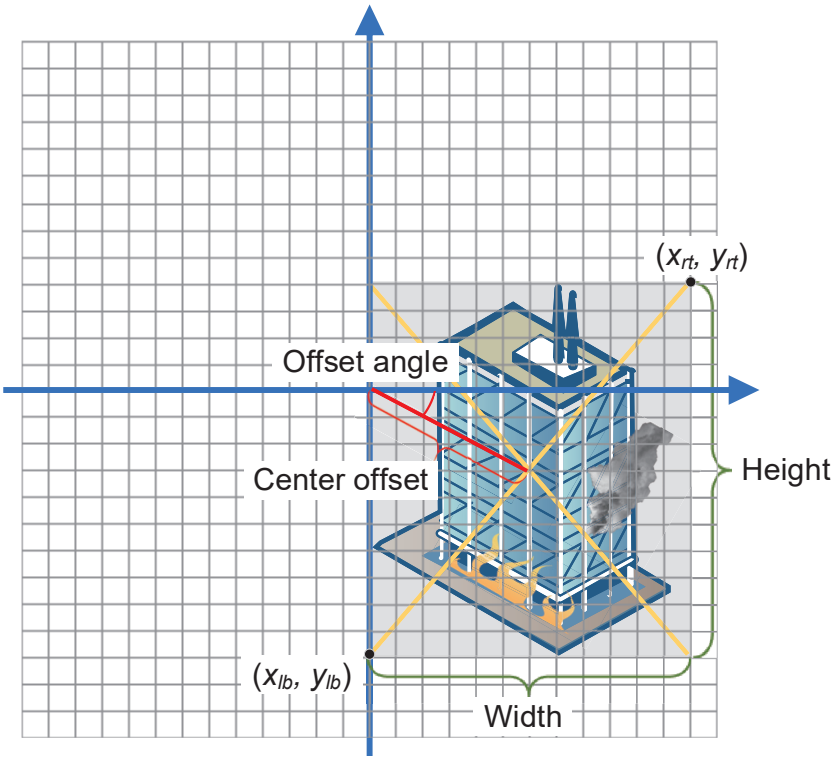


Fig. 4.6 An example of recognition result

making, but also avoid possible human errors. Even the simple situation shown in Fig. 4.6 may face the time cost on the fly adjustment. That is, during the period of changing direction or moving, new fly instruction can also take effect. I apply this dynamic on-air feedback to accurately grasp the conditions of disaster environment. Especially when accidents such as the movement of object or appearance of obstacle happen, UAVs can respond accordingly. As a result, I design a lightweight UAV navigation algorithm to help obtain the flight paths based on visual recognition results.

As shown in Algorithm 5, W and H are the size of the image with targeted object. $\alpha \in (0,1)$ is a ratio used for deciding whether to get close to the object or stay away. In the current loop of time-slot, task schedule module gives fly instructions by judging the location information of the target in the image being captured by gimbal camera. While there exist multiple UAVs ready for missions, I take turns in checking and updating the newest instructions. Changing directions & ascending/descending (line 6-18) are different from moving forward/backward (line 19-23) in practice. The former two can be done by o_c and o_a, and the latter one needs the comparison between w/h and $\alpha W/\alpha H$. The analysis of computational complexity is as follows. First for time complexity in Algorithm 5, without

Algorithm 5 LUNA: Lightweight UAV Navigation Strategy

Input: o_c, o_a // results obtained from visual recognition module
Output: fi_i // fly instructions made by task schedule module

loop
  $\tau \leftarrow \tau + 1$
  for $i = 1$ to n_u **do**
    if $r_i(\tau) = 1$ **then**
      $fi_i(\tau) \leftarrow 000$
      if $o_c \neq 0$ **then**
        if $|o_a| = 0$ or π **then**
          $fi_i(\tau) \leftarrow fi_i(\tau) + 010$
          Turn left once $|o_a| < \frac{\pi}{2}$, right once $|o_a| > \frac{\pi}{2}$
        else if $|o_a| = \frac{\pi}{2}$ **then**
          $fi_i(\tau) \leftarrow fi_i(\tau) + 100$
          Ascend once $o_a < 0$, descend once $o_a > 0$
        else
          $fi_i(\tau) \leftarrow fi_i(\tau) + 110$
          Turn left once $|o_a| < \frac{\pi}{2}$, right once $|o_a| > \frac{\pi}{2}$
          Ascend once $o_a < 0$, descend once $o_a > 0$
        end if
      end if
      if $w < \alpha W_i(\tau)$ && $h < \alpha H_i(\tau)$ **then**
        $fi_i(\tau) \leftarrow fi_i(\tau) + 001$, forward
      else if $w > \alpha W_i(\tau)$ || $h > \alpha H_i(\tau)$ **then**
        $fi_i(\tau) \leftarrow fi_i(\tau) + 001$, backward
      end if
    end if
  end for
end loop

the regard of the loop for time-slot, there exist only one for loop, then I have the $O(n_u)$. Second for space complexity, since n_u of r_i and fi_i are needed, I have $O(n_u)$.

4.4.2 Task Management Strategy for UAV-mounted MEC Using LoRaWAN

In this subsection, I design task management strategies for UAV-mounted MEC service using LoRaWAN in a post-disaster scenario.

To solve the two problems shown in (4.38) and (4.41), I design two algorithms in handling task assignment and task queue management during the operation of the UML system. First, I propose an algorithm for allocating the tasks collected by the LoRa concentrator to UAV

nodes in Service Tier. The central server plays the role of a backup service provider to take over the tasks in need of high computational resource consumption.

Algorithm 6 STAS: Single Task Assignment Strategy

1: $this\_task$ // the current task from u_i
2: d^*_{min} // the shortest distance between u_i and any idle m_j
3: $\{x^m(\tau), x^s(\tau)\}$ // the results whether the task can be assigned to Server Tier and Control Tier
4: **loop**
5: $\tau = \tau + 1$
6: $this\_task($ from $u_i) \leftarrow Q_t.pop()$
7: **if** $this\_task.x^m(\tau) = 1 \parallel this\_task.x^s(\tau) = 1$ **then**
8: **if** $find(m_j | m_j.state = 0$ && $dist(m_j, u_i) = d^*_{min})$ **then**
9: **if** $this\_task.x^m(\tau) = 1$ **then**
10: $m_j.state \leftarrow 0, this\_task.x^m(\tau) \leftarrow 0, this\_task.x^s(\tau) \leftarrow 0$
11: **else if** $C_s < C_{max}$ **then**
12: $C_s \leftarrow C_s + 1, this\_task.x^s(\tau) \leftarrow 0$
13: **end if**
14: **end if**
15: **end if**
16: **end loop**

Algorithm 6 describe the process of one task *this_task* being assigned to one of the UAV nodes or server according to decision parameters $[x^m(\tau), x^s(\tau)]$ with the current time-slot. Assuming that the user who sends *this_task* is u_i, I start by finding the nearest UAV node to ensure a stable connection. Line 9 judges if *this_task* is $\{1,1\}$ or $\{1,0\}$. That is, I aim at using UAV nodes at the edge as much as possible when computing resources are sufficient in the UML system. $m_j.state$ stands for the work status indicator, 1 for occupied and 0 for idle. Time complexity of Algorithm 6 is $O(n_\tau)$.

Second, to manage the FIFO task queue defined in Equation (4.31), I propose TQMS as shown in Algorithm 7.

Follow the steps in Algorithm 6, when *this_task* does not find an idle UAV node within this τ, *this_task* will be pushed into Q_t again to wait for the chance in next τ (line9). Then as shown in line 12-17, I push the new tasks *newtask* into Q_t until reaching maximum capacity C_Q. In my design in UML, besides performing the tasks assigned by STAS, UAV nodes can receive nearby user requests containing GPS position information at any time. The time complexity is $O(n_\tau(n\_task + n\_newtask))$.

With the help of STAS and TQMS, I am able to manage the requests collected by UAV nodes, and select service provider for each of them considering the time & energy cost as well as the work state of nodes and server.

Algorithm 7 TQMS: Task Queue Management Strategy

newtask // new tasks received within the current τ
loop
  $\tau = \tau + 1$
  $n\_task = Q_t.size$
  for $i = 1$ to n_task **do**
    $this\_task \leftarrow Q_t.pop()$
    task assignment
    if $this\_task.x^m(\tau) = 1 \parallel this\_task.x^s(\tau) = 1$ **then**
      $Q_t.push(this\_task)$
    end if
  end for
  for $i = 1$ to $newtask.size$ **do**
    if $Q_t.size < C_Q$ **then**
      $Q_t.push(newtask(i))$
    else
      break
    end if
  end for
end loop

4.5 Performance Evaluation

In this section, I carry out simulations and experiments for lightweight UAV navigation. There are two parts, airborne visual recognition test, and path-finding simulations based on recognition results.

4.5.1 Airborne Visual Recognition

First, in model training, the model is obtained from ssd_mobilenet_v1_coco after retraining. I would like to use this existed model acquired from MobileNet as a pre-model. I first build a post-disaster scenario by miniature models with 1/150 scale and take 658 photos as data for second training. I use a server with the configuration of Intel Core i7-7700, NVIDIA GTX 1080, 16 GB memory, for training 100,000 steps. As a result, I put a trained model of 22.75 MB at the offloading device. Compared with the pre-model of 29.11 MB, here I re-train by images after compression and further realize the lightweight of the deep learning model itself.

