Cooperative Resource Sharing in Mobile Cloud Computing

Wei Liu

January 9, 2015

Abstract

Cloud computing, which is aimed at providing infrastructures, platforms and software as services has been introduced and implemented in the last few years. It is widely recognized as the next generation of computing architecture. At the same time, wireless communication technologies have been extensively developed. The network connectivity and data throughput of mobile devices have been greatly improved. Therefore, a new paradigm for the integration of mobile devices with cloud computing promises to have a strong impact on our lives. It is named as Mobile Cloud Computing (MCC) and has attracted a great deal of attention from both industrial and academic communities. This thesis discusses cooperative resource sharing among mobile devices in mobile cloud computing. Particularly, this thesis presents resource discovery, local resource sharing and opportunistic resource sharing mechanisms. This thesis is composed of the following 5 chapters.

Chapter 1 introduces the motivation and background of this thesis.

Chapter 2 discusses an energy-efficient method of adaptive resource discovery against the background of MCC. According to different network environments, the proposed method transforms between centralized and flooding modes to save energy. Theoretical models of both energy consumption and the quality of response information are presented. A novel algorithm is provided to enable the proposed method to select energy-efficient discovery mode automatically according to different network environments. Simulation results have demonstrated the effectiveness of the strategy and the efficiency of the proposed method.

Chapter 3 discusses a design of an energy-efficient local resource cloud (LRC) to accelerate task completion against the background of MCC. It is constructed of an on-demand local resource platform to enable resource sharing among mobile devices in the local vicinity. The theoretical models and the problem formulations are presented. An efficient algorithm with low computational complexity is introduced to solve the problem. Simulation results have indicated that the proposed models are efficient in terms of both energy consumption and task completion time. The proposed algorithm has also been proved to be suitable for dynamic mobile wireless environments.

Chapter 4 discusses the concept and design of an opportunistic resource sharing

mechanism against the background of MCC. Mobile devices make use of contact opportunities between themselves to share resources and accelerate their task completion. The theoretical models and the problem formulations are presented. The efficiency of the proposed mechanism has been validated through formal proofs. Finally, simulation results have indicated that the proposed mechanism is efficient in terms of both resource consumption and task completion time.

Chapter 5 concludes this thesis and shows the future direction.

Contents

Chapter 1 Introduction 1
1.1 Motivation 1
1.2 Background 2
1.3 Composition of the thesis 8

Chapter 2 Adaptive Resource Discovery in Mobile Cloud Computing 11
2.1 Background 11
2.2 System model 12
2.3 Proposed method of adaptive resource discovery 14
2.4 Proposed prediction algorithm 22
2.5 Extension of flooding method 24
2.6 Simulation 26
2.7 Related work 39
2.8 Conclusion 41

Chapter 3 Local Resource Sharing in Mobile Cloud Computing 43
3.1 Background 43
3.2 Related work 44
3.3 Definitions of the local resource cloud 46
3.4 Proposed model of the energy-efficient LRC 50
3.5 Proposed heuristic algorithm 56
3.6 Simulation 58
3.7 Conclusion 64

Chapter 4 Opportunistic Resource Sharing in Mobile Cloud Computing 65
4.1 Background 65

4.2 Related work 67
4.3 System architecture 68
4.4 Proposed models for opportunistic resource sharing 71
4.5 Proposed algorithms for opportunistic resource sharing 76
4.6 Simulation 80
4.7 Proofs of theorems 89
4.8 Conclusion 93

Chapter 5 Conclusions and Future Directions 95

Acknowledgement 97

Publication 99

Bibliography 101

Abbreviations

CC Cloud Computing

CCC Common Control Channel

CE Cross Entropy

CPU Central Processing Units

CRB Central Resource Broker

DTN Delay Tolerant Networks

FLOPS Floating-points Operation per Second

FIFO First In First Out

IoT Internet of Things

LRC Local Resource Cloud

LRU Least Recently Used

LTE Long Term Evolution

MCC Mobile Cloud Computing

M2M Machine to Machine

NP-hard Non-deterministic Polynomial hard

NRS Networks Resource Status

PC Personal Computer

QoE Quality of Experience

QoS Quality of Service

RIA Resource Information Availability

RN Relay Nodes

RP Resource Provider

RS Resource Seeker

RTT Round Trip Time

3G Third Generation of Mobile Telecommunications Technology

TTL Time-to-Live

VANETs Vehicular ad-hoc networks

VM Virtual Machine

WIMAX Worldwide Interoperability for Microwave Access

WLAN Wireless Local Area Networks

WSNs Wireless Sensor Networks

Chapter 1

Introduction

1.1 Motivation

Cloud computing [1], which is aimed at providing infrastructures, platforms and software as services has been introduced and implemented in the last few years. It is widely recognized as the next generation of computing architecture. Wireless communication technologies have simultaneously been extensively developed. Different kinds of wireless networks like the third generation of mobile telecommunications technology (3G), Bluetooth, wireless local area networks (WLANs) and worldwide interoperability for microwave access (WIMAX) have become available in our daily lives. Users can choose different networks according to their requirements. The network connectivity and data throughput of mobile devices have been greatly improved. Therefore, the integration of mobile devices with cloud computing has attracted a great deal of attention from both industrial and academic communities because of its potential value. Mobile Cloud Computing (MCC) [2] has been widely accepted as one of the most important solutions to this issue.

In the conventional client-server based architecture of MCC, the data center that is also named as "the cloud" provides overall resource management for mobile devices. Mobile devices utilize resources in the cloud to enhance their functionality and improve their processing capabilities. Therefore, the service provided by the client-server based architecture highly depends on the processing capability of the cloud and the connectivity of wireless networks such as the throughput of cellular networks. However, the mobile data traffic is forecast to increase 13 times by 2017, with the volume climbing to 13.2 exabytes per month and the number of users approaching

approximately 5.2 billion [3]. This increase in traffic demand is overloading cellular networks, forcing them to operate close to and often beyond their capacity. Considering the surge in machine-to-machine (M2M) [4] and internet of things (IoT) [5], the sheer number of mobile devices/applications are also saturating resources in the cloud. Possible solutions to this problem are the upgrading of cellular networks and the centralized cloud, e.g., move from 3G to LTE or deploy more powerful servers in the cloud. However, these solutions may not be cost-effective from the operator's perspective due to the high cost for power supply, location rents, deployment, and maintenance.

At the same time, along with the development of hardware and software technologies, modern mobile devices like smart phones, smart vehicles, and wearable devices have greater resources of computation, communication, sensing and other functions than before [6]. These resources are not always fully utilized by their owners and increase along with the number of mobile devices. As a result, three shortcomings of the conventional client-server based architecture of MCC have emerged:

(1) Although resources in the cloud and bandwidth of cellular networks are saturated rapidly, available resources like computation and short-range communication capabilities in the mobile devices themselves are not efficiently utilized.

(2) Persistent connectivity to the remote cloud may not be available for mobile devices, e.g., in rural areas or due to damaged infrastructures in disasters.

(3) Long delays are caused by the long distance and relatively limited bandwidth between the cloud and mobile devices.

1.2 Background

The cooperation based architecture of MCC considers mobile devices to be part of the cloud to solve the problems described above. Thus, apart from utilizing resources in the cloud, mobile devices in the local vicinity pool and share idle resources among themselves through short-range communication networks, e.g., WLAN ad hoc, and Bluetooth. This approach not only makes use of pervasive resources but also enables resource sharing even when mobile devices are not able to be connected to the

remote cloud. Delays could also be reduced by devices benefiting from high throughput short-range communications and location proximity. The traditional cloud in the cooperation-based architecture plays a role as a scheduler in the collaboration by wireless devices. Of course, according to different contexts, the data center can also provide resources to mobile devices as it does in the client-server based architecture. Due to its huge potential value, the cooperation based architecture of MCC has become to be the most interesting and visible research area of MCC at present [7–12]. This architecture is also called "Fog Computing" [9], "wireless distributed computing" [10], "edge computing" [11] and the "vehicular cloud" in vehicular ad-hoc networks (VANETs) [12]. Figure 1.1 shows the architecture of cooperative mobile cloud computing.

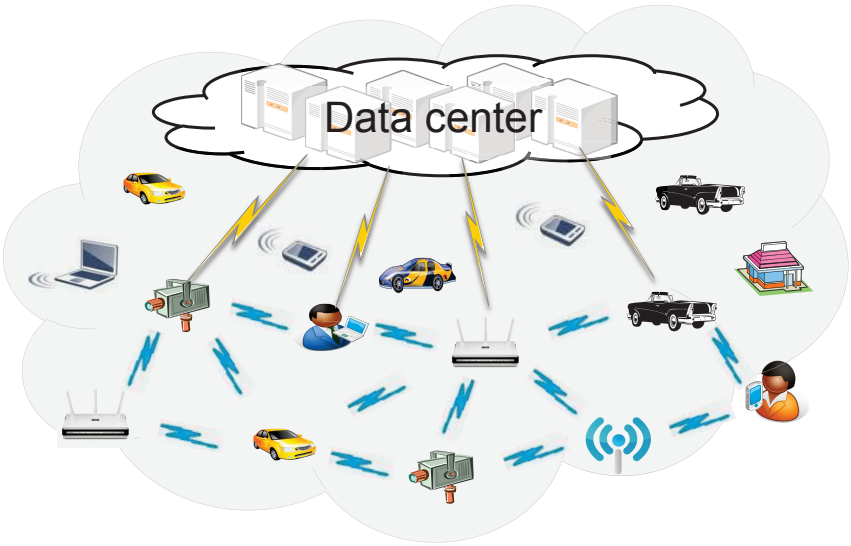


Fig. 1.1 Architecture of cooperative based MCC.

This thesis discusses resource sharing in the cooperative mobile cloud computing. Several terms are used to better clarify following contents:

Task (TS): A job generated by mobile users that consumes resources to finish it, e.g., data downloading consumes communication bandwidth resources and image processing consumes CPU's computational resources.

Resource seeker (RS): a mobile node that seeks available resources in other nodes to accelerate the processing of its tasks.

Resource provider (RP): a mobile node that provides resources to an RS.

A concept model of cooperative mobile cloud computing has been proposed in Fig. 1.2. A brief introduction to this model is given:

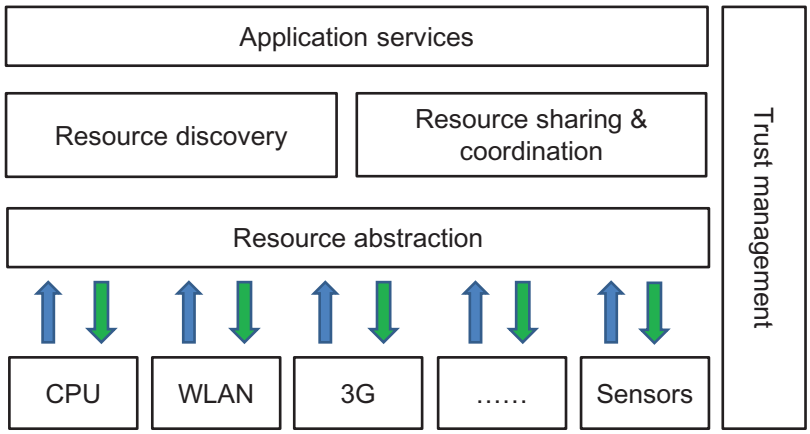


Fig. 1.2 Concept model of cooperative mobile cloud.

Resource abstraction

The resource abstraction component encapsulates physical details of the mobile devices. In detail, it is responsible for : (1) collecting resource information from different physical units of the mobile devices, and (2) allocating resources to process incoming tasks according to the instructions from upper layers.

Resource discovery

In order to implement resource sharing among mobile devices, an RS has to be able to discover other mobile devices that own idle resources first. This component helps the RS to discover candidates of RPs that could provide resources to it.

Resource sharing & coordination

After discovering candidates of RPs, the resource sharing & coordination component decides details of resource allocations and task processing, e.g., the selection of

appropriate RPs from all available candidates, and the allocation of tasks to different RPs according to their characteristics, e.g., amount of their idle resources, moving pattern, and security level.

Application services

Based on functions provided by the previous described components, the application services component provides required services to the mobile users. It encapsulates details of resource sharing among mobile devices from end users, but is capable of providing required services with impoved quality of service (QoS) and quality of experience (QoE) to satisfy users' requirements.

Trust management

Not every user of mobile devices wants to share his resources with unrelated people for security and economical reasons. However, mobile users may be motivated to share resources with acquaintances or people who share similar interests with them [8]. As a result, the devices that take part in resource sharing should be authenticated in advance. The trust management component provides related functions for mobile users. It should be noted that functions of trust management would spread over different components of the concept model, e.g., network trust managements based on hardware authentication, and application trust managements based on software authentication.

This thesis particularly discusses resource discovery and resource sharing & coordination components in the concept model. Detailed descriptions about these two components are given below.

1.2.1 Resource discovery

Resource discovery is one of the most important issues that need to be solved to achieve the goal of cooperative resource sharing in mobile cloud computing. Much research on resource discovery has been published. Conventional work generally adopts

one of the following two strategies to discover available resources:

(1) Centralized mode: As shown in Fig. 1.3(a), a resource registry is maintained in a central resource broker (CRB) in this mode. If an RS wants to allocate resources to process its tasks, it checks whether it has the resources itself. If not, it sends a resource request to the CRB. The CRB returns the identifications of mobile nodes in which the required resources are available.

(2) Flooding mode: As shown in Fig. 1.3(b), no resource registry is maintained in this mode. If an RS wants to discover resources to process its tasks, it checks whether it has the resources itself. If not, it floods a resource request in the area through short-range communications like WLAN ad-hoc. When a node that owns the required resource receives the request, it replies to the RS with its identification.

However, most of the existing work has adopted a fixed strategy for resource discovery and failed to adapt its strategy based on different network environments. At the same time, since battery capacity can not cope with the development of mobile applications, bottlenecks in energy consumption by mobile devices are unlikely to be solved in the near future. Therefore, many improved versions of energy-efficient methods based on both strategies have been proposed by researchers to reduce energy consumed in the process of resource discovery.

This thesis (Chapter 2) proposes a novel adaptive method of resource discovery from a different point of view to distinguish it from existing work. The proposed

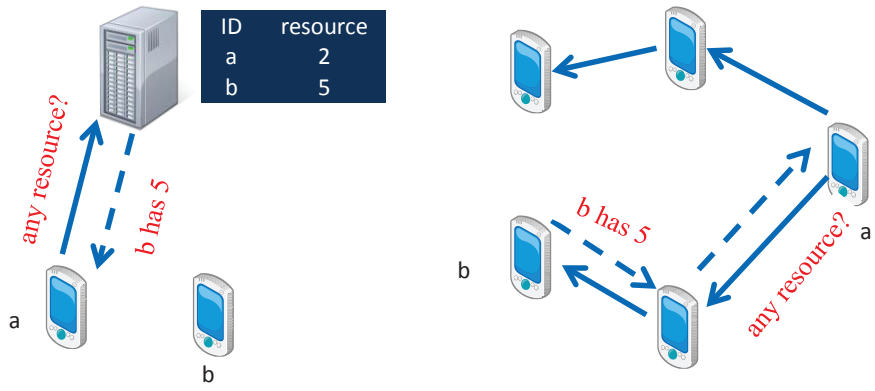


Fig. 1.3 Different strategies for resource discovery.

method automatically transforms between centralized and flooding strategies to save energy according to different network environments. Theoretical models of both energy consumption and the quality of response information are presented. A novel prediction algorithm is also designed to implement the new adaptive method of resource discovery.

1.2.2 Resource sharing & coordination

Resource sharing & coordination is the essential part of cooperative mobile cloud. After discovering candidates for RPs, the RS selects a group of RPs from all candidates and utilize their resources to accelerate its task completion. The selection of RPs depends on different characteristics, e.g., moving patterns of RPs, amount of idle resources in RPs, and security level of RPs. After selecting a group of RPs, resource coordination components is responsible for allocating appropriate (sub)tasks to selected RPs in accordance with their own characteristics and coordinating the following process of resource sharing.

Resource sharing & coordination in the mobile cloud computing could be roughly classified into two different categories according to their applied networks: (1) local resource sharing in consistently connected networks, and (2) opportunistic resource sharing in intermittently connected networks. In the local resource sharing, mobile devices in the local vicinity pool and share idle resources among themselves through consistently connected networks. In the opportunistic resource sharing, mobile devices share resources by making use of opportunistic contacts between themselves through intermittently connected networks. Both of these two approaches take mobile devices to be part of the resource cloud and make use of resources located at the "edge" of mobile networks to increase the scalability of information networks.

The previous work on local resource sharing has only concentrated on verifying the effectiveness of this concept to the best of our knowledge [13, 14]. Neither of them have referred to how to construct a local resource sharing platform, especially for the ones comprised of multiple RPs in dynamic mobile environments. Therefore, this thesis (Chapter 3) presents the design of a local resource sharing platform that

consists of multiple RPs and call it as a local resource cloud (LRC) to differentiate it from the traditional cloud which mainly refers to the centralized data center. Along with theoretical models and formal definitions of problems, a novel greedy algorithm with low computational complexity is proposed to construct LRCs that are efficient in terms of both energy consumption and task completion time.

The previous work on opportunistic resource sharing either analyzed simplified models that are not applicable to real situations or used heuristic algorithms without formal analysis [15–18]. Therefore, this thesis (Chapter 4) reports the concept and design of an opportunistic resource sharing mechanism that utilize resources in mobile devices through opportunistic contacts between them. It defines a data structure named opportunistic contact table (OCT) to maintain characteristics of opportunistic contacts between different pairs of mobile devices. Theoretical models of opportunistic resource sharing that are based on realistic parameters have been presented. The efficiency of the proposed mechanism has been validated through both formal proofs and extensive simulation.

1.3 Composition of the thesis

Chapter 2 discusses an energy-efficient method of adaptive resource discovery against the background of MCC. According to different network environments, it transforms between centralized and flooding modes to save energy. An extension of the flooding method is introduced as an optional choice. A prediction algorithm is provided to enable the proposed method to select energy-efficient discovery mode automatically according to different network environments. The simulation results show that the proposed adaptive solution to resource discovery is energy-efficient in different network environments due to its adaptivity.

Chapter 3 discusses a design of an energy-efficient local resource cloud (LRC) in cooperative MCC. It is constructed of an on-demand local resource platform to enable resource sharing among mobile devices. Theoretical models and formal definitions of problems are defined. An efficient greedy algorithm (GELRC) with low computational complexity is presented to solve the problem. Simulation results show that the

proposed model and method are efficient in terms of both energy consumption and task completion time. The proposed GELRC algorithm is also proved to be suitable for dynamic mobile wireless environments.

Chapter 4 discusses the concept and design of an opportunistic resource sharing mechanism. Mobile nodes make use of contact opportunities between themselves to share resources and accelerate their task completion. Theoretical models and formal definitions of problems are presented. Based on the theoretical models, two algorithms are designed to implement the proposed concept. The efficiency of the proposed mechanism has been validated through formal proofs. Finally, simulation results show that the proposed mechanism is efficient in terms of both resource consumption and task completion time.

Chapter 5 concludes this thesis and shows the future direction.

Chapter 2

Adaptive Resource Discovery in Mobile Cloud Computing

2.1 Background

As described in Chapter 1, to achieve cooperative resource sharing among mobile devices in MCC, it is quite important to find how available resources in nearby devices are discovered. Much research on resource discovery has been published [19–29]. However, most of the existing work has adopted a fixed strategy for resource discovery and failed to adapt this strategy based on different network resource statuses (NRSs), e.g., the degree of resource scarcity and the pattern of resource requirements.*1 Apart from that, more resources consume more energy. Battery capacity also becomes a bottleneck in wireless applications. Consequently, more and more research [23, 24, 28, 29] has aimed at providing energy-efficient solutions to resource discovery. However, there are two main problems in their research: (1) most of them have saved energy through sacrificing other important quality metrics like the accuracy and coverage of response information without formal quantitative analysis and (2) they have only taken into consideration resource discovery and energy consumption in homogeneous networks like 3G cellular or WLAN ad hocs alone. Obviously, energy consumption in heterogeneous networks is more realistic and more important for modern society.

This chapter proposes an energy-efficient method of resource discovery that auto-

*1 To differentiate it from "Network Status", which mainly refers to network traffic and bandwidth, Network Resource Status (NRS) is used in this chapter to represent the characteristics of resource distribution and usage in the network.

matically transforms between centralized and flooding strategies according to different NRSs. The three main contributions of this chaper are: (1) According to the best of our knowledge, this is the first proposal that has introduced an adaptive solution to resource discovery based on strategy transformations and the first work that has taken into consideration resource discovery in heterogeneous wireless networks. (2) We also established theoretical models of energy efficiency and quality of response information. (3) A prediction algorithm was designed to implement the proposal and it was proved to be energy-efficient through extensive simulations.

In the rest of this chapter, Section 2.2 introduces our system model. Our analysis of the proposed method of adaptive resource discovery is presented in Section 2.3. The prediction algorithm is introduced in Section 2.4. An extension of the proposed method is presented in Section 2.5. Section 2.6 explains how we evaluated the adaptive method through extensive simulations. Related work is discussed in Section 2.7. Conclusions are drawn and future work is discussed in the last section.

2.2 System model

We assumed that mobile nodes were in heterogeneous wireless networks including both 3G cellular and WLAN ad hoc in this chapter. Fig. 2.1 outlines the scenario. There is a central base station in the 3G cellular network that is able to communicate with all nodes in the area. We called it the central resource broker (CRB) in the background of resource sharing. Apart from that, every node can communicate with nearby nodes through an WLAN ad hoc. The assumed communication abilities are common in current smart phones and other devices. Nodes are assumed to be uniformly distributed throughout the area. [30] provides some application scenarios using this assumption. The effect of node mobility, which may violate this assumption, has been left for future work. Nodes either maintain a resource directory in the CRB through a widely-covered 3G network (a centralized mode) or flood resource requests in the area through a short-range WLAN ad hoc (a flooding mode) to discover resources.

