The Hong Kong Polytechnic University

Department of Computing

Data Dissemination and Sharing in Mobile Computing Environments

by

Xiaopeng FAN

A thesis submitted in partial fulfillment of the requirements for

the Degree of Doctor of Philosophy

June 2010

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_______________________(Signature)

Xiaopeng FAN (Name of Student)

Abstract

With the prevalence of mobile devices and recent advances in wireless communication technologies, mobile computing provides users with much easier access to the information and services available on the Internet, regardless of users' physical locations and movement behaviors. However, it is still challenging to improve the efficiency of data access in wireless mobile networks. First, wireless networks are notorious for the scarcity of communication bandwidth, energy, memory, and other resources. Thus mobile nodes cannot transmit too many packets for communication and store a mass of data locally. Second, dynamic topologies exacerbate the performance of the algorithms and protocols designed for static topologies. Third, many mobile applications, especially multimedia applications with video or audio, require transmitting much larger amount of data than before, which dramatically increase the traffic load of mobile networks.

To improve the efficiency of data access in wireless mobile networks, mobile data management is one of the most important topics in mobile computing. It mainly concerns the reliability and the efficiency of data access in mobile computing environments under the difficulties like intermittent connectivity, mobility and scarcity of resources. There are two key problems in mobile data management, namely data dissemination and data sharing. In this thesis, we investigate the challenging issues in designing algorithms and protocols to improve the performance of data dissemination and sharing in wireless mobile networks. The whole thesis is divided into three parts as follows.

In the first part of this thesis, we study data dissemination, i.e., how to disseminate a message to other mobile nodes reliably and timely. There are mainly three ways to disseminate a message to one or more nodes in a mobile network, namely unicast, multicast, and broadcast. In this thesis, we focus on how to use gossiping to provide reliable multicast, via building up mathematical models for gossiping to evaluate its reliability. We first investigate the fault tolerance problem of gossip-based reliable multicast protocols in mobile environments. We propose a generalized gossiping algorithm and develop a mathematical model based on generalized random graphs to evaluate the reliability of the gossiping protocol, answering questions like to what extent our proposed gossiping algorithm can tolerate node failures, while guaranteeing the specified message delivery ratio. We analytically derive the maximum ratio of failed nodes that can be tolerated without reducing the required degree of reliability. Next, we consider the typical hierarchical structure in an infrastructure-based wireless network and propose a generalized hierarchical gossiping algorithm, in which a multicast group is divided into smaller subgroups. We develop a mathematical model based on generalized random graphs to evaluate the reliability of hierarchical gossiping. We investigate the impact of the fanout distributions at the two levels of hierarchy on the reliability of hierarchical gossiping, and derive the critical condition for guaranteeing a gossiping message to be propagated from subgroups to the whole multicast group. Simulations in both above works have been carried out to validate the effectiveness of our analytical models in terms of the reliability of gossiping and the success of gossiping.

In the second part of this thesis, we study data sharing in mobile computing environments, i.e., how to share single or multiple data items with other mobile nodes efficiently and consistently. We consider data caching as the most important technique to share data items in mobile computing environments. We focus on the cache placement problem, i.e., how to select cache nodes to minimize total access cost in a mobile network. We deal with the cache placement problem in two cases: sharing single data item and multiple data items.

First, we consider the cache placement problem for sharing single data item in a mobile ad hoc network. We propose to achieve an optimal tradeoff between caching overhead and total access delay by properly selecting a subset of wireless nodes as cache nodes. Most of the existing cache placement algorithms use hop counts to measure the total cost of a caching system, but hop delay in wireless mobile networks varies due to the contentions among nodes and the traffic load on each link. Therefore, we propose to evaluate the per-hop delay using a metric defined on the contentions detected by a wireless node. We propose two heuristic cache placement algorithms, one centralized and another distributed. The two algorithms are named Centralized Contention-Aware Caching Algorithm (CCCA) and Distributed Contention-aware Caching Algorithm (DCCA) respectively. Both algorithms detect the variation of contentions and the change of the traffic flows to evaluate the benefit of selecting a node as a cache node. We apply a TTL-based cache consistency strategy to maintain the delta consistency among all the cache nodes. Simulation results show that the proposed algorithms achieve better performance than other alternative ones in terms of the average query delay, caching overhead, and query success ratio.

Second, we consider the cache placement problem for sharing multiple data items cooperatively in an Internet-based Mobile Ad Hoc Network (IMANET). In an IMANET, mobile nodes access a set of data items on the Internet through gateway nodes. To reduce the access delay from the Internet, mobile nodes can cooperatively cache some data items. Our objective is to let each mobile node select a subset of data items to cache cooperatively in its limited cache, such that the total access cost of all the nodes is minimized. This problem has been proved to be NP-hard. We propose a solution named Divide-and-Rule Cooperative Caching (DRCC), which divides the cache space of each node into two components: selfish and altruistic. In the selfish component, mobile nodes cache the most frequently accessed data items according to its own preference, showing the side of selfishness of a node. In the altruistic component, mobile nodes select data items in a randomized way, showing the side of altruism of a node. Given a specific access frequency distribution, we can find a near-optimal allocation solution to allocate cache sizes for the two components, aiming at minimizing the total access cost. Simulation results show that DRCC achieves much better performance than the existing best cooperative caching strategy in MANETs in terms of average query delay, caching overheads, and query success ratio. In particular, DRCC reduces caching overheads by 40% in average.

In the third part of this thesis, we apply our studies on mathematical modeling of gossiping in cooperative caching to design a novel solution named Gossip-based Cooperative Caching (GosCC) for data access in an IMANET. GosCC solves the cache placement problem, considering the sequential relation among data items. It makes use of the progress reports of mobile nodes assessing data items and the content in mobile nodes' caches to determine whether a data item should be cached at a mobile node. GosCC applies the gossiping scheme to guarantee that mobile nodes receive the accurate and timely information for making caching decisions. Simulation results show that GosCC achieves much better performance than other cooperative caching schemes, in terms of average interruption intervals and average interruption times, while sacrificing acceptable message cost to a certain degree.

Publications

Journal Papers

1. **Xiaopeng Fan**, Jiannong Cao, and Weigang Wu, Selfish or Altruistic: Divide-and-Rule Cooperative Caching in IMANETs, *IEEE Transactions on Parallel and Distributed Systems*, submitted.
2. **Xiaopeng Fan**, Jiannong Cao, and Weigang Wu, Contention-Aware Data Caching in Wireless Multi-hop Ad Hoc Networks, *Journal of Parallel and Distributed Computing, Elsevier*, submitted.
3. **Xiaopeng Fan**, Jiannong Cao, Weigang Wu, and Hui Cheng, Modeling Hierarchical Gossiping in Reliable Multicast Protocols, *Computer Communications*, Elsevier, under preparation.
4. **Xiaopeng Fan**, Jiannong Cao, Weigang Wu, and Michel Raynal, On Modeling Fault Tolerance of Gossip-Based Reliable Multicast Protocols, *Journal of Parallel and Distributed Computing, Elsevier*, under preparation.
5. Weigang Wu, Jiannong Cao, and **Xiaopeng Fan**, Design and Performance Evaluation of Overhearing-aided Data Caching in Wireless Ad Hoc Networks, *IEEE Transactions on Parallel and Distributed Systems*, submitted.
6. Hui Cheng, Jiannong Cao, and **Xiaopeng Fan**, GMZRP: Geography-aided Multicast Zone Routing Protocol in Mobile Ad Hoc Networks, *Mobile Networks and Applications, (ACM/Springer)*, Vol. 14, No. 2, pp. 165-177, 2009.

Conference Papers

7. **Xiaopeng Fan,** Jiannong Cao, and Weigang Wu, Gossip-based Cooperative Caching for Data with Sequential Relation in IMANETs, to be submitted.
8. **Xiaopeng Fan,** Jiannong Cao, and Weigang Wu, Contention-Aware Data Caching in Wireless Multi-hop Ad Hoc Networks, Proceedings

of the *sixth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'2009),* October 2009, Macau SAR, China.

9. **Xiaopeng Fan**, Jiannong Cao, Weigang Wu and Hui Cheng, Modeling Hierarchical Gossiping in Reliable Multicast Protocols, Proceedings of the *2nd International Conference on Future Generation Communication and Networking (FGCN'2008)*, December 2008, Sanya, Hainan Island, China, (Invited Paper).

10. Weigang Wu, Jiannong Cao, and **Xiaopeng Fan**, Overhearing-aided Data Caching in Wireless Ad Hoc Networks, Proceedings of the *6th IEEE ICDCS International Workshop on Wireless Ad hoc and Sensor Networks (WWASN'09)*, June, 2009, Montreal, Canada.

11. **Xiaopeng Fan**, Jiannong Cao, Weigang Wu, and Michel Raynal, On Modeling Fault Tolerance of Gossip-Based Reliable Multicast Protocols, Proceedings of the *37th International Conference on Parallel Processing (ICPP'08)*, September 2008, Portland OR, USA.

12. Hui Cheng, Jiannong Cao, and **Xiaopeng Fan**, GMZRP: Geography-aided Multicast Zone Routing Protocol in Mobile Ad Hoc Networks, Proceedings of the *5th International ICST Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine'08)*, July, 2008, Hong Kong.

13. Zhijun Wang, **Xiaopeng Fan**, and Jiannong Cao, Design a Hierarchical Cache System for Effective Loss Recovery in Reliable Multicast, Proceedings of the *7th International Symposium on Advanced Parallel Processing Technologies (APPT'2007)*, November, 2007, Guangzhou, China.

Acknowledgements

First and foremost, I want to thank Professor Jiannong Cao, my doctoral supervisor, for his encouragement and rigorous supervision of my research. His support and patience helped me overcome many difficult times during my PHD study. He unremittingly trained me to be a good researcher, including think carefully and express clearly. His vision, passion, and attitude towards the research deeply affected me. What I have learned and experienced will benefit me much in the future.

I am also deeply grateful to Dr. Weigang Wu for our fruitful collaborations and discussions. We explored ideas and wrote papers together. His kindness also helped me handle problems in my life satisfactorily. I am also very grateful to all my co-authors, Prof. Michel Raynal, Dr. Zhijun Wang, and Dr. Hui Cheng, for their insightful comments and constructive suggestions.

I own a special thank to Dr. Bin Tang, now an Assistant Professor at Wichita State University. He helped me to learn the NS2 simulation by sharing his works altruistically. My thanks also go to Yu Huang, Gang Yao, Jin Yang, Xuping Tu, Miaomiao Wang, Yan Sun, Yuan Zheng, and all other members of our research group that I cannot enumerate here. I thank them for interesting discussions and suggestions on my work.

Most importantly, I would like to thank my wife Haixia Mao. Her love, constant support, and endless patience help me enjoy the journey of my PHD study. Finally, I would like to express my heart-felt gratitude to my parents and my sisters, whose unwavering faith and confidence in me is what drives me forward to make this far, and to continue on the road ahead.

Table of Contents

List of Figures

List of Abbreviations

AF: Access Flow

BR: Best-effort Reliability

CA: Contention-Aware

CC: Cooperative Caching

DCP: Dynamic Cache Placement

DRCC: Divide-and-Rule Cooperative Caching

GosCC: Gossip-based Cooperative Caching

HF: Hedging Flow

IMANET: Internet-based Mobile Ad Hoc Networks

MH: Mobile Host

MANET: Mobile Ad Hoc Network

NH: Number of Hops

NM: Number of Messages

PCS: Personal Communication Systems

PR: Probabilistic Reliability

QoS: Quality of Services

RF: Reply Flow

SR: Strong Reliability

SRCP: Sequential Relation Cache Placement

UF: Update Flow

Chapter 1. Introduction

The main objective of this research is to investigate the issues and design novel algorithms, techniques, and mathematical models for efficient data dissemination and sharing in wireless mobile networks. In this thesis, a wireless mobile network refers to a Mobile Ad hoc NETwork (MANET) or its Internet extensions, i.e., an Internet-based Mobile Ad hoc NETwork (IMANET). With respect to the topic of data dissemination and sharing, we mainly discuss gossiping and data caching as two corresponding techniques. This chapter provides an introduction to our research, discussing the characteristics of mobile computing and the new challenges in developing applications for wireless mobile networks. The discussion serves as the motivation of our works. We also summarize the contributions and outline the organization of the thesis.

1.1. Motivation

Recent advances in wireless communication and portable devices have led to the rapid development of mobile computing technologies and applications. In the last decade, mobile computing [MM03] has emerged as a new computing paradigm. However, developing mobile computing applications faces new challenges: 1) mobile elements are resource-poor relative to static elements, such as mobile phones; 2) mobility is inherently hazardous. For example, portable computers are more vulnerable to loss or damage, even may be stolen much easier than workstation in office; 3) mobile connectivity is highly variable in performance and reliability and the quality of communication is easy to be affected by the surrounding environments; and 4) mobile devices rely on a finite energy source. Concerns of power consumption must span many levels of hardware and software to be fully effective.

Mobile data management [MM03, IB93] is one of the most important topics in mobile computing. This topic mainly concerns reliable and efficient data access in wireless mobile networks, targeting at overcoming the difficulties like intermittent connectivity, mobility and scarcity of resources. There are two key problems in mobile data management, namely “data dissemination” and “data sharing”. The first problem is about **how to disseminate a message to a set of mobile hosts reliably and timely**. The second problem is about **how to share one or multiple data items with a set of mobile hosts efficiently and consistently**.

It remains a challenging task to improve the efficiency of data access in MANETs. Firstly, wireless networks are notorious for the scarcity of communication bandwidth and other resources. Thus mobile nodes cannot transmit too many packets for communications. Moreover, due to the limited storage, mobile nodes cannot cache all the frequently accessed data items locally. Secondly, mobile nodes move freely and disconnections may occur frequently. Once a network partitions, mobile nodes in one partition cannot access the data items cached by nodes in another partition. Data accessibility in MANETs is then much lower than that in the conventional fixed networks. Thirdly, many mobile applications, especially multimedia applications with video or audio, require transmitting much larger amount of data than before, which dramatically increase the traffic load of wireless mobile networks.

1.2. Contributions of the Thesis

The contributions of this thesis mainly lie on designing novel algorithms, techniques, and mathematical models for data dissemination and sharing in wireless mobile networks. As illustrated in Figure 1.1, our contributions mainly focus on two topics: 1) with respect to data dissemination, we focus on gossiping, in which each node randomly selects one or a few of nodes to disseminate a message. We build up mathematical models for gossiping to evaluate the reliability of gossip-based reliable multicast; and 2) with respect to data sharing, we focus on data caching, in which data queries can be served by a set of cache nodes that are selected to hold some copies of data items. We address the cache placement problem in data caching with the objective of reducing the total access cost in MANETs.

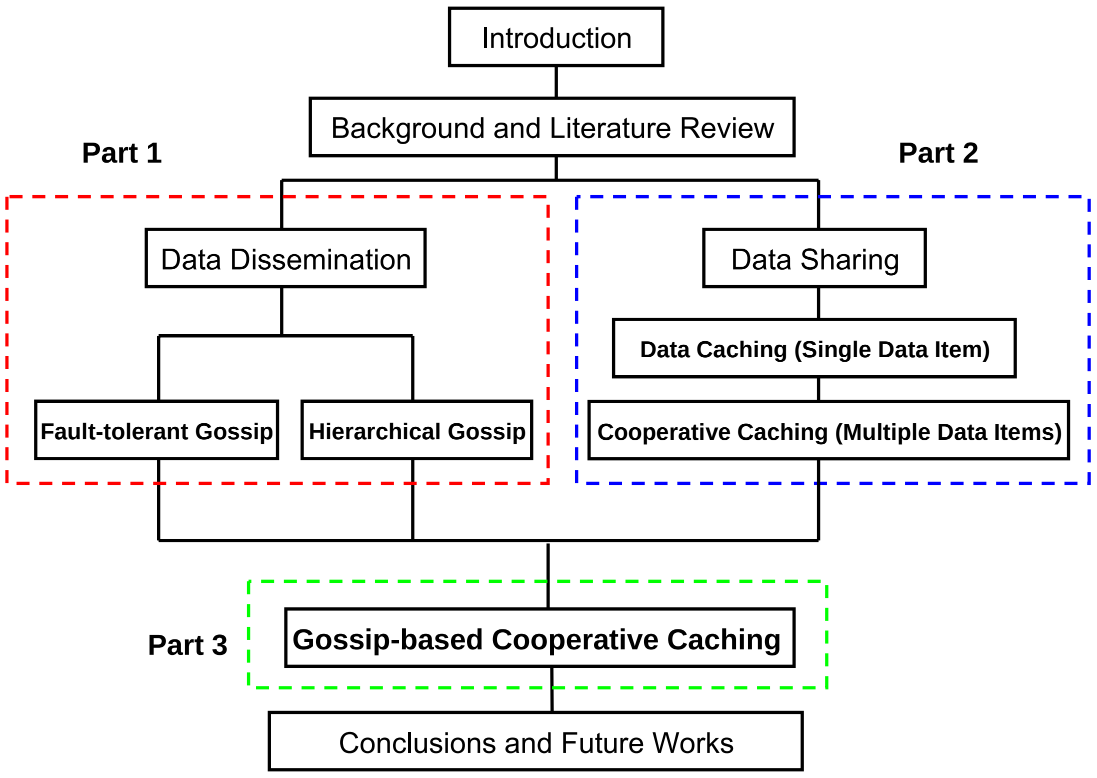

Introduction
Background and Literature Review
Part 1
Part 2
Data Dissemination
Data Sharing
Data Caching (Single Data Item)
Fault-tolerant Gossip
Hierarchical Gossip
Cooperative Caching (Multiple Data Items)
Part 3
Gossip-based Cooperative Caching
Conclusions and Future Works

Figure 1.1: An outline of the contributions in this thesis

1.2.1. Contributions in Data Dissemination

Gossiping [BHO$^+$99, KMG03] is one of the most important techniques to provide probabilistic reliability in reliable multicast. Gossip-based multicast protocols rely on a peer-to-peer interaction model for multicasting a message. They are scalable since the loads are distributed among all participating nodes. In a wireless mobile network, there are new challenging issues for designing gossip-based reliable multicast protocols. The new characteristics of wireless mobile networks and devices need be considered in the design. For example, we should consider the impacts of the weak capacity of mobile devices on the fault-tolerant property of gossip-based reliable multicast, while keeping its probabilistic reliability. As another example, we can take advantage of the inherent hierarchical structure of some wireless mobile networks to reduce the message cost of gossiping.