Fig. 4.7∼4.9 are three examples of airborne visual recognition results, in this part, I consider three kinds of targeted objects in disaster scenarios, high building, low building, and base station tower. Each one has different external features that can be collected, classified

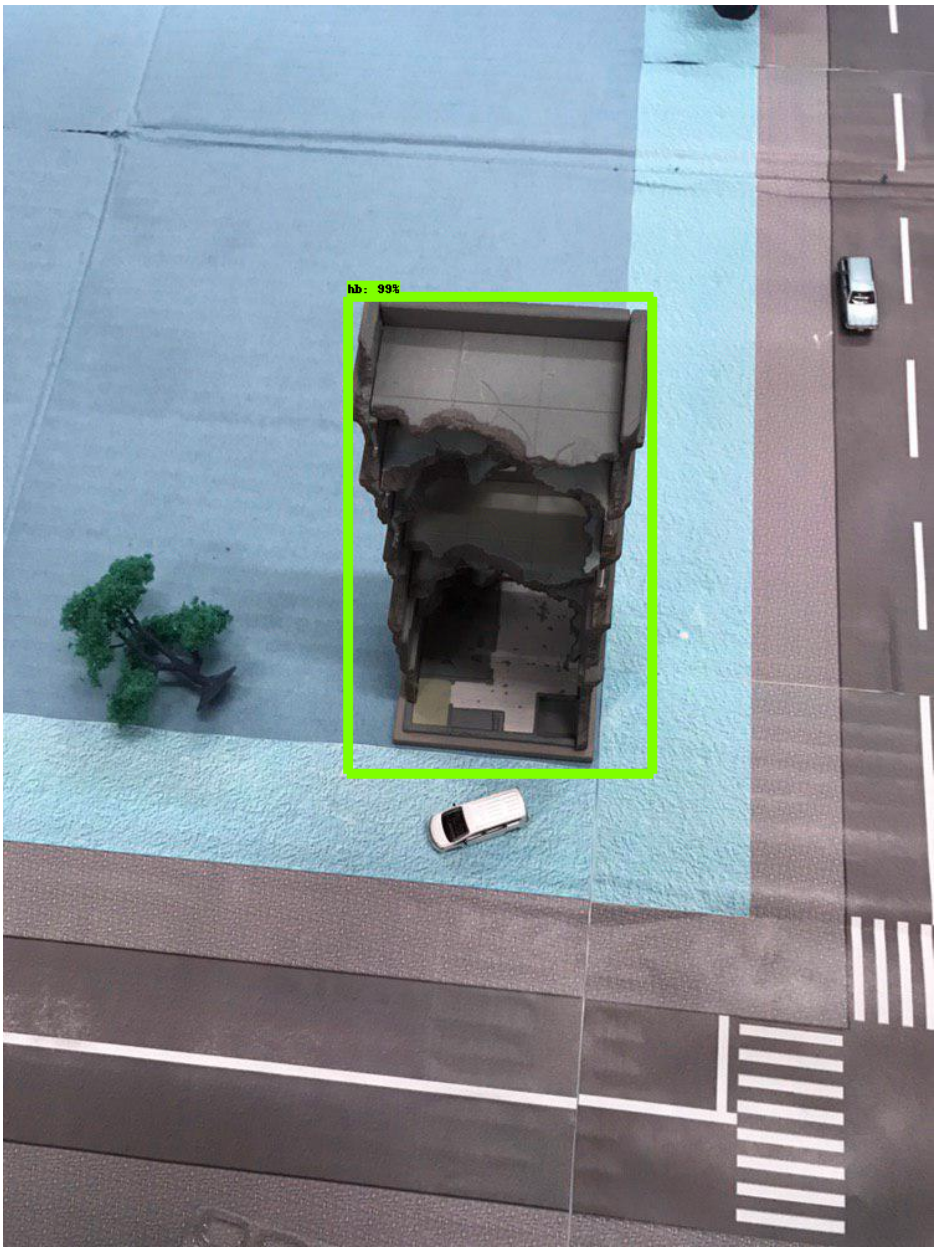


Fig. 4.7 Three examples of airborne visual recognition results: High building

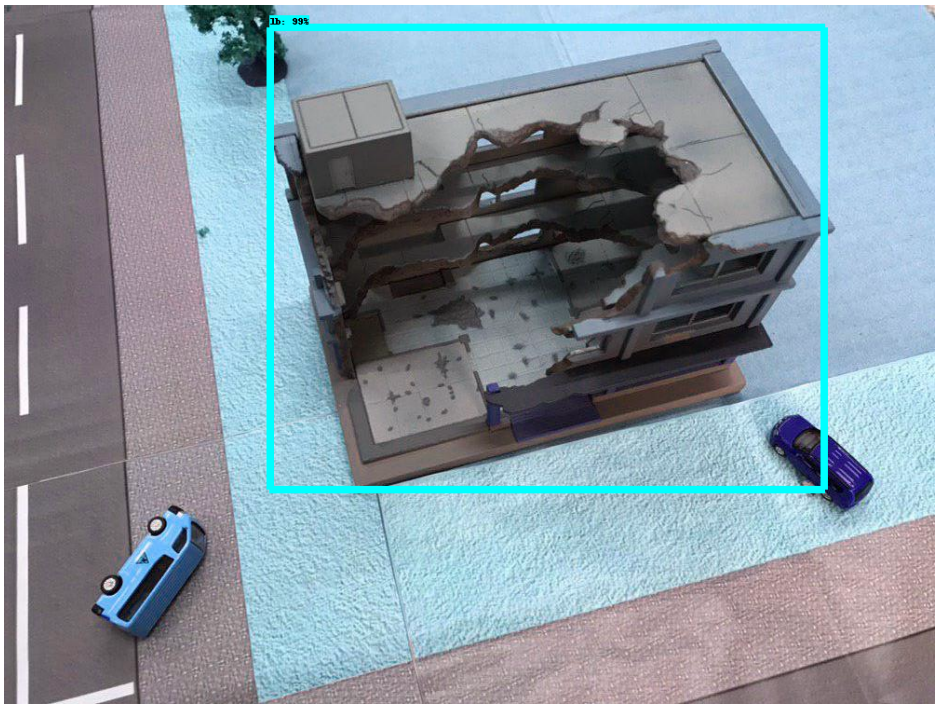


Fig. 4.8 Three examples of airborne visual recognition results: Low building

and combined through trained model. Thus, visual recognition module then can give the judgment on whether the wanted object is in the captured image.

Next, I take 224 more photos as test dataset which include all three types of objects. The actual appearance and number of objects appearing within the test images are both random. I calculate the receiver operating characteristic (ROC) curves of three types to show the diagnostic ability of the model when serving as binary classifiers to judging if one or more instances are existing.

In Fig. 4.10∼4.12, the ROC curves of high building, low building, and base station tower show different results. ROC curves can show the performance of binary classifiers in the prediction experiment. AUC (area under the curve) refers to the probability that binary classifiers rank positive instances (targeted objects exist) higher than negative ones (no targeted object) [22]. The value of AUC can serve as an indicator of the pros and cons of the model.

The x-axis is the false positive rate (FPR) or fall-out that refers to how many objects being wrongly recognized while there exists no instance in the test. And the y-axis is true positive rate (TPR) or recall that refers to how many objects being correctly recognized while there exist instances. The calculations of FPR and TPR are as follows