Different resource discovery modes consume different amounts of energy. We try

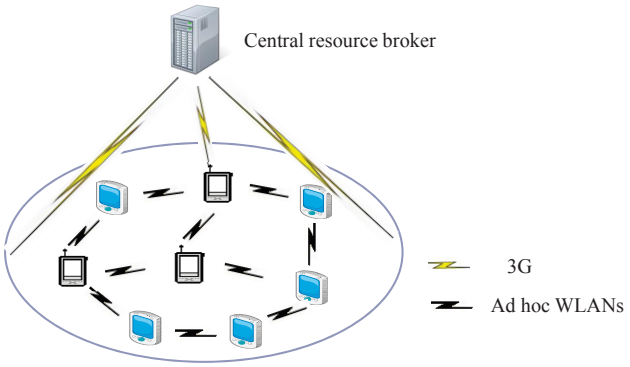


Fig. 2.1 System architecture.

to minimize energy consumption through transformations between the two modes according to different NRSs. Time is divided into consecutive time slots in our model. We define x_i as an indicator that specifies whether a centralized or flooding mode is selected in time slot i :

$$x_i = \begin{cases} 1 & \text{centralized mode is selected} \\ \\ 0 & \text{flooding mode is selected .} \end{cases} \tag{2.1}$$

Accordingly, $E_i(x_i)$ is defined as the energy consumed in time slot i based on different values of x_i. Without loss of generality, the period from time slot 1 to time slot Q is considered. The optimization problem is defined as the selection of an Q-dimensional vector comprised of x_i that minimizes the energy consumed by all 2^Q candidates, while keeping the expected value of resource information availability (RIA) no less than a threshold R_{thresh}. RIA is a quality metric of response information defined in Subsection 2.3.2 and R_{thresh} is a real value in [0,1].

$$\text{objective:} \quad \min \textstyle\sum_{i=1}^{Q} E_i(x_i) \quad ,$$

$$\text{subject to:}$$

$$x_i = 0 \text{ or } 1 \quad ,$$

$$E[RIA] \geq R_{thresh} \quad .$$

2.3 Proposed method of adaptive resource discovery

2.3.1 General description

As described in Section 2.2, nodes can discover available resources through both 3G cellular and WLAN ad hoc. There are two modes in the proposed method:

(1) Centralized mode: A resource directory is maintained in the CRB in this mode. If a node wants to allocate resources to its tasks, it first checks whether it has the resources itself. If not, it sends a resource request to the CRB through the 3G network. The CRB returns the identifications of nodes in which the required resources are available.

(2) Flooding mode: We adopted on-demand flooding [24, 31] for the flooding mode in this chapter. No resource information is maintained in the CRB in this mode. If node U wants to allocate resources to its tasks, it first checks whether it has the resources itself. If not, it sends a resource request with a unique sequence number as a broadcast package through the WLAN ad hoc, which is received by all the nodes within the wireless transmission range of U. When another node receives a new resource request, if it has the required resources, it replies to the resource seeker with its identification. Every receiving node decreases the time-to-live (TTL)[*2] value of the resource request by one. The node propagates this resource request by transmitting it as a broadcast package, if the remaining TTL value is positive. However, if this node receives a duplicate resource request that has already been dealt with, it discards the request to avoid duplicate propagations.

Nodes, automatically transform between the centralized and flooding modes to save energy in the proposed adaptive method of resource discovery based on different NRSs. Four sequential steps are executed. (1) Time is divided into consecutive time slots, (2) nodes send statistics to the CRB at the end of each time slot according to their experiences in the last time slot, (3) the CRB estimates the energy consumed

[*2] The time-to-live value is the number of hops that the package of a resource request can take through the network before it gets discarded.

by both modes based on the collected statistics, and (4) the CRB chooses the most energy-efficient method and notifies each node to use it in the next time slot.

2.3.2 Resource information availability and maintenance

Available resources in a node might change at certain times. This is affected by factors like task processing, environment changes, and remaining battery power. We have only taken into consideration the factor of task processing that is generated from two sources in this chapter.

(1) Nodes allocate resources to a task from themselves.
(2) Nodes allocate resources to a task from other nodes.

As defined in Section 1.2, a task is a job generated by users that consumes resources to finish it. Different tasks consume different kinds of resources, e.g., image processing consumes CPU resources (FLOPS) and data transmission consumes bandwidth resources (Mbps). Since the proposal in this chaper is not constrained by specific types of tasks or resources, we can encapsulate these details with the concept of the abstract size of tasks and units of resources in the following descriptions. Task processing occupies resources while the end of task processing releases occupied resources.

Resource information availability (RIA), which reflects the quality of response information, is defined as: the possibility that the responses to a request will accurately include all available resource information. This includes two aspects: accuracy and coverage. Energy is consumed to maintain RIA.

RIA in centralized mode

Every node benefits from the wide coverage of 3G network and can register its resource information with the CRB in the centralized mode. The coverage aspect of RIA is completely maintained. However, when the amount of available resources changes, nodes should update resource information in the CRB to maintain the accuracy aspect of RIA.

The number of resource information updates to maintain RIA must be calculated to estimate the energy consumed by the centralized mode. First, only one type of resource called A is considered. All the model parameters listed in Table 2.1 are for one time slot.

Table. 2.1 Parameters for RIA maintenance

S	Expected task size of a request for resource A
R	Sum of resources A in all nodes
λ_A	Number of generated tasks for resource A
λ_{A-o}	Number of generated resource requests for resource A
T	Length of one time slot
$T^{'}$	Processing time for one task with all resources A
N	Maximum number of tasks that can be processed
$F_{A-regist}$	Expected number of updates for resource A
F_{regist}	Expected number of updates for all resource types

The processing time for one task with all available resources A in the nodes is:

$$T^{'} = \frac{S}{R} \ . \tag{2.2}$$

The capacity of all nodes that indicates the maximum number of tasks that can be processed in one time slot is:

$$N = \frac{T}{T'} \ . \tag{2.3}$$

Depending on the relationship between λ_A and N, there are two situations for the number of updates in one time slot:

(1) $\lambda_A > N$: the number of tasks that can be processed in one time slot is constrained by the capacity of all nodes N. A node needs to update resource information at both when resources are allocated and released. Consequently, $F_{A-regist}$ can be calculated as:

$$F_{A-regist} = 2 \times N = \frac{2 \times R \times T}{S} \ . \tag{2.4}$$

(2) $\lambda_A \leq N$: the number of tasks that can be processed in one time slot is deter-

mined by the number of generated tasks λ_A. $F_{A-regist}$ is:

$$F_{A-regist} = 2 \times \lambda_A \ . \tag{2.5}$$

Obviously, the number of updates for all types of resources in one time slot is:

$$F_{regist} = \sum F_{I-regist} \ , \tag{2.6}$$

where I stands for different types of resources.

RIA in flooding mode

Because on-demand flooding is adopted, nodes in the flooding mode fully know what their available resources are when they receive a request. The accuracy aspect of RIA is completely maintained. However, the resource seeker has to set a large enough TTL value to ensure request packages reach every node in the area to maintain the coverage aspect of RIA.

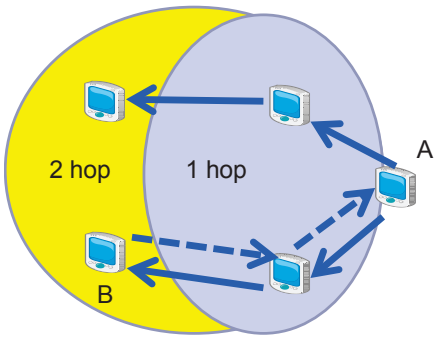


Fig. 2.2 RIA in flooding mode.
@2014 Elsevier

To estimate the appropriate TTL value, we define P_i as the expected percentage of extra nodes that will receive the requests when their TTL value increases from $i-1$ to i, e.g., $P_1 = 40\%$ in Fig. 2.2, since 40% of new nodes will receive the requests when their TTL value increases from 0 to 1. We define $P_0 = 1/N_{node}$, which is the percentage occupied by the seeker itself. N_{node} is the number of nodes in the area. As a result, TTL value k should satisfy the following equation asymptotically to provide

complete coverage of nodes:

$$\sum_{i=0}^{k} P_i = 1 \ , \tag{2.7}$$

where k is the minimum integer value that satisfies the equation, e.g., k is 2 for the network topology depicted in Fig. 2.2, since $P_0 + P_1 + P_2 = 1$. The calculation of P_i depends on three parameters of (1) the wireless transmission range of the nodes, (2) the number of nodes in the area, and (3) the size of the area (length and width). The detailed method and equations have not been explained in this chapter because they are beyond its scope. However, a detailed description is available in [32].

2.3.3 Tradeoff between RIA and energy consumption

We assumed the RIA was completely maintained in the previous model. However, nodes may also agree to a lower RIA to save energy. This subsection discusses our analysis of the tradeoff between the two metrics.

Tradeoff in centralized mode

A node updated resource information right after its available resources had changed in Subsection 2.3.2. Instead of that, it could wait for a period of time before the update. Because update frequency decreases, less energy is consumed with this strategy.

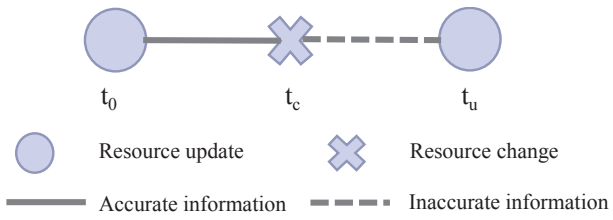


Fig. 2.3 Tradeoff in centralized mode.

As Fig. 2.3 shows, t_0 is the time for the previous resource update. The t_c is the time when the available resources changed. The node did not update information in the CRB until t_u. Resource requests generated between t_c and t_u were responded to

with inaccurate information by the CRB. If t_u is further from t_c, the expected RIA from t_0 to t_u decreases. If t_u is further from t_c, on the other hand, update frequency F_{regist} also decreases, and less energy is consumed to maintain the RIA.

Resource requests from other nodes are assumed to be a Poisson arriving process [33] in the following analysis. Let H represent a random variable defined as:

$$H = \begin{cases} 1 & \text{response to a request is accurate} \\ \\ 0 & \text{response to a request is inaccurate .} \end{cases} \tag{2.8}$$

We need to find expected value $E[H]$ for one request:

$$E[H] = \int_0^{T_{0-u}} f(t) \times E[H|T_{rc} = t]dt \ , \tag{2.9}$$

where T_{0-u} is the length of the period from t_0 to t_u, T_{rc} is a random variable of t_c, and $f(t)$ is the probability density function of T_{rc}. Assuming a Poission arriving process, the arrival of requests are uniformly distributed in T_{0-u} provided a request occurs within this period. Therefore, $f(t)$ is:

$$f(t) = \frac{1}{T_{0-u}} \ . \tag{2.10}$$

$E[H|T_{rc} = t_c]$ is the expected value of H provided that available resources changed at time t_c. Obviously, the requests generated between t_0 and t_c are responded to with accurate information. Based on a uniform distribution for the arrival time of requests, the expected value of H is calculated as:

$$E[H|T_{rc} = t_c] = \frac{t_c - t_0}{t_u - t_0} \ . \tag{2.11}$$

Nodes can keep $E[H|T_{rc} = t_c]$ constant through choosing t_u according to t_0 and t_c. Let $\alpha = (t_u - t_c)/(t_c - t_0)$. Based on Eqs. (2.9), (2.10), and (2.11), $E[H]$ can be computed as:

$$E[H] = E[H|T_{rc} = t_c] \int_0^{T_{0-u}} f(t)dt = \frac{t_c - t_0}{t_u - t_0} = \frac{1}{1+\alpha} \ . \tag{2.12}$$

To keep $E[H]$ no less than R_{thresh}, the choice of t_u should make α satisfy the inequality:

$$\alpha \leq \frac{1 - R_{thresh}}{R_{thresh}} \ . \tag{2.13}$$

The resulting expected number of updates in one time slot is reduced to:

$$F'_{regist} = F_{regist} \times \frac{t_c - t_0}{t_u - t_0} = \frac{F_{regist}}{1 + \alpha} \ , \tag{2.14}$$

since the update interval increases from $(t_c - t_0)$ to $(t_u - t_0)$.

Tradeoff in flooding mode

The TTL value of resource requests in Subsection 2.3.2 was assumed to be large enough to reach every node in the area. A larger TTL value means a larger probability of discovering the required resource. However, it also consumes more energy to propagate requests. If a lower RIA is acceptable, a smaller TTL value could be used to save energy.

The method of evaluation is nearly the same as that in Eq. (2.7). However, only an R_{thresh} fraction of nodes are assumed to be covered by the requests:

$$\sum_{i=0}^{k} P_i \geq R_{thresh} \ , \tag{2.15}$$

where k is the minimum integer value that satisfies the inequality, e.g., when R_{thresh} is 0.5, k is 1 for the network topology depicted in Fig. 2.2, since $P_0 + P_1 = 0.6$.

2.3.4 Energy consumption models

This subsection presents the energy consumption models for both modes. We define N_{resp} as the average number of responses to a resource request. λ_o is the number of resource requests for all types of resources in one time slot.

The energy consumed by the centralized mode in one time slot can be calculated as:

$$\begin{aligned} E_{central} = &(\lambda_o \times E_{3G-trans}) + (N_{resp} \times \lambda_o \times E_{3G-recv}) \\ &+ (F_{regist} \times E_{3G-trans}) \ , \end{aligned} \tag{2.16}$$

where $E_{3G-trans}$ and $E_{3G-recv}$ are the energy consumed by transmission and reception through the 3G interface. The total consumption of $E_{central}$ includes three parts: sending resource requests to the CRB, $\lambda_0 \times E_{3G-trans}$, receiving resource responses from the CRB, $N_{resp} \times \lambda_0 \times E_{3G-recv}$, and sending updates to the CRB to maintain the RIA, $F_{regist} \times E_{3G-trans}$.

In the flooding mode, the expected number of neighboring nodes that are within the wireless transmission range of a node, N_{neig}, is equal to the expected number of extra nodes that will receive the resource requests when their TTL value increases from 0 to 1:

$$N_{neig} = P_1 \times N_{node} \ . \tag{2.17}$$

The average hot count (HC) from the resource seeker to the providers is:

$$HC = \sum_{i=0}^{k} P_i \times i \ . \tag{2.18}$$

W_{trans} and W_{recv} are the energy consumed by transmission and reception through the WLAN ad hoc interface. Only P_i percentage of nodes that receive a resource request for the first time at hop i will propagate the request when i is not the last hop. The energy consumed by propagating the request is $(\sum_{i=0}^{k-1} P_i \times N_{node}) \times W_{trans}$. All the neighbors of propagating nodes will receive the request. The energy consumed by receiving the request is $(\sum_{i=0}^{k-1} P_i \times N_{node} \times N_{neig}) \times W_{recv}$. Only unicasting is needed when returning response messages. The energy consumed by relaying and receiving responses is $N_{resp} \times (W_{trans} + W_{recv}) \times HC$, where HC is derived from Eq. (2.18). The energy consumed by one request in the flooding mode is the sum of the three previous parts:

$$\begin{aligned} E^{'}_{flooding} = &\sum_{i=0}^{k-1} P_i \times N_{node} \times W_{trans} + \sum_{i=0}^{k-1} P_i \times N_{node} \\ &\times N_{neig} \times W_{recv} + N_{resp} \times (W_{trans} + W_{recv}) \times HC \ . \end{aligned} \tag{2.19}$$

Therefore, the energy consumed by all requests in one time slot is:

$$E_{flooding} = \lambda_o \times E^{'}_{flooding} \ . \tag{2.20}$$

2.4 Proposed prediction algorithm

This subsection explains how we implemented the proposed adaptive method through a prediction algorithm. According to the energy consumption models in the last section, three statistics are needed to estimate the energy consumed by two modes in a time slot. These are:

(1) The number of generated resource requests λ_o ,
(2) The number of generated resource updates F_{regist} , and
(3) The average number of responses for each request N_{resp} .

Distributed nodes send the previous three statistics to the CRB at the end of each time slot according to their experiences in that time slot. The CRB processes raw data like sums up λ_o from all the nodes, then stores the records of previous N_{slot} time slots in a list. After initialization, the $Check$ part tests whether the NRS is too dynamic to be predicted. If there are more than N_{trans} transformations of the discovery mode in the retained records, the NRS is assumed to be unpredictable. The centralized mode is used until the NRS becomes relatively regular. If the energy consumption ratio of a better mode to a worse mode is less than C_{thresh} in the $Large\_diff$ part, the better mode is directly chosen. If none of the previous conditions are satisfied, the CRB uses the average value of previous N_{trend} records to predict the NRS of the next time slot in the $Predict$ part. The CRB chooses an energy-efficient mode and sends the decision to every node. If a transformation is from the flooding mode to the centralized mode, nodes should update their resource information in the CRB first to ensure RIA. The pseudo-codes on the next page describe the algorithm.

The N_{slot} in the prediction algorithm defines how long the records of passed time slots are kept. The ratio of N_{trans} to N_{slot} indicates the degree of dynamicism of the NRS. This is used to prevent meaningless transformations when the network is too dynamic to be predicted. C_{thresh} is just a threshold value. If the discrepancy

---

Prediction_algorithm $(N_{slot}, N_{trans}, C_{thresh}, N_{trend}, P_{thresh})$

Initialize :

preprocess raw data and create a new record r.

insert r into the list and delete outdated records.

Check :

if (there are N_{trans} or more transformations in the record list)

choose centralized mode.

goto $Make\_choice$.

$Large\_diff$:

if (better consumption / worse consumption $\leq C_{thresh}$)

choose better mode.

goto $Make\_choice$.

Predict :

$//X_{i-j}$ means the statistic X_j in the i-th record

$\lambda_o = \frac{\sum_{i=1}^{N_{trend}} \lambda_{i-o}}{N_{trend}}$, $F_{regist} = \frac{\sum_{i=1}^{N_{trend}} F_{i-regist}}{N_{trend}}$,

$N_{resp} = \frac{\sum_{i=1}^{N_{trend}} N_{i-resp}}{N_{trend}}$.

use λ_o , F_{regist}, N_{resp} to estimate energy consumption.

if (better consumption / worse consumption $\geq P_{thresh}$)

remain the current mode.

else

choose the better mode.

$Make\_choice$:

send decision to every node.

---

between two modes is quite large, it is highly probable that the better mode will be chosen in the next time slot except for unpredictably heavy changes in status. N_{trend}

indicates the smoothness of NRS changes. Because transformation also consumes energy, P_{thresh} prevents transformation when the difference between the two modes is small.

2.5 Extension of flooding method

The proposed adaptive method in the previous sections only included basic centralized and flooding methods. However, it can also be provided with improved versions of both methods. A simple extended flooding method is introduced in this section as an optional choice to illustrate this.

Intuitively, if more nodes have the required resources, there is a larger chance of finding available providers of resources with a smaller TTL value. Moreover, if fewer tasks are generated by nodes, it is also easier to find available providers of resources with a smaller TTL value. This is due to the fact that most of the resources are not used. Consequently, nodes can send related statistics to the CRB. The CRB calculates a reasonable TTL value for resource requests based on the collected statistics.

First, the method of calculating the TTL value with the extended flooding method

Table. 2.2 Parameters for method of the extended flooding

RES_{A-dens}	Percentage of nodes that has resources A (resource density)
RES_{A-con}	Expected percentage of resources A occupied by tasks (consumption ratio)
$RES_{A-con-A}$	Average number of resources A occupied by each task
RES_{A-avai}	Expected percentage of available resources A
R_{A-avai}	Expected percentage of nodes having available resources A
N_{A-res}	Average number of resources A each node owns
λ_A	Number of generated tasks for resource A
k_{ex}	TTL value used by extended flooding method
k	TTL value used by basic flooding method

is analyzed. One type of resource called A is considered. Related parameters are summarized in Table 2.2. The resource consumption ratio RES_{A-con} is the ratio of occupied resources A to the number of resources A in the area:

$$RES_{A-con} = \frac{\lambda_A \times RES_{A-con-A}}{RES_{A-dens} \times N_{node} \times N_{A-res}} \ , \tag{2.21}$$

where N_{node} is the number of nodes in the area.

According to Eq. (2.21), it is obvious that $0 \leq RES_{A-con} \leq 1$. The percentage of available resources A that has not been used is:

$$RES_{A-avai} = 1 - RES_{A-con} \ . \tag{2.22}$$

As a result, the expected percentage of nodes that still owns available resources A satisfies the following inequality:

$$R_{A-avai} \geq RES_{A-dens} \times RES_{A-avai} \ . \tag{2.23}$$

To find at least one available provider of resources A, the TTL value should be large enough to cover more than $1 - R_{A-avai}$ percentage of nodes:

$$\sum_{i=0}^{k_{ex}} P_i > 1 - Min(R_{A-avai}) = 1 - RES_{A-dens} \times RES_{A-avai} \ , \tag{2.24}$$

where k_{ex} is the smallest integer value that satisfies this inequality.

According to the described model, the CRB notifies nodes of TTL value k_{ex} for resources A. If a node failed to discover any provider with k_{ex}, it again tries to discover resources A with the basic flooding TTL value, k. The energy consumption model of the extended flooding method is nearly the same as that of the basic flooding method except that two different TTL values k_{ex} and k are used. The energy consumption for resources A is:

$$E_{A-flooding} = E_{A-flooding}(k_{ex}) + E_{A-flooding}(k) \ . \tag{2.25}$$

The energy consumption for all kinds of resources is:

$$E_{flooding} = \sum E_{I-flooding} \ , \tag{2.26}$$

where I stands for different types of resources.

The prediction algorithm has to be slightly changed to enable the extended flooding method to be integrated into the proposed adaptive method of resource discovery. Because the number of resource requests with TTL value k is unavailable in the centralized mode,[*3] only k_{ex} is used to estimate the energy consumption of the flooding mode. Both k_{ex} and k are used for estimation in the flooding mode. Although the methods of estimating energy are slightly different for the two modes, because of the conservative estimates in Eqs. (2.23) and (2.24), the proposed method is rarely affected. This is proved by the simulation results presented in Section 2.6. Because different types of resources are associated with different values of k_{ex}, the statistics λ_o, F_{regist}, and N_{resp} should also be sent separately for each type of resource.

It should be noted that the extended flooding method still reflects the tradeoff between RIA and energy consumption. Although it is energy-efficient, it only intends to discover at least one available provider of resources rather than all potential providers in the area.

2.6 Simulation

We verified the proposed adaptive method and prediction algorithm through simulations. We compared the energy consumed by the proposed method with that consumed by the centralized and flooding methods. The proposed method was further divided into two sub-categories: (1) the basic adaptive method (adaptive-b), which was provided with the basic flooding method and (2) the extended adaptive method (adaptive-ex), which was provided with the extended flooding method. The extended flooding method relied on the CRB to calculate the TTL value. Consequently, the performance of the extended flooding method alone was meaningless. Therefore, this was not evaluated in the simulations. The energy consumption caused by different TTL values (k and k_{ex}) in the adaptive-ex method was not further distinguished. Requests with the basic TTL value k in all the following simulations, actually only

[*3] Since the extended flooding method is not executed, the number of resource requests that failed to discover any provider with TTL value k_{ex} is unknown.

occupied less than 5% of the overall energy consumption caused by the extended flooding method. Since the effectiveness of the proposed method mainly resulted from transforming discovery modes, we did not focus on improving the centralized or flooding method itself. However, the proposed method also benefitted from the integration of improved components. This was proved by the superior performance of the adaptive-ex method in the simulations that followed.