In this thesis, we propose two generalized gossiping algorithms, in which the fanout distribution of all the nodes can be an arbitrary distribution. The fanout of a node means the number of gossiping targets randomly selected from the network. Most existing algorithms are designed with a specific fanout distribution. With our generalized algorithms, we establish the corresponding mathematical models to evaluate the reliability of gossiping. The details are described in the following subsections.

1.2.1.1. Fault-tolerant Gossip Model

We aim to investigate the fault-tolerant property of gossip-based reliable multicast in a wireless mobile network. Mobile devices are prone to fail, due to either their weak capacity or their mobility. Moreover, wireless communication is also subject to intermittent disconnections. Firstly, we propose a generalized gossiping algorithm in which the fanout distribution of all the nodes can be an arbitrary probabilistic distribution. Then, with the generalized algorithm, we develop a mathematical model based on generalized random graphs to evaluate the reliability of gossiping, i.e., to what extent gossip-based protocols can tolerate node failures, yet guarantee the specified message delivery. We analytically derive the maximum ratio of failed nodes that can be tolerated without reducing the required degree of reliability. We also investigate the impact of the fanout distributions and the nonfailed member ratio, on the reliability of gossiping. Simulations have been carried out to validate the effectiveness of our analytic model in terms of the reliability of gossiping and the success of gossiping. The results obtained can be used to guide the design of fault tolerant gossip-based protocols.

1.2.1.2. Hierarchical Gossip Model

In some wireless networks, such as cellular networks or WLANs, the structures of such networks are inherently hierarchical. We reduce the message cost of gossiping by dividing a multicast group into smaller subgroups and using a small set of nodes in each subgroup to gossip among the subgroups. We propose a generalized hierarchical gossiping algorithm and develop a mathematical model based on generalized random graphs to evaluate the reliability of hierarchical gossiping. Using our mathematical model, we investigate the impact of the fanout distributions at the two levels of hierarchy on the reliability of hierarchical gossiping. We also discover the critical condition for guaranteeing the gossiping messages to be propagated from local subgroups to the whole group. Simulations have been carried out to validate the effectiveness of our analytic model in terms of the reliability of gossiping and the success of gossiping.

1.2.2. Contributions in Data Caching

Data caching [BO00, QPV01, CS02, KD02, KRW03, KRS00, LWY93, LGI+99, MHV+97, QPV01, RS02] is one of the most attractive techniques for sharing data items in wireless mobile networks, so as to reduce access delay and redundant data transmission. The data source transfers some copies of a data item to some client nodes, called cache nodes. Thus other client nodes can access the copies of the data item from these cache nodes instead of the data source so that total access cost can be reduced. Cooperative caching [YC06, TGD06, CLC07, ASM+08, DGV09] can be considered as an extension of data caching, which further explore the potential of data caching. A typical cooperative caching strategy allows the sharing and coordination of cached data items among multiple nodes.

There are mainly three sub-problems in data caching [W99, TCO01], *cache placement* [BI94], *cache replacement* [AAF$^+$95], and *cache consistency* [C03, HCW$^+$07, CZX$^+$07]. The cache placement problem is about how to select cache nodes to minimize total access cost. In this thesis, we mainly focus on the cache placement problem. In two of our works, we also address the cache replacement problem and the cache consistency problem respectively.

In wireless mobile networks, we need to address some challenging issues when considering the cache placement problem. For example, due to the shared nature of wireless medium, nodes may contend with each other to access data items. We should consider the impact of these contentions on the cost of data access. Moreover, if there is a set of data items to be accessed, we need consider what data items should be selected for caching under the constraint of limited cache sizes. Additionally, the relations among the data items can help us to improve the performance of data access.

In this thesis, we deal with the cache placement problem in two cases: sharing single data item and multiple data items. There are three contributions on this topic. Firstly, we propose the Contention-aware Caching Algorithm to share single data item, considering the impact of wireless contentions. Secondly, we propose the Divide-and-Rule Cooperative Caching to share multiple data items under the constraint of limited cache sizes. Thirdly, we propose Gossip-based Cooperative Caching to share multiple data items with sequential relation, under the constraint of limited cache sizes. The details are described in the following subsections.

1.2.2.1. Contention-aware Data Caching

We consider the cache placement problem in wireless multi-hop networks, investigating how to achieve an optimal tradeoff between caching overheads and total access delay by properly selecting a subset of wireless nodes as cache nodes when the network topology changes. We assume a data source updates a single data item to be accessed by other client nodes. Most of the existing cache placement algorithms use hop counts to measure the total cost of a caching system, but hop delay in wireless networks maybe varies due to the contentions among nodes and the traffic load on each link. Therefore, we evaluate the per-hop delay according to the contentions detected by a wireless node.

We propose two heuristic cache placement algorithms, named Centralized Contention-aware Caching Algorithm (CCCA) and Distributed Contention-aware Caching Algorithm (DCCA). Both can detect the variation of contentions and the change of the traffic flows in order to evaluate the benefit of selecting a node as a cache node. We also apply a TTL-based cache consistency strategy to maintain the delta consistency among all the cache nodes. Simulation results show that the proposed algorithms achieve better performance than other alternative ones in terms of average query delay, caching overheads, and query success ratio.

1.2.2.2. Divide-and-Rule Cooperative Caching

We consider the cache placement problem for sharing multiple data items cooperatively in an IMANET. Mobile nodes access a set of data items on the Internet through gateway nodes. To reduce the access delay from the Internet, mobile nodes can cooperatively cache some data items. Our objective is to let each mobile node select a subset of data items to cache in its limited cache such that the total access cost of all the nodes is minimized. This problem is NP-hard [BR01].

We propose a novel solution named Divide-and-Rule Cooperative Caching (DRCC), which divides the cache space of each node into two components: selfish and altruistic. In the selfish component, mobile nodes cache the most frequently accessed data items according to its own preference, showing the side of selfishness of a node. In the altruistic component, mobile nodes select data items in a randomized way, showing the side of altruism of a node. Given a specific access frequency distribution, we can find a near-optimal allocation solution to allocate cache sizes for the two components, aiming at minimizing the total access cost. Simulation results show that DRCC achieves much better performance than the existing best cooperative caching strategy in MANETs in terms of average query delay, caching overheads, and query success ratio. In particular, DRCC reduces caching overheads by 40% in average.

1.2.2.3. Gossip-based Cooperative Caching

We consider the relation among data items in designing cooperative caching in IMANETs. We assume each mobile node accesses all the data items in a sequential order. Most of the existing works does not consider the inherent relations among these data items. With more demands on sharing a video or other media contents, the relations among data segments (items) becomes much more important to improve the efficiency of data access. We present a novel solution named Gossip-based Cooperative Caching (GosCC) to address the cache placement problem, considering the sequential relation among data items. The current ID of the data item accessed by a mobile node is stored into a progress report. GosCC makes use of the information about the progress reports of mobile nodes and the content in mobile nodes' caches, in order to determine whether a data item should be cached at a mobile node. To obtain the aforementioned information reliably and in time, GosCC apply a gossip-based scheme to guarantee that mobile nodes receive the accurate and timely information for making caching decisions. Simulation results show that GosCC achieves better performance than BDC [TGD06] in terms of average interruption intervals and average interruption times, while sacrificing message cost to a certain degree.

1.3. Outline of the Thesis

The structure of this thesis is described in Figure 1.1. Chapter 1 is our introduction to this thesis. Chapter 2 briefly presents the literature review of the relevant topics and provides some necessary background knowledge for works reported in this thesis. Finally, we conclude the thesis with discussion on directions of our future works in Chapter 8. The main body of this thesis is divided into three parts from Chapter 3 to Chapter 7. The details are presented as follows.

In the first part, we mainly discuss mathematical models for gossiping. This part is composed of two chapters. In Chapter 3, we propose a generalized gossiping algorithm and a fault-tolerance gossiping model to describe the fault tolerant property of the proposed algorithm. In Chapter 4, we propose a generalized hierarchical gossiping algorithm and develop a mathematical model based on generalized random graphs to evaluate the reliability of hierarchical gossiping.

In the second part, we mainly discuss how to deal with the cache placement problem in wireless mobile networks. This part consists of two chapters. In Chapter 5, we investigate the cache placement problem for sharing one data item in wireless mobile networks. We define the cache placement problem on a dynamic network topology as Dynamic Cache Placement (DCP). We take the contentions among wireless nodes into consideration and present two heuristic algorithms named Centralized Contention-aware Caching Algorithm (CCCA) and its distributed version DCCA.

In Chapter 6, we investigate the cache placement problem for sharing multiple data items in cooperative caching. We take the distribution of data access frequencies into consideration and present an algorithm named Divide-and-Rule Cooperative Caching (DRCC) which improves the performance of cooperative caching by providing a tradeoff between caching data items for self and for others.

In the third part, we try to build up a cooperative caching system by applying gossiping to disseminate the information about caching. This part only includes one chapter. In Chapter 7, we consider the relation among data items in cooperative caching for sharing multiple data items in IMANETs. We present a novel solution named Gossip-based Cooperative Caching (GosCC) to address the cache placement problem, considering the sequential relation among data items.

Chapter 2. Background and Literature Review

In this chapter, we provide a literature review and some background knowledge related to the research in this thesis. The organization of this chapter is as follows. Firstly, Section 2.1 presents an overview of wireless mobile networks. Section 2.2 presents an overview of gossiping with its existing mathematical models. Section 2.3 describes the related works on the topic of data caching. Finally, Section 2.4 makes a brief introduction to the mathematical tool used in this thesis, i.e., the generalized random graph theory.

2.1. Wireless Mobile Networks

In this thesis, a wireless mobile network is a network that consists of Mobile Hosts (MHs), which communicate with each other by using wireless communication. Sometime, we use “wireless network” or “mobile network” to stand for a wireless mobile network in some chapters. There are many different wireless mobile networks proposed, such as cellular networks, Wireless LANs, and Wireless Mesh networks, etc [IK96, P97, V99, FZ94]. The architectures of all the wireless mobile networks can be classified into two categories: infrastructure-based networks and ad hoc networks [AM98, M01, M99]. An infrastructure-based network generally consists of a large number of MHs and relatively fewer but more powerful Mobile Support Stations (MSSs) that are connected by a wired or wireless backbone network. Messages among mobile users are normally relayed by one or more MSSs.

In this thesis, we mainly focus on MANETs, and its extension IMANETs [CMC99]. A MANET is an infrastructure-less network that consists of a collection of autonomous MHs communicating with each other through wireless channels. The signal coverage range and the distance between two hosts determine whether they are directly connected. Each host is a router and the communication between two hosts can be multiple hops. Both link and host failures may frequently occur. The topology of a MANET can dynamically change due to the mobility of MHs and link or host failures. MANETs provides an attractive solution for networking in the situation where network infrastructure is not available, such as battlefield, rescue operations and so on.

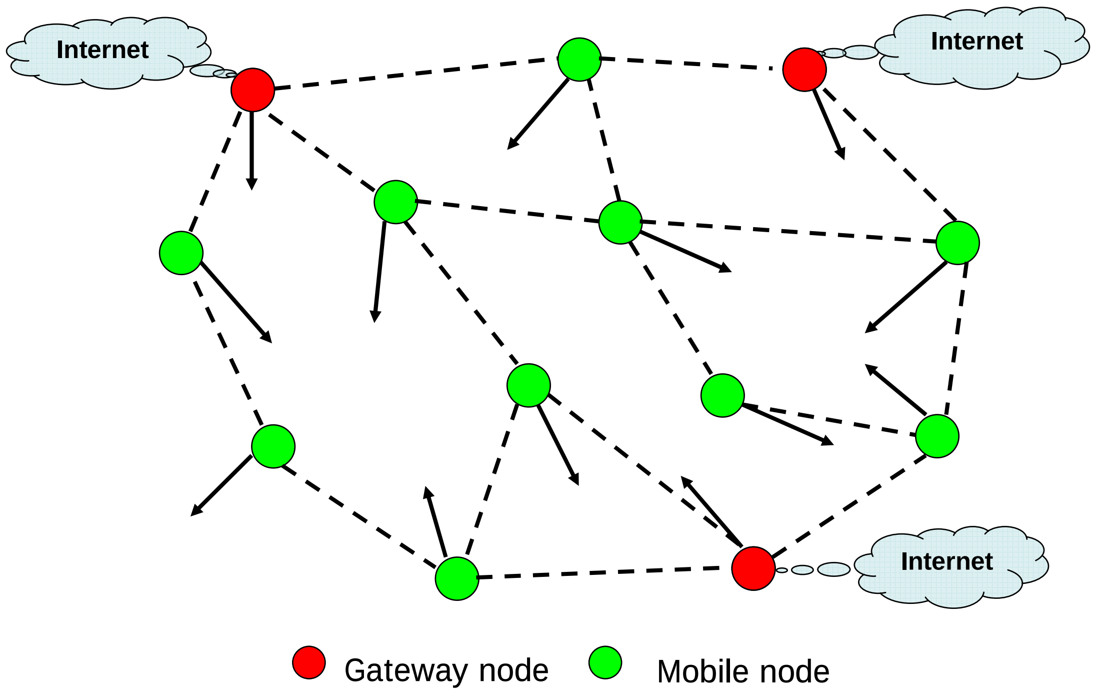


Figure 2.1: An Internet-based Mobile Ad hoc NETwork (IMANET)

In an IMANET as Figure 2.1 shows, mobile users access the Internet through some gateway nodes by multi-hop communications, without requiring any infrastructure in the users' proximity. In fact, an IMANET combines a MANET with the Internet to provide universal information accessibility. With the growth of mobile devices, an IMANET is well suited for many Internet applications in terms of facilitating flexible accessibility and information availability. The contents on the Internet can be provided to mobile users much easier.

2.2. Gossiping for Data Dissemination

2.2.1. Gossiping

In this thesis, data dissemination means the procedure that disseminates a message from a source to a set of destinations. Normally, data dissemination can be addressed by reliable multicast or broadcast [EGB$^+$03, DGH$^+$87, GT92, RMM98, BHO$^+$99, WFC07, FanCW$^+$08].

Existing multicast protocols guarantee one of the three types of reliability: Strong Reliability (SR), Best-effort Reliability (BR) and Probabilistic Reliability (PR) [BHO$^+$99]. Compared with strong reliability and best-effort reliability, probabilistic reliability does not always guarantee atomicity but can provide message delivery guarantee with some required probability. For example, Bimodal Multicast [BHO$^+$99] provides a bimodal delivery guarantee which changes the traditional “all or nothing” guarantee to the “almost all or almost none” guarantee.

Gossiping [BHO$^+$99, KMG03] is one of the most important techniques to provide probabilistic reliability in reliable multicast. Gossip-based multicast protocols rely on a peer-to-peer interaction model for multicasting a message, and they are scalable since the loads are distributed among all participating nodes. Redundant messages are used to achieve reliability and fault tolerance. A few pioneering works on gossiping have been done for both wired and wireless networks. In wired networks, many works can be found on data dissemination [KMG03], consistency management in replicated databases [DGH$^+$87], and failure detection [RMH98].

As we all know, much work has been done in the topic of reliable multicast in MANETs [CRB01, LEP03]. In wireless networks, gossip-based protocols have been proposed for multicast in mobile ad hoc networks (MANETs). A seminal approach is the Anonymous Gossip (AG) protocol [CRB01], which is a descendant of the pbcast

[BHO+99] protocol. The Route Driven Gossip (RDG) [LEP03] protocol uses a pure gossip scheme, by which messages, negative acknowledgments, and membership information is gossiped uniformly without requiring an underlying multicast primitive.

2.2.2. Mathematical Models of Gossiping

In this subsection, we briefly review the previous work on developing mathematical models of gossiping. Three different modelling approaches have been used, including the recurrence model, the epidemic model, and the random graph model.