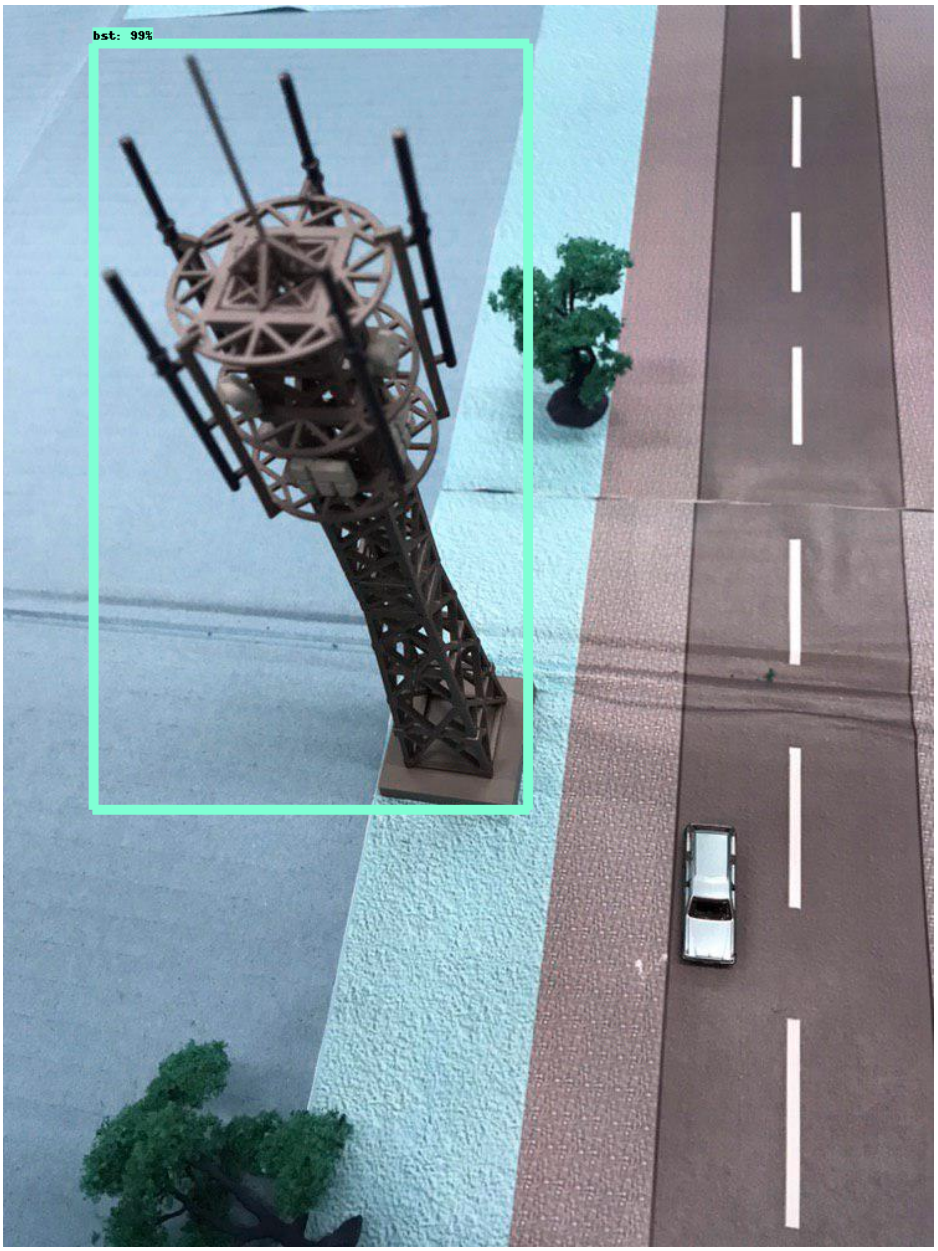


Fig. 4.9 Three examples of airborne visual recognition results: Base station tower

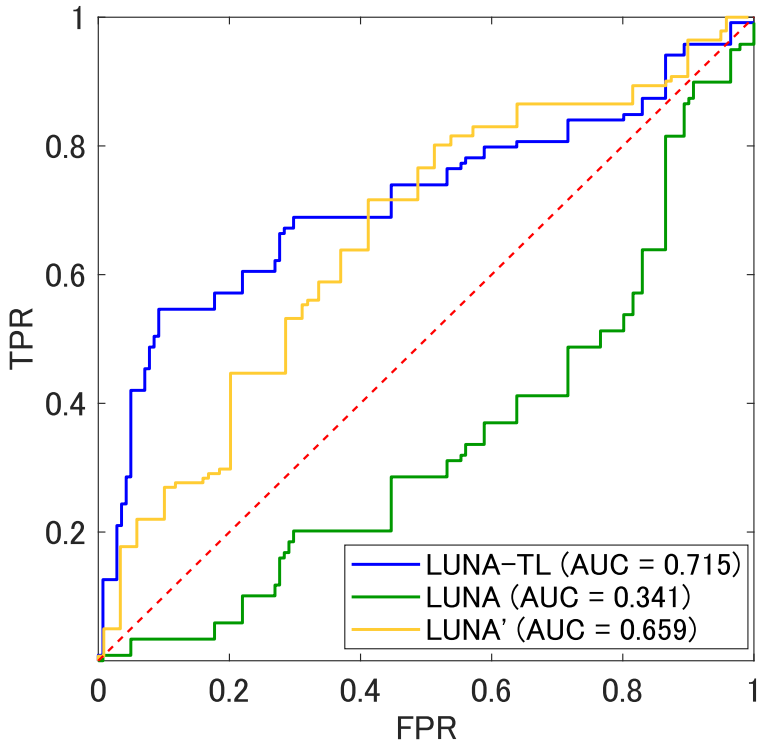


Fig. 4.10 ROC curves of airborne visual recognition results: (a) High building

$$FPR = \frac{FP}{FP+TN} \tag{4.44}$$

$$TPR = \frac{TP}{TP+FN} \tag{4.45}$$

where FP, TN, TP, and FN are short for false positive, true positive, true negative, and false positive, respectively.

Blue curves (LUNA-TL) are the proposed lightweight method with the consideration of transfer learning, while green ones (LUNA) are without that. I also draw the yellow curves (LUNA') which are obtained by reversing the decisions of green ones as comparisons. Red dotted lines are baselines for random guessing.

First in Fig. 4.10, without the consideration of the mixed model, the blue curve is even below the baseline which means the performance is worse than random guessing. By drawing its mirrored curve across the baseline, I can see that although sometimes LUNA-TL and LUNA' fluctuate high and low, I achieve a larger AUC in the mixed model. Then in Fig. 4.11, the gap between LUNA-TL and LUNA (LUNA') becomes larger. And in Fig. 4.12, LUNA without transfer learning bypasses its mirrored curve. In summary, for the recognition of

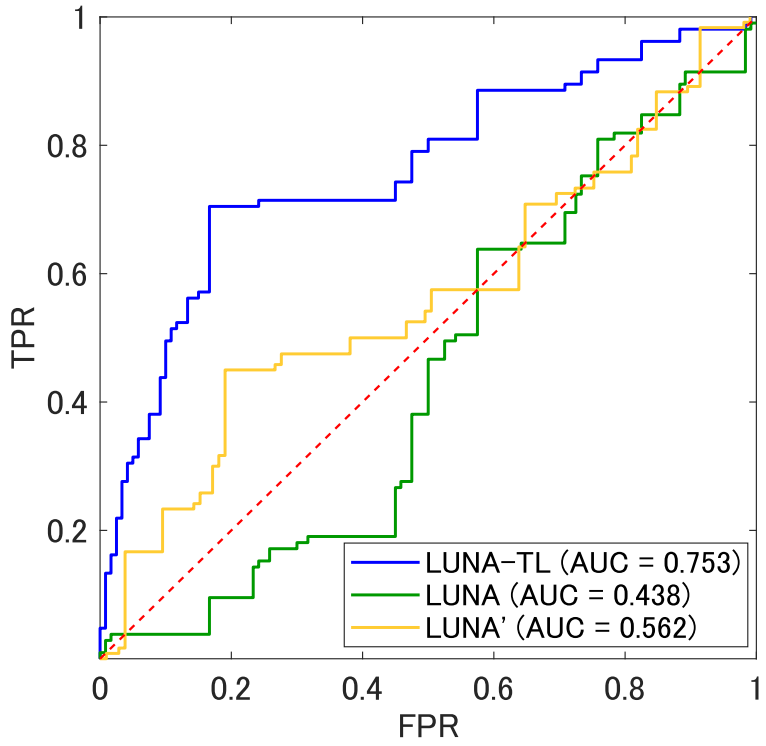


Fig. 4.11 ROC curves of airborne visual recognition results: (b) Low building

three objects, the model shows better performance in finding base station towers from the airborne vision. This can be explained by the object itself that base station tower has a big difference against the other two. The features of Fig. 4.12 are easy to distinguish on the outer contour.

Next, I verify the effectiveness of my proposed navigation algorithm based on the recognized results.

4.5.2 UAV Lightweight Navigation Based on Airborne Vision

Through the recognized results, the fly control module can obtain the instructions on changing directions, etc. I assume that there exists a 3-D simulation scenario with $1000 \times 1000 \times 200$ m^3. Once receiving a request with the position of the target spot, a UAV embedded with the navigation system will take off from (0, 0, 0) and head to the destination. After arriving in the nearby area of the target, once line 19-22 in Algorithm 5 is satisfied, UAV then starts the next task. I repeat the procedure of request 10000 times and calculate the average time cost on following different fly instructions.

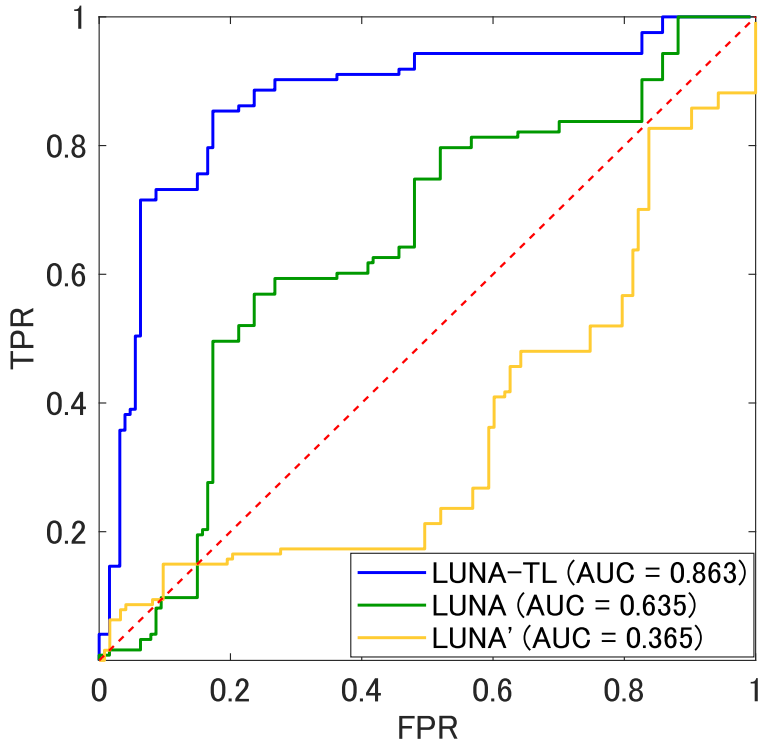


Fig. 4.12 ROC curves of airborne visual recognition results: (c) Base station tower