2.6.1 Parameters, definitions and assumptions

The nodes in the simulations were uniformly distributed in a rectangular area. The nodes could discover available resources through the 3G network in the centralized mode or the WLAN ad hoc in the flooding mode. We assumed one unit of resources could process one task within a time slot.[*4] Every node generated tasks with an equal

Table. 2.3 Basic simulation parameters

Rectangular area	$1000 \times 1000\ m$
Number of nodes	100
Resource density, RES_{dens}	100%
WLAN ad hoc range	$250\ m$
3G transmit, $E_{3G-trans}$	20
3G receive, $E_{3G-recv}$	10
WLAN transmit, W_{trans}	1
WLAN receive, W_{recv}	0.5
Size of resource requests	1 KB
Size of response identifications	0.1 KB
Size of statistics messages	0.1 KB
Size of decision notification messages	0.1 KB
Time slot length	30s

[*4] This means the 'time slot length' parameter in Table 2.3 was not sensitive in the simulation. We chose 30 s just because of the tradeoff between the duration of simulation and machine processing capabilities.

probability.

The size of both the statistics message and the decision notification message were set to 0.1 KB. This was a conservative approximation. Indeed, only three integer or float variables of the payload were needed for the statistics messages. Only one bit of payload was needed for the decision notification messages. Since only a comparison between different methods was concerned, we adopted the results in [34] where the ratio of energy consumption between 3G and WLAN ad hocs was 20/1 without a specific unit.

The proposed adaptive methods chose the centralized mode when they started up. The expected RIA was 0.95 for both modes. The prediction algorithm was initialized with $< N_{slot} = 5, N_{trans} = 3, C_{thresh} = 0.5, N_{trend} = 2, P_{thresh} = 0.9 >$. The parameters for the algorithm are further discussed in Subsection 2.6.4. Other basic simulation parameters are listed in Table 2.3.

The flooding mode consumed a great deal of bandwidth when the number of resource requests, λ_o, is large. However, when λ_o was large, in the proposed adaptive solution to resource discovery, the energy consumed by the flooding mode was much greater than that consumed by the centralized mode. The proposed methods chose the centralized mode automatically to prevent "flooding storm". Because of this, channel collisions were rare when the messages were short like those exchanged in the resource discovery process. Therefore, we assumed an idealized communication channel in the simulations that followed.

Fig. 2.4 plots the energy consumed in one time slot for different modes with the basic parameters listed in Table 2.3. Each node had five units of resources in this simulation (flooding: the basic flooding method and flooding-ex: the extended flooding method). The discontinuity of energy consumed by the extended flooding method was due to the discontinuity of calculated TTL value k_{ex} for different numbers of generated tasks.

We divided the NRS into three regions according to the energy consumed by the centralized and basic flooding methods:

(1) Flooding region (F-R): the energy consumed by the basic flooding method was less than that by the centralized method, e.g., the number of generated tasks was

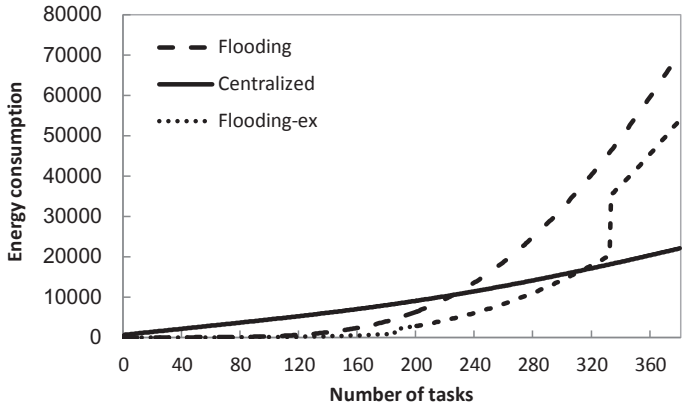


Fig. 2.4 Tradeoff in centralized mode.

within the range of [0 , 222) in Fig. 2.4.

(2) Centralized region (C-R): the energy consumed by the centralized method was less than that by the basic flooding method, e.g., the number of generated tasks was within the range of [245 , $+\infty$) in Fig. 2.4.

(3) Cross region (CR-R): the energy consumed by the two methods was nearly the same. This was defined as the number of generated tasks within a shift range of 5% based on the crossing point, e.g., the number of generated tasks was within the range of [222 , 245) in Fig. 2.4.

Due to the large differences in energy consumption in different simulation scenarios, it was difficult to unify the y-axes for energy consumption in all graphs. Since only comparisons in the same situation were concerned, this did not affect the simulation results. All the results in this section were an average value obtained from 100 simulation trials.

2.6.2 Performance in single region of NRS

We verified the energy consumed by the proposed methods in different regions of NRS separately. The energy consumed by the proposed methods should ideally satisfy three characteristics:

(1) If the NRS is in F-R, the energy consumption should be near that consumed by the flooding method and less than that by the centralized method.

(2) If the NRS is in C-R, the energy consumption should be near that consumed by the centralized method and less than that by the flooding method.

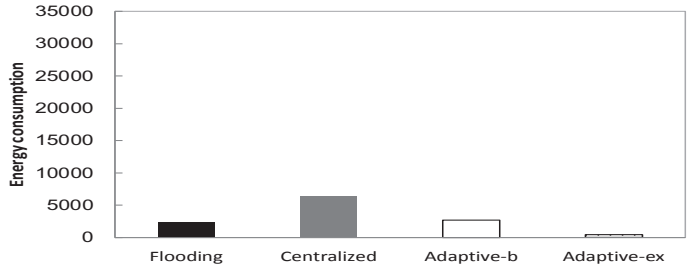
(3) If the NRS is in CR-R, the energy consumption should be near that consumed by both methods. This should prevent meaningless transformations, which only waste energy .

Every node had five units of resources in this simulation. The other parameters were the basic parameters summarized in Table 2.3. We chose 160 tasks to represent F-R, 300 tasks to represent C-R, and 240 tasks to represent CR-R. Fig. 2.4 shows the reasons for the choices. The average energy consumed by the different methods in one time slot is shown in Fig. 2.5. As the figures indicate, the proposed adaptive methods satisfied the previous three characteristics. When the flooding mode was selected, the adaptive-ex method consumed less energy than the adaptive-b method. This is due to the smaller TTL value chosen by the extended flooding method.

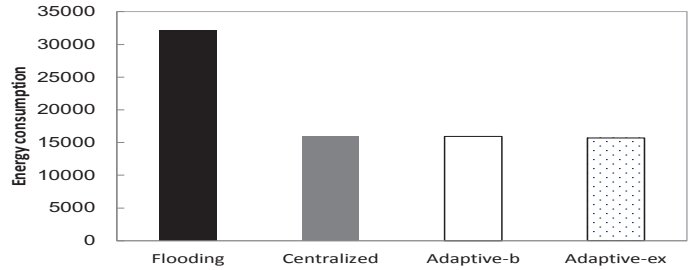
2.6.3 Performance in composite NRSs

The performance of the proposed methods was separately verified in different regions of the NRS, which was explained in Subsection 2.6.2. However, in reality, the NRS often exhibited various kinds of periodicities, e.g., in class and after class, on days and nights, and on weekdays and weekends. If the NRS had a period that included both F-R and C-R, the proposed methods automatically transformed between different modes to save energy. Two examples are given in this subsection to provide details on the energy consumed by the proposed methods in composite NRSs.

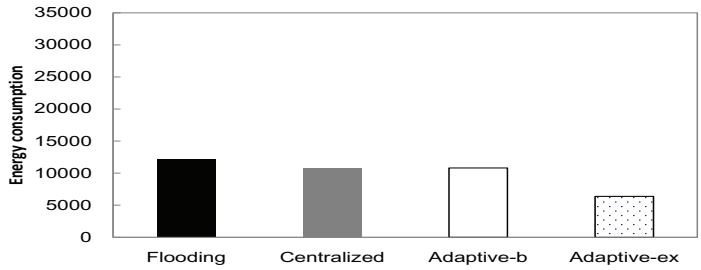
In example 1, the NRS transitions were caused due to the number of tasks generated by nodes. Every node had six units of resources in this example. In state S_1, 180 tasks were generated in a time slot. In state S_2, 360 tasks were generated in a time slot. As plotted in Fig. 2.8(c), S_1 belonged to F-R while S_2 belonged to C-R. The network stayed in S_1 for 10 time slots and then transformed to state S_2 through



(a) Energy consumed in F-R



(b) Energy consumed in C-R



(c) Energy consumed in CR-R

Fig. 2.5 Energy consumed in single region of NRS.
@2014 Elsevier

an intermediate state with 270 tasks generated in a time slot.[*5] Then, it stayed in S_2 for another five time slots before returning to S_1 in reverse. This process went on infinitely. The process had a period of 17 time slots. Fig. 2.6 shows the energy

[*5] 270 tasks was chosen for the intermediate state because it is the median of 180 and 360.

consumed by different methods in one period. The bars at left for both adaptive methods indicate the energy consumed by the proposed methods. The bars at right for these two methods indicate the energy consumed in an ideal situation. Nodes in the ideal situation were assumed to be able to predict future NRSs without any mistakes and make the right choices in advance. Of course, this is not realistic.

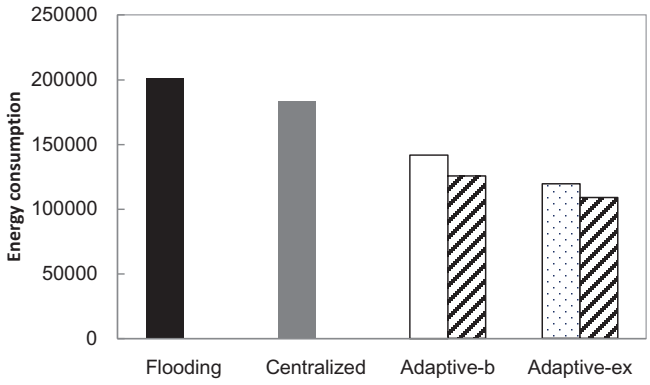


Fig. 2.6 Energy consumed in example 1.
@2014 Elsevier

As we can see in Fig. 2.6, both adaptive methods consumed less energy than the centralized and the basic flooding methods due to their adaptivity. The adaptive-b method consumed 77.06% of the energy of the centralized method and 70.36% of that of the flooding method. By benefiting from the extended flooding method, the adaptive-ex method performed better. It only consumed 64.98% of the energy of the centralized method and 59.33% of that of the flooding method. The energy consumed by both adaptive methods is close to the idealized consumption, i.e., adaptive-b (112.85%) and adaptive-ex (109.74%). The differences were due to prediction errors and the initial choice.

Two reasons caused the NRS transitions in example 2: (1) the number of tasks generated by nodes, and (2) different resource densities. There were two kinds of resources, A and B, in this example. Every node had five units of resource A. Only 60% of nodes had five units of resource B. In reality, resource A represented common resources like 3G. Resource B represented less popular resources like GPS or software applications. In state S_3, 100 tasks were generated for resource A while no tasks were

generated for resource B. In another state, S_4, 50 tasks were generated for resource B while no tasks were generated for resource A. The network stayed in S_3 for 10 time slots. Then, it changed to state S_4 for another five time slots. This process went on infinitely. Fig. 2.7 shows the energy consumed by the different methods in one period.

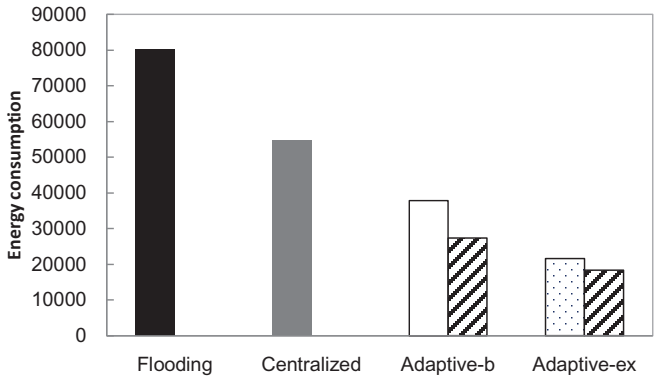
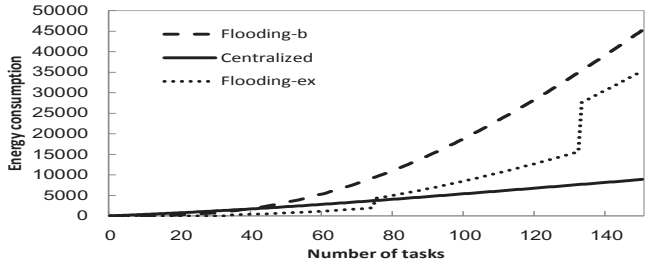


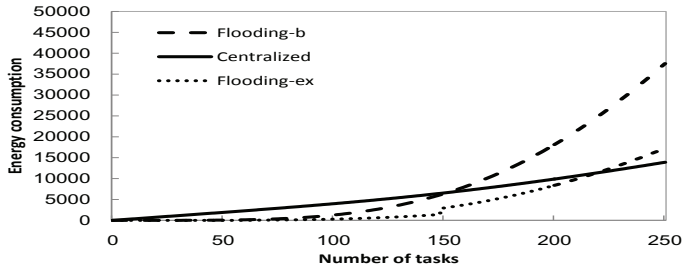
Fig. 2.7 Energy consumed in example 2.

Again, both the adaptive-b method (68.89% of the centralized method and 47.08% of the flooding method) and the adaptive-ex method (39.33% of the centralized method and 26.88% of the flooding method) consumed less energy than the centralized and basic flooding methods. The energy consumed by the adaptive-b method was 138.07% of that in the ideal situation. The energy consumed by the adaptive-ex method was 117.87% of that in the ideal situation. The differences were slightly larger than those in example 1 because there was no intermediate status in this example. Due to benefits from a smaller TTL value chosen in the flooding mode, the penalties for incorrect predictions were less expensive in the adaptive-ex method. Consequently, the energy consumed by the adaptive-ex method was closer to that of the ideal situation compared with the adaptive-b method.

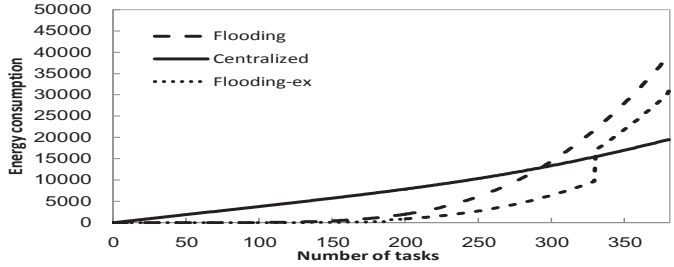
The results in this subsection indicate that the proposed methods could automatically choose an energy-efficient discovery mode according to different NRSs.



(a) 2 units of resources per node



(b) 4 units of resources per node



(c) 6 units of resources per node

Fig. 2.8 Relationship between number of resources and divisions in region.
@2014 Elsevier

2.6.4 Scalability

The basic parameters listed in Table 2.3 were used in the previous simulations. The scalabilities of the proposed methods were verified under different parameter settings

and are discussed in this subsection.

Relationship between number of resources and divisions in regions

The relationship between the number of resources each node had and the divisions in regions is first explained. If there were more resources in each node, with a fixed number of generated tasks, there was intuitively less chance for it to generate a resource request for available resources in other nodes. This meant little energy would be consumed in the flooding mode. However, the number of available resources in a node still changed even if the task was processed by the node itself. Energy was still consumed to maintain the resource directory accurate in the centralized mode. As the number of resources increased, the flooding mode was more preferable because there was no need to maintain a resource directory in it. This is proved in Fig. 2.8, where the range of F-R increases with the increasing number of resources in each node.

Node & resource densities

The performance of the proposed methods was verified under different node and resource densities instead of the default parameters in Table 2.3.

Fig. 2.9 plots the energy consumed by different methods under six different node densities. Because the ranges of energy consumed by different methods was too large, a 10 based log function for energy consumption was used as the y-axis of the figure. Every node in this simulation had five units of resources. A total of 250 tasks was generated by nodes in the area in each time slot. As the figure shows, the adaptive-b method always performed closest to the best choice for the centralized and basic flooding methods for different node densities. The adaptive-ex method consumed less energy than the others if the flooding mode was chosen because it benefitted from the extended flooding method. We can see that both adaptive methods chose the flooding mode when the number of nodes increased. This was due to the assumption that a fixed number of 250 tasks was generated in one time slot. The average number of tasks generated by each node decreased when the number of nodes increased under this assumption. Consequently, there was less chance for each node to ask the other nodes for resources. Therefore, the flooding mode was preferred.

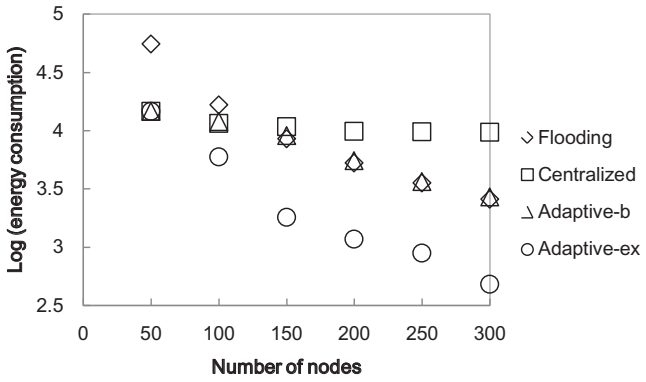


Fig. 2.9 Energy consumed under different node densities.

@2014 Elsevier

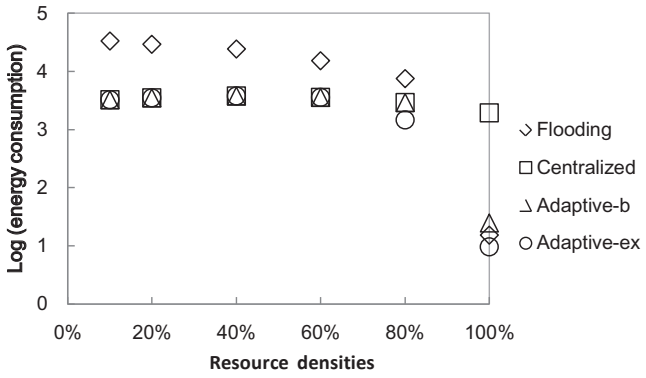


Fig. 2.10 Energy consumed under different resource densities.

@2014 Elsevier

The performance of the proposed methods with different resource densities, that represents the percentage of nodes that had the required resources, is plotted in Fig. 2.10. Again, a 10 based log function for energy consumption was used as the y-axis of the figure. A total of 500 units of resources was equally distributed in nodes that had the required resources in this simulation. 50 tasks were generated by nodes in the area in one time slot. As the figure shows, the adaptive-b method always performed closest to the best choice for the centralized and basic flooding methods for different resource densities. We can see that when the resource density was low, both adaptive-b and adaptive-ex methods chose the centralized mode. This is because many resource requests were generated by nodes without any resources. The centralized mode was

more energy-efficient than the flooding mode in this situation. The emergence of this situation was due to our assumption that every node generated tasks with an equal probability.

Parameters for proposed prediction algorithm

Finally, parameter settings for the proposed prediction algorithm are discussed. The scenario for example 1 described in Subsection 2.6.3 was used for the simulations that are explained in this subsection.

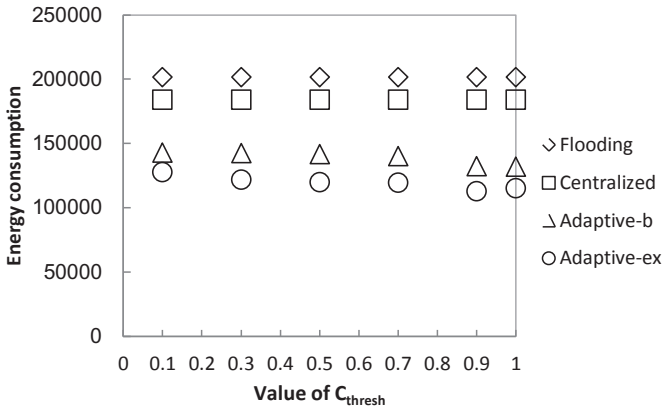


Fig. 2.11 Energy consumed with different values of C_{thresh}.
@2014 Elsevier

Fig. 2.11 plots the energy consumed by different methods with different values of C_{thresh}. Energy consumed by the centralized and basic flooding methods has been presented for the sake of convenience, although it was not influenced by the parameters presented in this part. Both adaptive-b and adaptive-ex methods performed best when the C_{thresh} value was near 1. This was due to the fact that the centralized mode consumed about $65\% - 85\%$ of the energy consumed by the flooding mode in the first time slot of C-R in the simulation scenario.[*6] As a result, if the value of C_{thresh} was larger than 0.85, the proposed methods transformed to the centralized mode quickly to adapt to the transition of NRS. Otherwise, the transformation was slightly delayed by using the average value of the previous N_{trend} (2 by default) time

[*6] The flooding mode consumed about $5\% - 10\%$ of the energy consumed by the centralized mode in the first time slot of F-R. Consequently, it was rarely affected by different values of C_{thresh}.

slots. Therefore, the value of C_{thresh} should be a clear signal that the NRS stably entered into the region of F-R or C-R. If the duration of NRS transitions is short, the value of N_{trend} should be kept small to adapt to transitions quickly. If the duration is long, the value of N_{trend} should become larger to filter out possible fluctuations during the process. The value of N_{trend} was assigned to 2 by default because the duration of transitions in the previous simulation scenarios were short, i.e., with one or no intermediate states.

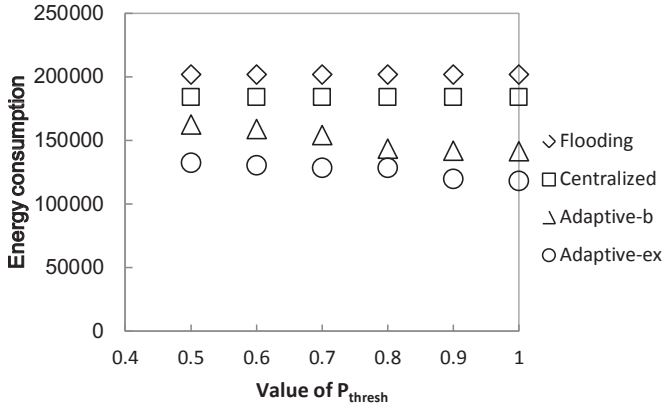


Fig. 2.12 Energy consumed with different values of P_{thresh}.
@2014 Elsevier

The energy consumed by different methods with different P_{thresh} values is plotted in Fig. 2.12. Because the priority of P_{thresh} is lower than that of C_{thresh} (0.5 by default) according to the algorithm, values larger than 0.5 were considered. The energy consumption decreased along with the increasing value of P_{thresh}, since a small value of P_{thresh} prevented transformations even if substantial amounts of energy were saved. Consequently, the value of P_{thresh} should be near 1 for this kind of scenarios in which the transitions of NRS are regular. A smaller value of P_{thresh} combined with the ratio of N_{trans} to N_{slot} are helpful in preventing transformations for some extreme scenarios, e.g., when the NRS is in CR-R, the preferred mode changes frequently and irregularly due to the stochastic uncertainty of nodes that generate tasks.