In pbcast [BHO+99], the analysis work shows how to calculate the bimodal delivery distribution for a given networking setting. The authors derive a recurrence relationship between the successive gossiping rounds of the protocol. However, due to its complexity, the model cannot determine the accurate value of the reliability of gossiping. Analysis based on this model only calculates an upper bound in round t on the probability that s_{t+1} nodes will receive the gossip message in the next round $t+1$. It does not show how to find a proper number of rounds required in gossiping.

The second approach is based on the epidemic model [JLW+07]. Two mechanisms, Local Retransmission and Gossiping (LRG), are combined to provide the high reliability of data delivery. Since the gossiping process is similar to the spreading of epidemic diseases, the authors use the so-called SI model in epidemiology to analyze LRG. In this model, the balance equations are developed to describe the process of spreading messages among Group Cluster Heads (GCHs). However, this model did not take the message losses and node failures into consideration.

More recently, the random graph theory [B01] has been used to model gossiping. The seminal work is the model proposed by Microsoft [KMG03], which aims at establishing the relationship between the success of the gossiping protocols and the key gossip parameters, including the fanout and the failure rate. The model considers the presence of arc {*x, y*} in a random graph $\varsigma(n, p_n)$ as saying that *x* gossips message to *y*. The success of gossiping means the existence of a directed path from the source node *s* to every other node in the random graph. The probability of the success of gossiping is denoted by $\pi(p_n, n)$. It is proved that the limit of $\pi(p_n, n)$ is $e^{-e^{-c}}$ if p_n is equal to *(log(n)+c+o(1)/n*, where *c* is a constant. If the proportion of the failed nodes is *ε*, that is, *n'=(1-ε)*n*, gossiping succeeds with the probability $e^{-e^{-c}}$ if $p_n = [\log n' + c + o(1)] / n'$. Although the success of gossiping, i.e., all of the group members receive the message, is important, we still need to know the probability that one node receives the message during gossiping if we cannot guarantee such a strong requirement as the success of gossiping in practice. Both of them are considered in detail by our mathematic model.

2.3. Data Caching for Data Sharing

2.3.1. Data Caching

Data caching [BO00, QPV01, CS02, KD02, KRW03, KRS00, LWY93, LGI+99, MHV+97, QPV01, RS02] is one of the most attractive techniques that can share one or more data items in wireless multi-hop ad hoc networks [NSC03, H01, TG07, TGD06, FCW09, WCF09]. The data source transfers some copies of a data item to some client nodes, called cache nodes. Thus other client nodes can access the copies of the data item from these cache nodes instead of the data source. By this way, total access delay is decreased because of the service provided by cache nodes. However, the data source is responsible for updating the copies at cache nodes periodically or aperiodically. This brings more traffic cost. Obviously, there is a trade-off between average access delay and overall traffic cost.

There are mainly three sub-problems in data caching [W99, TCO01], including cache placement [BI94], cache replacement [AAF+95], and cache consistency [C03, HCW+07, CZX+07]. The cache placement problem is how to select cache nodes to minimize total access cost. The cache replacement problem is how to replace data items in the cache when the cache is full. The cache consistency problem is how to maintain cache consistency, i.e., consistency among the source data owned by the data source and the cache copies held by a collection of cache nodes.

In this thesis, we mainly focus on the cache placement problem. We study two cases on the cache placement problem as follows.

Firstly, if a data source aims to share **single data item** with other mobile nodes and the data item is updated periodically or aperiodically, the cache placement problem becomes how to select cache nodes to obtain the optimal trade-off between total access delay and overall caching overhead.

Secondly, if a set of data sources aims to share **multiple data items** with other mobile nodes under the constraints of cache sizes, the cache placement problem becomes what data items should be cached in which mobile nodes under the constraint of cache space, so that total access delay is minimized. Normally, we use cooperative caching to address this problem. We will review the existing works in cooperative caching in Section 2.2.3.

2.3.2. Cache Placement for Sharing Single Data Item

In this subsection, we focus on the cache placement for sharing single data item in a wireless multi-hop ad hoc network. The key problem of determining the optimal cache placement in an arbitrary network topology has the similarity to two problems in graph theory viz. the facility location problem and the *k*-median problem [SK02]. As we all known, the two problems have been proved to be NP-Hard. In the facility location problem, setting up a cache at a node incurs a certain fixed cost, and the goal is to minimize the sum of total access cost and the fixed costs of all caches. On the other hand, the *k*-median problem is to minimize the total access cost under the constraint of the number of cache nodes, i.e., that at most *k* nodes can be selected as cache nodes. A number of constant factor approximation algorithms have been developed for each of the two problems [CG99], under the assumption of triangular inequality of edge costs. Without the triangular inequality assumption, either problem is as hard as approximating the set cover [JV01], and thus can be approximated better than $O(log|V|)$ unless $P=NP$.

In [LGI+99], the authors address the problem of proxy placement and employ dynamic programming to determine the optimal placement. They only consider the case of tree topology. In the context of wireless network, Nuggehailli et al. [NSC03] formulate the cache placement problem in ad hoc wireless networks as a special case of the connected facility location problem [SK02], named as the rent-or-buy problem. An existing facility is given, along with a set of locations at which further facilities can be built. Every location is associated with a service demand, denoted by p_k, which must be served by one facility. The cost of serving k by using facility j is equal to p_k*c_{kj}, where c_{kj} is the cost of connecting k to j. The fixed cost of opening a facility at any location is zero. Besides selecting the sites to build the facilities, all of the facilities should be connected by using a Steiner Tree with the given facility as root. To the best of our knowledge, the best solution for this problem is 2.92-approximation algorithm [EGR+08]. In [NSC03], the authors propose an algorithm named POACH, which is a 6-approximation-ratio algorithm for ad hoc wireless networks. However, POACH is designed for static topology such that it does not work properly in a distributed way and in a mobile ad hoc network. Moreover, the performance of POACH is evaluated by the numerical results, which cannot reflect the reality of wireless ad hoc networks. Although the author mentioned that POACH can be implemented in a distributed fashion, there is still no any real implementation in the original work and the extended version ECHO [NSC+06]. Another key point is that ECHO requires global topological information and disseminating such information in a low-cost manner. It is not a good choice in a mobile ad hoc network.

2.3.3. Cooperative Caching

When we discuss how to share multiple data items, mobile nodes may cache data items in a cooperative way in order to improve the efficiency of data access in wireless mobile networks. Cooperative caching [YC06, TGD06, CLC07, ASM$^+$08, DGV09] has been proved as an efficient way to improve the performance of data access. A typical strategy in cooperative caching works as follows. Data sources transfer some data copies to some nodes called *cache nodes*. Other nodes can access the data item from cache nodes instead of data sources. Consequently, access delay can be decreased because of the service provided by these cache nodes. In this way, each cache node not only serves the requests from its own but also the data requests from other nodes. Each cache node caches data items not only on behalf of its won needs but also on the behalf of others.

The major problem in cooperative caching is the cache placement problem [H01, YC04, YC06, TGD06, TG07, NSC03, TCC07], i.e., what data items should be cached at which mobile nodes under the constraint of cache space, so that total access cost is minimized. The existing solutions on this topic can be categorized into three categories, including selfish schemes, cooperative schemes and global schemes.

Firstly, selfish schemes make mobile nodes cache data items only by their own preferences. Hara [H01] proposes SAF (Static Access Frequency) scheme that makes each node cache the items most frequently accessed by self. Yin and Cao [YC04] proposes Greedy-S (Greedy Scheme) that considers the impact of the access frequencies and the sizes of data items on the caching decision. Data items with higher access frequencies and smaller sizes are preferred to be cached. Under such schemes, the performance is even worse when there are fewer client nodes and access frequencies are uniform for all the nodes.

Secondly, cooperative schemes consider the requirements from both self and other nodes. Hara [H01] presents Dynamic Access Frequency and Neighborhood (DAFN) that eliminates the replica duplication among neighboring mobile hosts in order to improve SAF. Dynamic Connectivity based Grouping (DCG) is the third cooperative caching schemes proposed in [YC04]. DCG aims at sharing replicas in larger group of mobile hosts than DAFN that shares replicas among neighboring nodes. However, it is not easy to find stable nodes to act as “central nodes” in order to collect neighboring information and determine caching placements, when there are frequent failures and movements in ad hoc networks. This is the same problem as the one in One-To-One Optimization (OTOO) [YC04] and Reliable Neighbor Scheme (RN) [YC04]. Yin and Cao [YC06] proposes three distributed caching schemes, viz. CacheData, CachePath, and HybridCache. CacheData caches the passing-by data items at each node. CachePath caches the path to the nearest cache of the passing-by data item. HybridCache caches the data item if its size is small enough, else caches the paths to the data item. These three schemes are simple but much efficient schemes. However, the only consideration is that CachePath depends on the modification of routing protocols and sometimes the node that modified the route should reroute the request to the original data center.

Thirdly, global schemes consider the impact of the benefit to the whole system when a node intends to evaluate the result of its caching decision. Tang et al. present Benefit-based Data Caching (BDC) scheme [TGD06] in order to maximize the benefit (i.e., the reduction in total access cost) instead of minimizing the total access cost. To the best of our knowledge, it is the best solution that presents approximation algorithms for the general cache placement problem with multiple data items under memory constraints. Thus, we also use BDC as our counterpart in simulations.

However, BDC has similar but better performance compared with other schemes in mobile environments and it cannot determine the value of the benefit threshold in the distribution version. Moreover, data servers periodically broadcast to the entire network the latest cache list, which results in much message cost.

2.4. Generalized Random Graphs

The generalized random graph [NSW01] is the most important mathematical tool used in this thesis. We model gossiping by using the generalized random graph theory in Chapter 3 and Chapter 4. Additionally, we also use the bipartite random graphs to model the relation between mobile nodes and data items in Chapter 6. With respect to GosCC in Chapter 7, we also use our mathematical model for gossiping to analyze the performance of GosCC. Therefore, we must make a brief introduction to random graph and the generalized random graph theory in this subsection.

Although the theory of random graphs is one of the youngest branches of graph theory, it is second to none in importance. In the 1940s and 1950s, there are some sporadic papers written by Erdos, in which Erdos used random methods to show the existence of graphs with seemingly contradictory properties [B01]. The great discovery of Erdos was that we can use probabilistic methods to demonstrate the existence of the designed graphs without actually constructing them. This is a revolutionary work in both graph theory and combinatorics.

The systematic study of random graphs was started by Erdos and Renyi in 1959. They laid the foundation of a rich theory of random graphs, proving many of the fundamental results in a series of papers. In this thesis, we are mainly interested in the phenomena of phase transition, i.e., while a random graph with n nodes and a certain number of edges is unlikely to have the property P at hand, a random graph with a few more edges is very likely to have the property P. This term “phase transition” is borrowed from physics. The most dramatic example of a phase transition discovered by Erdos and Renyi is the appearance of the giant component during the process of increasing the probability of connecting two nodes. The giant component is the biggest connected subgraph formed after a phase transition happens. It has a size of order at least $n^{2/3}$, while the sizes of other components are of order at most $n^{2/3}/2$.

In this thesis, we mainly use the second probabilistic space in the random graph theory [B01], i.e., if the total number of nodes is *n*, the space $\varsigma(n,p)$ is defined for $0 \le p \le 1$. To get a random element of this space, we select the edges independently with the probability *p*. In another way, the ground set of $\varsigma(n,p)$ is the set of all 2^N graphs ($N=C(n,2)$), and the probability of a graph *H* with *m* edges in this space is $p^m(1-p)^{N-m}$.

Each edge on a random graph appears independently with the probability *p*. If a random graph has *n* nodes and there are average *z* edges for each node, we know $p=z/(n-1)$. Let p_k be the ratio of the nodes with *k* degrees in $\varsigma(n,p)$. We can obtain the following equation.

$$p_k = \binom{n}{k} p^k (1-p)^{n-k} \approx \frac{z^k e^{-z}}{k!} \tag{2.1}$$

Therefore, p_k follows the Poisson distribution *Po(z)*. This is the traditional ER model [B01] in the random graph theory.

With the development of the random graph theory, the node degree distribution can be arbitrary distribution [NSW01], other than the Poisson distribution. Newman et al [NSW01] proposed a new way to model the random graphs with arbitrary degree distributions by generating functions. This is called the generalized random graph theory.

On unipartite undirected graphs with arbitrary probability distribution of the degrees of their nodes, the new method provides a formalism for calculating a variety of quantities, both local and global. In all respects other than their degree distribution of their nodes, these graphs are assumed to be entirely random. This means that the degrees of all nodes are independent identically-distributed random integers drawn from a specified distribution. For a given choice of these degrees, also called "degree sequence," the graph is chosen uniformly at random from the set of all graphs with that degree sequence. All properties calculated in this thesis are average over the ensemble of graphs generated in this way. This also means the results by the new way are statistical.

(1) Generating functions

The new method to describe random graphs with arbitrary degree distributions is based on the generating functions, among which the most fundamental one is he generating function $G_0(x)$ for the probability distribution of node degree k. We can obtain the following equation.

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k \qquad \textbf{(2.2)}$$

where p_k is the probability that a randomly chosen node on the graph has degree k. The distribution p_k is assumed correctly normalized, so that $G_0(1)=1$. Because the probability distribution is normalized and positive definite, $G_0(x)$ is also absolutely convergent for all $|x|\leq 1$, and hence has no singularities in this region. All the calculations of this generalize random graph model will be confined to the region $|x|\leq 1$.

(2) Derivatives

The probability p_k is given by the k-th derivatives of G_0 by the following equation.

$$p_k = \frac{1}{k!}\frac{d^k G_0}{dx^k}\Big|_{x=0} \tag{2.3}$$

Thus the one function *$G_0(x)$* encapsulates all the information contained in the discrete probability distribution p_k. Thus, we say the function *$G_0(x)$* "generates" the probability distribution p_k.

(3) Moments

The average over the probability distribution generated by a generating function, for example, the average degree *z* of a node in the case of *$G_0(x)$*, can be obtained by the following equation.

$$z = \langle k \rangle = \sum_k k p_k = G_0'(1) \tag{2.4}$$

This means if we can calculate a generating function, we can also calculate the mean of the probability distribution which it generates. Higher moments of the distribution can be calculated from higher derivatives as follows.

$$\langle k^n \rangle = \sum_k k^n p_k = \left[\left(x\frac{d}{dx}\right)^n G_0(x)\right]_{x=1} \tag{2.5}$$

(4) Powers

If the distribution of a property *P* of an object is generated by a given generating function, the distribution of the total of *P* summed over *m* independent realizations of the object is generated by the *m*-th power of that generating function. For example, if we select *m* nodes at random from a random graph, the distribution of the sum of the degrees of these nodes is generated by *$[G_0(x)]^m$*. We take the simple case of two nodes. The square *$[G_0(x)]^2$* of the generating function for a single node can be expanded as

$$[G_0(x)]^2 = \left[\sum_k p_k x^k\right]^2 = \sum_{jk} p_j p_k x^{j+k} \tag{2.6}$$

The coefficient of the power of x^n in this expression is the sum of all powers p_jp_k such that $j+k=n$. It shows that this property extends also to all higher powers of the generating function.

There is another important quantity is the distribution of the degree of the nodes that we arrive at a node with the probability proportional to the degree of the node, and the node therefore has a probability distribution of degree proportional to kp_k. The correct normalized distribution is generated by the following equation.

$$\frac{\sum_k kp_k x^k}{\sum_k kp_k} = x\frac{G_0'(x)}{G_0'(1)} \tag{2.7}$$

If we start at a randomly selected node and follow each of the edges of this node to reach the k nearest neighbours, each of the nodes arrived have the distribution of remaining outgoing edges generated by this function, less one power of x, to allow for the edge that we arrived along. Thus the distribution of outgoing edges is generated by the following function:

$$G_1(x) = \frac{G_0'(x)}{G_0'(1)} = \frac{1}{z}G_0'(x) \tag{2.8}$$

where z is the average node degree.

In Chapter 6, we refer to the bipartite random graphs. We use it to model the relation between mobile nodes and data items. There are many other examples, such as the collaboration graphs of scientists, company directors, and movie actors. In [NSW01], the authors also provide a similar model to investigate the theory of bipartite random graphs, by taking moving actors as an example. Consider a bipartite random graph of M movies and N actors, in which each actor has appeared in an average of μ movies and each actor has appeared in an average of v actors. The relation among the four parameters can be explained in the following equation.

$$\frac{M}{\mu} = \frac{N}{\upsilon} \tag{2.9}$$

Let p_j be the probability distribution of the degree of actors, i.e., the number of movies that an actor joins. Let q_k be the probability of the degrees of movies, i.e., the number of actors in a movie. Therefore, the tow generating function for the two probabilities are describes as follows:

$$f_0(x) = \sum_j p_j x_j \quad g_0(x) = \sum_k q_k x_k \tag{2.10}$$

All the important results are derived by these two generating functions. More details can be found in [NSW01].