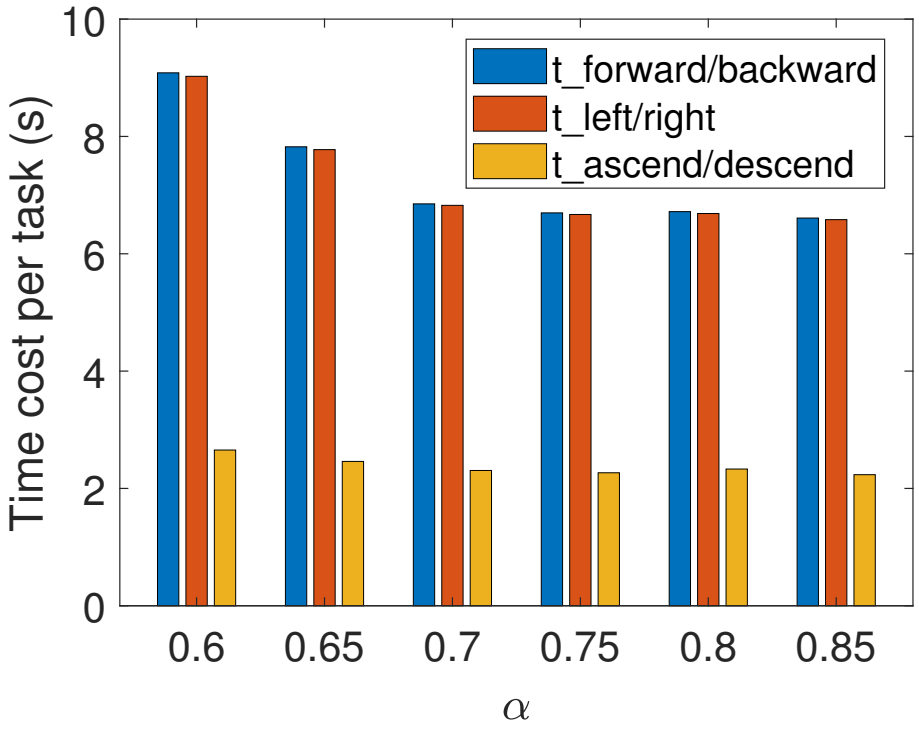


Fig. 4.13 UAV lightweight navigation results: (a) Three types of time cost per task

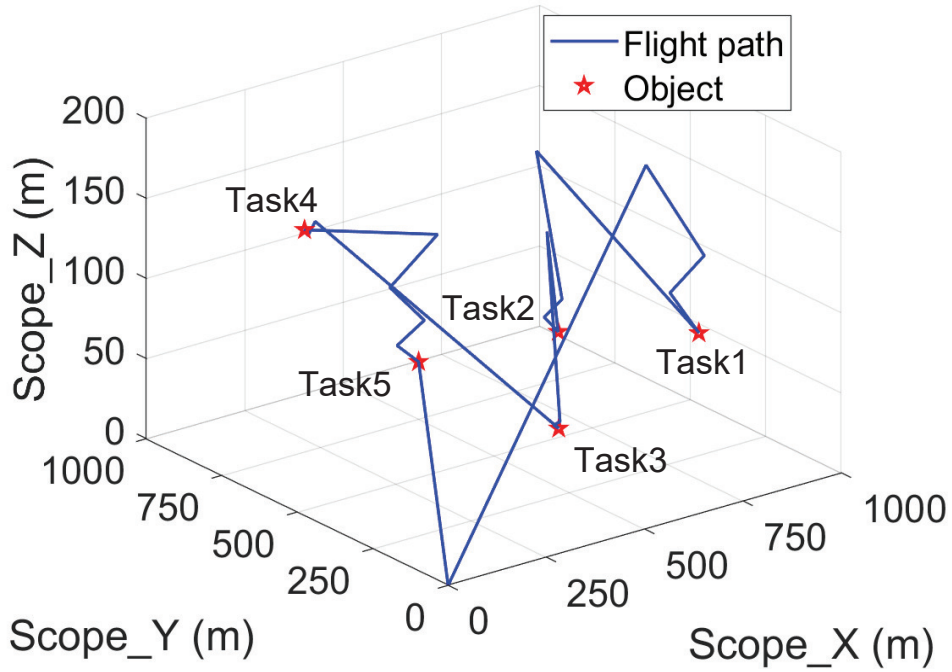


Fig. 4.14 UAV lightweight navigation results: (b) An example of flight path

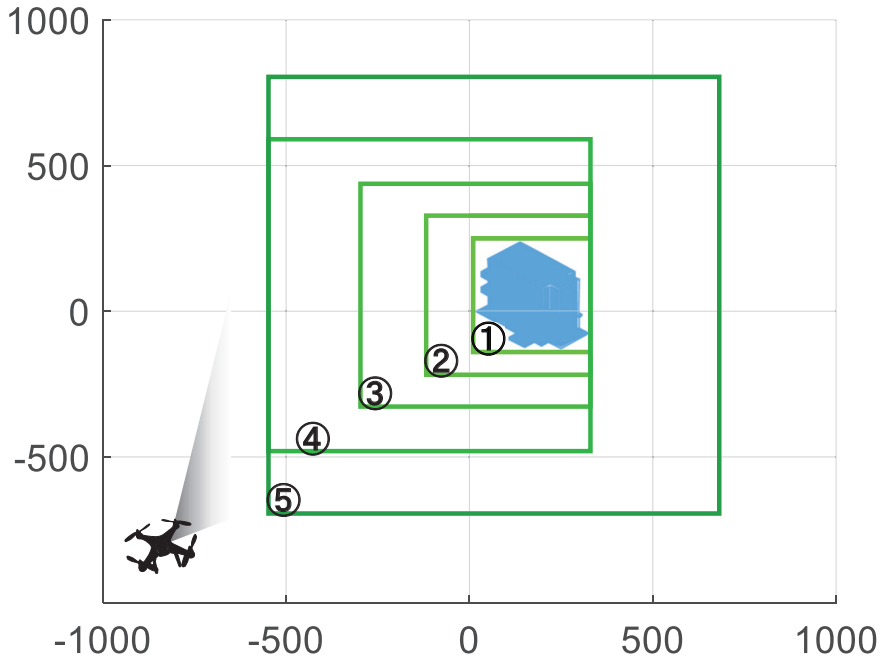


Fig. 4.15 UAV lightweight navigation results: (c) An example of navigation procedure

Table 4.2 Experimental settings of UAV-assisted edge computing

Parameter	Value	Parameter	Value
Number of users	100	d_{fs}, d_0	5, 100 m
Number of UAVs	50	$\gamma_{los}, \gamma_{nlos}$	2, 2.5
Flight altitude of UAVs	100 m	X_g^{los}, X_g^{nlos}	5, 20 dB
Bit rate of LoRa	50 $kbps$	α, β	15.27, -0.88
Frequency of LoRa	1 GHz	γ_{ld}	2
Bandwidth of LoRa	500 kHz	X_g^{ld}	0 dB
Transmit Power of LoRa	100 mW	σ_g^2	-100 dBm

As shown in Fig. 4.13, here I compare the results of different α. Blue, red and yellow bars stand for the time cost on moving forward/backward, turning left/right and ascending/descending. A larger α means the distance of judging whether to approach or apart from target. Compared with forward/backward and left/right, the time cost on adjustments of flight altitude changes not much. To further demonstrate the principles of the lightweight navigation strategy, I add Fig. 4.14 and 4.15. Fig. 4.14 gives an example that one UAV is assigned to complete five tasks at different places. From the figure, I can know that before reaching in front of the target, UAV constantly fine-tuning the flight posture based on the recognition of captured images. And for each movement, Fig. 4.15 gives the overlayed result showing how a UAV is flying near the target while receiving feedback from the camera lens. Boxes 1 to 5 are targets been found out in five in time order. For example, from box 1 to box 2, I need a fly instruction 011 according to Fig. 4.1 that turning left and approaching. Then after several rounds of feedback, from box 4 to box 5, I can judge that a right-and-forward is sent to fly control module. As a result, in this way, I can realize the lightweight navigation based on airborne vision.

In summary, I carry out experiments of testing the UAV onboard visual recognition and simulate the procedure of autonomous navigation according to the real-time recognition. The results show that my design on visual recognition can increase the performance and solution on UAV lightweight navigation is feasible.