As we can see, there is redundancy in the proposed algorithm, viz., (1) both N_{trend} and C_{thresh} control the sensitivity of transformations and (2) both the ratio of N_{trans}

to N_{slot} and P_{thresh} prevent meaningless transformations. Redundancy is mainly retained for two reasons: (1) it keeps the proposed algorithm robust in extreme scenarios and (2) it reserves mechanisms that automatically optimize parameters for the proposed algorithm in future work.

Generally speaking, parameter settings for the algorithm are not universal and depend on different characteristics of networks. However, the effectiveness of the proposed method mainly resulted from transformations of discovery modes according to different NRSs. The values of parameters just control the degree of sensitivity and robustness for transformations, while they have limited impact on overall performance except for extremely impractical settings. This is the reason that we chose a common parameter setting rather than optimize parameters separately for different scenarios in this chapter.

According to all the simulation results in this section, it can be concluded that: (1) When the NRSs only included a single region, the energy consumed by the adaptive-b method was always closest to the best choice for the centralized and basic flooding methods because of its adaptivity. (2) When the NRSs included both F-R and C-R, the adaptive-b method consumed less energy than either the centralized or flooding methods because of its adaptivity. Although the degree of reduced consumption depended on different NRS parameters, the energy consumed by the adaptive-b method was close to the ideal situation. (3) When the proposed method was provided with an improved version of components like adaptive-ex, energy consumption was further reduced while maintaining the previous advantages. (4) The previous conclusions were not sensitive to different NRSs. As a result, the proposed adaptive solution to resource discovery was energy-efficient in different network environments due to its adaptivity.

2.7 Related work

There is a great deal of related work that exists due to the importance of resource discovery. This work can be roughly divided into two categories: directory-aided and directory-less. Resource directories were used to facilitate resource discovery in

directory-aided strategies [19–22, 25–27]. Two modes were defined for resource discovery in [19], i.e., service searching and service browsing. Service searching allowed a client to formulate a query containing the required attributes of the service. A client in the service browsing mode could send a generic query and obtain a list of all the services of a specific provider. However, it only supported one-hop discovery due to the limits of Bluetooth. A Chord based resource discovery method was introduced against the background of wireless mesh networks (WMNs) in [22]. It used location-awareness ID assignment and a cross-layer strategy to facilitate resource discovery. A centralized and homogeneous naming mechanism of global resource discovery for the Internet of Things (IoT) was presented in [25]. A context-aware service discovery framework based on the virtual personal space (VPS) was introduced in [26]. A personal operating middleware (POM) in the framework was responsible for providing personalized response information. A framework of semantic service discovery for ubiquitous computing was proposed in [27]. Although different issues in service discovery and reciprocal work with current ontology languages were discussed, no theoretical model or prototype system was mentioned by the authors. In [35], several clusters named as rings of resource providers were formed based on semantic proximity and physical proximity. A service access point (SAP) in each ring was responsible for resource registrations and resource requests. The semantic summaries of each ring were also exchanged by different SAPs to provide global resource discovery. The [20] was the only work that took into consideration the combination of different discovery modes. However, their method always chose the centralized mode when it was available and did not take into account adaptivity to different network environments.

Flooding based methods were used in the directory-less strategies. However, the energy consumed by flooding exponentially increased along with the increasing number of resource requests. Several improved strategies like probability based [23] and location based [29] resource discovery methods have been proposed to solve this problem. A node that received a resource request that it was not able to fulfill forwarded it with a probability that decreased with the number of hops the request had already travelled in [23]. However, this method decreased the coverage aspect of RIA without a formal analysis of tradeoff. Resource providers periodically sent resource advertise-

ments along cross-shaped trajectories in [29]. A resource seeker only sent a request along a path that intersected with any one of the trajectories. The intersecting node of the advertising and requesting trajectories answered the request. Because it only utilized relay nodes in four directions, the hit ratio of resource discovery was low in a sparse network. Crossing-layer strategies were adopted by [24] and [28]. The resource discovery protocols in these strategies were integrated with routing protocols. Although energy consumption was reduced, the compatibility of resource discovery methods was greatly limited. Different implementations were needed for different routing protocols. It should be mentioned that, improved centralized and flooding methods are easy to integrate into the proposed adaptive method when they become mature. Since they perform better than their basic versions, the energy efficiency of our adaptive method can also be improved. The simulation results from the adaptive-ex method proved this. [31] provided a comprehensive survey of published work in this area.

As a result, we found there was no existing work that was similar to our proposed method of adaptive resource discovery. However, the importance of adaptive resource discovery based on method or mode transformations was also emphasized in [31].

2.8 Conclusion

This chapter presented an energy-efficient method of adaptive resource discovery against the background of MCC. According to different network environments, it transforms between centralized and flooding modes to save energy. An extension of the flooding method was introduced as an optional choice. A prediction algorithm was provided to implement the proposed method. The effectiveness of the new approach is proved through extensive simulations.

Our work was only the first step toward utilizing the proposed adaptive strategy. As discussed in the previous sections, it could be integrated with other improved methods to replace their basic counterparts. It could also be used to optimize other metrics like response time and RIA. Mechanisms that automatically optimize parameters for the proposed algorithm are interesting. Finally, when node mobility is considered,

nodes may not be distributed uniformly throughout the area. An appropriate model or algorithm still needs to be found to estimate energy consumption in the flooding mode.

Chapter 3

Local Resource Sharing in Mobile Cloud Computing

3.1 Background

After the phase of resource discovery described in Chapter 2, a resource seeker (RS) is aware of surrounding mobile devices that have idle resources. Therefore, this chapter focuses on how to construct a local resource sharing platform based on these discovered candidates of resource providers (RPs). The resource platform that is comprised of surrounding surrogate devices is called a local resource cloud (LRC) in this chapter to differentiate it from the traditional cloud that mainly represents the centralized data center.

Error-prone transmission channels, diverse node capabilities, the mobility of nodes, and limited apriori knowledge of environments have a significant impact on the availability and reliability of devices running in mobile wireless networks. Therefore, compared with wired networks, we argue that four additional properties need to be considered when constructing LRCs in mobile wireless networks: (1) Multiple resource providers (RPs) are needed. Although resources in mobile devices have increased rapidly, they are still limited compared to traditional computing devices like PCs, servers, and clusters. As a result, resources from multiple RPs need to be gathered to facilitate task processing. (2) Physical network conditions should be integrated. Error-prone channels, locations and mobility of nodes significantly impact the quality of resource sharing. (3) The computational effectiveness of algorithms is important. Mobile wireless networks are highly dynamic due to the mobility of nodes. The LRCs

must be in "real-time" and established to utilize available resources. (4) Energy consumption should also be taken into account. Increasingly more "heavy applications", such as image processing for video games, natural language processing, and streaming media, are currently favored by mobile users. These applications are expensive in terms of energy consumption. Considering the trends in mobile devices and batteries, bottlenecks in energy consumption by mobile devices are unlikely to be solved in the near future.

However, previous work has only concentrated on verifying the effectiveness of the cooperative MCC to the best of our knowledge [13, 14]. Neither of them have referred to how LRCs are constructed, especially for the ones comprised of multiple RPs in dynamic mobile environments. Therefore, this chapter presents the design of an LRC that selects multiple RPs to construct local resource platforms in MCC. The three main contributions of this chapter are: (1) It presents the concept and design of an energy-efficient LRC in mobile networks that accelerates task completion through resource sharing among mobile devices in the local vicinity. (2) Theoretical models and a formal problem definition of an energy-efficient LRC are proposed. (3) A heuristic algorithm is presented that we prove to be efficient through extensive simulations.

The remainder of this chapter is organized as follows. Based on a literature review in Section 3.2, Section 3.3 defines the concept of an LRC. Our analysis of the LRC and formal problem definitions are presented in Section 3.4. The heuristic algorithm is introduced in Section 3.5. Section 3.6 presents the simulation results. Section 3.7 concludes the chapter.

3.2 Related work

3.2.1 Mobile cloud computing

Much research has recently focused on the cooperative architecture of MCC. Huerta-Canepa and Lee [13] and Fitzek et al. [14] demonstrated the feasibility of cooperative resource sharing by surrounding surrogate mobile devices through experiments. The former researchers [13] presented the motivation and preliminary design for a frame-

work to create virtual ad hoc cloud computing providers. They established prototype system based on mobile phones. Their preliminary results indicated the efficiency of task completion time through cooperation by mobile devices. The later researchers [14] considered an approach in which a centralized cellular network dynamically and collaboratively interacted with a local distributed network connected over short-range links that aimed at achieving better use of energy and spectra resources. However, neither of them referred to how local resource platforms were to be constructed, especially in dynamic mobile environments.

Because battery capacity can't cope with the development of mobile applications, many researchers have proposed different energy-efficient solutions in the background of MCC. Cuervo et al. [36] implemented an mechanism that dynamically offloaded applications from mobile phones to surrounding infrastructures to conserve energy. Chun et.al [37] adopted VM migration to offload part of their application workload to a server with resources. Fernando et al. [8] surveyed related work in MCC .

3.2.2 Conventional service selection algorithm

There has been much research on the selection of RPs in wired networks mainly in the field of Web services [28, 38–40]. Selecting RPs that are constrained by multiple requirements is usually an NP-hard problem [41, 42]. Many heuristic approaches have been proposed to strike a trade-off between the level of computational complexity and optimization. However, the network considered by [28, 38, 39] was a wired and fix network that is assumed to have constant routes and round trip time. Although Gu et al. [40] explicitly modelled link delay and availability in large wired networks, the optimized Dijkstra algorithm provided by them required a global view of the whole network. It is obvious that the assumption of constant routes or global view of the whole network hardly holds in mobile wireless networks.

Yang et al. [41] and Luo et al. [43] took into account the selection of RPs in mobile wireless environments to satisfy quality of service (QoS) requirements. The former researchers [41] modelled availability (stability) of links based on the mobility of end nodes and took it into consideration when making decisions. The later researchers

[43] proposed a heuristic method based on the Cross Entropy (CE) algorithm that preferred RPs moving slowly under the same conditions. Formal analysis of the impact of mobility was left for future work in [43]. Compared with our proposal in this chapter, they focused on selecting of a single RP of best quality while neglecting the possibility of utilizing resources in multiple RPs to improve the quality of resource sharing.

3.3 Definitions of the local resource cloud

3.3.1 System model

We should take into account three steps to achieve resource sharing by mobile devices: (1) Authentication of mobile devices. Not every user of mobile devices wants to share his resources with unrelated people for security and economical reasons. However, users may be motivated to share resources with acquaintances or people who share similar interests with them [8]. As a result, the devices that take part in resource sharing should be authenticated in advance. (2) Resource discovery. Resource discovery is aimed at finding potential candidates for RPs that own idle resources. Routes to access these candidates should also be discovered in this phase. (3) Selection of RPs. The RS selects a group of RPs from candidates and utilizes their resources to process its task. Since selection of RPs is independent of different authentication and resource discovery protocols, we focus on the selection phase in this chapter. Cho et al. [44] and Ververidis et al. [31] surveyd related work on trust management and resource discovery protocols in mobile networks. The terminologies and abbreviations that are used throughout this chapter have been summarized in Table 3.1 for convenience.

Here, we have assumed all mobile nodes have some kind of short-range communication capabilities, e.g., WLAN ad hoc, or Bluetooth. After the resource discovery phase, the RS is aware of all the surrounding mobile nodes that have idle resources that are required (candidates of RPs) and routes to access these nodes. Multi-hop routes in mobile networks without any infrastructure are also composed of mobile nodes. These nodes are called relay nodes (RNs) in this chapter. Then, the RS se-

Table. 3.1 Terminologies & abbreviations

LRC	Local resource cloud that is a resource platform composed of local mobile devices through short-range communications.
RS	A resource seeker that generates an LRC to accelerate processing of its task.
RP	A resource provider that provides resources to an RS in an LRC.
RN	A relay node that contributes to the LRC by relayng packets between the RS and RP.
Task	A job generated by mobile users that consumes resources to finish it, e.g., data downloading (communication bandwidth resources) and image processing (CPU's computational resources).

lects a group of RPs from all candidates to utilize their resources. As a result, the RS, selected RPs, and RNs along the routes from RS to RPs constitute a resource sharing platform through short-range wireless communications. As previously described, we called it an LRC since all the participating nodes are within the local area of RS.

Fig. 3.1(a) shows an example scenario that four candidates of RPs named a, b, c, and d were discovered after the resource discovery phase. Without loss of generality, candidates a, b, and d are assumed to be super candidates (S-candidates) that have plenty of idle resources and candidate c is assumed to be a common candidate (C-candidate) that has few idle resources. The routes between RS and a, b, d are relatively stable (S-Routes), while that between RS and c are unstable (U-Route).[*1] Intuitively, the RS prefers S-candidates that are connected by S-Routes for the sake of more available resources and less maintenance cost. Therefore, the RS prefers a, b, d over c in this scenario. Fig. 3.1(b) indicates the resulting LRC if the RS selected candidates a, b, and d as its RPs. Theoretical models that are applicable to more

[*1] It depends on the characteristics of mobile nodes that constitute the route, e.g., moving speed and wireless transmission range.

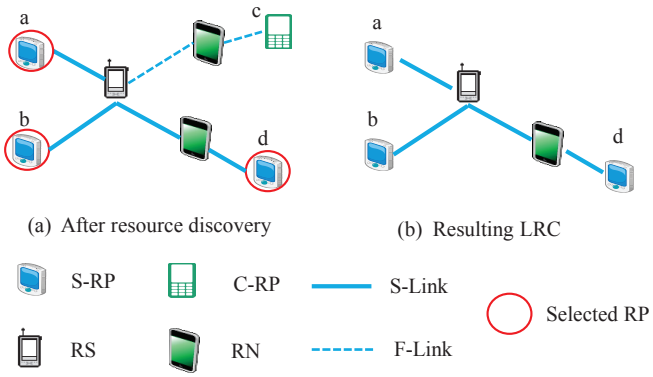


Fig. 3.1 System architecture.

@2014 IEICE

complex scenarios are formalized in Section 3.4.

Two characteristics of an LRC need to be noted: (1) On-demand creation. The LRC is only created when an RS wants to utilize more resources than it owns to accelerate processing of its task. (2) Limited lifetime. After a generated task has finished, the RS releases the generated LRC to make resources in RPs available to other nodes. It is possible for multiple LRCs to simultaneously coexist in the physical networks.

3.3.2 Problem statement

We mainly considered the sharing of computational and communication resources among mobile devices in this chapter.[*2] The concept of LRCs efficiently utilizes distributed resources in surrounding mobile devices to accelerate task completion. Compared with existing solutions that only take into consideration a single RP, the LRC proposed in this chapter is able to utilize more resources to achieve superior time efficiency of task processing since it comprises multiple RPs. However, extra energy is consumed by LRCs in the process of resource sharing compared with conventional computing strategies in which a user processes a generated task by itself only.

[*2] Information resources, as described by Nishio et al. [45], can be regarded as an alternative of communication and computational resources, e.g., if a Web page is cached by a node, the node does not need to consume bandwidth to receive it from Web servers.

One of such consumption is the energy consumed by maintaining the structure of an LRC. The RS has to keep connected to RPs in the LRC during the process of resource sharing to utilize resources in them, e.g., if an RS that lacks the capability of Internet access wants to utilize 3G or LTE resources in other nodes, it has to send (receive) data packages to (from) RPs that own these resources through the LRC. An RS also has to coordinate and synchronize computation processes among different RPs through the LRC to share computational resources [10]. However, because nodes are mobile, routes composed of RNs are dynamic. When a route in the LRC becomes unavailable, an alternative route has to be discovered by routing protocols to guarantee communications between RS and the lost RP. Energy is consumed in discovering an alternative route. Another kind of extra energy consumption in the LRC is the energy consumed by RNs to relay packets between RS and RPs. It increases along with the increasing number of RNs (hops) along the routes from RS to RPs. It is important to minimize these two kinds of extra energy consumption while retaining time efficiency benefitted from resource sharing when constructing an LRC.

We mainly aimed at reducing the energy consumed by maintaining the structure of an LRC to improve its energy efficiency in this chapter. To achieve this objective, the proposed solution prefers to select RPs connected by stable routes rather than select those connected by unstable routes. What is more, since a route with more RNs (more hops) tends to be more dynamic due to the mobility of nodes, the proposed solution still prefers to select RPs that are connected by routes with fewer RNs under the same conditions. This implicitly reduces the amount of energy consumed by relaying packets in the resulting LRC. Unlike wireless sensor networks (WSNs) [46, 47], the lifetime of an LRC is much shorter and the depleted battery can be replaced by users in good time, e.g., at home or office. RPs and RNs can also decide whether to join an LRC according to their own statuses. As a result, we leave sustainability and fairness of LRCs for future work, e.g., the influence of remaining battery power of RPs and RNs. Fig. 3.2 indicates the main research focus of this chapter as well as its relationship with previous work.

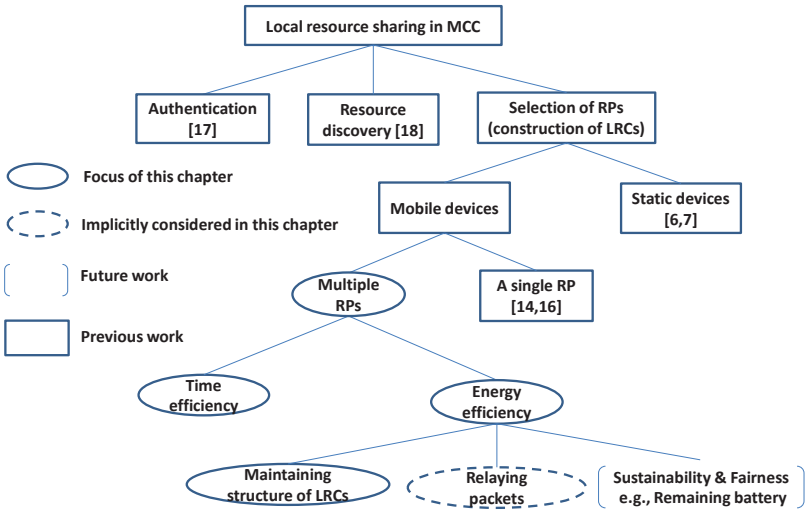


Fig. 3.2 Research focus of this chapter.

3.4 Proposed model of the energy-efficient LRC

Theoretical models of an energy-efficient LRC are established in this section. Two metrics used in the models are introduced in the first two subsections: first, we define availability, which represents the dynamics of an LRC and determines the energy efficiency of an LRC; then, we define task latency, which measures the time spent in processing a task and illustrates the time efficiency of an LRC. Finally, the problem of constructing an energy-efficient LRC while retaining its time efficiency is formalized based on these two metrics.

3.4.1 Availability of LRC

As described in Section 3.3, energy consumed by maintaining an LRC is greatly related to the dynamics of the network. The availability of a link in mobile networks is greatly related to the mobility of its end nodes. If one node i moves beyond the transmission range of its neighboring node j, the link between nodes i and j becomes unavailable. The availability of a link also expresses the availability of a route. Consequently, the availability of an LRC depends on the availabilities of links

and routes in it.

Availability of links

The definition for the availabilty of a link in this chapter is the same as it was in Yang et al. [41]. We briefly introduce it here for completeness and convenience. N_i and N_j are two end nodes of a link. The transmission range of a node is R.[*3] Every node moves randomly and its moving range is a circle with radius r. d is the distance between the two nodes. We assume that transmission range R is known and every node knows its location coordinates (e.g., through GPS or peer based techniques [48, 49]). As a result, distance d between two nodes can be calculated with the Euclidean distance formula:

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{3.1}$$

where (x_i, y_i) and (x_j, y_j) are the coordinates of nodes N_i and N_j.

Finally, radius r of the moving range for a node during the process of resource sharing is its moving speed v multiplied by the period t in maintaining an LRC. Speed v of a mobile node can be calculated based on its moving distance during a period from t_1 to t_2 [50]:

$$v = \sqrt{(x_{t_1} - x_{t_2})^2 + (y_{t_1} - y_{t_2})^2}/(t_2 - t_1) \ , \tag{3.2}$$

where (x_{t_1}, y_{t_1}) and (x_{t_2}, y_{t_2}) are coordinates of a node at time t_1 and t_2. The duration of period from t_1 to t_2 represents the frequency of calculating speed. Therefore,

$$r = v \times t. \tag{3.3}$$

As Fig. 3.3 indicates, the possibility of node N_i staying within the communication range of node N_j is equal to the area of node N_i's moving range inside the transmission range of node N_j divided by the overall area of node N_i's moving range. Consequently, the availability of a link composed of two nodes is:

$$A_l = \frac{Area_{inter}}{Area_{mov}} \ , \tag{3.4}$$

[*3] Although bilateral communications were assumed in this chapter, this model can be applied to scenarios in which nodes have different transmission ranges.

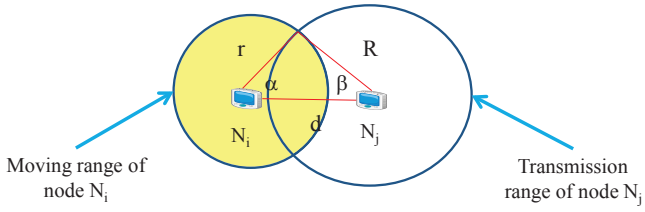


Fig. 3.3 Availability of links.

@2014 IEICE

where $Area_{inter}$ is the intersecting area of two circles and $Area_{mov}$ is the area of N_i's moving range. Equation (3.4) is calculated by:

$$\alpha = \arccos\left(\frac{r^2 + d^2 - R^2}{2 \times r \times d}\right) , \tag{3.5}$$

$$\beta = \arccos\left(\frac{R^2 + d^2 - r^2}{2 \times R \times d}\right) , \tag{3.6}$$

$$A_l = \frac{\beta R^2 + \alpha r^2 - (R^2 \sin\beta\cos\beta) + r^2 \sin\alpha\cos\alpha)}{\pi \times v^2 \times t^2} . \tag{3.7}$$

The proofs of Eqs. (3.5), (3.6), and (3.7) are in Yang et al. [41] and Namuduri et al. [51].

Availability of routes

Routes in an LRC can be divided into three subcategories of serial, parallel, and hybrid routes. Fig. 3.4 outlines different subcategories of routes.

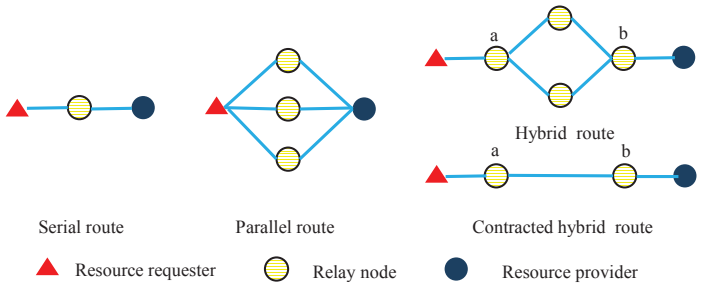


Fig. 3.4 Availability of routes.