Chapter 3. Modeling Fault-Tolerance for Gossip-based Reliable Multicast

In this chapter, we introduce the proposed generalized gossiping algorithm with its fault-tolerant mathematical model. This chapter is organized as follows: Section 3.1 is the overview to this work. Section 3.2 provides the network model, the generalized gossiping algorithm, and the preliminaries for the theory of generalized random graphs. In Section 3.3, we present a mathematical model for analyzing fault tolerance of our gossiping algorithm. Simulation results are reported in Section 3.4. Finally, Section 3.5 concludes this chapter.

3.1. Overview

Reliable multicast is one of the most important techniques to disseminate data items throughout all kinds of networks. It is very essential to design distributed systems and applications, such as publish/subscribe systems [EGH$^+$03], distributed databases [DGH$^+$87], consistency management [GT92], and distributed failure detection [RMH98]. There are two challenging issues when we evaluate the performance of a reliable multicast protocol. The first issue is how to evaluate the scalability of the protocol. Traditional solutions applicable in small-scale settings are not scalable and reliable in large distributed systems. How to design multicast protocols guaranteeing specified reliability in large-scale systems has become a challenging problem for researchers. The second issue is how to evaluate the fault-tolerance of these reliable multicast protocols. Wireless nodes are prone to fail due to the limited resources. Moreover, mobile nodes move freely such that disconnections may occur frequently. In this chapter, we mainly focus on the second issue. Our objective is to propose a mathematical model to describe and analyze the perforamce of fault-tolerance for gossip-based reliable multicast protocols.

As we all know, there are three ways for data dissemination in wireless mobile networks, i.e., unicast, multicast, and broadcast. Although broadcast can be used to broadcast a message among the network by taking advantage of the broadcast nature of wireless communication, this will results in congesting wireless medium if all the nodes uses the wireless medium at the same time. On the other hand, gossiping can be considered as a controlled form of flooding. In wireless mobile networks, it is difficult to build up and maintain a structure like a tree, due to dynamic topologies. Gossiping is independent to the network topology. Moreover, the peer-to-peer interaction model in gossiping can provide better scalability and resilience to node and link failures in a wireless mobile network.

In this chapter, we first describe a generalized gossiping algorithm, which differs from existing algorithm, and allows each node to generate a random number of gossiping targets by following a specified probability distribution. In the traditional gossiping algorithms, each node normally has a fixed number of gossiping targets. With respect of our proposed gossiping algorithm, we develop a mathematical model to analyze its fault tolerance property by using the generalized random graph theory [NSW01]. We observe that the process of generating a random graph is very similar to the process of gossiping a message in a multicast group. Thus, we use the size of the giant component in a random graph to represent the probabilistic reliability of gossiping, in the sense that nodes in the giant component can be reached by the source node with a very high probability. We consider node failures and analyze the performance of gossiping in terms of the reliability and the success of gossiping.

Compared with the Microsoft's work [KMG03], our proposed model in this chapter is with three advantages. First, we propose a novel way to represent the reliability of gossiping by the size of the giant component [NSW01] on a generalized random graph, which provides a much simpler way to describe the reliability of gossiping. Second, we discuss not only the success of gossiping, but also the reliability of gossiping, i.e., the percentage of non-failed nodes that receive the message. Third, the analysis in our model can be performed for various fanout distributions besides the Poisson distribution. Gossiping tailored for different applications over various types of overlays or physical topologies may need to use different fanout distributions in order to improve the performance of gossip-based reliable multicast protocols.

3.2. Preliminaries

In this subsection, we first introduce our system model and describe a generalized gossiping algorithm. Then we provide the preliminaries on the theory of generalized random graphs.

In our system model, a multicast group G is composed of n members, which have the same interest to share the same message m. Each member has a unique ID. We consider a fail-stop failure model, where failed nodes will not gossip messages they receive any more, and they fail only by crashes. Moreover, we assume the source never fails. In real applications, we assume that a scalable membership protocol is available, such as [GKM01, ADH05], which can be applied to gossip-based reliable multicast protocols in large-scale systems. Membership protocols are beyond the scope of this chapter and will not be discussed further.

We propose a generalized gossiping algorithm as shown in Figure 3.1, which considers various distributions of the fanouts of the gossiping nodes. When a member receives the message m for the first time, it generates a random number f_i by following a specified probability distribution P. Then the node chooses f_i gossip targets from its own membership view and sends the message out. If a member receives the message again, it discards it immediately.

The reliability of gossiping is defined as the ratio of the number of non-failed members that received the message *m* to the total number of non-failed members in the group *G*. We denote the reliability of gossiping as *R(q, P)*, the probability that a non-failed member can receive the message *m* after one execution of our gossiping algorithm. The success of gossiping is defined as all of the non-failed members receive the message *m* at least once after *t* executions of our gossiping algorithm, denoted by *S(q, P, t)*. In this chapter, we focus on the relationship between the parameters of the gossiping algorithm and the reliability of gossip-based multicast protocols. The key parameters in gossiping are listed as follows:

- *P: Fanout Distribution*, the probability distribution of the fanout of members;
- *q: Non-failed Member Ratio*, the ratio of the number of the non-failed members to the number of the total members.

```
Algorithm for each node in a multicast group $G$

Upon member $i$ receiving the message $m$ for the first time
{
    Member $i$ generates a random number $f_i$ by following a specified
probability distribution $\boldsymbol{P}$.
    Member $i$ selects $f_i$ nodes uniformly at random from its membership
view.
    Member $i$ sends the message $m$ to the selected $f_i$ nodes.
}
```

Figure 3.1: The generalized gossiping algorithm

Compared with the traditional Poisson random graph model, generalized random graph [NSW01, N03], which has been previously applied in Physics, is a more general model for random graphs. It is applicable to arbitrary degree distribution in a random graph. Before we introduce our mathematical model for gossiping, we briefly introduce some fundamental results from the generalized random graph theory: **Degree Distribution***,* **Component**, **Phase Transition**, and **Giant Component**.

Degree Distribution denotes the probability distribution of the degrees of nodes in a generalized random graph. A **Component** is a set of nodes that can reach each other along the paths on the graph. A **Phase Transition** refers to the phenomenon that, while a random graph with n nodes and a certain number of edges is unlikely to have one special property, a random graph just with a few more edges is very likely to have this property. A good example of a phase transition is the critical point of the connectivity of a random graph. The critical point means the point at which the connectivity of a random graph grows dramatically. The **Giant Component** is the biggest component formed after a phase transition happens. It has a size of order at least $n^{2/3}$, while the sizes of other components are of order at most $n^{2/3}/2$.

In our gossiping algorithm, the number of each member's gossip targets is a random variable and these random variables are independent and identically-distributed random variables. In fact, it is similar to the case that we draw some samples from the total population but without putting back any of them, because each nonfailed member only gossips once in our algorithm. The execution of the gossiping algorithm dies out when the nonfailed members that received the message m are in the same connected component. Therefore, the distribution of the fanout has the most important impact on the reliability of gossiping and the success of gossiping.

3.3. A Fault-tolerant Gossip Model

In this subsection, we first propose an analytical model for fault-tolerant gossiping by using the theory of generalized random graphs. Here the term of "fault-tolerant" means that if a proposed gossiping algorithm aims to achieve the required reliability, it should take node failures into consideration. Then we show how to use this model to analyze the performance of the gossiping algorithm by taking the Poisson fanout distribution as an example.

3.3.1. Model Definition

A gossiping model *Gossip(n, P, q)* consists of *n* members to participate in gossiping with the fanout distribution *P*. Only a ratio *q* of all of the members can work correctly and other nodes may fail by crashes during gossiping. We consider two cases of failures which are treated as the same case. Members may crash either before receiving the message, or after receiving the message but not yet forwarding it to others. *P* is the fanout distribution of non-failed members that participate in gossiping. As mentioned before, we assume that the source node that initiates gossiping never fails. We use the terms "source member" and "source node" interchangeably in this section.

Let $\varsigma(n,\mathrm{P})$ be the space of generalized random graphs generated by gossiping, consisting of *n* nodes in the group *G*, and each node chooses its gossiping targets from its own membership view. Let p_k be the probability that a randomly chosen node from one element in $\varsigma(n,\mathrm{P})$ has the degree *k*, and q_k be the probability that a node with the degree *k* is also a non-failed node. The degree distribution in the random graph $\varsigma(n,\mathrm{P})$ can be generated by the following generating function [NSW01]:

$$F_0(x)=\sum_{k=0}^{\infty} p_k q_k x^k \tag{3.1}$$

In the above model, p_k can be any probability distribution. But we investigate the special case of uniform probability for q_k because we assume the setting for each member gossiping is the same one. We set $q_k=q$ for all k, i.e. all of the nodes fail with the same probability $(1-q)$. Note that the total number of non-failed nodes is equal to $n*q$.

3.3.2. Analysis of Gossiping

The methodology in this work is to investigate the properties of generalized random graphs to analyze the performance of the generalized gossiping algorithm. We consider two of the most important problems, including how to evaluate the reliability of gossiping and how to guarantee the success of gossiping. There are n members in the group G and the total number of the non-failed members is denoted by $n_{nonfailed} = [n*q]$. We define n_{rece} as the number of non-failed members that receive the message m after one execution of the algorithm. The reliability of gossiping $R(q, P)$ can be defined as $R(q, P)= n_{rece}/ n_{nonfailed}$. The number of non-failed members that receive the message m at least one time after t executions of the algorithm is referred to as n^t_{rece}. The success of gossiping $S(q, P, t)$ can be defined as $Pr(S(q, P, t)) = Pr(n^t_{rece} = n_{nonfailed})$, where all of the non-failed members receive the message m at least one time after t executions of the proposed algorithm.

3.3.2.1. Reliability of Gossiping $R(q, P)$

Since the giant component changes the connectivity of a random graph, the probability that a randomly chosen node belongs to this component is increased dramatically. With the size of the giant component growing, the probability that nodes receive the message m is also increased, which means more and more members can receive the message m sent from the source node in gossiping.

Firstly, the condition for the appearance of the giant component can be obtained in the following steps. According to the generalized random graph theory, the mean size <s> of the components in the random graph $\varsigma(n, \mathrm{P})$ is defined as follows:

$$\langle s \rangle = q\left[1 + \frac{qG_0'(1)}{1 - qG_1'(1)}\right] \tag{3.2}$$

where $G_0(x) = \sum_{k=0}^{\infty} p_k x^k$ is defined as the generating function for the probability distribution of nodes degree *k*, and $G_1(x) = G_0'(x) / G_0'(1)$ is the generating function of the probability distribution of the number of outgoing edges. Eq. (3.2) diverges where the equation $1 - qG_1'(1) = 0$ is satisfied, which is also the critical point at which a random graph achieves the giant component. According to the result above, the non-failed member ratio at the critical point is

$$q_c = \frac{1}{G_1'(1)} \tag{3.3}$$

where *c* is the critical point at which the generated random graph begins to improve its connectivity dramatically.

Secondly, we refer to *S* as the size of the giant component, which means the ratio of the total number of the non-failed nodes in the giant component to the total number of non-failed nodes in the random graph. *S* can be calculated by the following equation:

$$S = F_0(1) - F_0(u) \tag{3.4}$$

where *u* is the solution of the self-consistency condition $u=1-F_1(1)-F_1(u)$, and $F_1(x)$ is defined as $F_0(x) / G_0'(1)$ [CNS+00].

3.3.2.2. Probability of Success of Gossiping: *Pr(S(q, P, t))*

The success of gossiping is the result that all of the non-failed group members receive the message *m*. In one execution of our gossiping algorithm, we can increase the probability of gossiping success by increasing the fanout of members. However, this method is not a pragmatic one in the implementation of real applications. For example, one node only has one physical neighbour but it still needs to send the message *m* to all of the other nodes at one execution. Therefore, we consider another means to increase the probability of the success of gossiping by increasing the number of executions of our gossiping algorithm, in order to guarantee the required probability of the success of gossiping.

In the repeated executions, each execution can be viewed as one independent Bernoulli trial. So *t* times of executions can be considered as a *t* times Bernoulli trials. We define *X* as the number of executions in which a non-failed member receives the message m during *t* executions. We do not consider how many times for each non-failed member to receive the message *m* in one execution. We use p_r to denote the requirement of the reliability *R(q, P)*, and it is obvious that *X* follows a Binomial distribution $B(t, p_r)$. The distribution of *X* is in the following:

$$P(X = k) = C_t^k (p_r)^k (1 - p_r)^{t-k}, \text{ k = 0, 1, 2, …, t}$$

The probability of the success of gossiping *S(q, P, t)* can be calculated by the following:

$$\Pr(S(q, P, t)) = P(X \geq 1) = 1 - (1 - p_r)^t \qquad \textbf{(3.5)}$$

If the requirement for the success of gossiping is denoted by the probability p_s, we can obtain the requirement of *t* as follows:

$$t \geq \lg(1 - p_s) / \lg(1 - p_r) \quad t \in N \qquad \textbf{(3.6)}$$

3.3.3. Case Study: Poisson Fanout Distribution

In this subsection, we take the Poisson distribution as an example of the fanout distributions to show how to apply our mathematical model in the performance analysis of gossiping.

The fanout distribution is specified by a Poisson distribution *Po(z)*, where *z* is the mean of Poisson distribution *Po(z)*, and is also the average fanout. Then, the gossiping model can be defined as *Gossip(n, Po(z), q)*. This gossiping model can be modelled by a random graph model $\varsigma(n, Po(z))$. Let p_k be the probability that a randomly chosen node from $\varsigma(n, Po(z))$ has the degree *k*, and *q* be the non-failed node ratio. It is important to notice that, although the distribution of node degrees may be changed by node failures, it is always a Poisson distribution but with a smaller mean fanout $q*z$ [CNS$^+$00].

Theorem 1: For a Poisson distribution, the new generating function for the random graphs after removing failed nodes is $F_0(x) = e^{zq(x-1)}$.

Proof: Let $F_0(x)$ be the new generating function. Then, $F_0(x)$ can be derived in the following steps:

$$F_0(x) = \sum_{k=0}^{\infty} P'(k) x^k$$

$$= \sum_{k=0}^{\infty} \sum_{k_0=k}^{\infty} \frac{f^{k_0} e^{-z}}{k_0!} \binom{k_0}{k} q^k (1-q)^{k_0-k} x^k$$

$$= \sum_{k=0}^{\infty} x^k e^{-z} \frac{q^k}{(1-q)^k} \sum_{k_0-k=0}^{\infty} \frac{z^{k_0}(1-q)^{k_0}}{k!(k_0-k)!}$$

$$= \sum_{k=0}^{\infty} x^k e^{-z} \frac{q^k}{(1-q)^k} \frac{(z-fq)^k}{k!} \sum_{k_0-k=0}^{\infty} \frac{(z-zq)^{k_0-k}}{(k_0-k)!}$$

$$= \sum_{k=0}^{\infty} e^{-zq} \frac{(zq)^k}{k!} x^k$$

$$= e^{zq(x-1)}$$

It is clear that these new generated graphs are still random graphs with the Poisson degree distribution, only with a smaller mean degree. □

We can obtain the following generating functions for $\varsigma(n, Po(z))$:

$$F_0(x) = \sum_{k=0}^{\infty} p_k q x^k = e^{zq(x-1)} \tag{3.7}$$

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k = e^{z(x-1)} \tag{3.8}$$

$$G_1(x) = \frac{G_0'(x)}{G_0'(1)} = e^{z(x-1)} \tag{3.9}$$

According to Eq. (3.3), the critical point q_c can be obtained by $q_c = 1/G_1'(1) = 1/z$ .This means that, to the guarantee on the reliability of gossiping, the non-failed member ratio q should be greater than *1/z*, i.e.:

$$q > 1/z \tag{3.10}$$

It is trivial that $G_0(x) = G_1(x) = e^{z(x-1)}$ in case of the Poisson distribution. By following Eq. (3.4), we can obtain the size of the giant component by

$$S = 1 - e^{-zqS} \tag{3.11}$$

Eq. (3.11) shows that the reliability of gossiping *R(q, Po(z))* can be improved if we increase the fanout *z* or *q*.