4.5.3 UAV-assisted Edge Computing for Disaster Management

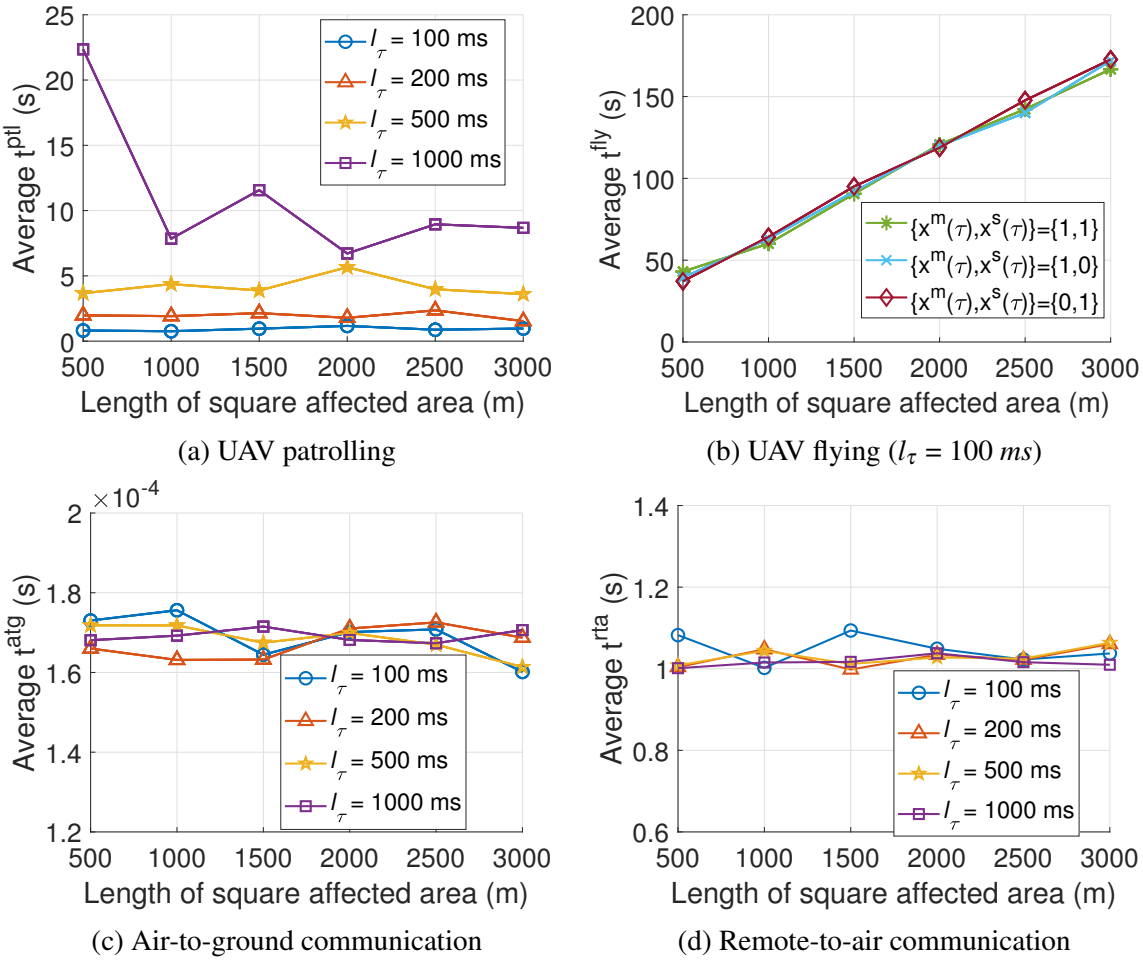
Here I evaluate my proposed strategies by simulation experiments in achieving UAV-mounted mobile edge computing for disaster management.

As shown in TABLE 4.2, there exist 100 users in a square affected area $\mathbb{R}^2$ after the disaster. I have 50 UAVs as carriers for providing MEC services to users in need. UAVs fly at an altitude of 100 m and receive requests from user devices at any time. LoRa concentrator at the center of the area plays the role of collecting user requests and handing them over to the central server for task management and assignment. I set the carrier frequency, bandwidth and maximum transmit power of LoRa to 1 GHz, 500 kHz and 100 mW (20 dBm). And for communication in the air-to-ground part, I use the parameters in 802.11n. I choose the suburban environment model [36] for α and β used in calculating the probability of LoS/NLoS path loss.

In each τ, devices in User Tier send requests to UAV nodes in the patrol model or passing by. With the help of long-range low-cost LoRa connection, UAV nodes can forward requests up to concentrator immediately together with the work state information $y^m(\tau)$. Central server decides the task performers of each task in queue according to the users' locations $\{\phi, \lambda, h\}$ and specific requirements $x^m(\tau)$. Central server also participates in task processing when $x^s(\tau) = 1$. At last, the tasks can not find idle UAV nodes or available computational resources in the server will be pushed back into Q_t with new ones received in this τ. I set different l_τ from 100 ms to 1 s and repeat each group 10000 times. The results are as follows.

The reason I choose l_τ as a variable in performance evaluation is that, as I choose the Markov chain model to describe the system state transition, it is especially important when defining discrete time-slot. In the case that the other experiment settings are the same, the length of τ may affect the overall state change frequency and then determine performance parameters such as SNR and channel capacity under different conditions. In addition with the situation that user density ρ_u changes in x-axis while the total number stays the same, I would like to explore what kind of increase in burden may be brought to UAVs in use, and how LoRa can enhance the UML system such as no excessive fluctuations in time or energy consumption.

Fig. 4.16 shows the time cost on each task being solved in the UML system. First, in Fig. 4.16a, four colored polylines stand for the average patrol time with different lengths of τ. In order to take advantage of the high mobility of UAVs, patrol mode is able to make use of the idle UAV nodes. The average time cost on this mode is consistent with l_τ from high to low in numerical value. Purple line with l_τ = 1000 ms is far higher than the results under other conditions. And there exist significant fluctuations as the user density changes. In my system model design, large l_τ may reduce the system state transition speed. And in relation to the time spent on UAV patrolling, it is possible to perform multiple task assignments within the same time (such as flying to an area that has not been visited and then picking up new tasks), and only a few commands can be executed when l_τ is large. Fluctuations here



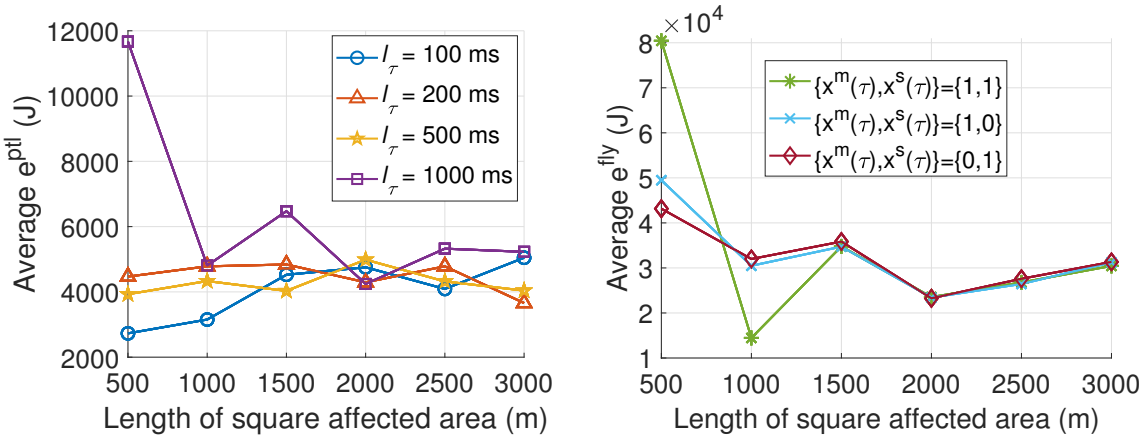
(a) UAV patrolling

(b) UAV flying (l_τ = 100 *ms*)

(c) Air-to-ground communication

(d) Remote-to-air communication

Fig. 4.16 Simulation results of time cost on UAV-based mobile edge computing using LoRaWAN



(a) UAV patrolling

(b) UAV flying (l_τ = 100 ms)

Fig. 4.17 Simulation results of energy cost on UAV-based mobile edge computing using LoRaWAN

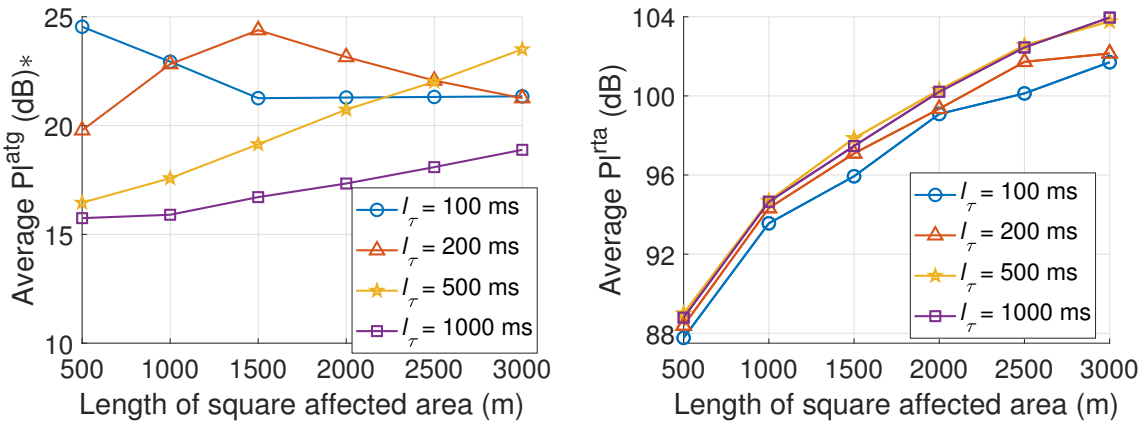
can be explained that when the affected is small, there may be no obvious differences in responsibility areas among UAVs. m_1 may have done what m_2 should have done, and m_2 has to stay in patrol mode longer, similar situations like this will bring uncertainty to the results.