@2014 IEICE

a) Serial routes

All the links in serial routes are connected in sequence. A serial route is available

if and only if all links in the route are available. Therefore, the availability of a serial route is:

$$A_r = \prod_{i=1}^{k} A_{l-i} \ , \tag{3.8}$$

where k is the number of links in the route and A_{l-i} is the availability of the i-th link in the route.

b) Parallel routes

There are multiple links between two end nodes in parallel routes. A parallel route is available if and only if one of the links in the routes is available. The availability of a parallel route is:

$$A_r = 1 - \prod_{i=1}^{k} (1 - A_{l-i}) \ , \tag{3.9}$$

where k is the number of links in the route and A_{l-i} is the availability of the i-th link in the route.

c) Hybrid routes

Both serial and parallel routes coexist in hybrid routes. To calculate the availability of a hybrid route, we first contract all the parallel routes to a serial link as Fig. 3.4 indicates. The availability of the contracted link is calculated with Eq. (3.9). After that, the hybrid route is reduced to a serial route. Its availability can be calculated with Eq. (3.8).

Availability of LRC

The LRC generated by an RS is composed of routes to all selected RPs. As a result, the availability of an LRC is :

$$A_o = \prod_{i=1}^{q} A_{r-i} \ . \tag{3.10}$$

where q is the number of routes in the LRC and A_{r-i} is the availability of the i-th route.

3.4.2 Task latency

As presented in the last subsection, period t in maintaining an LRC is a key variable in its availability. If t increases, the availability of an LRC decreases. Therefore, more energy would be consumed to maintain the LRC. According to the discussion in Section 3.3, an LRC is constructed when a task is generated by an RS and is released after the task is finished. As a result, we formally name the period t in maintaining an LRC as task latency since it illustrates the effectiveness of task processing from a user's perspective. Task latency includes three parts: the time to process the task, the transmission delay of an LRC, and the delay in maintaining routes.[*4]

Time for task processing

Intuitively, the time consumed to process a task decreases if the utilized resources increase. It could be simplified to the following equation:

$$T_{proc} = \frac{W}{R} \ , \tag{3.11}$$

for computational and communication resources, where W is the size of the task and R is the number of resources. For example, when using a link with throughput R bps to transmit a package with W bits, the processing time can be calculated with Eq. (3.11). The processing time can still be calculated with Eq. (3.11) for computational resources, when using CPUs that perform R operations per second to process a computing task with W operations.

Transmission delay of LRC

The transmission delay of an LRC also contributes to task latency. To utilize computational and communication resources in RPs, the RS distributes subtasks to different RPs through different routes at the beginning. After the subtasks have finished, RPs return results to the RS. Consequently, a period is spent on the round

[*4] In this chapter, we assumed an idealized situation: the RS partitioned and distributed subtasks to RPs seamlessly. Communications between nodes through the LRC were parallel to their computational process.

trip time (RTT) for each route. Obviously, the transmission delay of an LRC is the maximum RTT value for routes in it:

$$D_{LRC} = Max_{i=1}^{q} RTT_i \ , \tag{3.12}$$

where q is the number of routes in the LRC and RTT_i is the RTT value for the i-th route. The transmission delay of an LRC can be measured in the resource discovery phase.

Delay in maintaining routes

Task latency is also influenced by the time spent to re-discover a route from the RS to a lost RP when it became unaccessible. Since the length of delay depends on different routing protocols, this variable uses a history-based method for calculation, viz., the average value of past statistics:

$$D_{route} = \frac{\sum_{i=1}^{n} D_{route-i}}{n} \ , \tag{3.13}$$

where $D_{route-i}$ is the value of the i-th record of statistics and n is the number of previous records. As a result, task latency of a generated task is:

$$t = T_{proc} + D_{LRC} + D_{route} \ . \tag{3.14}$$

D_{LRC} and D_{route} (tens or hundreds of milliseconds) are generally much shorter than T_{proc} (tens of seconds or minutes). In that case, T_{proc} approximately presents task latency.

3.4.3 Problem definition of energy-efficient LRC

An RS generally wants to utilize resources in RPs to reduce the task latency of its generated task [45]. The RS should select more RPs in the LRC to obtain more resources to achieve this goal. As the task latency decreases, the lifetime of the LRC also decreases. Consequently, energy consumed to maintain the LRC during its lifetime may decrease. However, if more RPs are selected, more routes to access these RPs need to be maintained in the same time. Therefore, energy consumed to maintain the LRC may also increase.

We assume N potential candidates of RP are discovered after the resource discovery phase and the i-th candidate has r_i units of the required resource. We define x_i as an indicator to show whether the i-th candidate is selected by the RS to be part of the LRC:

$$x_i = \begin{cases} 1 & \text{the } i\text{-th candidate is selected} \\ \\ 0 & \text{the } i\text{-th candidate is not selected.} \end{cases} \tag{3.15}$$

As a result, every possible choice of an LRC is represented by an N dimentional vector X comprised of x_i. Then, the task latency of each choice is:

$$\begin{aligned} t &= T_{proc} + D_{LRC} + D_{route} \\ &= \frac{W}{\sum_{i=1}^{N} x_i \times r_i} + D_{LRC} + D_{route} \ . \end{aligned} \tag{3.16}$$

The availability of every possible choice of LRC can be calculated by substituting Eq. (3.16) into Eqs. (3.4), (3.8), (3.9), and (3.10). Therefore, the problem of selecting RPs to comprise an energy-efficient LRC is defined as: establish an LRC with maximum availability from all possible choices with a constraint that resulted LRC having enough resources to finish the generated task within a time threshold, T_{thresh}.

Objective: Maximize availability of the established LRC.

Constraints: $t \leq T_{thresh}$, $x_i = 0$ or 1 .

This is a non-linear 0-1 programming problem [52] that is also NP-hard. If the optimal number of RPs that need to be selected is known in advance, this problem is reduced to an NP-hard QoS-aware service composition problem [38][42]. A greedy heuristic algorithm is introduced in the next section to solve it.

3.5 Proposed heuristic algorithm

As presented in the previous section, if N potential candidates were discovered in the resource discovery phase, there are 2^N different combinations of RPs. It is not

feasible to compare all available choices when N is large. A greedy heuristic algorithm is proposed in this section to solve this problem. It is named as GELRC, short for Greedy algorithm for constructing Energy-efficient Local Resource Clouds.

Set C contains all discovered candidates of RP. The RS maintains set S that includes selected RPs in the LRC. It is initialized by an empty set. The availability of S is represented by A_{o-S}. First, to satisfy the constraint of task latency, the RS continues to select the i-th candidate that makes A_{o-S} the largest of possible candidates in C. If the constraint of task latency could not be satisfied even if C became empty, the task fails due to a lack of resources. After the constraint is satisfied, the RS continues to select candidates from C if and only if A_{o-S} does not decrease. Finally, the LRC denoted by S is used to process the generated task.

Taking the scenario outlined in Fig. 3.1 as an example. At the beginning of executing the GELRC algorithm, the set S was empty, $S = \emptyset$. The set C contained all discovered candidates of RPs, $C = \{a, b, c, d\}$. The RS firstly selected candidate a into S if the availability of S, $A_{o-\{a\}}$, was larger than $A_{o-\{b\}}$, $A_{o-\{c\}}$, and $A_{o-\{d\}}$. Then $S = \{a\}$ and $C = \{b, c, d\}$. Assuming that the task latency of resulting S was still larger than the value of threshold, the RS had to select RPs from remaining candidates to satisfy the constraint. Therefore, candidate b was selected if the availability of S, $A_{o-\{a,b\}}$, is larger than $A_{o-\{a,c\}}$ and $A_{o-\{a,d\}}$. After that $S = \{a, b\}$ and $C = \{c, d\}$. Assuming that the constraint of task latency was satisfied after selecting b into S, the RS continued to search for additional resources in other RPs to further reduce the task latency with a constraint that selecting that RP into S did not decrease the availability of the resulting LRC. It was possible because although an additional route from RS to the selected RP had to be maintained, additional resources in the selected RP also reduced the maintaining period of all the routes in the LRC. As a result, candidate d was selected into S if $A_{o-\{a,b,d\}}$ was larger than $A_{o-\{a,b,c\}}$ and $A_{o-\{a,b\}}$. Then, $S = \{a, b, d\}$ and $C = \{c\}$. At last, assuming that $A_{o-\{a,b,c,d\}}$ was less than $A_{o-\{a,b,d\}}$, the RS gave up candidate c and used an LRC comprised of $\{a, b, d\}$ to process its task.

The asymptotic computational complexity of the proposed GELRC algorithm is $O(K \times N^2)$, where K is the maximum length of routes in the hop count. An RS

GELRC algorithm (C)

$S = \emptyset$;
C = all candidates ;
$A_{o-S} = 0$;

while (the constraint of task latency is not satisfied)
  if($C == \emptyset$)
    task fails due to lack of resources ;
    return ;
  select i-th candidate that maximize A_{o-S} from C;
  add i-th candidate to S and delete it from C ;

$S^{'} = S$; //$S^{'}$ is a temporary variable.

while ($A_{o-S} \leq A_{o-S'}$)
  $S = S^{'}$;
  select i-th candidate that maximize $A_{o-S'}$ from C;
  add i-th candidate to $S^{'}$ and delete it from C ;

//make choice
S denotes resulting LRC ;

generally searches for local resources with a constant hop limit, K (TTL value). In that case, the computational complexity of the method is $O(N^2)$. The pseudo-codes describe the algorithm.

3.6 Simulation

Simulation results executed on Qualnet 5.2 [53] are presented in this section. Nodes were distributed randomly in a rectangular area at the beginning. Every node generated a task with an equal probability. Since the proposed algorithm for constructing an LRC was not sensitive to bandwidth, IEEE 802.11b and 2 Mbps of link bandwidth

were adopted to verify its performance in a constrained environment. Because only comparisons of different methods were concerned, we assumed that transmitting a byte consumed one unit of energy and receiving a byte consumed 0.6 unit of energy for energy consumed to maintain an LRC. The transmission and energy settings were the same as those in Ahmed et al. [54].

Mobile nodes in the area were divided into three categories according to their amounts of resources:

(1) Super nodes (SN): Nodes that had 70 units of resources.
(2) Common nodes (CN): Nodes that had 30 units of resources.
(3) Relay nodes (RN): Nodes that did not have any resource.

Four different methods of constructing an LRC were compared in the simulations. The first two were the proposed methods based on the energy-efficient model described in Section 3.4.

(1) GELRC: The RS solved the energy-efficient model with the proposed heuristic algorithm.
(2) Optimal: The RS solved the energy-efficient model with a brute force strategy. The upper bound of the proposed model was obtained with this method.
(3) Random: The RS selected RPs randomly until the constraint for task latency was satisfied.
(4) LOSSA: The RS selected a single RP with best availability from all candidates that had enough resources to satisfy the constraint for task latency. This was proposed by Yang et al. [41].*5

Of the four methods, the random method was unaware of the availability of the resulting LRC. The LOSSA method only concentrated on a single SN node with best

*5 It was a weighted average value in Yang et al. [41]. We eliminated the "price of service" and "reliability of service" by assigning zero to their weights because these two properties were beyond the scope of this chapter.

Table. 3.2 Parameters for simulations

Simulation area	500 × 500 m
Size of task	600
Number of SNs	5
Number of CNs	20
Number of RNs	25
Number of resources in SNs	70
Number of resources in CNs	30
Latency threshold	10 s
Routing protocol	DSR
MAC protocol	IEEE 802.11b
Link bandwidth	2 Mbps
Transmission range of nodes	150 m
Energy of transmission	1 unit / byte
Energy of reception	0.6 unit / byte
Mobility model	Random way point
Mobility speed	1, 5, 10, 15, 20 (m/s)
Pause time	0 s
Interval of task generation	5 s
Simulation period	1000 s

availability since only SN nodes were able to independently finish a task within the threshold of task latency. Compared with the previous two methods, both optimal and GELRC methods were able to utilize resources in multiple SN and CN nodes to increase the availability of the resulting LRC as well as reducing task latencies. The parameters used in the simulations are summarized in Table 3.2.

3.6.1 Computation cost

The first series of simulations were aimed at comparing the four methods previously described with respect to the computational overhead involved in their construction phase of an LRC. We measured the computation cost of selecting RPs to construct

Table. 3.3 Computation cost

GELRC	Optimal	Random	LOSSA
0.002 s	285.318 s	$\leq$ .001 s	$\leq$ 0.001 s

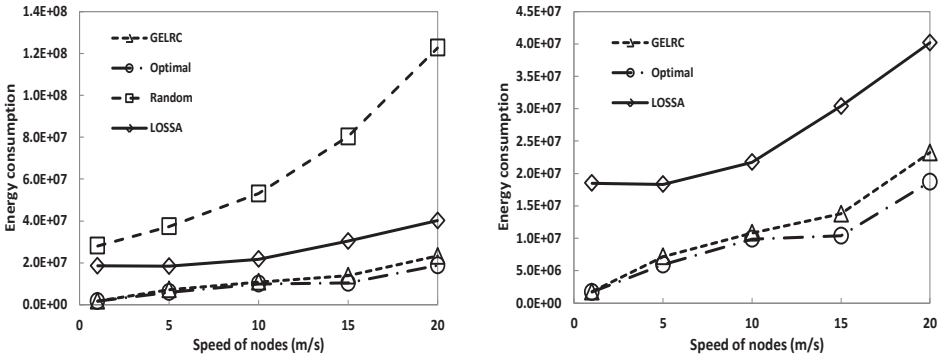
an LRC from 25 candidates with different approaches. We executed each method 10 times for each test case on an PC with an Intel Core i7-2640M CPU @ 2.80 GHz and 8.00 GB of RAM on Windows 7. The average computation costs are summarized in Table 3.3.

As listed in Table 3.3, although the optimal method achieved the upper bound for the energy-efficient model, it cost hundreds of seconds to solve it. However, nodes in realistic scenarios continue to move during this period of time. Therefore, the resulting optimal point became meaningless in the real-time distribution of nodes. The importance of the optimal method in this chapter was that it specified a benchmark to measure the gap between the GELRC method and the upper bound of the proposed model.

3.6.2 Energy consumption & task latency

The energy consumed to maintain an LRC under different speeds of nodes is plotted in Fig. 3.5. As Fig. 3.5(a) indicates, the random method consumed much more energy than the other methods. This is because the selection of RPs with the random method did not take into consideration the availablity of the resulting LRC. We eliminated data for the random method in Fig. 3.5(b) to make the comparison in other three methods clearer. As it indicates, the GELRC method consumed less energy than the LOSSA method, i.e., 42% - 88% less according to different speeds. This is because the GELRC method utilized available resources in multiple RPs to reduce the task latency of an LRC. The differences between the GELRC and optimal methods were small, i.e., 7% - 36% according to different speeds.

This is also proved in Fig. 3.6 that plots the task latencies of the four methods. The task latencies of both the GELRC and optimal methods were much less than



(a) (b)

Fig. 3.5 Energy consumed by different methods.

@2014 IEICE

those of the other two methods especially when the speeds of nodes were low. The availability of an LRC remained high even if more RPs were selected when the speeds of nodes were low. Therefore, utilizing available resources in these RPs accelerated task processing while preventing from extra energy being consumed to maintain the structure of LRCs. As a result, unlike the other two methods, both methods based on the energy-efficient model were able to construct a reasonable scale LRC that was adaptive to different levels of mobility.

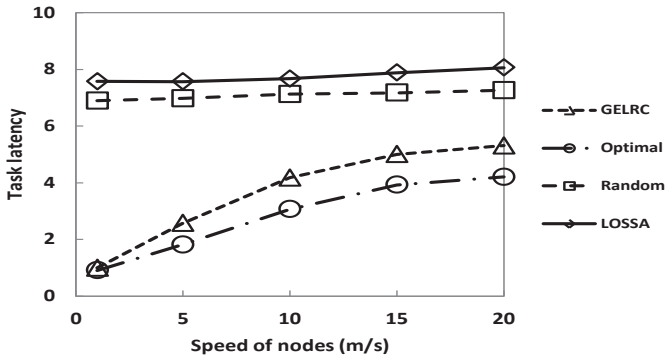


Fig. 3.6 Task latency of different methods.

@2014 IEICE

3.6.3 Failure of RPs

We assumed that all RPs in an LRC remained "on-line" from the begining to the end of task processing in previous simulations. However, in reality, mobile RPs may suddenly go "off-line" due to reasons like system crashes or moving out of communication range. The RS is usually not capable of predicting this kind of "sudden failure of RPs" or is not notified by RPs in advance. In this kind of scenario, the constraint for task latency can't be guaranteed. As a result, we defined a metric called "success ratio" to measure the capabilities of fault tolerance by different methods. In this chapter, the "success ratio" is defined as the percentage of tasks that are finished within the threshold of task latency without re-establishing a new LRC. The lifetime of a node is defined as the duration over which it is capable of sharing resources with other nodes in the following descriptions.

We assumed the lifetimes of both SN and CN nodes conformed to an exponential distribution. RNs were not taken into account because they have no resources. The failure of RN nodes was the same as the failure of simulated routes discussed in the previous subsection. Nodes were disabled after their lifetimes expired. If a disabled node acted as an RP in an LRC, the RS of the LRC detected the node had failed through a keep-alive timer. The keep-alive timer was set to 2 s in the simulation.[*6] After the disabled node was detected, we assumed the RS could ideally distribute remaining subtasks for the disabled node to all remaining RPs including itself. A new node with the same number of resources was enabled at a random location to replace the disabled node to keep node densities constant. The resulting success ratios of different methods are plotted in Fig. 3.7 when the speed of nodes is 10m/s. According to the results, both the GELRC and optimal methods outperformed the other two methods in dynamic environments. This mainly benefitted from the fact that both GELRC and optimal methods reduced task latencies as shown in Fig. 3.6. As a result, (1) the probability of the failure of RPs within a shorter period of task

[*6] Two seconds was enough for this simulation because the RTTs of most routes were within 200 ms. This value could be adjusted according to different network statuses.

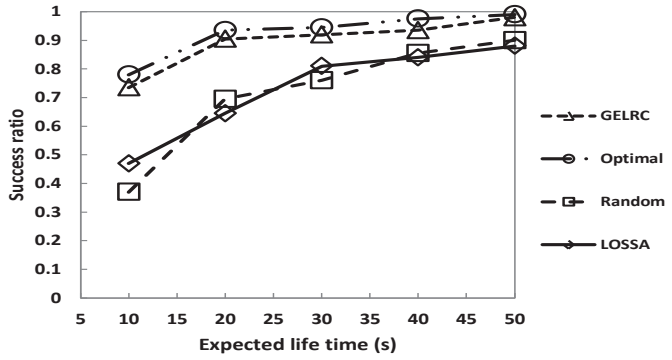


Fig. 3.7 Success ratios for different methods.

processing time decreased, and (2) a larger room was reserved for the failure of RPs, e.g., even if task latencies were delayed by 4 seconds due to the failure of RPs, both GELRC and optimal methods were able to finish it within the threshold (10 seconds) as shown in Fig. 3.6. LOSSA and random methods failed to do this.

Based on previous simulation results, it can be concluded that: (1) The proposed model and methods are able to construct an LRC that is efficient in terms of both energy consumption and task completion time in mobile wireless networks. (2) When the failure of nodes was taken into consideration, the proposed methods were capable of higher success ratios than the conventional methods. (3) The proposed GELRC method performed close to the upper bound of the energy-efficient model while dramatically decreasing the compuation cost.

3.7 Conclusion

The design of an energy-efficient local resource cloud (LRC) in cooperative MCC was proposed in this chapter. It is constructed of an on-demand local resource platform to enable resource sharing among mobile devices. Simulation results validate that the proposed model and methods are efficient in terms of both energy consumption and task completion time. For future work, explicit considerations of energy consumed by relaying packets, sustainability and fairness of LRCs are interesting research areas in local resource sharing.

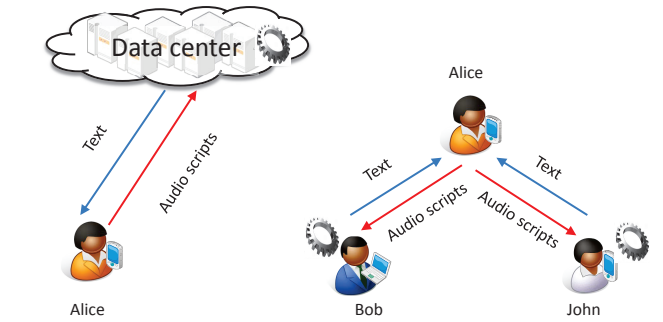
Chapter 4

Opportunistic Resource Sharing in Mobile Cloud Computing

4.1 Background

Chapter 3 mainly focused on local resource sharing in consistently connected networks, i.e., there exists at least one available route between RSs and RPs during the process of resource sharing. However, this assumption may not be valid in real short-range communication networks due to the mobility of mobile devices, e.g., in sparse networks, there may be no enough relay nodes to constitute a route between RSs and RPs all the time. In this kind of scenarios, mobile devices, from time to time, make contact with other devices that could provide resources to them. However, such contacts are intermittent and of indeterminate duration. To distinguish from consistently connected networks, researchers call this kind of networks as "opportunistic networks" [55] or "delay tolerant networks" [56]. This chapter discusses the concept and design of an resource sharing mechanism that utilize resources in mobile devices through opportunistic contacts between them.

A typical example application of opportunistic resource sharing could be the speech-to-text application as discussed in [18, 57]. Assume that a student named Alice wants to translate audio scripts into text for language study, while her smart phone lacks required resources, such as a speech-to-text application or powerful CPUs. Without opportunistic resource sharing, she has to rely on centralized solutions in which the remote data center translates the audio scripts for her, as shown in Fig. 4.1(a). Opportunistic resource sharing solves this problem from a different approach, as shown



(a) Client-server based MCC (b) Opportunistic resource sharing in MCC

Fig. 4.1 An example scenario of opportunistic resource sharing.

in Fig. 4.1(b). If Alice knows that she will encounter her friends Bob and John in a short time[*1] and both of their devices own but seldom use the required resources, she could allocate the audio scripts to them to help her finish the translation task. Therefore, the overhead in terms of network traffic and resource consumption of the data center will be drastically reduced since only local communications like WLAN ad hoc or Bluetooth, and local resources in end devices are utilized. This strategy also enables other interesting applications such as intelligent transportation systems, crisis management, and pervasive healthcare as described in [58, 59].

However, the diverse capabilities and connectivities of mobile devices make the problem of opportunistic resource sharing very complicated [58, 59]. Moreover, previous work in this area either analyzed simplified models that are not applicable to real situations or used heuristic algorithms without formal analysis. As a result, this chapter proposes a novel mechanism to coordinate opportunistic resource sharing in the MCC. The three main contributions of this chapter are: (1) It presents the concept and design of an opportunistic resource sharing mechanism that utilizes resources in mobile devices through opportunistic contacts between them. (2) Theoretical models of opportunistic resource sharing that are based on multiplex parameters, such as inter-contact time, contact duration, contact bandwidth and workload of RPs, have been presented. (3) Algorithms are designed to implement the theoretical models.