Then, given the reliability of gossiping (represented by *S*) and the non-failed node ratio *q*, the mean fanout *z* of the Poisson distribution can be obtained as follows:

$$z = \ln(1-S)/(-qS) \tag{3.12}$$

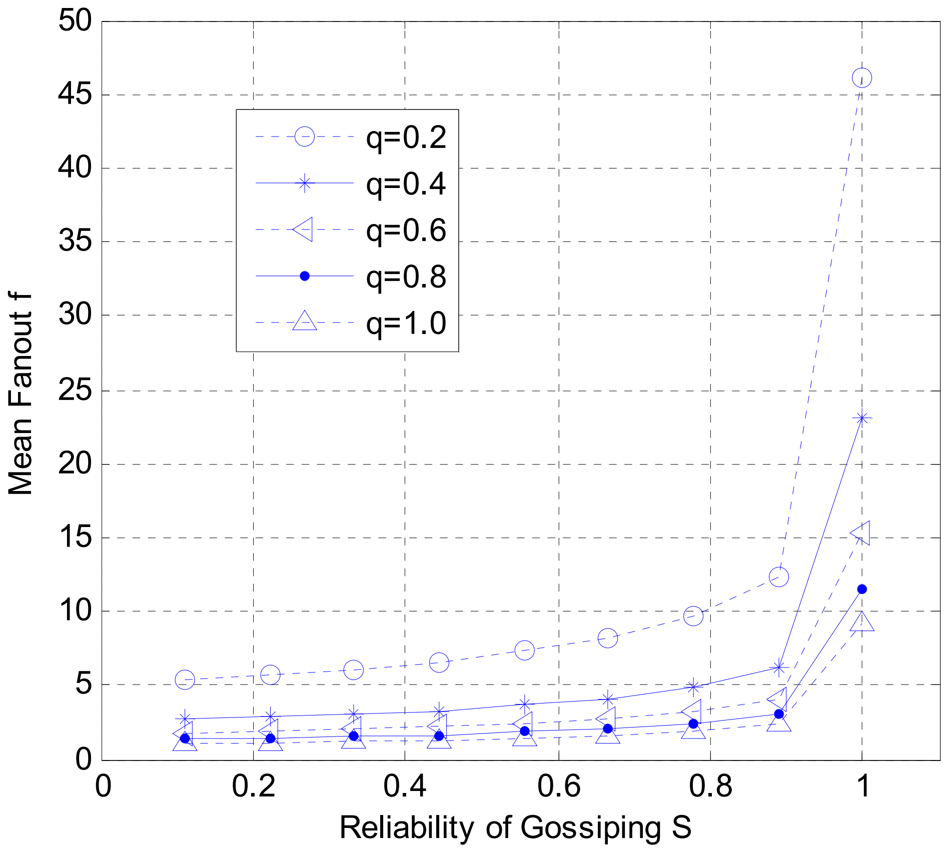


Figure 3.2: Mean fanout vs. Reliability of Gossiping under various non-failed node ratio.

Figure 3.2 shows the numerical results of z against S under various q. With these results, we can determine the proper mean fanout for the Poisson distribution. However, remember that Eq. (3.10) should still be held. The reliability of gossiping ranges from 0.111 to 0.999.

Since the reliability of gossiping can be evaluated by the size of the giant component S, the condition for the success of gossiping $S(q, Po(z), t)$ in Eq. (3.6) can be revised as follows:

$$t \geq \lg(1-p_s)/\lg(1-S) \quad t \in N.$$

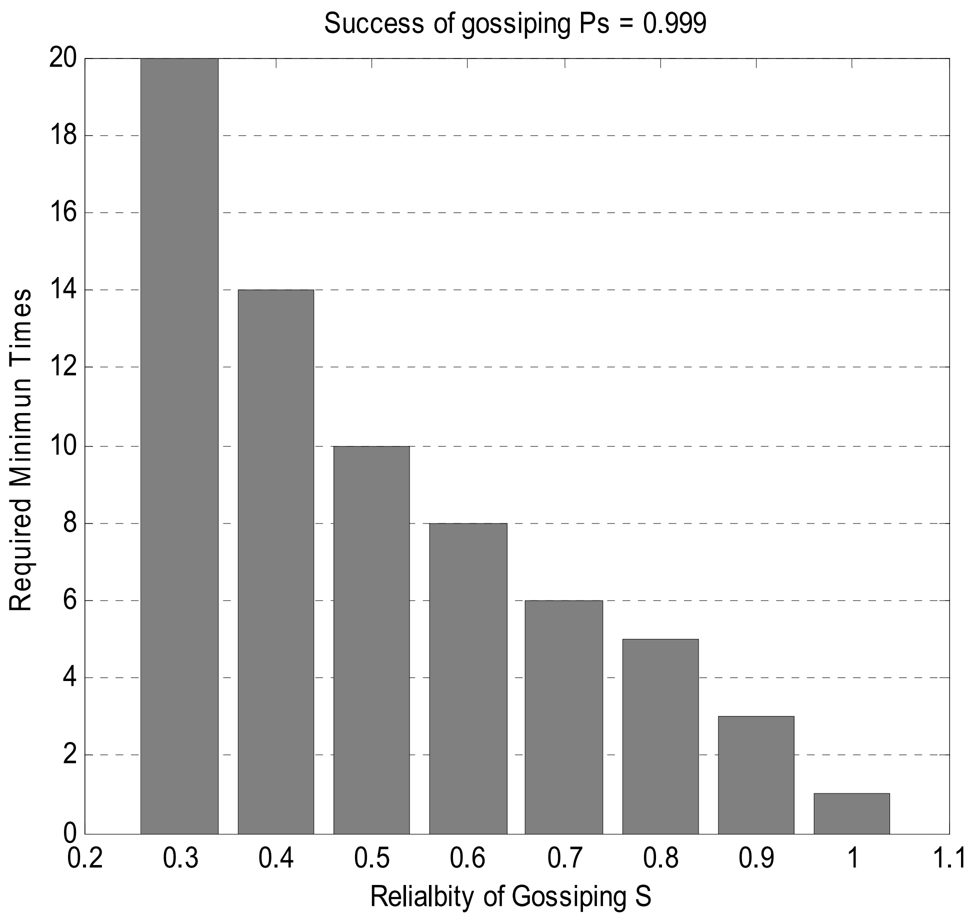


Figure 3.3: Minimum times of executions for the required probability of gossiping success.

Figure 3.3 shows the analytical results of the minimum number of executions with a specified probability of the success of gossiping.

3.4. Simulations

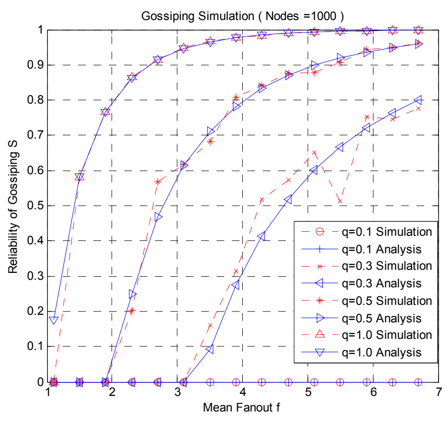
To examine the effectiveness of our analytic model, we have carried out extensive simulations. We evaluate the performance of our gossiping algorithm according to the following metrics:

- The reliability of gossiping
- The success of gossiping

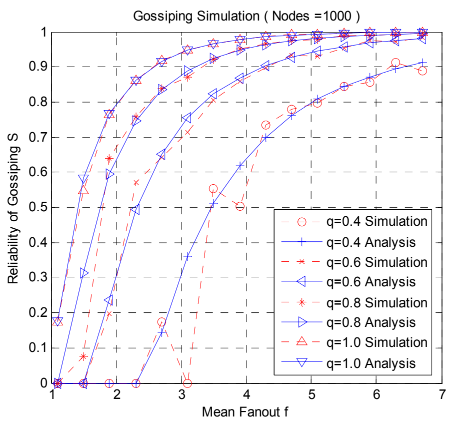
We use MATLAB 7.0 to implement and execute our gossiping algorithm. We evaluate the performance of the algorithm with two different group sizes of the groups 1000 and 5000 members. The key parameters, i.e., the fanout distribution and the non-failed member ratio, are varied to evaluate their impact. In our simulation, we take the Poisson distribution as an example for both the simulations and our analysis. Compared with the analytical results obtained by our mathematical model, the simulation results are well consistent.

3.4.1. Reliability of Gossiping

We set the non-failed member ratio *q* to be 0.1, 0.2, 0.3, …, 1.0 respectively. The mean fanout *f* for the fanout distribution is varied from 1.10 to 6.7 with an incremental step 0.4. The reason why we select this range is the value of the reliability is almost covered from 0 to 1. For each pair of {*f, q*}, we run our gossiping algorithm 20 times and report the average results of the reliability of gossiping. In addition, we calculate the size of giant component for each case.



(a)



(b)

Figure 3.4: Reliability in a group with 1000 nodes. (a) q=0.1, 0.3, 0.5, and 1.0; (b) q=0.4, 0.6, 0.8, and 1.0.

Figure 3.4 and Figure 3.5 show the results in simulation and analysis for configurations with 1000 and 5000 nodes respectively. To see the results clearly, we divide each figure into two plots by a group of different *q*. In each plot, each dotted line denotes the simulations, while the continuous line represents the size of giant components using our mathematical model solved by Eq. (3.11).

We first observe that all of the critical points for each fanout are held under the condition that the non-failed member ratio *q* should be greater than the reciprocal of the mean fanout *f*. For each fanout in our simulation, the reliability of gossiping can be guaranteed under the above condition. Figure 3.4 also shows that the results of simulations tally with the analytical results except very few points. The curves in Figure 3.5 are very similar to those in Figure 3.4. However, the simulation results tally with the analytical results better than in Figure 3.4, which indicates that our modeling works much better in larger scale systems.

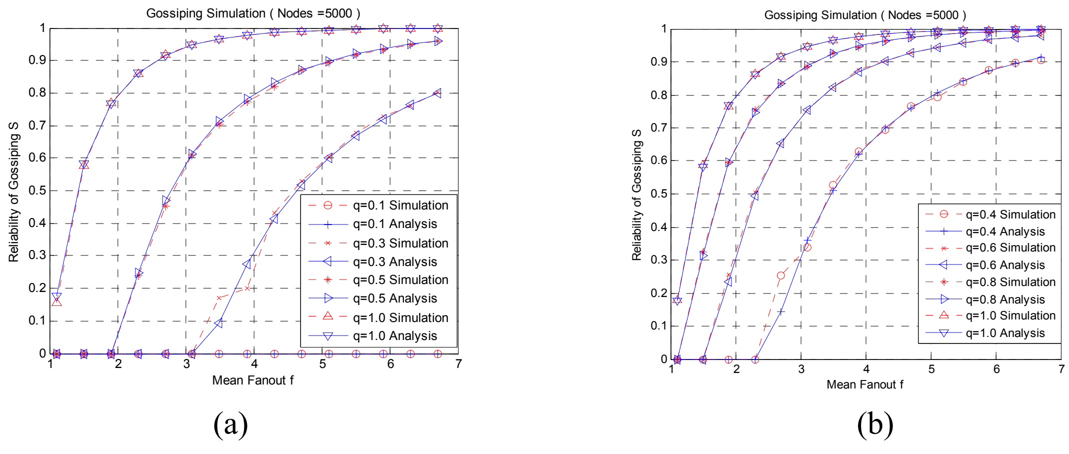


Figure 3.5: Reliability in a group with 5000 nodes. (a) q=0.1, 0.3, 0.5, and 1.0; (b) q= 0.4, 0.6, 0.8, and 1.0.

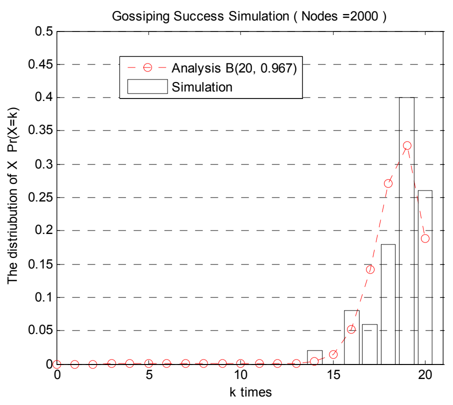
3.4.2. Success of Gossiping

Besides reliability, we also measure the success of gossiping. We select two pairs of key parameters {*f*, *q*} as follows: {4.0, 0.9}, and {6.0, 0.6}. The requirement for the success of gossiping is set to 0.999, the same value as in our analysis. For each pair of parameters, we run our gossiping algorithm for 20 times in one simulation, and each simulation is repeated for 100 times. Then we report the distribution of the number *X*, i.e. the number of gossiping succeeds among 20 executions. If *X* is approximately follows a binomial distribution *B(20, R(q, Po(z)))*, this means the calculation in Eq. (3.6) is valid.

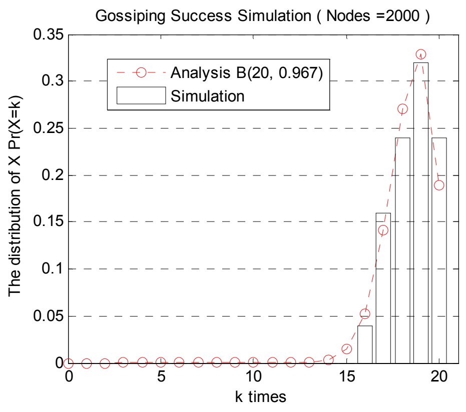
Figure 3.6 plots the results of the simulations and analysis in a group with 2000 nodes. In the figure, each bar represents the simulation result of the probability *Pr(X=k)* where *k* ranges from 0 to 20, while the continuous line represents the value of *Pr(X=k)* from *X~B(20, R(q, Po(z)))*. According to Eq. (3.6), we can obtain the required number of executions as follows:

$$t \geq \lg(1-0.999)/\lg(1-0.967) \quad t \in N$$

Obviously, *t* should be greater than three. Figure 3.6 shows the simulation results tally with our analytic results well.



(a)



(b)

Figure 3.6: The distribution of Gossiping Success with a pair of {f, q}. (a) f=4.0, q=0.9; (b) f=6.0, q=0.6.

It is interesting to notice that the gossiping with {4.0, 0.9} and {6.0, 0.6} can obtain the same reliability of gossiping in one execution as 0.967 because the product of $f*q$ are the same one. However, their corresponding distributions of gossiping success are not exactly identical. This is because the mean fanout and the nonfailed node ratio have different impact factors on the probability of the success of gossiping.

3.5. Summary

Based on the generalized random graph theory, we develop a mathematical model to analyze the performance of the generalized gossiping algorithm, in terms of the reliability of gossiping and the success of gossiping. We focus on the fault tolerance of gossiping by taking node failures into consideration. We propose to represent the reliability of gossiping by using the size of the giant component in a random graph for the first time. Our model can be resolved by the generalized random graph theory and derive the relationship between the parameters of gossiping, and the reliability of gossiping and the success of gossiping. Our model shows that there exists a threshold value of the number of the non-failed nodes ratio for guaranteeing a specific reliability in gossiping. We have carried out extensive simulations to validate our proposed model. The simulation results tally with our analytic results very well. Therefore, our analytic model is effective and accurate.

The analytical results we obtained from our fault-tolerant gossip model can be used to design a gossip-based reliable multicast protocol for data dissemination in a wireless mobile network. First, we should know the failure model of mobile devices so that we can know the nonfailed node ratio. Second, we should consider the membership view provided by a membership protocol so that we can use a proper fanout distribution for such a membership view. The best case is we can find enough gossiping targets for each node. We should deal with the problem how the fanout distribution can be adaptive to the views provided by the membership protocols.

Third, there is one point we can confirm, i.e., we are going to make full use of the broadcast nature of wireless communication. The gossip message can be broadcast to let all the neighboring nodes receive it. Fourth, our proposed generalized gossiping algorithm is not complex so that it can be easily implemented to a network protocol.

Chapter 4. Modeling Hierarchical Gossiping in Reliable Multicast Protocols

In this chapter, we introduce the proposed hierarchical gossiping algorithm with its mathematical model. This chapter is arranged as follows. Firstly, Section 4.1 is the overview to this work. Section 4.2 describes the system model, the generalized hierarchical gossiping algorithm, and the preliminaries for the theory of generalized random graphs. In Section 4.3, we present our mathematical model for analyzing the performance of hierarchical gossiping. Simulation results are reported in Section 4.4. Finally, Section 4.5 concludes this chapter.

4.1. Overview

Considerable research in gossip-based reliable multicast protocols [EGH$^+$03] has been done because of the importance of scalability and fault-tolerance in distributed systems. In these protocols, each node forwards messages to a small set of gossip targets, chosen from the entire or partial view of the multicast group. Gossip-based protocols probabilistically deliver the message to all the group members. Compared with the strong reliability of traditional protocols [BHO$^+$99], a gossip-based protocol can deliver much higher scalability and fault tolerance with probabilistic guarantees. However, the attractive reliability properties are derived from a high degree of message redundancy. This results in a large number of messages, which may be expensive in a wide-area network setting.

In some wireless networks, such as wireless LANs and wireless mesh networks [AWW05], the infrastructures of these networks are hierarchical inherently. There are two distinct sets of entities, including a large number of mobile hosts (MHs) and relatively fewer but more powerful mobile support stations (MSSs). MSSs are interconnected using a wired or wireless network, while MHs are connected to MSSs using wireless communications. Each MSS is in charge of a cell. A cell is a logical or geographical coverage area under a MSS. Each MH that has identified itself with a particular MSS is considered to be local to the MSS. A MH can directly communicate with a MSS (and vice versa) only if the MH is physically located within the cell serviced by the MSS. In this chapter, we aim to take advantage of these characteristics of infrastructure-based wireless networks.

As we know, hierarchical gossiping [KMG03] divides the whole group into smaller subgroups according to some network proximity, such as physical topology or RTT, to reduce the messages cost by using fewer messages between subgroups. Within each subgroup, members gossip the message in the same way as flat gossiping [KMG03] does. However, each node in a small set Q within each subgroup maintains a remote view of some group members in other subgroups. Once receiving the gossiping message, a node in Q sends the message out to other subgroups by choosing its gossip targets from its remote view. Compared with flat gossiping [KMG03], it means that hierarchical gossiping can achieve the same reliability but use fewer messages.