Fig. 4.16b gives the average fly time of a single task being solved. I choose task indicators of two tiers as variable for t^{fly}, that is, task with $\{x_i^m(\tau), x_i^s(\tau)\} = \{1,1\}$, $\{1,0\}$ and $\{0,1\}$. As user density decreases, the UAV flying time required for each task increases linearly. Moreover, the overlapped results indicate that my UML system treats tasks of different needs equally.

Since the time costs on air-to-ground (802.11n) and remote-to-air (LoRa) are in the same order of magnitude, I divide the results of t^{hov} into Figs. 4.16c and 4.16d. As shown in the figures, t^{atg} and t^{rta} under the current experimental settings are relatively stable. It only needs about 0.17 ms for a single task in air-to-ground communication and about 1 s for remote-to-air. As a result, I can get the ratio of time cost between two parts as 1.7×10^{-4}.

Second, I obtain the results of energy consumption in different modes. First, the part of e^{ptl}, l_τ = 1000 ms in purple line keep the similar pattern. Especially when the user density ρ_u = 400 km^{-2}, each task costs about 12000 J on UAV patrolling averagely. Situations of the other l_τ show some irregularity. That is, compared to time cost, energy consumption may be weaker in correlation with different lengths of time-slot.

Then in the case of UAV flying, energy consumption also appears different from time Fig. 4.16. As shown in Fig. 4.17b, the result of $\{x_i^m(\tau), x_i^s(\tau)\} = \{1,1\}$ in green line behaves significant fluctuations when length of square area is 500/1000 m. Since tasks of $\{1,1\}$ can be satisfied by either UAV nodes in Service Tier or central server in Control Tier, they may



(a) Air-to-ground communication

(b) Remote-to-air communication

Fig. 4.18 Simulation results of path loss on UAV-based mobile edge computing using LoRaWAN

not have advantages in the priority of task assignment. According to Equation (4.29), since e^{atg} and e^{rta} only relate respective time cost, I skip this part of results.

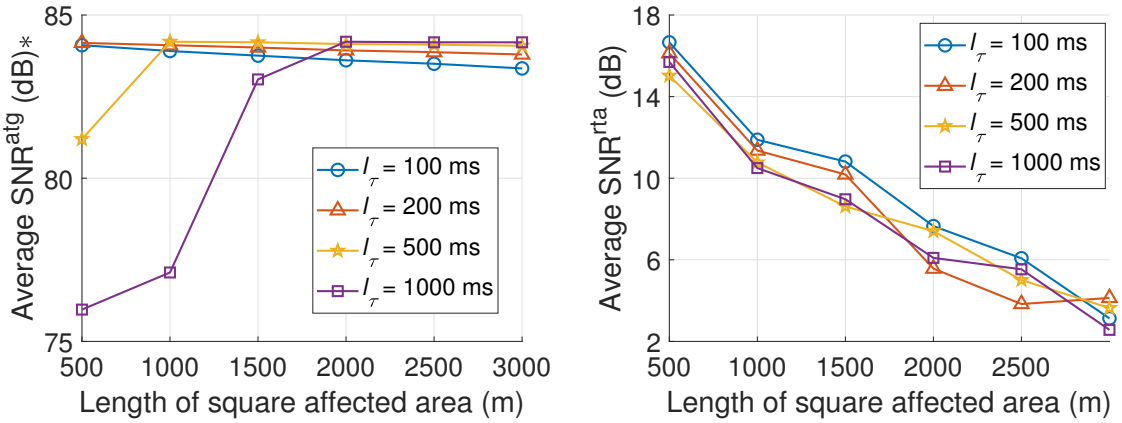
Besides two main metrics, I also consider the path loss (dB), SNR (dB) and channel capacity (Mbit/s) of air-to-ground and remote-to-air communications.

Figs. 4.18a and 4.18b show the path losses of two parts. $*$ marker in Figs. 4.18a stands for the modification in setting y-axis. I calculate the original numerical values by $(Pl_{org} - 105.4109117583) \times 10^{12}$. As a result, the path loss in LoS/NLoS model of air-to-ground communication stays at 105.41 dB. And for remote-to-air part in log-distance model of LoRaWAN, path loss increases with sparser user density and larger l_τ

The same with Fig. 4.18a, I calculate the original values by $(SNR_{org} - 14.58908824167) \times 10^{13}$. The SNR of air-to-ground also stays at 14.59 dB. For remote-to-air part of LoRaWAN, SNR decreases with sparser user density.

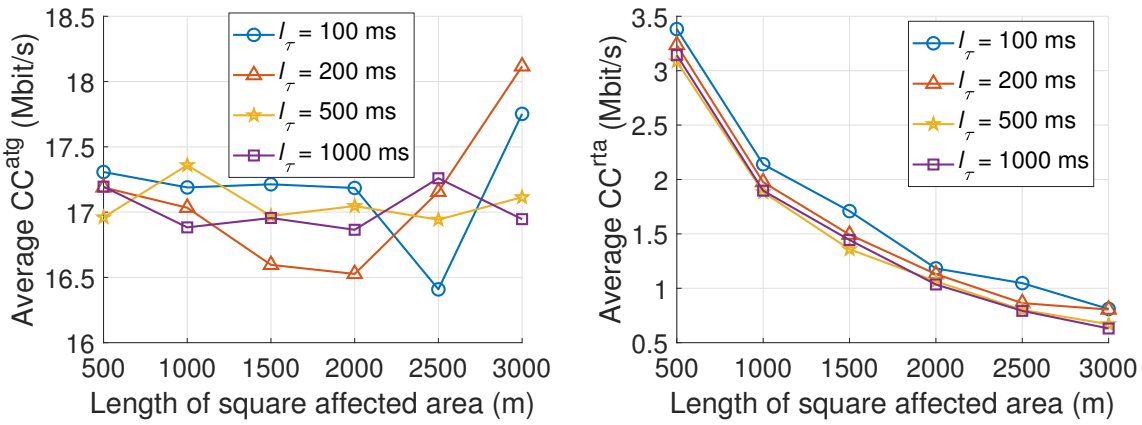
Lastly, I calculate the channel capacity by Equations (4.16) and (4.19). Air-to-ground part in 802.11n has larger but unstable channel capacity as user density or l_τ changes. And remote-to-ground of LoRa shows regular variation that channel capacity decreases with sparser user density and larger l_τ.

In summary, through the simulation results of time and energy cost, path loss, SNR and channel capacity, my proposed task management strategies can provide low-cost MEC services based on UAV and LoRaWAN to users in the affected area after a disaster.



(a) Air-to-ground communication (b) Remote-to-air communication

Fig. 4.19 Simulation results of SNR on UAV-based mobile edge computing using LoRaWAN



(a) Air-to-ground communication (b) Remote-to-air communication

Fig. 4.20 Simulation results of channel capacity on UAV-based mobile edge computing using LoRaWAN

Chapter 5

Conclusions and Future Directions

This chapter draws the conclusions of this thesis and give some future works. Through Chapter 2 4, I introduce my solutions of applying different emerging technologies in building the next generation disaster response system.

In the first part of emergency networking, I design an information-centric fog computing architecture to fast build a temporary emergency network while the original ones can not be used. I focus on solving name-based routing for disaster relief by applying the idea from six degrees of separation theory. I first put forward a 2-tier information-centric fog network architecture under the scenario of post-disaster. Then I model the relationships among ICN nodes based on delivered files, and propose a name-based routing strategy to enable fast networking and emergency communication. I compare with DNRP under the same experimental settings and prove that my strategy can achieve higher work performance.

In the second part of efficiency optimization, I introduce the idea of edge caching in prolong the lifetime of rebuilt network. I focus on how to improve energy efficiency of edge caching using in-memory storage and processing. Here I build a 3-tier heterogeneous network structure and propose two edge caching methods using different TTL designs & cache replacement policies. I use total energy consumption and backhaul rate as the two metrics to test the performance of in-memory caching method and compare with conventional method based on disk storage. The simulation results show that in-memory storage and processing can help save more energy in edge caching and share considerable workload in percentage.

In the third part of coverage expansion, I apply UAV technology and real-time image recognition in user search and autonomous navigation. I focus on the problem of designing a navigation strategy based on the airborne vision for UAV disaster relief. After the survey of related works on UAV fly control in disaster management, I find that in consideration of the current UAV manufacturing technology and actual demand on unmanned search &

rescue, a lightweight solution is in urgent need. As a result, I design a lightweight navigation strategy based on visual recognition using transfer learning. In the simulation part, I evaluate my solutions using 1/150 miniature models and test the feasibility of the navigation strategy. The results show that my design on visual recognition has the potential for a breakthrough in performance and the idea of UAV lightweight navigation can realize real-time flight adjustment based on feedback. There exist some drawbacks in my work such as the precision in navigation, and the lack of real-world verification. In the future, since my work still stays in the experimental stage, I am going to consider more technical details and make efforts to achieve the implementation of UAV autonomous navigation in disaster management.

References

[1] Adamic, L. A. and Huberman, B. A. (2002). Zipf's law and the internet. *Glottometrics*, 3(1):143–150.