[*1] E.g., they have the same next class or go to attend the same meeting.

The efficiency of the proposed algorithms was validated through formal proofs and extensive simulation.

Following the literature review in Section 4.2, Section 4.3 explains the proposed system model of opportunistic resource sharing. Analyses of theoretical models and formal problem definitions are presented in Section 4.4. The proposed algorithms are described in Section 4.5. Section 4.6 discusses the simulation results. Section 4.7 presents mathematical proofs of the three theorems listed in Section 4.5. Conclusions are drawn and future work is discussed in the last section.

4.2 Related work

Much research has recently focused on resource sharing in opportunistic networks. Pasarella et al. [15] analyzed resource sharing by modeling opportunistic contacts between resource seekers and providers. Using their developed model, they investigated the optimal number of task replicas that should be offloaded onto encountered resource providers to minimize the task completion time. However, they considered an ideal homogeneous network, i.e., one in which every resource provider has the same amount of resources and the inter-contact times between nodes are identically distributed. Therefore, their model is not suitable for real situations in which the capabilities and movement patterns of mobile devices diversifies. Sadiq et al. [16] and Tamhane et al. [17] developed architectures of service composition in opportunistic networks. However, they only took into consideration the inter-contact time, thus neglecting other factors that greatly affect the quality of resource sharing like contact duration and contact bandwidth. Both work focused on selecting the best-qualified resource provider while neglecting the possibility of utilizing resources in multiple resource providers to improve the quality of resource sharing. Shi et al. [18, 57] described a task scheduling scheme called Serendipity that enables mobile resource seekers to offload tasks to other contacted mobile devices opportunistically. Different heuristic algorithms were used depending on whether a common control channel (CCC) was existed to forecast future contacts. Their preliminary results indicated the effectiveness of opportunistic resource sharing on two example applications, i.e., face

recognition and speech-to-text. In contrast with the proposal in our chapter, their scheme did not use historical information to predict future contact opportunities when an CCC was not available and left it for future work. Moreover, they did not present formal analysis of their heuristic algorithms. Conti et al. [58, 59] described research background and motivation for studying in this area.

4.3 System architecture

4.3.1 Opportunistic contact table (OCT)

In this chapter, all mobile nodes are assumed to have some kind of short-range communication capability such as WLAN ad hoc or Bluetooth and emit a beacon signal periodically to advertise their presence. In practice, when another node senses the beacon, these two nodes establish a contact relationship by exchanging IDs and other relevant information. Bidirectional contacts are assumed in this chapter, viz., if a contact exists between nodes N_a and N_b, it also exists between N_b and N_a. Also, each node is assumed to maintain a timer, which is used for recording contact information.

The concept of opportunistic contact table (OCT) is introduced in this section to capture characteristics of contact opportunities between different pairs of mobile nodes. Several terms are used to better clarify the study of OCT.

Inter-contact time (IT): the time interval between two contacts of the same pair of mobile nodes. The timer is used to record the time elapsed since the two nodes last encountered each other.

Contact duration (CD): the time during which a pair of mobile nodes are close enough to communicate with each other. The timer is used to record the time elapsed from the beginning to the end of their current contact.

Contact bandwidth (CB): the average bandwidth of short-range wireless communications between a pair of mobile nodes during their contact, e.g., the bandwidth with WLAN ad hoc differs from that with Bluetooth. In practice, any method like those proposed by Renesse et al. [60] and Sarr et al. [61] may be used to measure it.

Table. 4.1 Opportunistic contact table for node N_a

Node	IT	CD	CB	AR
N_a	0 s	$+\infty$	$+\infty$	500 units
N_b	1000 s	100 s	500 Kbps	1000 units
N_c	5000 s	1000 s	11 Mbps	5000 units
N_d	300 s	100 s	6 Mbps	100 units

Available resources (AR): the amount of idle resources in a mobile node that can be shared with other nodes, e.g., unused CPU cycles and cellular bandwidth. It depends on the node's hardware/software configuration as well as on its current workload. This information may be exchanged along with their IDs, when two nodes establish their contact relationship.

Every mobile node in this system model maintains an opportunistic contact table (OCT) for recording these four variables for other nodes that it has encountered. Consider a couple of mobile nodes N_a and N_b. When N_a and N_b make contact, they exchange information of their available resources and probe their contact bandwidth. Both nodes also calculate inter-contact time and contact duration of the current contact by using their timers. They then update the related entries in their OCT using a moving window average strategy:

$$Q_{t+1} = \alpha \times Q_t + (1 - \alpha) \times Q_c \ , \tag{4.1}$$

where the variable Q represents the four variables (CD, CB, IT, and AR) and α is a constant parameter between 0 and 1. Q_t is the historical value in the OCT, and Q_c is the value for the current contact. As a result, the updated value after this contact, Q_{t+1}, is a moving window average of the historical and current values. An example of OCT maintained by N_a is shown in Table 4.1 in which it has had contact with nodes N_b, N_c, and N_d. Since N_a can utilize its own resources without any communication cost, the values of its CB and CD are set to $+\infty$.

It can easily be proved that, as time passes, the moving window average strategy ensures that each entry in the OCT converges to the average value of the corresponding

variables and this proof is independent of different α or initial values of the table entries [62]. It is actually a random process that fluctuates around the average values. There is a trade-off in the value of α. A larger value of α reduces errors, but increase the time it take to reach the steady state and vice vesa. In general, α should be set in accordance with the application requirements. Moreover, it may be difficult for a mobile node to manage all contacted nodes in its OCT because of their huge amount in reality. In that case, outdated entries may be removed to maintain the size of the OCT at a reasonable level, e.g., remove the least recently used (LRU) entry. Finally, since a mobile node makes contact with different groups of mobile nodes at different time and in different environments, different OCTs may be kept for different scenarios, e.g., an OCT for work time and an OCT for home time.

4.3.2 Overview of opportunistic resource sharing

The mechanism of opportunistic resource sharing presented in this chapter focuses on a resource seeker wishing to accelerate task completion by using available resources in other opportunistically contacted devices. The explanation of the mechanism uses four terms to improve clarity.

Task (TS): a job generated by a resource seeker that consumes resources to finish it. A task is described by $TS = (TS_i, TS_e, TS_o)$, where the size of the input parameters TS_i, the size of the execution body TS_e, and the size of the output results TS_o are three key variables that determine the task completion time. Specifically, we define a task with a zero value for these three variables as a dummy task, TS_{dummy}.

Resource seeker (RS): a mobile node that seeks available resources in other nodes to accelerate the processing of its tasks.

Resource provider (RP): a mobile node that provides resources to an RS.

Task latency (TL): the length of time from when a task is generated to when it is finished. It representes the effectiveness of task processing from users' perspective.

Different tasks consume different kinds of resources, e.g., image processing consumes CPU resources (FLOPS) and Internet accessing consumes cellular bandwidth resources (Mbps). Since the proposed mechanism is not constrained by specific types

of tasks or resources, these details can be encapsulated by abstracting the size of tasks and units of resources in the following descriptions.

Task latency is comprised of three parts: uploading input parameters, processing task and downloading output results. When an RS generates a task, it first partitions the task into subtasks and uploads the parameters for the subtasks to RPs through opportunistic communications. Once an RP receives the required input parameters for its assigned subtask, it starts processing the subtask using its available resources. After the RP finishes its subtask, the RS downloads the results from it when they encounter again. Finally, the RS combines all the results received from the RPs to finish its task. Since both uploading the parameters and downloading the results may take multiple contacts to complete,[*2] an upload or download progress that was not completed is resumed from where it was stopped at the next contact. An example flow of opportunistic resource sharing is shown in Fig. 4.2.

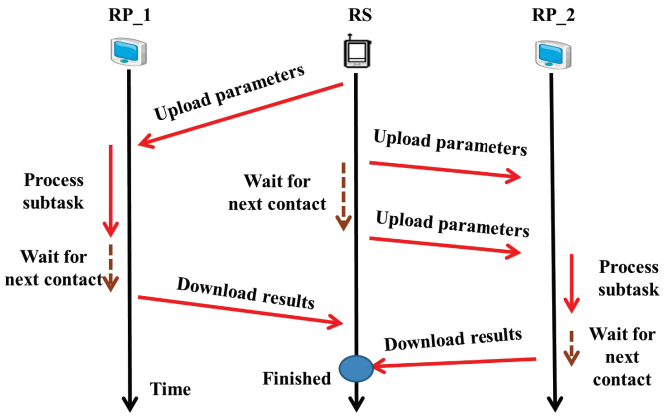


Fig. 4.2 Example flow of opportunistic resource sharing.

4.4 Proposed models for opportunistic resource sharing

In this section, theoretical models of calculating task latency are first analyzed. The problem of minimizing task latency through opportunistic resource sharing is

[*2] It depends on the CD and CB between the mobile nodes.

then formalized. Without loss of generality, a situation in which an RS (N_i) wants to offload a task (ts) to an RP (N_j) is assumed. Since there is no need to process a dummy task, the task latency of TS_{dummy} is always zero. Therefore, only non-dummy tasks are considered in this section.

4.4.1 Time consumed by uploading input parameters

As described above, due to the limited bandwidth and contact duration between mobile nodes in opportunistic networks, uploading input parameters may take one or more contacts to complete. Therefore, the time consumed by uploading input parameters is composed of two parts: (1) the time consumed by transmitting the input parameters, $T_{transmit}$, and (2) the time consumed by waiting for contact opportunities, T_{wait}.

$T_{transmit}$ depends on the size of the input parameters as well as the average contact bandwidth between nodes N_i and N_j.

$$T_{transmit} = \frac{ts_i}{CB_j}, \tag{4.2}$$

where ts_i is the size of the input parameters of task ts and CB_j is the contact bandwidth between nodes N_i and N_j which is obtained from the OCT maintained in N_i.[*3]

The estimation of T_{wait} is more complicated. First, the size of the input parameters that can be uploaded in one contact is given by

$$T_{up-one} = CB_j \times CD_j \ , \tag{4.3}$$

where CD_j is the contact duration between nodes N_i and N_j. The value of T_{wait} depends on whether N_i is in contact with N_j when task ts is generated.

(1) N_i is not in contact with N_j

In this case, N_i has to wait for the next contact with N_j to begin the uploading pro-

[*3] CD_j, IT_j, and AR_j described in this section are also obtained from the OCT maintained in N_i.

cess. Under the assumption that a task is generated uniformly in time, the expected length of the waiting period is one half of their inter-contact time.

$$T_{first-wait} = \frac{1}{2} \times IT_j \ , \tag{4.4}$$

where IT_j is the inter-contact time between nodes N_i and N_j. After the first contact, if the process of uploading is not finished, the expected size of the remaining input parameters is given by

$$ts_{i-remaining} = ts_i - T_{up-one}. \tag{4.5}$$

Since N_i is expected to be contacting with N_j with an interval of IT_j, and T_{up-one} input parameters will be uploaded during each contact, the remainning waiting time for uploading the input parameters is given by

$$T_{remaining} = \lceil \frac{ts_{i-remaining}}{T_{up-one}} \rceil \times IT_j \ , \tag{4.6}$$

where $\lceil \ \rceil$ is a ceiling function. Therefore, the total expected waiting time is given by

$$\begin{aligned} T_{wait} &= T_{first-wait} + T_{remaining} \\ &= \frac{1}{2} \times IT_j + \lceil \frac{ts_i}{T_{up-one}} - 1 \rceil \times IT_j \ . \end{aligned} \tag{4.7}$$

(2) N_i is in contact with N_j

In this case, N_i is able to upload the input parameters to N_j as soon as the task is generated. However, under the assumption of uniform distribution of task generation, the expected remaining contact time after the generation of tasks is one-half of the expected contact duration between nodes N_i and N_j. Consequently, the size of the input parameters that can be uploaded during the first contact is given by

$$T_{up-first} = CB_j \times \frac{1}{2} \times CD_j = \frac{1}{2} \times T_{up-one} \ . \tag{4.8}$$

After the first contact, if the process of uploading has not been finished, the expected size of remaining input parameters is given by

$$ts_{i-remaining} = ts_i - T_{up-first} \ . \tag{4.9}$$

Since N_i is expected to be contacting with N_j with an interval of IT_j, and T_{up-one} input parameters will be uploaded during each contact, the waiting time for uploading input parameters in this case is given by

$$T_{wait} = \lceil \frac{ts_{i-remaining}}{T_{up-one}} \rceil \times IT_j = \lceil \frac{ts_i}{T_{up-one}} - \frac{1}{2} \rceil \times IT_j \ . \tag{4.10}$$

Finally, the total time consumed by uploading the input parameters is the sum of $T_{transmit}$ and T_{wait},

$$T_{upload} = T_{transmit} + T_{wait} \ . \tag{4.11}$$

4.4.2 Time consumed by processing tasks

Intuitively, the time consumed by processing a task decreases if the utilized resources increase. The model proposed by Nishio et al. [45] is used to calculate it here

$$T_{proc} = \frac{ts_e}{AR_j} \ , \tag{4.12}$$

where ts_e is the execution body size of ts and AR_j is the estimated available resources in node N_j according to the OCT in N_i, e.g., if using a CPU that perform AR_j operations per second to process a task with ts_e operations, the processing time can be calculated using Eq. (4.12).

4.4.3 Time consumed by downloading output results

Since the process of downloading the output results is the reverse of uploading input parameters analyzed in Section 4.4.1, we can substitute ts_i in Eqs. (4.2), (4.7), and (4.10) with the size of the output results ts_o to get the time consumed by downloading results, $T_{download}$. Eq. (4.7) or (4.10) is used depending on whether N_i is in contact with N_j when the task ts is finished. As a result, the total task latency of ts is

$$TL\ (ts) = T_{upload}\ (ts_i) + T_{proc}\ (ts_e) + T_{download}\ (ts_o) \ . \tag{4.13}$$

4.4.4 Problem definitions

An RS generally wants to utilize resources in RPs to reduce the latencies of its tasks. Therefore, it has to partition its task into subtasks and assign these subtasks to appropriate RPs. Here, an opportunistic network in which an RS may make contact with N RPs (including the RS itself) is assumed, and x_k is defined as the percentage of the original task assigned to the $k-th$ RP, i.e., the subtask.[*4] The subtask is dummy if x_k is zero. As a result, the problem of minimizing task latency for a generated task ts is formalized as

P.1:

$$\text{Min. } max_{k=1}^{N} TL_k\ (x_k \times ts)\ ,$$

$$\text{S.t.} \quad \sum_{k=1}^{N} x_k = 1\ ; \quad x_k \geq 0\ \ (k = 1, \ldots, N)\ .$$

where $TL_k\ (x_k \times ts)$ is the latency of the subtask assigned to the $k-th$ RP. The objective function minimizes the maximum task latency for N RPs since the task is finished only when the RS has received the results from all RPs. The constraint conditions ensure a complete partition of tasks.

Solution vector X is defined as an N dimensional vector $(x_1, x_2, x_3, ..., x_N)$ that satisfies the constraint conditions of problem *P.1*. The subset of coordinates in X that are positive ($x_k > 0$) is called the support of X. A solution vector is full support if all its coordinates are positive. A particular reordering of X, $(x_{m1}, x_{m2}, x_{m3}, ..., x_{mN})$, such that the inter-contact time of x_{mk} is in a nondecreasing order is an order vector of X and is represented by X_{order}. The objective value of problem *P.1* represents the expected latency of ts when it is partitioned and allocated according to X. Without loss of generality, $t(X)$ is used to represent it.

[*4] An idealized situation in which tasks can be partitioned seamlessly is assumed in this chapter. Also assumed is that the execution bodies of tasks are proportional to the size of the input parameters and output results, e.g., a subtask with an x_k value of 0.5 contains 50% of the input parameters, execution body, and output results of the original task.

4.5 Proposed algorithms for opportunistic resource sharing

4.5.1 LP-based approximation algorithm

It is difficult to solve optimization problem *P.1* directly due to the discontinuities of the ceiling functions in Eqs. (4.6), (4.7), and (4.10). Therefore, an approximation **a**lgorithm based on **l**inear **p**rogramming (LP) [52] is proposed to solve it and is referred to as "ALP".

First, an auxiliary function called TL' is defined to approximate TL defined in Eq. (4.13):

$$TL'\ (ts) = \frac{ts_i + ts_o}{CB_j} + \frac{ts_e}{AR_j} + \frac{ts_i + ts_o}{T_{up-one}} \times IT_j\ , \tag{4.14}$$

where T_{up-one} is defined in Eq. (4.3). It is easy to verify that $TL'(TS_{dummy})$ is zero. As a result, optimization problem *P.1* is transformed into:

P.2:

$$\text{Min. } max_{k=1}^{N} TL_k^{'}\ (x_k \times ts)\ ,$$

$$\text{S.t.} \quad \sum_{k=1}^{N} x_k = 1\ ; \quad x_k \geq 0\ \ (k = 1, \ldots, N)\ .$$

The definitions of solution vectors, supports of solution vectors and order vectors are the same as those for the problem *P.1*. The objective value of problem *P.2* is represented by $t^{'}(X)$. It is easy to see that both the definitions of TL' and the constraint conditions in *P.2* are convex. As a result, problem *P.2* is a piecewise-linear minimization problem and is equivalent to the LP problem

P.3:

$$\text{Min. } v\ ,$$

$$\text{S.t.} \quad TL'_k\ (x_k \times ts) \leq\ v\ \ (k = 1, \ldots, N)\ ;$$

$$\sum_{k=1}^{N} x_k = 1\ ; \quad x_k \geq 0\ \ (k = 1, \ldots, N)\ ;.$$

where v is an auxiliary variable.

Theorem 1. *Application of the same solution vector X to problems P.1 and P.2 limits*

the error between the objective values to within the maximum inter-contact time of the support of X, $IT_{max-support}(X)$, viz., $|t(X) - t^{'}(X)| \leq IT_{max-support}(X)$.

Proof. In Section 4.7. □

From Theorem 1, it is easy to see that $t^{'}(X) + IT_{max-support}(X)$ is the worst-case bound of $t(X)$, i.e., $t_{worst-bound}$. However, we can only get a solution vector X with full support by solving *P.3* directly due to the limitation of LP approximation. This defect may cause a large $t_{worst-bound}$ in some cases, e.g., a small subtask is assigned to an RP with a large inter-contact time that is rarely encountered. As a result, if the support of the optimal solution vector $X_{optimal}$ for problem *P.1* can be found and *P.3* can be solved on the support of $X_{optimal}$ the error is further decreased.[*5]

Theorem 2. *Suppose that $(x_{o1}, x_{o2}, x_{o3}, ..., x_{oN})$ is the optimal solution vector $X_{optimal}$ to the problem P.1 and that (x_{m1}, x_{m2}, x_{m3}, ..., x_{mN}) is its corresponding order vector X_{order}. If x_{mk} is in the support of $X_{optimal}$, then all coordinates before x_{mk} in X_{order} are also in the support of $X_{optimal}$.*

Proof. In Section 4.7. □

From Theorem 2, there are N candidate supports for $X_{optimal}$, viz., (x_{m1}), (x_{m1}, x_{m2}), (x_{m1}, x_{m2}, x_{m3}) ... $(x_{m1}, x_{m2}, x_{m3}, ..., x_{mN})$. Problem *P.3* is solved on N candidate supports, and the one that minimizes $t_{worst-bound}$ is selected as the result of the ALP algorithm, namely X_r.

Theorem 3. *Application of X_r and $X_{optimal}$ to problem P.1 restricts the error between $t(X_r)$ and $t(X_{optimal})$ to within twice the maximum inter-contact time of the support of $X_{optimal}$, $IT_{max-support}(X_{optimal})$.*

Proof. In Section 4.7. □

[*5] Here, solving *P.3* on the support of a solution vector X means set x_k to be zero in the constraint conditions of *P.3* if it is not in the support of X.

ALP algorithm

X_r = Solve *P.3* on the support (x_{m1}, x_{m2}, x_{m3}, ... , x_{mN}) ;
$X_{temp} = \emptyset$; $i = 1$;

for (i = 1 ; i < N ; ++i)
  X_{temp} = Solve *P.3* on the support (x_{m1}, x_{m2}, x_{m3}, ... , x_{mi}).
  if ($t_{worst-bound}(X_{temp}) < t_{worst-bound}(X_r)$)
    $X_r = X_{temp}$;
return X_r

4.5.2 Algorithm for accidental contacts

An RS uses the ALP algorithm to partition the original task into subtasks and assigns them to a group of appropriate RPs. However, due to the randomness of contact opportunities, an RS may encounter an RP that has idle resources but was not scheduled any subtask by the ALP algorithm. The RS should still be able to utilize the idle resources in that RP to accelerate its task completion. An **a**lgorithm is introduced here to deal with this kind of **a**ccidental **c**ontacts. It supplements the ALP algorithm and is called "AAC".

The basic idea of AAC is simple: the RS estimates the duration of the current contact and assigns the encountered RP a subtask that it can finish within that time.[*6] Since estimating contact duration is not the focus of this chapter, a simple model in which two nodes move from their current locations to their destinations in line is used. As a result, the duration of their contact is calculated in accordance with their traces, i.e., the period during which their distance apart is within the wireless transmission range of each other. Location information can be obtained through GPS or peer-based techniques [48]. It should be noted that other improved methods of es-

[*6] Since the result of ALP indicates that the current encountered RP is not a good candidate, it may take a long time to make contact with the encountered RP again.

timating contact duration can be smoothly integrated into AAC.

The last problem is which part of the remaining (unfinished) subtasks should be allocated to the current encountered RP. There are two types of remaining subtasks:

(1) The subtask whose input parameters have not been completely uploaded to the RP assigned by the ALP algorithm (type one);

(2) The subtask whose input parameters have already been uploaded to the RP assigned by the ALP algorithm. However, output results of the subtask have not been received yet (type two).

The following two rules are used to determine the priority of subtasks allocated to the current encountered RP:

(1) Subtasks of type one have a higher priority than those of type two;

(2) In the same type of remaining subtasks, the subtask previously assigned to the RP with largest inter-contact time has the highest priority, since an RS meets RPs with larger inter-contact time less frequently.

It is clear that subtasks of type two may be parallel processed by both the current encountered RP and the RP assigned by the ALP algorithm. Two copies of each subtask are parallel processed in the worst case. Therefore, using of the AAC algorithm can result in duplicate task processing.

Remark: When two mobile nodes make contact first time, entries for each other are created in their OCTs with a large initial value of IT and small initial values of CD, CB, and AR. Since this means large expected task latencies, at the beginning, these two nodes seldom schedule their tasks to each other based on the ALP algorithm. They mainly share resources through the AAC algorithm at this stage. As time passes, variables in the OCTs converge to their averages which promotes deep-level resource sharing between them step by step. A timer can be set to delete outdated entries to prevent failure of nodes. In that case, two nodes treat each other as a new node when they encounter again.