In this chapter, we focus on developing a mathematical model to analyze the reliability of hierarchical gossiping. We first describe a generalized hierarchical gossiping algorithm, where the number of gossiping targets of a node follows any given probability distribution. In traditional gossiping algorithms, each node normally has a fixed number of gossiping targets. Targeted at this proposed algorithm, we develop a mathematical model to analyze the reliability of hierarchical gossiping using the generalized random graph theory [NSW01]. We observe that the process of generating a random graph is similar to the process of gossiping a message in a multicast group. Consequently, we use the size of the giant component in a random graph to represent the reliability of gossiping firstly in [FCW+08], in the sense that each node in the giant component can be reached by the source node with a very high probability. In our previous work [FCW$^+$08], we focus on the fault-tolerance of flat gossiping by considering the node-failure model. Compared with our previous work [FCW$^+$08], this chapter focuses on analyzing the reliability of **hierarchical gossiping** under the critical condition to propagate a message from local subgroups to the whole group.

4.2. Preliminaries

In this section, we first introduce our system model and describe a generalized hierarchical gossiping algorithm. Then we provide the preliminaries on the theory of generalized random graphs.

```
Algorithm for each node in each subgroup in a multicast group $G$

Upon Member $i$ receiving the message $m$ for the first time
{
  Member $i$ generates a random number $f_i$ by following the Level-1 fanout distribution $P_{intra}$.
  Member $i$ selects $f_i$ nodes uniformly at random from its local view $V_{local.}$
  Member $i$ sends the message to the selected $f_i$ nodes.

  If Member $i \in Q$
  {
    Member $i$ generates a random number $k_i$ by following the Level-2 fanout distribution $P_{inter}$
    Member $i$ selects $k_i$ nodes from its remote view $V_{remote}$.
    Member $i$ sends the message to the selected $k_i$ nodes.
  }
}
```

Figure 4.1: The hierarchical gossiping algorithm

In our system model, a multicast group G is composed of n members, which have an interest to share the same message m. Each member has a unique ID. We assume that members do not fail and a scalable hierarchical membership protocol is available for a large-scale system, such as the protocols in [GKM02, DGH$^+$87]. The whole group is divided into n_{subg} subgroups, in each of which the number of group members is n_{i_subg}, $i=1, 2,\ldots, n_{subg}$. We also assume that n_{subg} is much smaller than n_{i_subg}. In each subgroup, the membership protocol maintains a small partial view V_{local} of the subgroup for each group member. Additionally, there is a small set Q among the members in each subgroup and nodes in Q are provided with a remote view V_{remote} that contains members in other subgroups. We denote the size of Q as $q*n_{i\_subg}$, where q is named as the remote view ratio.

Our mathematical model of gossiping is based on a generalized hierarchical gossiping algorithm as shown in Figure 4.1, which allows for various distributions of the fanouts of nodes.

The reliability of hierarchical gossiping is defined as the ratio of the number of members that receive the message m to the total number of members in the group G. We denote the reliability of hierarchical gossiping as $R(P_{intra}, P_{inter})$, i.e., the probability that a member receives the message m after one execution of our proposed algorithm. We investigate the impact of the two distributions P_{intra} and P_{inter} on the reliability of gossip-based multicast protocols. We refer to the process of gossiping within each subgroup and among subgroups as the Level-1 and Level-2 gossiping respectively. The key parameters in hierarchical gossiping are listed as follows:

- P_{intra}: the fanout distribution at the Level-1, i.e., the probability distribution of the fanout of members within one subgroup.
- P_{inter}: the fanout distribution at the Level-2, i.e., the probability distribution of the fanout of members between different subgroups.

Compared with the traditional Poisson random graph model, a generalized random graph [NSW01] [N03] is a more general model for random graphs. It is applicable to arbitrary degree distributions in a random graph. Some preliminaries can be found in our previous work [FCW^{+}08].

4.3. A Hierarchical Gossip Model

In this section, we first propose an analytical model for hierarchical gossiping by using the theory of generalized random graphs. Then, we show how to use this model to analyze the performance of the hierarchical gossiping algorithm by taking the Poisson fanout distribution as an example.

4.3.1. Model Definition

A hierarchical gossiping model $HGossip(n, n_{subg}, P_{intra}, P_{inter})$ consists of n members, grouped into n_{subg} subgroups, to participate in gossiping with two fanout distributions P_{intra} and P_{inter}. Each member in the same subgroup selects the number of its gossip targets by following the fanout distribution P_{intra}. Moreover, if we consider each subgroup as one super node, each subgroup selects the number of its gossip targets by following the fanout distribution P_{inter}. We aim to investigate the two key problems as follows:

- What is the critical condition for guaranteeing the gossiping message to be propagated from Level 1 to Lever 2?
- Given P_{intra} and P_{inter}, how to evaluate the reliability of hierarchical gossiping?

We describe the completed process that the message m is propagated as follows.

4.3.1.1. Level-1 Gossiping

First, the source node *s* in some subgroup sends the gossiping message *m* to its gossip targets, which are selected from its local view by following the fanout distribution P_{intra}. Then the message *m* is propagated within the subgroup. If some nodes in *Q* receive the message, they send the message *m* to other subgroups outside. In subgroup G_{sub_i}, we describe the result of gossiping as one of the elements in the random graph space $\varsigma(n_{i\_sub}, P_{\text{int}ra})$.

Let p_{k_intra} be the probability that a randomly chosen node from $\varsigma(n_{i\_sub}, P_{\text{int}ra})$ has the degree *k*. The degree distribution in the random graph space $\varsigma(n_{i\_sub}, P_{\text{int}ra})$ can be generated by the following generating function [NSW01]:

$$G_{0\_\text{int}ra}(x) = \sum_{k=0}^{\infty} p_{k\_\text{int}ra} x^k \tag{4.1}$$

where G_{0_intra} is absolutely convergent for all $|x| \leq 1$.

4.3.1.2. Level-2 Gossiping

At the Level-2, we consider each subgroup as one super node in our hierarchical gossiping model. The message *m* is propagated from one super node to another. The result of gossiping can be modeled as one of the elements in the random graph space $\varsigma(n_{subg}, P_{\text{int}er})$. Let p_{k_inter} be the probability that a randomly chosen node from one element in $\varsigma(n_{subg}, P_{\text{int}er})$ has the degree k. The degree distribution in the random graph space $\varsigma(n_{subg}, P_{\text{int}er})$ can be generated by the following generating function [NSW01]:

$$G_{0\_\text{int}er}(x) = \sum_{k=0}^{\infty} p_{k\_\text{int}er} x^k \tag{4.2}$$

4.3.2. Analysis of Hierarchical Gossiping

We analyze the reliability of hierarchical gossiping through the properties of the corresponding generalized random graphs. We discuss two important problems in our hierarchical gossiping algorithm: how to guarantee the message to be propagated from the Level-1 to the Level-2, and how to evaluate the reliability of hierarchical gossiping. With the size of the giant component growing, the probability that nodes receive the message *m* is increased, which means more and more members can receive the message *m* sent from the source node *s* in gossiping.

4.3.2.1. Critical Condition of Propagating Message from Level 1 to Level 2

Obviously, the Level-2 gossiping depends on the Level-1 gossiping. If the Leve-1 gossiping fails, it is impossible for other subgroups to receive the message *m*. Therefore, the critical condition for hierarchical gossiping includes two sub-conditions. The first sub-condition is that at least one node in the set *Q* of each subgroup receives the gossiping message *m*. The second sub-condition is that the fanout between subgroups should be great enough to guarantee the appearance of the giant component in the Level-2 gossiping.

According to the generalized random graph theory, the mean size $<s_{intra}>$ of the components on one random graph in $\varsigma(n_{i\_subg}, P_{\mathrm{int}ra})$ is

$$\langle s_{\mathrm{int}ra} \rangle = 1 + \frac{G'_{0\_\mathrm{int}ra}(1)}{1 - G'_{1\_\mathrm{int}ra}(1)} \tag{4.3}$$

where $G_{1\_\mathrm{int}ra}(x) = G'_{0\_\mathrm{int}ra}(x) / G'_{0\_\mathrm{int}ra}(1)$ is the generating function of the probability distribution of number of outgoing edges. Eq. (4.3) diverges where the equation $1 - G'_{1\_\mathrm{int}ra}(1) = 0$ is satisfied, which is also the critical point at which a random graph achieves the giant component.

We refer to S_{intra} as the size of the giant component within each subgroup in the Level-1 gossiping, which means the ratio of the number of nodes in the giant component to the total number of nodes in the random graph that represents the result of the execution of our hierarchical gossiping algorithm in each subgroup. S_{intra} can be calculated by the following equation:

$$S_{\mathrm{int}ra} = 1 - G_{1\_\mathrm{int}ra}(u) \tag{4.4}$$

where u is the smallest non-negative real solution of $u=G_{1\_intra}(u)$ [NSW01]. We define q as the ratio of the number of nodes in Q to the number of nodes in each sub-group. As we mentioned before, q is also the remote view ratio. Therefore, it is necessary to guarantee that the gossiping message m should be propagated to any node in the set of Q. So the first sub-condition can be described exactly as follows:

$$S_{intra} > 1 - q \tag{4.5}$$

Next, we consider the second sub-condition. According to the generalized random graph theory, the mean size $<S_{inter}>$ of the components on one random graph in $\varsigma(n_{subg}, P_{\mathrm{int}er})$ is

$$\langle s_{\mathrm{int}er} \rangle = 1 + \frac{G'_{0\_\mathrm{int}er}(1)}{1 - G'_{1\_\mathrm{int}er}(1)} \tag{4.6}$$

where $G_{1\_\mathrm{int}er}(x) = G'_{0\_\mathrm{int}er}(x) / G'_{0\_\mathrm{int}er}(1)$ is the generating function of the probability distribution of number of outgoing edges. Eq. (4.6) diverges where the equation $1 - G'_{1\_\mathrm{int}er}(1) = 0$ is satisfied, which is also the critical point at which a random graph achieves the giant component.

4.3.2.2. Reliability of Hierarchical Gossiping *R(P*

First, we define n_{rece} as the number of members that receive the message m after the execution of the hierarchical gossiping algorithm. The reliability of gossiping $R(P_{intra}, P_{inter})$ can be defined as follows:

$$R(P_{intra}, P_{inter}) = n_{rece}/n \tag{4.7}$$

Let S_{inter} be the size of the giant component in the random graph that is the result of the execution of our hierarchical gossiping algorithm in the Level-2 gossiping. Therefore, S_{inter} also represents the probability that each subgroup in Level 2 receives the gossiping message *m*. It can be obtained by the following equation:

$$S_{inter} = 1 - G_{1\_inter}(u) \tag{4.8}$$

where u is the smallest non-negative real solution of $u=G_{1\_inter}(u)$ and $G_{1\_inter}(x) = G'_{0\_inter}(x)/G'_{0\_inter}(1)$.

We refer to S as the reliability of hierarchical gossiping $R(P_{intra}, P_{inter})$ at the two levels, which means the ratio of the total number of the nodes in the two giant components to the total number of nodes on the two random graphs.

$$S = \frac{S_{inter}\sum_{i=1}^{n_{subg}} S_{intra} \bullet n_{i\_subg}}{n} = S_{intra}S_{inter} \tag{4.9}$$

4.3.3. Case Study: Poisson Fanout Distribution

In this subsection, we take the Poisson distribution as the example of the fanout distribution, in order to show how to apply our mathematical model in the performance analysis of hierarchical gossiping.

We assume that P_{intra} and P_{inter} are two Poisson fanout distributions, which are denoted as $Po(z_1)$, $Po(z_2)$ respectively. Then, the gossiping model can be defined as $HGossip(n, n_{subg}, Po(z_1), Po(z_2))$. Let p_k be the probability that a randomly chosen node from $\varsigma(n, Po(z_1))$ or $\varsigma(n_{subg}, Po(z_2))$ has the degree *k*. We can obtain the following generating functions:

$$G_{0\_intra}(x)=\sum_{k=0}^{\infty}p_k x^k=e^{z_1(x-1)} \tag{4.10}$$

$$G_{0\_inter}(x)=\sum_{k=0}^{\infty}p_k x^k=e^{z_2(x-1)} \tag{4.11}$$

It is trivial that we find $G_{0\_intra}(x)$ is equal to $G_{1\_intra}(x)$ according to the definition $G_{1\_intra}(x)=G'_{0\_intra}(x)/G'_{0\_intra}(1)$, and $G_{0\_inter}(x)$ is the same case. Let S_1 and S_2 be the sizes of giant components at the two levels respectively. According to the theory of generalized random graphs [NSW01], we obtain the size of giant components at the two levels as follows:

$$S_1=1-e^{-z_1 S_1} \tag{4.12}$$

$$S_2=1-e^{-z_2 S_2} \tag{4.13}$$

4.3.3.1. Critical Condition of Propagating Success

To simplify our model, we define the number of nodes in each subgroup is equal to $m=n/n_{subg}$. As we mentioned in Section 4.2, the ratio q for the small set Q can be obtained by z_2/m. Therefore, the condition for guaranteeing the message m to be propagated from Level 1 to Level 2 can be described as the following theorem.

Theorem 1: Hierarchical Gossiping Model $HGossip(n,\ n_{subg},\ Po(z_1),\ Po(z_2))$ guarantees the gossiping message m to be propagated from Level 1 to Level 2 if and only if $z_1>\log(z_2/m)/(z_2/m-1)$ and $z_2>1$.

Proof: we denote A as the event that the message m cannot be propagated into the set Q for any subgroup. If A happens, it means that the message m cannot reach any node that can send the message m out of the subgroup, i.e., hierarchical gossiping fails. From the view of generalized random graphs, none of the nodes in the set Q is covered by the giant component. Therefore, we obtain the probability of the event A as follows:

$$P(A) = 1 - S_1 \tag{4.14}$$

Next, we obtain the first sub-condition that S_1 should be greater than $1-q$. We describe it as follows:

$$S_1 > 1-q \tag{4.15}$$

We substitute q with z_2/m and Eq. (4.15) can be written as follows:

$$S_1 > 1 - \frac{z_2}{m} \tag{4.16}$$

According to Eq. (4.12) and Eq. (4.16), we obtain the first sub-condition as follows:

$$z_1 > \log(\frac{z_2}{m}) / (\frac{z_2}{m} - 1) \tag{4.17}$$

As for the second sub-condition, it is equal to guarantee that the giant component appears in the random graph. The mean size <*s*> of the components on any of the random graphs in $\varsigma(n_{subg}, Po(z_2))$ is

$$\langle s \rangle = 1 + \frac{G'_{0\_inter}(1)}{1 - G'_{1\_inter}(1)} \tag{4.18}$$

where $G_{1\_inter}(x) = G'_{inter}(x) / G'_{inter}(1)$ is the generating function of the probability distribution of number of outgoing edges. Eq. (4.18) diverges where the equation $1 - G'_{1\_inter}(1) = 0$ is satisfied, which is also the critical point at which a random graph achieves the giant component. Note $G'_{1\_inter}(1) = z_2$. Therefore, according to the theory of generalized random graphs, we obtain the second sub-condition in the following:

$$z_2 > 1 \tag{4.19}$$

4.3.3.2. Reliability of Hierarchical Gossiping: $R(Po(z_1), Po(z_2))$

We refer to S_{total} as the reliability of hierarchical gossiping $R(Po(z_1), Po(z_2))$. S_{total} can be easily calculated as follows:

$$S_{total} = S_1 * S_2 \tag{4.20}$$

4.4. Simulations

To demonstrate the effectiveness of our analytic model, we carry out the following extensive simulations. We evaluate the performance of our gossiping algorithm according to the following metrics:

- The reliability of gossiping
- The success of gossiping

We use MATLAB 7.0 to implement and execute our gossiping algorithm. We evaluate the performance of the proposed algorithm with two different group sizes with 1000 and 5000 members respectively. The key parameters, i.e., the fanout distribution and the non-failed member ratio, are evaluated in our simulations to investigate their corresponding impact on the reliability of hierarchical gossiping. In our simulations, we take Poisson distribution as an example for both the simulations and our analysis.

The mean fanout f in Level 1 is also the mean value z_1 of the fanout distribution $Po(z_1)$. The mean fanout k in Level 2 is also the mean value z_2 of the fanout distribution $Po(z_2)$. For the critical condition, we choose the value of the mean fanout $f=log(q)/(q-1)$ according to Theorem 1, which is the minimum value at the critical point. The value of the size of Q is calculated by $q = k/1000$. Therefore, the proper value of the mean fanout f in Level 1 can be calculated by the value of the mean fanout k in Level 2. Compared with the analytical results obtained by our mathematical model, the simulation results are well consistent. For each pair of $\{f,$

k}, we run our hierarchical gossiping algorithm for 15 times and report the average results of the reliability of hierarchical gossiping. In addition, we calculate the size of giant component for each case to be compared with the results of simulations.