[2] Al-Hourani, A. and Gomez, K. (2018). Modeling cellular-to-uav path-loss for suburban environments. *IEEE Wireless Communications Letters*, 7(1):82–85.

[3] Al-Hourani, A., Kandeepan, S., and Lardner, S. (2014). Optimal lap altitude for maximum coverage. *IEEE Wireless Communications Letters*, 3(6):569–572.

[4] Andrews, J. G., Baccelli, F., and Ganti, R. K. (2011). A tractable approach to coverage and rate in cellular networks. *IEEE Transactions on Communications*, 59(11):3122–3134.

[5] Baccelli, F. and Blaszczyszyn, B. (2010). Stochastic geometry and wireless networks: Volume ii applications. *Foundations and Trends in Networking*, 4(1-2):1–312.

[6] Beneventi, F., Bartolini, A., Cavazzoni, C., and Benini, L. (2017). Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 1038–1043.

[7] Bettstetter, C., Resta, G., and Santi, P. (2003). The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269.

[8] Boccardi, F., Heath, R. W., Lozano, A., Marzetta, T. L., and Popovski, P. (2014). Five disruptive technology directions for 5g. *IEEE Communications Magazine*, 52(2):74–80.

[9] Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA. ACM.

[10] Bor, M. C., Roedig, U., Voigt, T., and Alonso, J. M. (2016). Do lora low-power wide-area networks scale? In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pages 59–67, New York, NY, USA. ACM.

[11] Buzzi, S., I, C., Klein, T. E., Poor, H. V., Yang, C., and Zappone, A. (2016). A survey of energy-efficient techniques for 5g networks and challenges ahead. *IEEE Journal on Selected Areas in Communications*, 34(4):697–709.

[12] Chen, M., Hao, Y., Qiu, M., Song, J., Wu, D., and Humar, I. (2016). Mobility-aware caching and computation offloading in 5g ultra-dense cellular networks. *Sensors (Basel, Switzerland)*, 16(27347975):974.

[13] Chen, M., Mozaffari, M., Saad, W., Yin, C., Debbah, M., and Hong, C. S. (2017). Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience. *IEEE Journal on Selected Areas in Communications*, 35(5):1046–1061.

[14] Chen, S. W., Shivakumar, S. S., Dcunha, S., Das, J., Okon, E., Qu, C., Taylor, C. J., and Kumar, V. (2017). Counting apples and oranges with deep learning: A data-driven approach. *IEEE Robotics and Automation Letters*, 2(2):781–788.

[15] Chung, K. and Park, R. C. (2016). P2p cloud network services for iot based disaster situations information. *Peer-to-Peer Networking and Applications*, 9(3):566–577.

[16] DJI. Dji - the future of possible.

[17] DJI. Matrice 100 specs.

[18] Dong, M., Ota, K., Lin, M., Tang, Z., Du, S., and Zhu, H. (2014). Uav-assisted data gathering in wireless sensor networks. *The Journal of Supercomputing*, 70(3):1142–1155.

[19] Dong, M., Ota, K., Yang, L. T., Liu, A., and Guo, M. (2016). Lscd: A low-storage clone detection protocol for cyber-physical systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(5):712–723.

[20] Erdelj, M., Natalizio, E., Chowdhury, K. R., and Akyildiz, I. F. (2017). Help from the sky: Leveraging uavs for disaster management. *IEEE Pervasive Computing*, 16(1):24–32.

[21] Ernst, C., Mladenow, A., and Strauss, C. (2017). Collaboration and crowdsourcing in emergency management. *International Journal of Pervasive Computing and Communications*, 13(2):176–193.

[22] Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874. ROC Analysis in Pattern Recognition.

[23] Feeney, L. M. and Nilsson, M. (2001). Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 3, pages 1548–1557 vol.3.

[24] Franco, C. D. and Buttazzo, G. (2015). Energy-aware coverage path planning of uavs. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, pages 111–117.

[25] Gabry, F., Bioglio, V., and Land, I. (2016). On energy-efficient edge caching in heterogeneous networks. *IEEE Journal on Selected Areas in Communications*, 34(12):3288–3298.

[26] Gai, K., Choo, K. R., Qiu, M., and Zhu, L. (2018a). Privacy-preserving content-oriented wireless communication in internet-of-things. *IEEE Internet of Things Journal*, 5(4):3059–3067.

[27] Gai, K. and Qiu, M. (2018). Blend arithmetic operations on tensor-based fully homomorphic encryption over real numbers. *IEEE Transactions on Industrial Informatics*, 14(8):3590–3598.

[28] Gai, K., Qiu, M., Xiong, Z., and Liu, M. (2018b). Privacy-preserving multi-channel communication in edge-of-things. *Future Generation Computer Systems*, 85:190 – 200.

[29] Ge, X., Yang, B., Ye, J., Mao, G., Wang, C., and Han, T. (2015). Spatial spectrum and energy efficiency of random cellular networks. *IEEE Transactions on Communications*, 63(3):1019–1030.

[30] Giusti, A., Guzzi, J., Cireşan, D. C., He, F., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Caro, G. D., Scaramuzza, D., and Gambardella, L. M. (2016). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667.

[31] Gregori, M., Gómez-Vilardebó, J., Matamoros, J., and Gündüz, D. (2016). Wireless content caching for small cell and d2d networks. *IEEE Journal on Selected Areas in Communications*, 34(5):1222–1234.

[32] Han, G., Yang, X., Liu, L., Guizani, M., and Zhang, W. (2018). A disaster management-oriented path planning for mobile anchor node-based localization in wireless sensor networks. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1.

[33] Hayajneh, A. M., Zaidi, S. A. R., McLernon, D. C., Di Renzo, M., and Ghogho, M. (2018). Performance analysis of uav enabled disaster recovery networks: A stochastic geometric framework based on cluster processes. *IEEE Access*, 6:26215–26230.

[34] Hemmati, E. and Garcia-Luna-Aceves, J. J. (2015). A comparison of name-based content routing protocols. In *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 537–542.

[35] Hemmati, E. and Garcia-Luna-Aceves, J. J. (2018). Making name-based content routing more efficient than link-state routing. *CoRR*, abs/1804.02752.

[36] Holis, J. and Pechac, P. (2008). Elevation dependent shadowing model for mobile communications via high altitude platforms in built-up areas. *IEEE Transactions on Antennas and Propagation*, 56(4):1078–1084.

[37] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications.

[38] Hu, Z., Zheng, Z., Song, L., Wang, T., and Li, X. (2018). Uav offloading: Spectrum trading contract design for uav-assisted cellular networks. *IEEE Transactions on Wireless Communications*, 17(9):6093–6107.

[39] Kimberling, C. (1994). Central points and central lines in the plane of a triangle. *Mathematics Magazine*, 67(3):163–187.

[40] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[41] Leishman, G. J. (2006). *Principles of helicopter aerodynamics with CD extra*. Cambridge university press.

[42] Li, H., Ota, K., and Dong, M. (2018a). Eccn: Orchestration of edge-centric computing and content-centric networking in the 5g radio access network. *IEEE Wireless Communications*, 25(3):88–93.

[43] Li, H., Ota, K., and Dong, M. (2018b). Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101.

[44] Li, H., Ota, K., and Dong, M. (2019). Deep reinforcement scheduling for mobile crowdsensing in fog computing. *ACM Trans. Internet Technol.*, 19(2):21:1–21:18.

[45] Li, L., Ota, K., and Dong, M. (2018c). Deep learning for smart industry: Efficient manufacture inspection system with fog computing. *IEEE Transactions on Industrial Informatics*, pages 1–1.

[46] Li, L., Ota, K., and Dong, M. (2018d). Deepnfv: A lightweight framework for intelligent edge network functions virtualization. *IEEE Network*, 33(1):136–141.

[47] Li, L., Ota, K., and Dong, M. (2018). Human in the loop: Distributed deep model for mobile crowdsensing. *IEEE Internet of Things Journal*, 5(6):4957–4964.

[48] Li, L., Ota, K., Dong, M., and Borjigin, W. (2017). Eyes in the dark: Distributed scene understanding for disaster management. *IEEE Transactions on Parallel and Distributed Systems*, 28(12):3458–3471.

[49] Liu, D., Chen, B., Yang, C., and Molisch, A. F. (2016). Caching at the wireless edge: design aspects, challenges, and future directions. *IEEE Communications Magazine*, 54(9):22–28.

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

[51] Liu, Z., Dong, M., Zhou, H., Wang, X., Ji, Y., and Tanaka, Y. (2016). Device-to-device assisted video frame recovery for picocell edge users in heterogeneous networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6.

[52] LoRa Alliance Technical Committee (2017). Lorawan 1.1 specification.

[53] Motlagh, N. H., Bagaa, M., and Taleb, T. (2017). Uav-based iot platform: A crowd surveillance use case. *IEEE Communications Magazine*, 55(2):128–134.

[54] Mozaffari, M., Saad, W., Bennis, M., and Debbah, M. (2018). Communications and control for wireless drone-based antenna array. *IEEE Transactions on Communications*, pages 1–1.

[55] Muldoon, S. F., Bridgeford, E. W., and Bassett, D. S. (2016). Small-world propensity and weighted brain networks. *Scientific Reports*, 6:22057.