AAC algorithm

$CD_{current}$ = calculate length of current contact;
ts_{pri} = select a remaining subtask based on priority rules;

// $y_k \times ts_{pri}$ can be finished within $CD_{current}$
assign a fraction of $y_k \times ts_{pri}$ to the current encountered RP.

remaining subtask is $(1 - y_k) \times ts_{pri}$.

4.6 Simulation

A custom discrete-time simulator implemented in C/C++ was used to evaluate the proposed algorithms. A landmark based opportunistic network model [63] was used in the simulation. Mobile nodes were at first uniformly distributed in a rectangular area. There were four square areas called "landmarks" in the area. Each mobile node usually moved in the landmark area with which it was associated. The mobile nodes were assumed to have deviation probability p. That is, every node has a probability $1 - p$ to visit a destination in the landmark with which it was associated, and visit a destination in another landmark area with probability $p/3$.

Mobile nodes were divided into three categories according to their amounts of resources. One unit of resources is assumed to process one unit of task execution body in one second.

(1) Super node (SN): node with 100 units of resources.
(2) Common node (CN): node with 20 units of resources.
(3) Poor node (PN): node with no resources.

Without loss of generality, there was one SN, two CNs, and two PNs associated with each landmark. Every node generated a task with an equal probability. Each

(sub)task was served according to a first-in-first-out (FIFO) policy by RPs. Five seconds were preserved for mobile nodes to detect each other at the begining of each contact. The value of α for generating OCTs in Eq. (1) was set to 0.9. The simulation first run without generating any task for 20,000 seconds so that the nodes could establish their OCTs. This is called the training period of OCTs.[*7] After that, 100 tasks were generated at 200-seconds intervals by the mobile nodes and their metrics

Table. 4.2 Parameters for simulation

Simulation area	3000 × 3000 m
Size of landmarks	1000 × 1000 m
Number of landmarks	4
Upper left corner of landmarks	(1000, 1000), (1000, 3000), (2000, 2000), (0, 2000),
Number of SNs	4
Number of CNs	8
Number of PNs	8
Size of execution body, ts_e	10,000 units
Size of input parameters, ts_i	100 Mbits
Size of output results, ts_o	100 Mbits
Link bandwidth	6 Mbps
Transmission range of nodes	150 m
Mobility model	Random way point
Mobility speed	10 (m/s)
Pause time	0 s
Deviation probability	0.1
α for generating OCTs	0.9
Training period of OCTs	20,000 s

[*7] With the default value of α, 20,000 seconds is enough for OCTs to reach the steady state in following simulation. The impacts of different values of α and training period are further discussed in Section 4.6.3.

were measured. The results presented in this section are the average values obtained for 100 trials. Without specifically pointing out, the basic parameters summarized in Table 4.2 were used in the following simulation.

Four different methods were compared. The first two were proposed in Section 4.5. Since the AAC algorithm relies on the ALP algorithm, it was not evaluated alone.

(1) ALP: The RS partitions the original task into subtasks and assigns them to a group of RPs in accordance with the proposed ALP algorithm.

(2) ALP-AAC: Apart from the ALP algorithm, the RS uses the AAC algorithm to make use of accidental contact opportunities.

(3) Epidemic: The RS replicates the original task to every encountered RP until it receives the result from one. To reduce resource consumption, an RS cancels duplicated tasks in encountered RPs if it has already received the results of this task.

(4) Best-qualified RP (BQR): The RS assigns the original task to the best-qualified RP, i.e., the one expected to minimize task latency. This is an improved version of the method proposed by Shi et al. [18, 57].*8

Four methods were compared on the basis of two criteria.

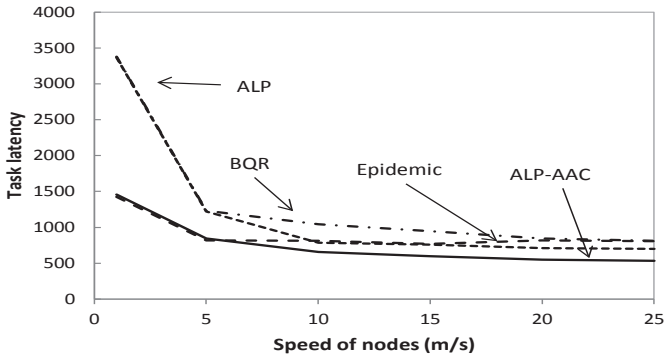
Task latency (TL): described in Section 4.3.2.

Duplication rate (DR): since both ALP-AAC and Epidemic methods generate duplicate (sub)tasks, this metric measures the ratio between the size of the task being processed and the size of the original task. Small DR is preferred to prevent saturating RPs' resources. Since ALP and BQR methods do not generate any duplicate task, their values of DR are always one. As described in Section 4.5.2, the ALP-AAC method generates at most two copies of each subtask. As a result, its value of DR is no larger than two in any case.

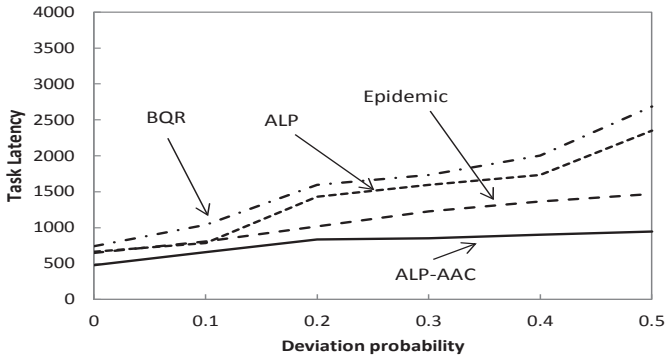
*8 The method proposed in [18, 57] select one best-qualified RP without considering about time consumed by uploading the parameters and downloading the results when common control channel (CCC) does not exist.

4.6.1 Parameters of the mobility model

The proposed algorithms were first evaluated for different parameters of the mobility model. The average latencies for one task for different node speeds are shown in Fig. 4.3(a). When the speed was slow, the values of inter-contact time and contact duration were large. That made it harder for an RS to encounter a specified RP. However, it also indicates that if an RS encountered an RP, the large contact duration ensured a high probability of finishing the task within this contact. As a result, both ALP

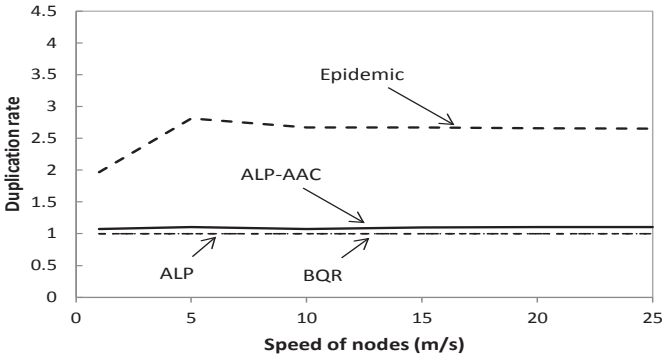


(a) Speeds of nodes

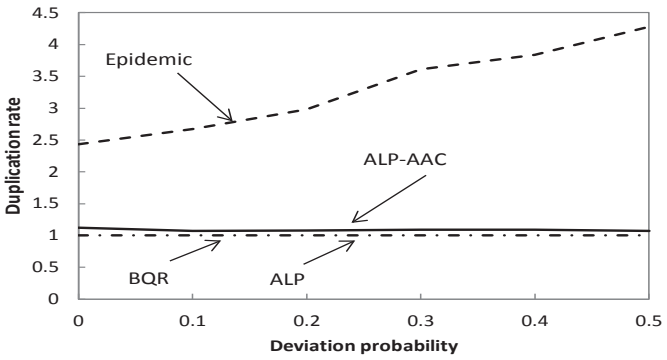


(b) Deviation probability

Fig. 4.3 Task latencies for different parameters of the mobility model.



(a) Speeds of nodes



(b) Deviation probability

Fig. 4.4 Duplication rates for different parameters of the mobility model.
@2014 IEICE

and BQR performed worse because they only assign (sub)tasks to a group of specified RPs. Both ALP-AAC and Epidemic performed better since they enabled an RS to utilize resources in any RP that is encountered first. The average latencies for one task with different deviation probabilities are shown in Fig. 4.3(b). The average latency increased with the deviation probability. This was because a larger deviation probability results in the mobile nodes moving in a wider area rather than spending most of their time in the landmark area with which they were associated. Therefore, it was more difficult for mobile nodes to make contact with each other.

Figs. 4.4(a) and 4.4(b) show the duplication rates for one task for different node

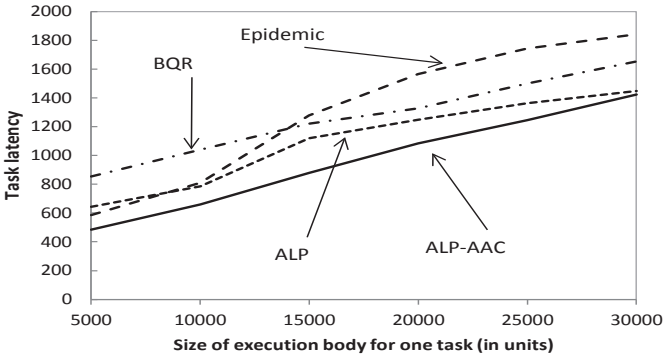
speeds and different node deviation probabilities. ALP and BQR methods did not generate any duplicate task. Therefore, a single line of value one is used to represent their values in the figures. Since Epidemic blindly replicated the original task to every encountered RP, it had a much larger DR than the other three methods. ALP-AAC only utilized idle resources in a contacted RP. As a result, its DR values were close to one in both figures.

It is obvious from these results that the proposed ALP-AAC method had the smallest task latencies of the four methods for the different parameters of the mobility model. Compared with the Epidemic method, it had dramatically lower DR values. Moreover, the proposed ALP method performed better than the BQR method in most cases even for a constant DR value of one.

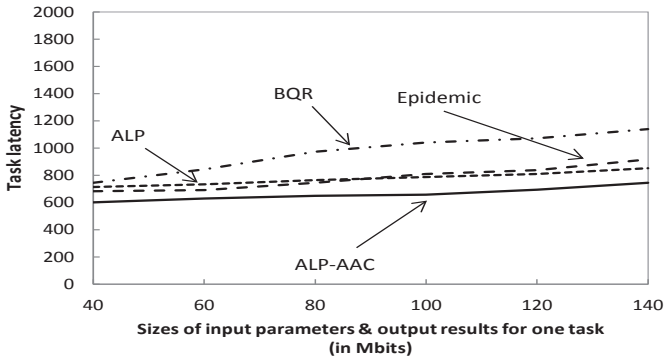
4.6.2 Parameters of tasks

The proposed algorithms for different parameters of tasks were evaluated. The average latencies for one task for different execution body sizes (ts_e) are shown in Fig. 4.5(a). The average latency increased with t_{se}. This was because mobile nodes need more time to process a larger task, as shown by Eq. (4.12). Along with the increasing size of ts_e, the workload of the RPs increased. As a result, there was less chance of encountering an RP with idle resources and thus to utilize the AAC algorithm. This is why the performance of the ALP algorithm converged to that of the ALP-AAC algorithm as ts_e increased. The average latencies for one task for different input parameter sizes (ts_i) and output result sizes (ts_o) are shown in Fig. 4.5(b). The sizes of ts_i and ts_o were assumed to be the same in the simulation. The average task latency increased with ts_i and ts_o. This was because more time was consumed by uploading the input parameters and downloading the output results.

Fig. 4.6(a) and Fig. 4.6(b) show duplication rates for different parameters of tasks. It seems counterintuitive that the DR for Epidemic decreased with ts_i and ts_o, as shown in Fig. 4.6(b). This was because a larger ts_i meant a longer process of uploading the parameters which may take one or more contacts to complete. An RS had a higher probability of receiving results from another RP during this period.



(a) Execution bodies

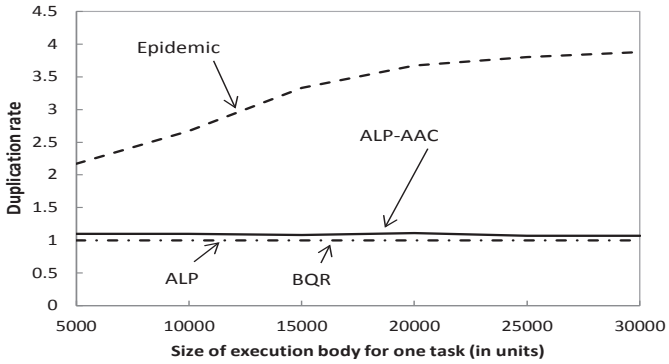


(b) input parameters & output results

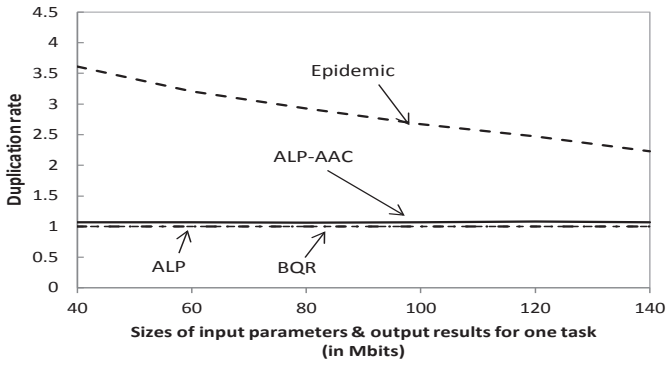
Fig. 4.5 Task latencies for different parameters of tasks.
@2014 IEICE

Since the task had been finished, the RS cancled duplicate tasks in the remaining RPs when they encountered, which reduced the values of DR. Nevertheless, the DR results for Epidemic were still much larger than those for the other methods in both figures.

It is obvious from these results that the proposed ALP-AAC method had the smallest task latencies for the different parameters of generated tasks. It had a much smaller DR even when there was a mechanism of cancling tasks in the Epidemic method. The proposed ALP method performed better than the BQR method when those methods with a DR value of one are considered.



(a) Execution bodies



(b) input parameters & output results

Fig. 4.6 Duplication rates for different parameters of tasks.
@2014 IEICE

It should be noted that simulation on increasing of ts_e was equivalent to simulation on decreasing the amount of resources in the mobile nodes since both of them increased the duration of task processing. Similarly, simulation on increasing of ts_i (ts_o) was equivalent to simulation on decreasing the bandwidth between the mobile nodes since both of them increased the time consumed by data transmission.

4.6.3 Parameters of generating OCTs

Finally, to determine the impact of accuracy of OCTs on the proposed methods, different parameter settings for generating OCTs are discussed. Fig. 4.7 shows aver-

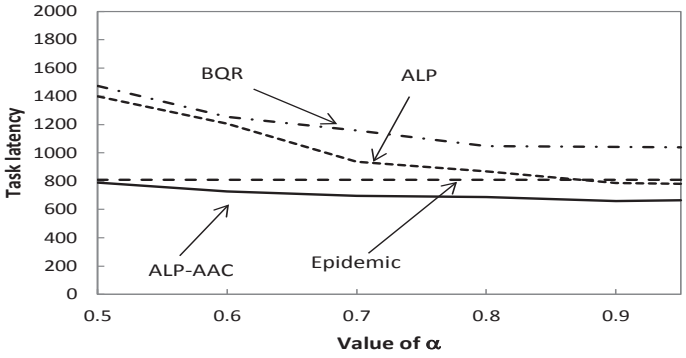


Fig. 4.7 Task latencies for different values of α.

@2014 IEICE

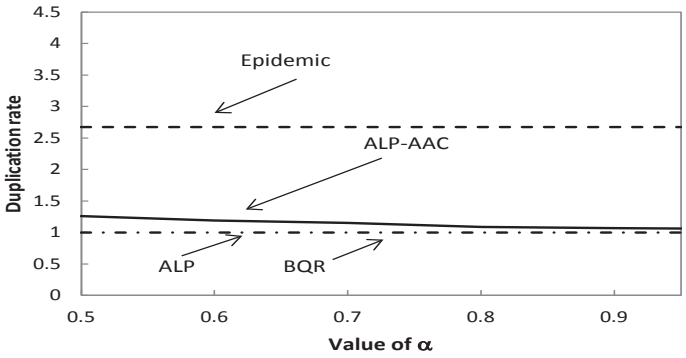


Fig. 4.8 Duplication rates for different values of α.

@2014 IEICE

age task latencies for one task under different values of α. Since the historical part usually has a higher weight in the moving window average strategy, only values larger than 0.5 were considered. Task latencies for the Epidemic method are presented for the sake of convenience, although they were not influenced by statistics in the OCTs. As discussed in Section 4.3.1, the accuracy of OCTs increases with the value of α. Therefore, BQR, ALP, and ALP-AAC performed worse when the value of α is small, since they allocated tasks according to inaccurate statistics in OCTs. The proposed ALP method performed better than the BQR method, even if both of them suffered from inaccurate statistics and did not generate any duplicate task. The proposed ALP-AAC method was more tolerant to inaccurate OCTs, since it used duplication strategies to compensate for inaccurate task allocations. This was proved by Fig.

4.8 in which the duplication rate of ALP-AAC became larger when the value of α decreased.

The accuracy of OCTs also depends on its training period. As discussed in Section 4.3.1, a larger value of α requires a longer training period to generate accurate OCTs. Simulation results under different training periods were similar to those under different values of α shown in Figs. 4.7 and 4.8, since both of them resulted in inaccurate OCTs when their values were small. Therefore, they are not shown here.

It is obvious from these results that accurate user statistics are important to the concept of opportunistic resource sharing. A simple moving window average strategy is used in this chapter to illustrate the superiority of our proposal. However, how to extract knowledge from existing data, such as life-logs of mobile users, to generate accurate OCTs is still an important open issue that needs further investigation.

4.7 Proofs of theorems

Theorem 1. *Application of the same solution vector X to problems P.1 and P.2 limits the error between the objective values to within the maximum inter-contact time of the support of X, $IT_{max-support}(X)$, viz., $|t(X) - t^{'}(X)| \leq IT_{max-support}(X)$.*

Proof. X is assumed to be a solution vector to task ts. Without loss of generality, the *k-th* coordinate of X, x_k, is considered. When x_k is not in the support of X (x_k is zero), both $TL_k(x_k \times ts)$ defined in problem *P.1* and $TL_k^{'}(x_k \times ts)$ defined in problem *P.2* are zero. Consequently, the difference between their values is zero.

When x_k is in the support of X, there are four cases for $TL_k(x_k \times ts)$ depending on when the subtask is generated and finished: (1) the subtask is generated when RS is in contact with the *k-th* RP (GC), (2) the subtask is generated when RS is not in contact with the *k-th* RP (GNC), (3) the subtask is finished when RS is in contact with the *k-th* RP (FC), and (4) the subtask is finished when RS is not in contact with the *k-th* RP (FNC). A simple result that is used in the following proof is

$$-1 < A - \lceil A \rceil \leq 0 \ . \tag{4.15}$$

(1) GC & FC

In this case, the time spent waiting for contacting opportunities, T_{wait}, for uploading input parameters and downloading output results is calculated using Eq. (4.10).

$$\begin{aligned} TL_k^{'}(x_k \times ts) - TL_k(x_k \times ts) = (&\frac{ts_i \times x_k}{T_{up-one}} - \frac{1}{2} - \lceil \frac{ts_i \times x_k}{T_{up-one}} - \frac{1}{2} \rceil \\ &+ \frac{ts_o \times x_k}{T_{up-one}} - \frac{1}{2} - \lceil \frac{ts_o \times x_k}{T_{up-one}} - \frac{1}{2} \rceil + 1) \times IT_k \ , \end{aligned} \tag{4.16}$$

where IT_k is the inter-contact time between RS and the $k-th$ RP.

(2) GNC & FNC

In this case, the time spent waiting for contact opportunities, T_{wait}, for uploading input parameters and downloading output results is calculated using Eq. (4.7).

$$\begin{aligned} TL_k^{'}(x_k \times ts) - TL_k(x_k \times ts) = (&\frac{ts_i \times x_k}{T_{up-one}} - 1 - \lceil \frac{ts_i \times x_k}{T_{up-one}} - 1 \rceil \\ &+ \frac{ts_o \times x_k}{T_{up-one}} - 1 - \lceil \frac{ts_o \times x_k}{T_{up-one}} - 1 \rceil + 1) \times IT_k \ . \end{aligned} \tag{4.17}$$

(3) GC & FNC

In this case, the time spent waiting for contact opportunities, T_{wait}, for uploading input parameters is calculated using Eq. (4.10), while that for downloading output results is calculated using Eq. (4.7).

$$\begin{aligned} TL_k^{'}(x_k \times ts) - TL_k(x_k \times ts) = (&\frac{ts_i \times x_k}{T_{up-one}} - \frac{1}{2} - \lceil \frac{ts_i \times x_k}{T_{up-one}} - \frac{1}{2} \rceil \\ &+ \frac{ts_o \times x_k}{T_{up-one}} - 1 - \lceil \frac{ts_o \times x_k}{T_{up-one}} - 1 \rceil + 1) \times IT_k. \end{aligned} \tag{4.18}$$

(4) GNC & FC

In this case, the time spent waiting for contact opportunities, T_{wait}, for uploading input parameters is calculated using Eq. (4.7) while that for downloading output results is calculated using Eq. (4.10).

$$\begin{aligned} TL_k^{'}(x_k \times ts) - TL_k(x_k \times ts) = (&\frac{ts_i \times x_k}{T_{up-one}} - 1 - \lceil \frac{ts_i \times x_k}{T_{up-one}} - 1 \rceil \\ &+ \frac{ts_o \times x_k}{T_{up-one}} - \frac{1}{2} - \lceil \frac{ts_o \times x_k}{T_{up-one}} - \frac{1}{2} \rceil + 1) \times IT_k. \end{aligned} \tag{4.19}$$

Application of Eq. (4.15) to Eqs. (4.16), (4.17), (4.18), and (4.19) gives that, for all cases,

$$-IT_k < TL_k^{'}(x_k \times ts) - TL_k(x_k \times ts) \leq IT_k \ . \tag{4.20}$$

Since $t(X)$ and $t^{'}(X)$ are the maximum values of $TL_k(x_k \times ts)$ and $TL_k^{'}(x_k \times ts)$ for all coordinates x_k in X, without loss of generality, $t(X) = TL_m(x_m \times ts)$ and $t^{'}(X) = TL_n^{'}(x_n \times ts)$ are assumed. Since $TL_m(x_m \times ts) \geq TL_n(x_n \times ts)$ and $TL_n^{'}(x_n \times ts) \geq TL_m^{'}(x_m \times ts)$,

$$TL_n^{'}(x_n \times ts) - TL_m(x_m \times ts) \leq TL_n^{'}(x_n \times ts) - TL_n(x_n \times ts) \leq IT_n. \quad (4.21)$$

$$TL_n^{'}(x_n \times ts) - TL_m(x_m \times ts) \geq TL_m^{'}(x_m \times ts) - TL_m(x_m \times ts) > -IT_m. \quad (4.22)$$

Finally, since both x_n and x_m are in the support of X,

$$|t(X) - t^{'}(X)| = |TL_m(x_m \times ts) - TL_n^{'}(x_n \times ts)| \leq IT_{max-support}(X)\ . \quad (4.23)$$

The proof is complete. □

Theorem 2. *Suppose that* $(x_{o1}, x_{o2}, x_{o3}, ..., x_{oN})$ *is the optimal solution vector* $X_{optimal}$ *to the problem P.1 and that* (x_{m1}, x_{m2}, x_{m3}, ..., x_{mN}) *is its corresponding order vector* X_{order}. *If* x_{mk} *is in the support of* $X_{optimal}$, *then all coordinates before* x_{mk} *in* X_{order} *are also in the support of* $X_{optimal}$.