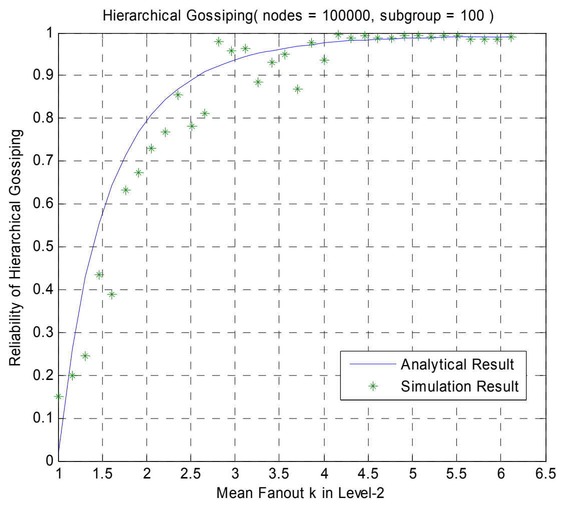


Figure 4.2: Mean Fanout k at Level 2 vs. Reliability of Hierarchical Gossiping, $n=100000$, $n_{subg}=100$

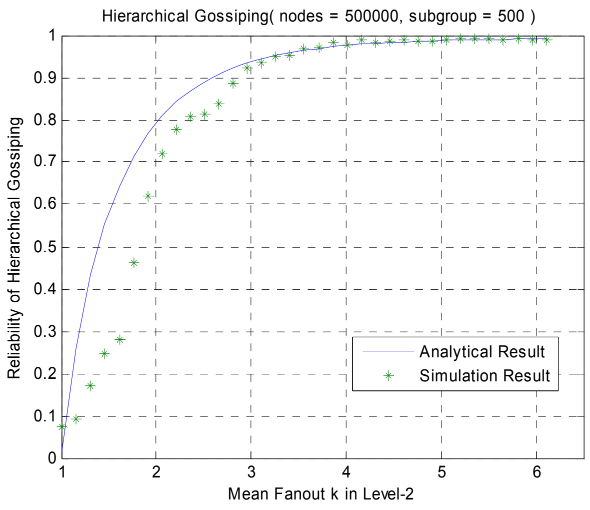


Figure 4.3: Mean Fanout k at Level 2 vs. Reliability of Hierarchical Gossiping, $n=500000$, $n_{subg}=500$

We first observe that all of the critical conditions for each pair of fanouts {f, k} are held when the mean fanout k in Level 2 is greater than 1.0. For each pair fanouts {f, k} in our simulation, the reliability of hierarchical gossiping can be guaranteed under the above condition. Figure 4.2 also shows that the results of simulations tally with the analytical results except for very few points. The curves in Figure 4.3 are very similar to those in Figure 4.2. However, the simulation results tally with the analytical results better than that in Figure 4.2, which indicates that our modeling works better in large-scale systems.

4.5. Summary

Based on the generalized random graph theory, we have developed a mathematical model to analyze the performance of the hierarchical gossiping algorithm in terms of the reliability of hierarchical gossiping. Our model can be solved by the generalized random graph theory and we derive the relationship between the fanout distributions at the two levels of gossiping and the reliability of hierarchical gossiping. Using our model, we discover there is a critical condition that is related to the fanout distributions at two levels, under which we can guarantee the gossiping message to be propagated from subgroups to the whole group. We have carried out simulations to validate our proposed model. The simulation results tally with our analytic results.

The analytical results we obtained from our hierarchical gossip model can be used to design a gossip-based reliable multicast protocol for data dissemination in a wireless mobile network. First, we should provide members in a multicast group with local view, remote view, or both by using a hierarchical membership protocol. In some infrastructure-based wireless networks, it is easy to organize the members of a multicast group into a hierarchical structure. Normally, we should propose a method based on clustering. Second, we should select a subset of nodes in each sub-group to propagate a gossiping message to other sub-groups. The relation between the fanouts in both levels should agree with our analytical results from our mathematical model.

Third, our proposed hierarchical gossiping algorithm is not complex so that it can be easily implemented to a network protocol.

Chapter 5. Contention-Aware Data Caching in Wireless Multihop Ad Hoc Networks

In this chapter, we introduce the proposed contention-aware data caching algorithms. This chapter is organized as follows. Firstly, Section 5.1 is the overview to this work. Section 5.2 describes the system model, the cache consistency model, and the problem formulation. In Section 5.3, we present the main ideas of our algorithms and describe the corresponding details. In Section 5.4, we analyze the performance of the proposed algorithms by developing a mathematical model. Simulation results are reported in Section 5.5. Finally, Section 5.6 concludes this chapter.

5.1. Overview

In a wireless network, due to the shared nature of wireless medium, nodes may contend with each other to access the data items. Consequently, the per-hop delay on each node is different from each other [YK06]. Some previous works conclude that the per-hop delay mainly depends on the contention delay in the 802.11-based wireless ad hoc networks [YK06]. Moreover, when more than one nodes access the same data item at one cache node, it dramatically increases the probability of access collisions. Thus, more collisions bring more retransmissions of data requests so that total access delay is increased sharply. As a result, it is necessary to consider the impact of contentions at wireless nodes on the performance of a caching system.

In this chapter, our cache placement problem is much different from the previous problems in the following way. There are three important points: 1) each hop has different weight in a wireless multihop ad hoc network due to the contentions; 2) the strategy of selecting cache nodes results in the changes on the contentions of wireless nodes because of the introduced or saved traffic flows; and 3) the contentions can be also changed due to the movement of mobile nodes. Therefore, our problem can be described as the cache placement problem in a network with dynamic topologies. We define such a problem as Dynamic Cache Placement (DCP). We prove that DCP is NP-hard. Due to the intractability of DCP, we address it by proposing our heuristics, which dynamically collects the contention variation on wireless nodes from the MAC layer, and selects the nodes that are with lower contentions and reduce more traffic flows as cache nodes. Our goal is to find an algorithm to select the set of cache nodes in order to minimize total cost, which includes caching overheads and total access delay. To the best of our knowledge, this is the first time to take the contentions into consideration for a caching system in a wireless multi-hop ad hoc network.

The proposed heuristic algorithms, Centralized Contention-Aware Caching Algorithm (CCCA) and Distributed Contention-aware Caching Algorithm (DCCA), provide a sub-optimal solution to DCP. Our algorithms have the following desirable properties: 1) it is a polynomial time algorithm; 2) it is a cross-layer designed algorithm because it makes caching decisions based on the contention information collected from the MAC and routing layer; 3) it can be easily applied and adapted to any arbitrary dynamic network topology; and 4) it can be easily implemented in a distributed asynchronous fashion.

5.2. Problem Formulation

In this section, we first introduce the system model for our work. Then we propose a TTL-based cache consistency strategy in our system. Finally, we formulate the cache placement problem in wireless multi-hop ad hoc networks as **Dynamic Cache Placement** (DCP).

5.2.1. System Model

Let *G(V, E)* be a connected graph representing a wireless multihop ad hoc network with *n* *(n=|V|)* nodes connected by *m* *(m=|E|)* links. Two network nodes communicate directly with each other, which is represented by an edge in the graph, whereas two nodes that cannot communicate with each other may still contend directly with each other, due to the shared nature of wireless medium. The radius of the contention range *r* depends on the type of wireless networks. Normally, the radius of the contention range in 802.11-based ad hoc networks is two times as long as the radius of the transmission range. The data rate of channel is defined as *W*. We describe all the notations used in the system model in Table 5.1.

Table 5.1: Notations used in system model

V	The set of mobile nodes in the network
E	The set of links in the network
n	The number of nodes
m	The number of links
N_i	The i-th wireless node in the network
r	The radius of the contention range of a wireless node
W	The data rate of wireless channel
d	The single data item
S	The data source that updates the data item *d*.
s_d	The size of data item *d*.
s_r	The size of a data request
f_u	The update frequency for the data source updating *d*
$f_a(i)$	The access frequency for N_i access *d*
V_C	The vector $V_C = \{c_1, c_2, \ldots, c_n\}$ that describes the solution of a caching strategy, where $c_i = 1$ if one copy of the data item *d* is cached on the node N_i. Otherwise, c_i is equal to 0.
S^t	The version number of d_i at the data source
C^t_j	The version number of d_i at the cache node N_j at time *t*.
T	The expiration time *T* for each copy at a cache node

For some Internet services or other mobile applications, a group of wireless nodes need to access the data item d maintained by the data source S. Let s_d be the size of data item d. The data source S updates the data item d with a given frequency f_u, i.e., the data item d changes with the mean update time T_u. Node N_i accesses the data item d with the access frequency $f_a(i)$. The size of a data request is defined as s_r. In order to reduce the access delay, the data source caches the copies of the data item d at some cache nodes in the network. We define two phases of our caching system in this chapter, including *the update phase* and *the access phase*. The first phase is that the data source updates cache copies in the network, and the second phase means that the client nodes send their queries for the data item d to their corresponding closest cache nodes. We define a cache placement strategy by the vector $V_C = \{c_1, c_2, \ldots, c_n\}$, where $c_i = 1$ if one copy of the data item d is cached on the node N_i. Otherwise, c_i is equal to 0.

To model our problem, we have the following assumptions.

- The data source S always possesses a cache copy of data item d.
- Links are bidirectional.
- Wireless nodes never fail.

5.2.2. TTL-based Consistency Management

In this chapter, we apply a TTL-based cache consistency strategy to maintain the delta consistency among all the cache nodes [CZX$^+$07]. In the proposed TTL-based caching strategy, each data copy is associated with an expiration time T, i.e., the Time-To-Live (TTL) value. If a query arrives before the expiration time, the copy is considered valid and the reply message is generated locally. Otherwise, the copy is considered invalid until the next update from the data source. The query should be replied by the data source S or other cache nodes that holds a valid cache copy. However, client nodes are not aware of the validation of cache copies before sending queries.

If a cache copy is invalid, the corresponding cache node will not serve any request but forward it to the nearest cache node from its backup cache node list, until next updates from the data source. If an access request is forwarded to the nearest cache node by a cache node that hold an invalid copy, there is no any cooperation between the two cache nodes. Therefore, the first cache node is not aware of the validation of the cache copy at the second cache node.

There are several reasons why we use a TTL-based caching consistency strategy in a wireless multihop ad hoc network. First, in a TTL-based strategy, there is no need for the data source to keep track of the locations of cache nodes. Therefore, it is resilient for mobile nodes to join or leave the set of cache nodes and thus it is suitable for highly dynamic systems. Second, each node determines the validity of cache copy autonomously so that validations of expired cache copies can be performed with either the data source or other mobile nodes with valid copies. Third, the proposed TTL-based strategy guarantees that the staleness of the requested data item is time-bounded. It offers the flexibility to use different TTL values to cater for different consistency requirements that may vary with the nature of shared contents and user tolerance.

Next, we define the delta consistency model as follows. Let S^t denote the version number of the source data and C^t_j be the version number of the cached copy on cache node N_j at time t. Initially, the version number of the source data is set to zero and then increased one upon each subsequent update. The version number of the cache copy is set to that of the source data at the time when it is synchronized. Thus, the delta consistency [CZX+07] can be defined as follows:

$$\forall t, \forall j, \exists \tau, 0 \le \tau \le \delta, s.t.\, S^{t-\tau} = C_j^t \qquad (5.1)$$

In the delta consistency model, any read of the data item *d* is never out of date by more than δ time. In this chapter, the parameter δ can be replaced by the expiration time *T* to show the degree of the users' requirements.

5.2.3. Problem Formulation

Given a set of graphs $G^k=\{G_0, G_1, G_2, ..., G_k\}$, which describes the topology changes for a wireless multi-hop ad hoc network *G*. We define a set $V=\{N_1, N_2, ..., N_n\}$ as *n* wireless nodes that are located by a piecewise functions $\{g_1, g_2, ..., g_n\}$, where g_i, $1 \leqq i \leqq n$ maps the time interval $[0,T]$ to G_k. Given *k* functions $f_1, f_2, \ldots, f_k$, f_i:$G_i \rightarrow [0,1]$, we select the cache nodes for G_i such that at given moment $t \in [0,T]$, f_i forms a sup-problem SP_i for all the mobile nodes located by $\{g_1, g_2, ..., g_n\}$. We describe all the notations used in problem formulation in Table 5.2.

Table 5.2: Notations used in problem formulation

G^k	$\{G_0, G_1, G_2, ..., G_k\}$, which describes the topology changes for a wireless multi-hop ad hoc network *G*.
SP_i	Ond of the sup-problem of the dynamic cache placement problem
$PATH(i, j)$	The directed path from the sender N_i to the receiver N_j
$w(i, j)$,	The sum of the total weight of each hop on the path $PATH(i, j)$.
$Cost(G_i, V_c(i))$	The total cost for the sub-problem SP_i, given a strategy $V_c(i)$
$w_{min}(c, S)$	The minimum weight among all of the paths from cache node *c* to the data source *S*.
$d_c(x)$	The contention delay of node *x*.
C_u	The traffic cost that the data source *S* sends the updates to all the cache nodes.
C_a	The traffic cost for client nodes sending their queries to their corresponding closest cache nodes, and cache nodes (including the data source) send replies to these client nodes
D_a	The access delay that all the requesting nodes experience.
β	The parameter in the objective function for SP_i to indicate the relative importance of access delay and caching overheads.

We divide the cache placement problem in wireless multi-hop ad hoc networks into smaller sub-problems. For each sup-problem SP_i, we also provide a formal formulation for the cache placement problem. Given a directed connected graph $G_i(V,E)$ and the weight of each link depends on the contentions of the sender. Let C be the set of cache nodes. Each message is sent from the sender N_i to the receiver N_j by following a directed path $PATH(i, j)$. The weight of the path, $w(i, j)$, is the sum of the total weight of each hop on this path. The weight of each hop (x, y) is measured by the per-hop delay of the sender x, $D(x)$. As we mentioned before, $D(x)$ is mainly determined by the contention delay of the node x in a contention-based MAC protocol. Let $d_c(x)$ be the contention delay of node x. Then we substitute the contention delay $d_c(x)$ for the per-hop delay $D(x)$ in this chapter. We obtain $w(i, j)$ as follows:

$$w(i,j) = \sum_{x} d_c(x) \quad x \in PATH(i,j) \; and \; x != j$$

Next, let $w_{min}(c, S)$ be the minimum weight among all of the paths from cache node c to the data source S. Let $w_{min}(i, c)$ be the minimum weight among all of the paths from the node N_i to the cache node c that **holds a valid cache copy of the data item *d***.

As we mentioned in Section 5.2.1, the cost in the update phase is defined as C_u, i.e., the traffic cost that the data source S sends the updates to all the cache nodes. The cost in the access phase includes: 1) the traffic cost, C_a, i.e. client nodes send their queries to their corresponding closest cache nodes, and cache nodes (including the data source) send replies to these client nodes, and 2) the access delay D_a that these nodes experience.

Therefore, given a strategy $V_c(i)$ for the sub-problem SP_i, we define the cost $Cost(G_i, V_c(i))$ as follows:

$$Cost(G_i, V_c(i)) = C_u + C_a + \beta D_a \tag{5.2}$$

with β indicating the relative importance of access delay and caching overheads. We can model different network scenarios and quality of service requirements by varying the parameter β.

To explain more details in Eq. (5.2), we have two extreme cases. Firstly, if all the nodes are cache nodes, the access traffic cost C_a and the access delay D_a will be zero. However, the data source *S* has to update all the copies at all the nodes. The update traffic cost will be maximized. Secondly, if the data source *S* decides not to cache the data item in any of the other nodes, the access traffic cost and the access delay will be maximized, while the update traffic cost will be zero.

Therefore, the cache placement problem for the sub-problem SP_i is to select a set of cache nodes $V_c(i)$ to minimize the cost $Cost(G_i, V_c(i))$:

$$\begin{aligned} Cost(G_i, V_c(i)) = & \sum_{j \in C} w_{\min}(j,S) \times s_d \times f_u \\ & + \sum_{j \in V} \sum_{c \in C} w_{\min}(j,c) \times s_r \times f_a(j) \\ & + \beta \sum_{j \in V} \sum_{c \in C} w_{\min}(j,c) \times f_a(j) \end{aligned} \tag{5.3}$$

The total cost for DCP is defined as C_{total}, which can be calculated as follows:

$$C_{total} = \sum_{i=0}^{k} Cost(G_i, V_c(i)) \tag{5.4}$$

Each solution for DCP is to find the *k* functions to minimize the total cost C_{total}. DCP can be proved NP-hard as follows:

Theorem 1: The computation of DCP is NP-hard.

Proof: If the topology of the network is not changed, it is proved that DCP is the rent-or-buy problem, the special case of the connected facility location problem. It has been proved NP-hard [SK02].

If the network topology changes due to the mobility of nodes, DCP is one of the special cases of the mobile facility location problem [BBK$^+$00]. The mobile facility location problem is proposed for continuously moving points. Given a set *ST* of *n* demand points in d-dimensional space ($d \geq 1$), find a set *P* of *k* supply points so that the dispersion defined as the sum of the L_p distance between demand points and their nearest supply pints in *P* is minimized. If we consider the cache nodes as facilities, DCP can be transformed to a mobile facility location problem. Similar to the rent-or-buy problem, DCP in mobile networks considers the cost of updates. The mobile facility location problem is NP-hard. Therefore, DCP is also a NP-hard problem. □

As mentioned in [BBK$^+$00], the data structures and algorithms that have been developed for the static problems (i.e. network nodes are not mobile) are not directly applicable to the setting of moving nodes when the motion of the facilities must satisfy natural constraints. Therefore POACH [NSC03] is not expected as efficient as in mobile networks. Our simulations also demonstrate this point.