[56] Naqvi, S. A. R., Hassan, S. A., Pervaiz, H., and Ni, Q. (2018). Drone-aided communication as a key enabler for 5g and resilient public safety networks. *IEEE Communications Magazine*, 56(1):36–42.

[57] Ota, K., Dao, M. S., Mezaris, V., and Natale, F. G. B. D. (2017). Deep learning for mobile multimedia: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, 13(3s):34:1–34:22.

[58] Ota, K., Dong, M., Gui, J., and Liu, A. (2018). Quoin: Incentive mechanisms for crowd sensing networks. *IEEE Network*, 32(2):114–119.

[59] Paper, C. W. (2018). Cisco global cloud index: Forecast and methodology, 2016–2021 white paper. Technical report, Cisco Systems, Inc.

[60] Paper, E. W. (2015). Mobile edge computing: A key technology towards 5g. Technical report, European Telecommunications Standards Institute (ETSI).

[61] Paper, G. W. (2016). 5g vision-the 5g infrastructure public private partnership: the next generation of communication networks and services. Technical report, The 5G Infrastructure Public Private Partnership (5G PPP).

[62] Parvez, I., Rahmati, A., Guvenc, I., Sarwat, A. I., and Dai, H. (2018). A survey on low latency towards 5g: Ran, core network and caching solutions. *IEEE Communications Surveys Tutorials*, 20(4):3098–3130.

[63] Patel, M., Naughton, B., Chan, C., Sprecher, N., Abeta, S., Neal, A., et al. (2014). Mobile-edge computing introductory technical white paper. *White paper, mobile-edge computing (MEC) industry initiative*, pages 1089–7801.

[64] Powers, D. M. W. (1998). Applications and explanations of Zipf's law. In *New Methods in Language Processing and Computational Natural Language Learning*.

[65] Prajzler, V. (2015). Lora, lorawan and loriot.io.

[RAK] RAK. Lora module rak811.

[67] Ramaswamy, L., Ling Liu, and Iyengar, A. (2005). Cache clouds: Cooperative caching of dynamic documents in edge networks. In *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 229–238.

[68] Rousseau, R. (2002). George kingsley zipf: life, ideas, his law and informetrics. *Glottometrics*, 3:11–18.

[69] Sardellitti, S., Scutari, G., and Barbarossa, S. (2015). Joint optimization of radio and computational resources for multicell mobile-edge computing. *IEEE Transactions on Signal and Information Processing over Networks*, 1(2):89–103.

[70] Sekander, S., Tabassum, H., and Hossain, E. (2018). Multi-tier drone architecture for 5g/b5g cellular networks: Challenges, trends, and prospects. *IEEE Communications Magazine*, 56(3):96–103.

[71] Series, M. (2015). Imt vision–framework and overall objectives of the future development of imt for 2020 and beyond. *Recommendation ITU*, pages 2083–0.

[72] Shao, X., Asaeda, H., Dong, M., and Ma, Z. (2019). Cooperative inter-domain cache sharing for information-centric networking via a bargaining game approach. *IEEE Trans. Network Science and Engineering*, 6(4):698–710.

[73] Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646.

[74] Simsek, M., Aijaz, A., Dohler, M., Sachs, J., and Fettweis, G. (2016). 5g-enabled tactile internet. *IEEE Journal on Selected Areas in Communications*, 34(3):460–473.

[75] Sánchez-García, J., Reina, D., and Toral, S. (2019). A distributed pso-based exploration algorithm for a uav network assisting a disaster scenario. *Future Generation Computer Systems*, 90:129 – 148.

[76] Vaquero, L. M. and Rodero-Merino, L. (2014). Finding your way in the fog: Towards a comprehensive definition of fog computing. *SIGCOMM Comput. Commun. Rev.*, 44(5):27–32.

[77] Vespignani, A. (2018). Twenty years of network science. *Nature*, 558:528–529.

[78] Wan, L., Lou, W., Abner, E., and Kryscio, R. J. (2016). A comparison of time-homogeneous markov chain and markov process multi-state models. *Communications in Statistics: Case Studies, Data Analysis and Applications*, 2(3-4):92–100.

[79] Wang, X., Yang, L. T., Kuang, L., Liu, X., Zhang, Q., and Deen, M. J. (2019). A tensor-based big-data-driven routing recommendation approach for heterogeneous networks. *IEEE Network*, 33(1):64–69.

[80] Wang, X., Yang, L. T., Liu, H., and Deen, M. J. (2018). A big data-as-a-service framework: State-of-the-art and perspectives. *IEEE Transactions on Big Data*, 4(3):325–340.

[81] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393:440.

[82] Wikipedia (2018). 2018 hokkaido eastern iburi earthquake.

[83] World Confederation for Physical Therapy (2016). What is disaster management?

[84] Wu, J., Dong, M., Ota, K., Li, J., and Guan, Z. (2017). Fcss: Fog computing based content-aware filtering for security services in information centric social networks. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1.

[85] Wu, J., Dong, M., Ota, K., Li, J., Yang, W., and Wang, M. (2019). Fog-computing-enabled cognitive network function virtualization for an information-centric future internet. *IEEE Communications Magazine*, 57(7):48–54.

[86] Yang, L. T., Wang, X., Chen, X., Han, J., and Feng, J. (2017). A tensor computation and optimization model for cyber-physical-social big data. *IEEE Transactions on Sustainable Computing*, pages 1–1.

[87] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Kong, J., and Jue, J. P. (2019). All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289 – 330.

[88] Zappone, A., Björnson, E., Sanguinetti, L., and Jorswieck, E. (2017). Globally optimal energy-efficient power control and receiver design in wireless networks. *IEEE Transactions on Signal Processing*, 65(11):2844–2859.

[89] Zappone, A. and Jorswieck, E. (2015). *Energy Efficiency in Wireless Networks via Fractional Programming Theory*. now.

[90] Zeng, Y. and Zhang, R. (2017). Energy-efficient uav communication with trajectory optimization. *IEEE Transactions on Wireless Communications*, 16(6):3747–3760.

[91] Zhang, H., Chen, G., Ooi, B. C., Tan, K., and Zhang, M. (2015). In-memory big data management and processing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1920–1948.

[92] Zhang, X., Chen, G., Wang, W., Wang, Q., and Dai, F. (2017). Object-based land-cover supervised classification for very-high-resolution uav images using stacked denoising autoencoders. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(7):3373–3385.

[93] Zhao, C., Dong, M., Ota, K., Li, J., and Wu, J. (2019). Edge-mapreduce-based intelligent information-centric iov: Cognitive route planning. *IEEE Access*, 7:50549–50560.

[94] Zhao, N., Lu, W., Sheng, M., Chen, Y., Tang, J., Yu, F. R., and Wong, K. (2019). Uav-assisted emergency networks in disasters. *IEEE Wireless Communications*, 26(1):45–51.

[95] Zhou, Z., Ota, K., Dong, M., and Xu, C. (2017). Energy-efficient matching for resource allocation in d2d enabled cellular networks. *IEEE Transactions on Vehicular Technology*, 66(6):5256–5268.

Publications

Journals

1. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "Fast Networking for Disaster Recovery," IEEE Transactions on Emerging Topics in Computing (TETC), In Press.

2. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "A Real Plug-and-Play Fog: Implementation of Service Placement in Wireless Multimedia Networks," China Communications, vol. 16, no. 10, pp. 191-201, October 2019.

3. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "Energy Efficient Hybrid Edge Caching Scheme for Tactile Internet in 5G," IEEE Transactions on Green Communications and Networking (TGCN), vol. 3, no. 2, pp. 483-493, June 2019.

4. Jianwen Xu, Kaoru Ota, Mianxiong Dong, Anfeng Liu and Qiang Li, "SIoTFog: Byzantine Resilient IoT Fog Networking," Frontiers of Information Technology & Electronic Engineering (FITEE), vol. 19, no. 12, pp. 1546-1557, December 2018. (Highlight Article)

5. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "Real-Time Awareness Scheduling for Multimedia Big Data Oriented In-Memory Computing," IEEE Internet of Things Journal, vol. 5, no. 5, pp. 3464-3473, October 2018.

6. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "Saving Energy on the Edge: In-Memory Caching for Multi-Tier Heterogeneous Networks," IEEE Communications Magazine, vol. 56, no. 5, pp. 102-107, May 2018.

Proceeding of International Conference

1. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "LUNA: Lightweight UAV Navigation Based on Airborne Vision for Disaster Management," The 12th IEEE International Conference on Cyber, Physical and Social Computing (CPSCom 2019), Atlanta, GA, USA, July 14-17, 2019. (Best Paper Award)

2. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "Information-Centric Fog Computing for Disaster Relief," The 3rd International Conference on Smart Computing and Communication (SmartCom 2018), Tokyo, Japan, December 10-12, 2018. (Best Innovative Paper Award)

3. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "Plug-and-Play for Fog: Dynamic Service Placement in Wireless Multimedia Networks," IEEE/CIC International Conference on Communications in China (ICCC 2018), Beijing, China, August 16-18, 2018. (Best Paper Award)

Under Review

1. Jianwen Xu, Kaoru Ota and Mianxiong Dong, "Big Data on the Fly: UAV-mounted Mobile Edge Computing for Disaster Management," IEEE Transactions on Network Science and Engineering (IEEE TNSE), Major Revision