Proof. Suppose x_{mi} is a coordinate before x_{mk} that is not in the support of $X_{optimal}$. Without loss of generality, the subtask assigned to the $n-th$ RP, $x_n \times ts$ is assumed to determine the value of $t(X_{optimal})$. However, a fraction of the subtask can be offloaded from x_n to x_{mi} to include x_{mi} in the support of a new solution vector, X_{new}. The fraction of offloaded subtask is represented by y_n. The remaining subtask assigned to the $n-th$ RP is $(1-y_n) \times x_n \times ts$, and the subtask assigned to the $mi-th$ RP is $y_n \times x_n \times ts$. As a result, the time consumed by the $n-th$ provider, $TL_n((1-y_n) \times x_n \times ts)$, is reduced. At the same time, the time consumed by the $mi-th$ provider $TL_{mi}(y_n \times x_n \times ts)$ is increased because no subtask is assigned to it in $X_{optimal}$.

Since tasks are assumed to be partitioned seamlessly in this chapter, y_n is a continuous variable. Since x_{mi} is before x_{mk} in X_{order}, there always exists a small enough y_n to satisfy the inequation

$$TL_{mi}(y_n \times x_n \times ts) < TL_{mk}(x_{mk} \times ts)\ , \quad (4.24)$$

where $x_{mk} \times ts$ is the subtask assigned to the $mk-th$ provider. This means that the $mi-th$ provider is able to finish its subtask before the $mk-th$ provider while the $n-th$ provider is able to finish its subtask in less time compared with the previous allocation in $X_{optimal}$. Since coordinates other than x_n and x_{mi} in $X_{optimal}$ remain the same, X_{new} is a better solution vector than $X_{optimal}$, which contradicts the hypothesis. Thus, x_{mi} is in the support of $X_{optimal}$ and the proof is complete. □

Theorem 3. *Application of X_r and $X_{optimal}$ to problem P.1 restricts the error between $t(X_r)$ and $t(X_{optimal})$ to within twice the maximum inter-contact time of the support of $X_{optimal}$, $IT_{max-support}(X_{optimal})$.*

Proof. Two inequations can be derived from Theorem 1:

$$t(X_r) \leq t^{'}(X_r) + IT_{max-support}(X_r) \ , \tag{4.25}$$

$$t(X_{optimal}) \leq t^{'}(X_{optimal}) + IT_{max-support}(X_{optimal}) \ . \tag{4.26}$$

Since $X_{optimal}$ minimizes the objective value of problem *P.1*,

$$t(X_{optimal}) \leq t(X_r) \ . \tag{4.27}$$

Since X_r minimizes $t_{worst-bound}$ for all candidate supports of $X_{optimal}$ by the ALP algorithm,

$$t^{'}(X_r) + IT_{max-support}(X_r) \leq t^{'}(X_{optimal}) + IT_{max-support}(X_{optimal}) \ . \tag{4.28}$$

Combining Eqs. (4.25), (4.26), (4.27), (4.28) produces

$$t(X_{optimal}) \leq t(X_r) \leq t^{'}(X_{optimal}) + IT_{max-support}(X_{optimal}) \ , \tag{4.29}$$

leading to

$$\begin{aligned} t(X_r) - t(X_{optimal}) &\leq t^{'}(X_{optimal}) - t(X_{optimal}) + IT_{max-support}(X_{optimal}) \\ &\leq 2 \times IT_{max-support}(X_{optimal}). \end{aligned} \tag{4.30}$$

The final induction step of Eq. (4.30) is based on Theorem 1. The proof is complete.

□

4.8 Conclusion

This chapter presented the concept and design of an opportunistic resource sharing mechanism. Mobile nodes make use of contact opportunities between themselves to share resources and accelerate their task completion. Theoretical models and formal definitions of problems were presented. Formal proofs were presented to illustrate the efficiency of the proposed approximation algorithm. Simulation results also validate that the proposed mechanism is efficient in terms of both resource consumption and task completion time. Interesting future work includes investigating discrete task models, improved methods to generate accurate OCTs, and resource sharing with indirectly encountered RPs.

Chapter 5

Conclusions and Future Directions

This thesis discussed cooperative resource sharing among mobile devices in mobile cloud computing. Each chapter focused on particular technical problems related to cooperative resource sharing as below:

Chapter 2 discussed an energy-efficient method of adaptive resource discovery against the background of MCC. Mobile nodes in the system either maintain a resource directory in the CRB through a widely-covered 3G network (a centralized mode) or flood resource requests in the area through a short-range WLAN ad hoc network (a flooding mode) to discover resources. According to different network environments, our proposed adaptive discovery method transforms between centralized and flooding modes to save energy. Theoretical models of both energy consumption and quality of response information were presented in this chapter. A prediction algorithm was provided to enable the proposed method to select energy-efficient discovery mode automatically. Simulation results validate that the proposed adaptive solution to resource discovery is energy-efficient in different network environments due to its adaptivity.

Our work was only the first step toward utilizing the proposed adaptive strategy. As discussed in the chapter, this model could be integrated with other improved centralized and flooding methods to replace their basic counterparts. It could also be used to optimize other metrics like response time and RIA. Mechanisms that automatically optimize parameters for the proposed algorithm are interesting. Finally, when node mobility is considered, nodes may not be distributed uniformly throughout the area. An appropriate model or algorithm still needs to be found to estimate energy consumption in the flooding mode.

Chapters 3 and 4 discussed resource sharing and coordination mechanisms against the background of MCC. Particularly, chapter 3 considered local resource sharing among nearby mobile devices in the consistently connected networks. We focused on the construction of an LRC comprised of multiple RPs in dynamic mobile environments. Along with theoretical models based on the availability (energy-efficiency) and task latency (time-efficiency) of an LRC, the problem was formulated to minimize extra energy consumption of the constructed LRC under a constraint of its task latency. An efficient greedy heuristic algorithm with low computational complexity was also presented. Simulation results validate that the proposed models are efficient in terms of both energy consumption and task completion time. The proposed method (GELRC) with low computational complexity is suitable for dynamic mobile wireless environments.

For future work, explicit considerations of energy consumed by relaying packets, sustainability and fairness of LRCs are interesting research areas in local resource sharing. Privacy and security issues are also important to realize this design.

Chapter 4 presented the concept and design of opportunistic resource sharing among mobile devices in the intermittently connected networks. Mobile nodes make use of contact opportunities between themselves to share resources and accelerate their task completion. Theoretical models that capture the characteristics of contact opportunities were presented. Algorithms were proposed to minimize the task completion time through opportunistic resource sharing. Theoretical analyses between the optimal bound and the proposed approximation algorithm were given. Simulation results validate that the proposed mechanism is efficient in terms of both resource consumption and task completion time.

For future work, since an idealized continuous task model was considered in this chapter, design of a mathematical model for opportunistic resource sharing on discrete task models is interesting. The method for generating accurate knowledge from existing data, such as life-logs of mobile users, is important. Finally, rather than among a single-hop away, directly contacted mobile users, mechanisms for opportunistic resource sharing among multi-hops away, indirectly contacted mobile users need to be proposed.

Acknowledgement

The author would like to extend my gratitude to a number of people, since the accomplishment of this thesis depends on their kind help and effort:

Special acknowledgment to my supervisor, Professor Tatsuro Takahashi, Graduate School of Informatics, Kyoto University, for his expert instructions, farsighted suggestions on my research and warm-hearted help in my daily life for the past three years.

I am deeply grateful to Professor Hiroshi Harada and Professor Ken Umeno, Graduate School of Informatics, Kyoto University, for their insightful suggestions and comments on this thesis.

I would like to express my sincere gratitude to Associate Professor Ryoichi Shinkuma, Graduate School of Informatics, Kyoto University. Without his patient supervision, constructive suggestions and valuable discussions across my Ph.D. studies, this thesis would not be possible to be finished.

I would like to express my gratitude to Asistant Professor Takayuki Nishio, Graduate School of Informatics, Kyoto University, especially for his kind help during my preparation for entrance examination and the early stage of my research.

Special thanks to Professor Shuyu Chen, School of Software Engineering, Chongqing University, China. His supervising during my M.E. study and on-going encouragement lay the foundation for my study in Kyoto University, Japan.

Thanks to the stuff members and all students in the Takahashi laboratory for their help in both academic areas and my daily life. Particularly, to the members of Innovative team, many thanks for your helpful advices which contribute a lot to this thesis.

I wish to give my thanks to all of my friends (both Japanese and Chinese) for their selfless help, support, and care during my difficult time.

At last, I would like to express my sincere gratitude to my family, especially my father Tianping Liu and my mother Xin Wei, for their warm support and encouragement in my life.

Publication

Journal papers

1. W. Liu, R. Shinkuma, and T. Takahashi, "Opportunistic resource sharing in Mobile Cloud Computing," IEICE Transactions on Communications, vol. E97-B, no. 12, Dec. 2014 (to be appear).
2. W. Liu, R. Shinkuma, and T. Takahashi, "A Local Resource Sharing Platform in Mobile Cloud Computing," IEICE Transactions on Communications, vol. E97-B, no. 9, pp. 1865-1874, Sep. 2014.
3. W. Liu, T. Nishio, R. Shinkuma, and T. Takahashi, "Adaptive resource discovery in Mobile Cloud Computing," Computer Communications, vol. 50, pp. 119-129, Sep 2014.

International conferences

1. W. Liu, R. Shinkuma, and T. Takahashi, "Opportunistic Resource Sharing in Mobile Cloud Computing: the Single-copy Case," Proc. IEEE APNOMS 2014, pp. 1-6, Hsinchu, Taiwan, Sep. 2014.
2. W. Liu, R. Shinkuma, and T. Takahashi, "A design of energy-efficient resource sharing overlay network in Mobile Cloud Computing," Proc. IEEE APNOMS 2013, pp. 1-3, Hiroshima, Japan, Sep. 2013.
3. W. Liu, T. Nishio, and R. Shinkuma, "System design and numerical analysis of adaptive resource discovery in wireless application networks," Proc. ITU Kaleidoscope conference 2013, pp. 1-8, Kyoto, Japan, April 2013.

Workshops and Conferences

1. W. Liu, R. Shinkuma, and T. Takahashi, "Opportunistic Resource Sharing in Mobile Cloud Computing: the Single-copy Case," IEICE Technical Report, CQ2013-33, pp.63-68, Ishinomaki, Japan, Sep. 2014.
2. W. Liu, R. Shinkuma, and T. Takahashi, "MRSON: A Design of Energy-

efficient Resource Sharing Overlay Networks in Mobile Cloud Computing," IEICE Technical Report, CQ2013-33, pp.29-34, Kanazawa, Japan, Sep. 2013.

3. R. Hu, W. Liu, R. Shinkuma, and T. Takahashi, "Modeling of Fair Resource Sharing between MVNOs, " IEICE Technical Report, CQ2014-22, pp. 39-44, Osaka, Japan, July. 2014.
4. R. Hu, W. Liu, R. Shinkuma, and T. Takahashi, "Resource Sharing Model for MVNOs," Proceedings of IEICE General Conference 2014, BS-1-11, IEICE, Niigata, Japan, Mar. 2014.
5. R. Hu, W. Liu, R. Shinkuma, and T. Takahashi, "Modeling of Resource Sharing among MVNOs, " IEICE Technical Report, CQ2013-68, pp. 7-10, Tokyo, Japan, Jan. 2014.

Bibliography

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.*, "A view of cloud computing," Communications of the ACM, vol.53, no.4, pp.50–58, 2010.

[2] "Mobile cloud computing forum." http://www.mobilecloudcomputingforum.com.

[3] "Cisco visual networking index: Global mobile data traffic forecast update, 2012-2017." http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white paper c11 -520862.pdf, 2013.

[4] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. Johnson, "M2m: From mobile to embedded internet," Communications Magazine, IEEE, vol.49, no.4, pp.36–43, April 2011.

[5] R.H. Weber and R. Weber, Internet of Things, Springer, 2010.

[6] K. Vanthournout, G. Deconinck, and R. Belmans, "A taxonomy for resource discovery," Personal and Ubiquitous Computing, vol.9, no.2, pp.81–89, 2005.

[7] L. Guan, X. Ke, M. Song, and J. Song, "A survey of research on mobile cloud computing," Computer and Information Science (ICIS), 2011 IEEE/ACIS 10th International Conference on, pp.387–392, IEEE, 2011.

[8] N. Fernando, S.W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," Future Generation Computer Systems, vol.29, no.1, pp.84–106, 2013.

[9] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp.13–16, ACM, 2012.

[10] D. Datla, X. Chen, T. Tsou, S. Raghunandan, S.S. Hasan, J.H. Reed, C.B. Dietrich, T. Bose, B. Fette, and J. Kim, "Wireless distributed computing: A survey of research challenges," Communications Magazine, IEEE, vol.50, no.1, pp.144–152, 2012.

[11] G.A. Lewis, "Mobile computing at the edge: Keynote summary," Proceedings of the 2013 ACM Workshop on Mobile Development Lifecycle, MobileDeLi '13, New York, NY, USA, pp.29–30, ACM, 2013.

[12] S. Olariu, I. Khalil, and M. Abuelela, "Taking vanet to the clouds," International Journal of Pervasive Computing and Communications, vol.7, no.1, pp.7–21, 2011.

[13] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, p.6, ACM, 2010.

[14] F.H. Fitzek, M. Katz, and Q. Zhang, "Cellular controlled short-range communication for cooperative p2p networking," Wireless Personal Communications, vol.48, no.1, pp.141–155, 2009.

[15] A. Pasarella, M. Kumar, M. Conti, and E. Borgia, "Minimum-delay service provisioning in opportunistic networks," Parallel and Distributed Systems, IEEE Transactions on, vol.22, no.8, pp.1267–1275, 2011.

[16] U. Sadiq, M. Kumar, A. Passarella, and M. Conti, "Modeling and simulation of service composition in opportunistic networks," Proceedings of the 14th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems, pp.159–168, ACM, 2011.

[17] S.A. Tamhane, M. Kumar, A. Passarella, and M. Conti, "Service composition in opportunistic networks," Green Computing and Communications (GreenCom), 2012 IEEE International Conference on, pp.285–292, IEEE, 2012.

[18] C. Shi, M.H. Ammar, E.W. Zegura, and M. Naik, "Computing in cirrus clouds: the challenge of intermittent connectivity," Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp.23–28, ACM, 2012.

[19] S. Bluetooth, "Specification of the bluetooth system, version 1.1," http://www.bluetooth. com, 2001.

[20] E. Guttman and J. Veizades, "Service location protocol, version 2," 1999.

[21] F. Sailhan and V. Issarny, "Scalable service discovery for manet," Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on, pp.235–244, IEEE, 2005.

[22] C. Canali, M. Renda, P. Santi, and S. Burresi, "Enabling efficient peer-to-peer

resource sharing in wireless mesh networks," Mobile Computing, IEEE Transactions on, vol.9, no.3, pp.333–347, 2010.

[23] Z. Gao, X. Yang, T. Ma, and S. Cai, "Ricffp: an efficient service discovery protocol for manets," Embedded and Ubiquitous Computing, pp.786–795, 2004.

[24] J. Garcia-Macias and D. Torres, "Service discovery in mobile ad-hoc networks: better at the network layer?," Parallel Processing, 2005. ICPP 2005 Workshops. International Conference Workshops on, pp.452–457, IEEE, 2005.

[25] A.J. Jara, P. Lopez, D. Fernandez, J.F. Castillo, M.A. Zamora, and A.F. Skarmeta, "Mobile digcovery: A global service discovery for the internet of things," Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on, pp.1325–1330, IEEE, 2013.

[26] K.L. Park, U.H. Yoon, and S.D. Kim, "Personalized service discovery in ubiquitous computing environments," Pervasive Computing, IEEE, vol.8, no.1, pp.58–65, 2009.

[27] R.C. Wang, Y.C. Chang, and R.S. Chang, "A semantic service discovery approach for ubiquitous computing," Journal of Intelligent Manufacturing, vol.20, no.3, pp.327–335, 2009.

[28] J. Liang, J. Chen, and T. Zhang, "An adaptive low-overhead resource discovery protocol for mobile ad-hoc networks," Wireless Networks, vol.17, no.2, pp.437–452, 2011.

[29] J.B. Tchakarov and N.H. Vaidya, "Efficient content location in wireless ad hoc networks," Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on, pp.74–85, IEEE, 2004.

[30] P. Vellore, P. Gillard, and R. Venkatesan, "Performance analysis of bittorrent enabled ad hoc network routing," Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly, pp.196–200, ACM, 2009.

[31] C. Ververidis and G. Polyzos, "Service discovery for mobile ad hoc networks: a survey of issues and techniques," Communications Surveys & Tutorials, IEEE, vol.10, no.3, pp.30–45, 2008.

[32] P. Vellore, P. Gillard, and R. Venkatesan, "Probability distribution of multi-

hop multipath connection in a random network," Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE, pp.1–5, IEEE, 2009.

[33] C. Gardiner, Stochastic methods, Springer-Verlag, Berlin–Heidelberg–New York–Tokyo, 1985.

[34] N. Ristanovic, J. Le Boudec, A. Chaintreau, and V. Erramilli, "Energy efficient offloading of 3g networks," Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, pp.202–211, IEEE, 2011.

[35] M. Klein, B. Konig-Ries, and P. Obreiter, "Service rings-a semantic overlay for service discovery in ad hoc networks," Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on, pp.180–185, IEEE, 2003.

[36] E. Cuervo, A. Balasubramanian, D.k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," Proceedings of the 8th international conference on Mobile systems, applications, and services, pp.49–62, ACM, 2010.

[37] B.G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," Proceedings of the sixth conference on Computer systems, pp.301–314, ACM, 2011.

[38] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," Software Engineering, IEEE Transactions on, vol.30, no.5, pp.311–327, 2004.

[39] E.M. Maximilien and M.P. Singh, "A framework and ontology for dynamic web services selection," Internet Computing, IEEE, vol.8, no.5, pp.84–93, 2004.

[40] X. Gu, K. Nahrstedt, and B. Yu, "Spidernet: An integrated peer-to-peer service composition framework," High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on, pp.110–119, IEEE, 2004.

[41] K. Yang, A. Galis, and H.H. Chen, "Qos-aware service selection algorithms for pervasive service composition in mobile wireless environments," Mobile Networks and Applications, vol.15, no.4, pp.488–501, 2010.

[42] G. Canfora, M. Di Penta, R. Esposito, and M.L. Villani, "An approach for qos-aware service composition based on genetic algorithms," Proceedings of the 2005

conference on Genetic and evolutionary computation, pp.1069–1075, ACM, 2005.

[43] Y.s. Luo, K. Yang, Q. Tang, J. Zhang, and B. Xiong, "A multi-criteria network-aware service composition algorithm in wireless environments," Computer Communications, vol.35, no.15, pp.1882–1892, 2012.

[44] J.H. Cho, A. Swami, and R. Chen, "A survey on trust management for mobile ad hoc networks," Communications Surveys & Tutorials, IEEE, vol.13, no.4, pp.562–583, 2011.

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

[46] F.L. Lewis, "Wireless sensor networks," Smart environments: technologies, protocols, and applications, pp.11–46, 2004.

[47] N. Bulusu and S. Jha, Wireless sensor networks, Artech House Boston, 2005.

[48] R. Mayrhofer, C. Holzmann, and R. Koprivec, "Friends radar: towards a private p2p location sharing platform," in Computer Aided Systems Theory–EUROCAST 2011, pp.527–535, Springer, 2012.

[49] I. Constandache, X. Bao, M. Azizyan, and R.R. Choudhury, "Did you see bob?: Human localization using mobile phones," Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking, MobiCom '10, New York, NY, USA, pp.149–160, ACM, 2010.

[50] Y.B. Ko and N.H. Vaidya, "Location-aided routing (lar) in mobile ad hoc networks," Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, pp.66–75, ACM, 1998.

[51] K. Namuduri and R. Pendse, "Analytical estimation of path duration in mobile ad hoc networks," Sensors Journal, IEEE, vol.12, no.6, pp.1828–1835, 2012.

[52] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *et al.*, Introduction to algorithms, MIT press Cambridge, 2001.

[53] Q.N. Simulator, "Scalable network technologies," Inc.[Online]. Available: www.qualnet.com.

[54] A. Ahmed, K. Yasumoto, N. SHIBATA, and T. KITANI, "Hdar: Highly dis-

tributed adaptive service replication for manets," IEICE transactions on information and systems, vol.94, no.1, pp.91–103, 2011.

[55] C.M. Huang, K.c. Lan, and C.Z. Tsai, "A survey of opportunistic networks," Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on, pp.1672–1677, IEEE, 2008.

[56] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: Social-based forwarding in delay-tolerant networks," Mobile Computing, IEEE Transactions on, vol.10, no.11, pp.1576–1589, 2011.

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

[58] M. Conti and M. Kumar, "Opportunities in opportunistic computing," Computer, vol.43, no.1, pp.42–50, 2010.

[59] M. Conti, S. Giordano, M. May, and A. Passarella, "From opportunistic networks to opportunistic computing," Communications Magazine, IEEE, vol.48, no.9, pp.126–139, 2010.

[60] R. De Renesse, M. Ghassemian, V. Friderikos, and A. Aghvami, "Qos enabled routing in mobile ad hoc networks," 3G Mobile Communication Technologies, 2004. 3G 2004. Fifth IEE International Conference on, pp.678–682, 2004.

[61] C. Sarr, C. Chaudet, G. Chelius, and I.G. Lassous, "Bandwidth estimation for ieee 802.11-based ad hoc networks," Mobile Computing, IEEE Transactions on, vol.7, no.10, pp.1228–1241, 2008.

[62] H. Dang and H. Wu, "Clustering and cluster-based routing protocol for delay-tolerant mobile networks," Wireless Communications, IEEE Transactions on, vol.9, no.6, pp.1874–1881, 2010.

[63] Q. Yuan, I. Cardei, and J. Wu, "Predict and relay: An efficient routing in disruption-tolerant networks," Proceedings of the Tenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc '09, New York, NY, USA, pp.95–104, ACM, 2009.