5.3. A Heuristic Solution

In this section, we first explain how to calculate the per-hop delay for each node in order to evaluate the weight of each path. Then we propose our two contention-aware caching algorithms, i.e., Centralized Contention-aware Caching Algorithm (CCCA) and Distributed Contention-aware Caching Algorithm (DCCA), which provide two heuristic rules to minimize the total cost in Eq. (5.4), i.e., balance caching overheads with total access delay.

In our proposed algorithms, we use two key parameters to implement our heuristic rules, including *Hedging Flow (HF)* and *Contention Coefficient (CC)*. The reason why we propose the two parameters is that the total cost is related to the changes of the network topology and the related traffic flows. We will describe the details in the following sub-sections.

5.3.1. Per-hop Delay

Since we consider the difference among hops in wireless multi-hop ad hoc networks, we explain how to obtain the per-hop delay at wireless nodes to describe the weight of paths, along which each node sends its data packets. To model the per-hop delay, we make use of the previous work proposed in [YK06]. The per-hop delay consists of the queuing delay, the contention delay, and the transmission delay. However, the work [YK06] shows that average transmission delay is fixed, and average queuing delay is determined by both the mean and the variance of contention delay. Thus the per-hop delay is determined by the contention delay.

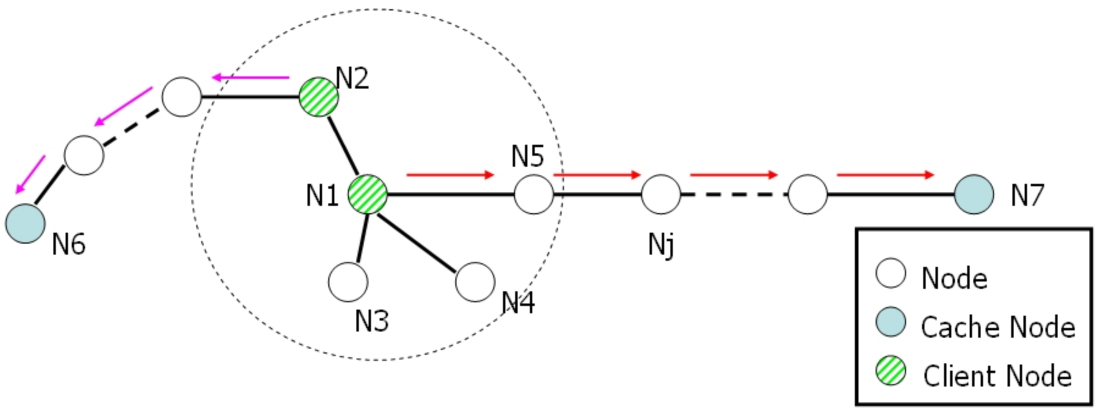


Figure 5.1: Example to show how to calculate the contention delay of node N1 in a 802.11-based wireless network

The contention delay is the interval between the time that the packet becomes the Head of the Line (HOL) in the node's queue, and the time that the packet actually starts to be transmitted on the physical medium. In this chapter, we focus on the contention delay in the 802.11-based wireless ad hoc networks, but the method can be also applicable for any contention-based MAC protocols. We define *CR(i)* as the set of nodes in the contention range of the mobile node N_i. The contention delay *d(i, c)* at the node N_i can be calculated by the following equation:

$$d(i,c) = DIFS + R_b(w_i\varepsilon + m_iT_d) \tag{5.5}$$

where *DIFS* is defined in 802.11 as DCF Inter-Frame Space, R_b is the probability that the channel state is busy, w_i is the number of back-off slots at mobile node N_i. The parameter ε is the value of back-off slot, m_i is the number of data packets transmitted by the neighbouring nodes during N_i's back-off process, and T_d is the duration of a successful data transmission. Next, we present a simple example to explain how to calculate the contention delay in 802.11 as follows.

In Figure, the contention range *CR(1)* is referred to as the node set $\{N_2, N_3, N_4, N_5\}$, each of which is in the contention range of N_1. The parameter a_{ji} is the discount factor of concurrent influence to the node N_i from the node N_j [YK06]. Thus the contention delay of the node N_1 can be calculated as follows:

$$d(1,c) = DIFS + (w_1\varepsilon + m_1 T_d)\sum_{j\in CR(1)} \frac{a_{j,1} f_a(j)}{W} + \frac{f_a(1)}{W} \tag{5.6}$$

In our simulation works, we collect the related parameters in Eq. (5.6) by interposing a detector in the source code of IEEE 802.11 MAC in NS2 [FV97]. The contention information is stored in the common header of each packet in NS2.

5.3.2. Two Heuristic Rules

In this subsection, we explain our heuristic rules in detail. As shown in Eq. (5.4), our objective is to minimize the total cost of a caching system. When we select a cache node, the first important point is that how much traffic flows can be saved and added. The best case is that we maximize the total traffic flows, in which we consider the flows saved as positive and the flows introduced as negative. Thus, we define such a kind of flows as *Hedging Flow (HF)*, which is borrowed from the financial industry. Those nodes with higher *HFs* can be considered as the candidates of cache nodes. The second important point is how to minimize the cost of sending and receiving data packets. Since the contention delay is used to describe the weight of each path, those nodes with more contentions are not suitable to act as cache nodes. Therefore, we propose the parameter *Contention Coefficient* to describe the level of contentions on each node.

Before we introduce the first heuristic rule, we should clarify the traffic flows in a caching system as follows. There are mainly three kinds of traffic flows in a caching system. The first one is the *Access Flow (AF)*, which is defined as the flows that traverse a node for data access. The second one is the *Reply Flow (RF)*, which is referred to as the flows that traverse a node to reply data requests from cache nodes or the data source *S*. The third one is the *Update Flow (UF)*, which is defined as the flows that traverse a node to update cache copies on cache nodes from the data source *S*. Therefore, it is important to select some special nodes as the candidates for cache nodes. By selecting these candidates, we aim to reduce the access flows and the reply flows as many as possible, while increasing the update flows as few as possible. As a result, we make the length of paths for data access and update as short as possible. As we mentioned, *Hedging Flow* describes the variation of the traffic flows if we select some node as a cache node. We denote the hedging flow of the node N_i as *HF(i)*, which can be calculated as follows:

$$HF(i) = \Delta AF(i) + \Delta RF(i) - UF(i) \tag{5.7}$$

Given a node N_i that we intend to select as a cache node, we obtain the reduced access flow *ΔAF(i)*, the reduced reply flow *ΔRF(i)*, and the introduced update flow *UF(i)*, three of which traverse N_i. In the above calculation, note that both the access flows and the reply flows include the traffics flows that traverse N_i, besides the traffic flows generated by N_i. Therefore, the first rule of our heuristic algorithm is given as follows:

Rule 1: When selecting the candidates of cache nodes, those nodes with their HFs higher than a given threshold Θ_1 should be selected.

To explain this rule, we use the following example shown in Figure 5.2. In this example, we combine the access flows with the reply flows for simplicity. That means we use the access flow represents the two flows, and we assume the reply flow is sent out along the reverse path from the requestor to its closest cache node. In this example, N_1 is the data source S. If we plan to select N_3 as the cache node, the reduced access flow is calculated by the following equation:

$$\Delta AF(3) = (f_a(4) + f_a(5) + f_a(6) + f_a(7) + f_a(8)) \times (s_r + s_d) \times w(3,1) \quad \textbf{(5.8)}$$

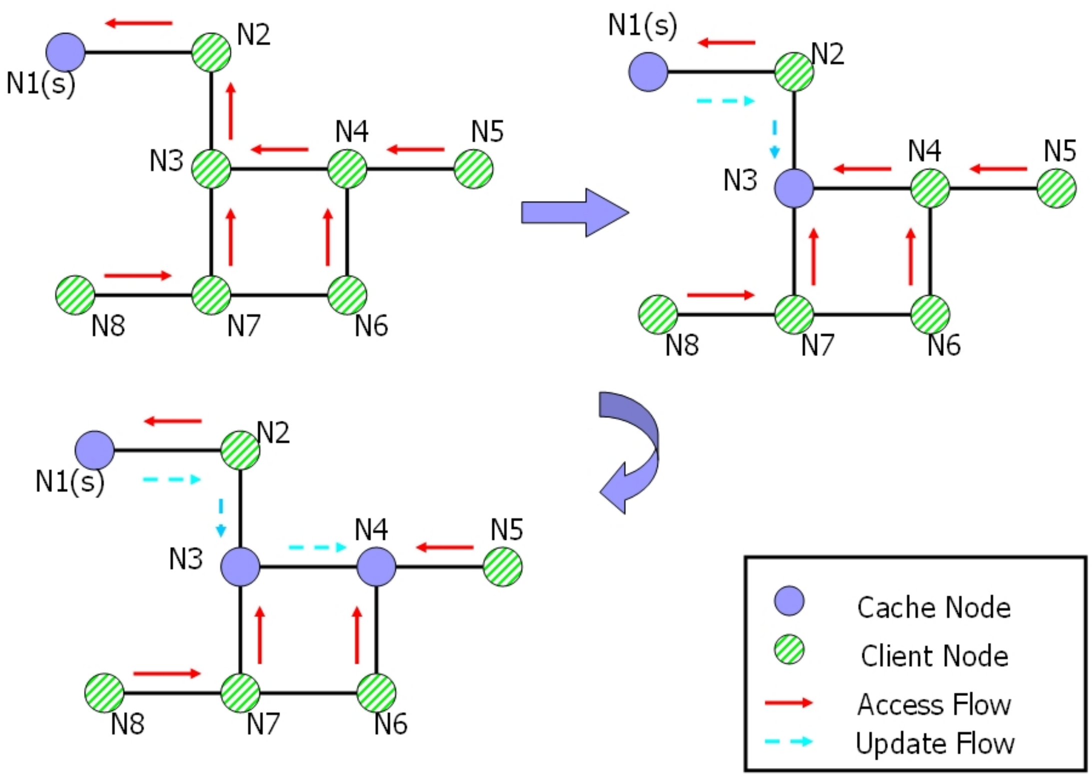


Figure 5.2: Example to show how to calculate the hedging flow

This means the reduced access flow is the sum of the access flows that traverse N_3, which should be forwarded to N_1 within two hops. Then the introduced update flow can be obtained by the following equation:

$$UF(3) = f_U \times s_d \times w(1,3) \quad \textbf{(5.9)}$$

According to the definition of hedging flow, the hedging flow of the node N_3, *HF(3)*, can be obtained as follows:

$$HF(3) = \Delta AF(3) - UF(3) \tag{5.10}$$

In generally, the first heuristic rule aims at optimizing the length of the paths from client nodes to their corresponding closest cache nodes, i.e., client nodes access the data item with much smaller cost. However, this operation changes the traffic loads locally so that the contentions in the network are also changed. Therefore, the second heuristic rule is designed to select the cache nodes from these candidates with fewer contentions.

To select the cache nodes from these candidates, we detect the variation of contentions on all the candidates. If the contentions are lower enough, it can decrease the access delay dramatically. Therefore, we present our second heuristic rule as follows:

Rule 2: When selecting a cache node from the set of the candidates, those nodes with their contention coefficients smaller than Θ_2 should be selected.

When we design our algorithms to implement **Rule 2**, we use the parameter *Contention Coefficient* of N_i, *CC(i)*, to describe the level of contentions on the node N_i. We define *F(i)* as the set of the nodes that have traffic flows traversing N_i. Thus we can obtain *CC(i)* by the following equation:

$$CC(i) = d(i,c) \times (f_a(i) + \sum_{j \in F(i)} f_a(j)) \tag{5.11}$$

In a contention-based MAC protocol, we can collect the contention information from the MAC layer every T_{con} seconds, in order to calculate the variation of contentions on each node. For example, from the time *t* to the time $(t+T_{con})$, we count the number of back-offs on N_i, and the number of the packets sending by N_i's neighbors during

N_i's back-offs. Then the *Contention Coefficient CC(i)* can be calculated by Eq. (5.11). Normally a dramatic variation of contentions between two sequential periods means that traffic flows are changed dramatically, or the network topology is changed around nodes. We detect the variation of contentions regardless of the mobility model. This is also one of the merits of our solution.

```
Algorithm 1: Centralized Contention-aware Caching Algorithm
INPUT:
 Graph G[n][n], Contention Coefficients CC[n]
OUTPUT:
 Cache Node Set C[n], Nearest Cache Node Set NC[n]
BEGIN
 The data source S converts the adjacency matrix G to a directed weighted graph
G' by CC[n].
  C = {S} and NC = {S}
  while (V-C) != Ø
    for each node $x \in (V-C)$
      S calculates the hedging flow HF(x).
      while ($y \in RoutingTable(x)$ & $y \notin C$ & HF(y) has not calculated before)
       if( $w_{min}(y, x) < w_{min}(y, NC[y])$)
         then $HF(x) += (w_{min}(y,NC[y])-w_{min}(y, x))*(s_d+s_r)*f_a(y)$
     end while
      $HF(x) -= w_{min}(S, x)*s_d*f_u$
      if $HF(x) > \Theta_1$, then add x to the cache candidates set H.
   end for
   if H == Ø, break.
   for each node $a \in H$
     if $CC[a] < \Theta_2$ ,then add a to the  cache node set C.
   end for
    S re-calculates contention coefficients CC[n].
    S converts G to another directed weighted graph G' by CC[n].
  for each node $z \in V$ & $z \notin C$
    $NC[z] = \arg\min_{b \in C} w(z,b)$
  end for
 end while // V-C
END.
```

Figure 5.3: Centralized Contention-aware Caching Algorithm

5.3.3. Centralized Contention-aware Caching Algorithm

In this subsection, we present a centralized algorithm named Centralized Contention-aware Caching Algorithm (CCCA) to address DCP. CCCA is mainly designed for the case of static topology, such as in a wireless sensor network. CCCA starts with the case that only the data source *S* has the data item *d*. Other nodes access the data item *d* from *S*. Then nodes collect their neighboring sets and calculate their corresponding contention coefficients. Next, each node sends these information to the data source *S*. Then the data source *S* calculates the hedging flows for these nodes and selects the set of cache nodes. Then the data source *S* finds the closest cache node for each node in an iterative way. Simultaneously, the data source *S* updates the date item *d* periodically. Other client nodes access the data item from its nearest cache node or the data source *S*.

CCCA is divided into two steps. In the first step, we select the candidates for the cache nodes. These candidates are selected one by one with **Rule 1**, which means we select nodes whose hedging flows are greater than the threshold Θ_1. In the second step, we select those whose contention coefficients are smaller than Θ_2 as cache nodes with **Rule 2**. Since the set of cache nodes changes with the movement of wireless nodes, the traffic flows are also changed consequently. Then we re-calculate the contention coefficients with the changed traffic flows. It is noted that Θ_2 is normalized in the interval [0, 1]. Let *TF(i)* be the total flows that traverse node N_i. When the data source *S* re-calculates contention coefficients, *TF(i)* is changed to *TF'(i)* and *CC(i)* is changed to *CC'(i)*. We can obtain *CC'(i)* by Eq. (5.12).

$$CC'(i) = \frac{TF'(i)}{TF(i)} CC(i) \tag{5.12}$$

5.3.4. Distributed Contention-aware Caching Algorithm

In this subsection, we describe a localized and distributed implementation of CCCA, namely Distributed Contention-aware Caching Algorithm (DCCA). One of the advantages of DCCA is that it can adapt itself to dynamic traffic conditions in a mobile wireless ad hoc network with low communication overheads, in order to select proper nodes as the cache nodes. We describe DCCA by two algorithms, including the **Algorithm 2** at the side of the data source and the **Algorithm 3** at the client nodes.

5.3.4.1. DCCA at the data source

The data source is responsible for sending the latest updates to all the cache nodes, and maintaining cache consistency among all the cache copies by applying a TTL-based strategy.

There are two key points that should be considered in the design of DCCA. Firstly, the set of cache nodes is changing with the dynamic network topology. Thus, the data source *S* should know which node is in the current set of cache nodes in order to reduce the message cost for sending the redundant updates to the node that was one of the cache nodes but is not a cache node now. Under such a condition, each cache node should notify the event that it resigns as a cache node. Therefore, if a cache node decides not to be a cache node after detecting the condition of contentions around itself, it sends a *CacheLeave* message to the data source *S*.

Secondly, the TTL value maybe expires. How to deal with the queries sent to a cache node that holds an invalid copy of the data item is an important issue. In DCCA, the data source should make each cache node know some of the other cache nodes, i.e., the backup cache nodes, which are also maintained as a node list at each cache node. If a client node sends a query to its closest cache node that holds an invalid cache copy, the query should be forwarded to one of the backup cache nodes, which maybe hold a valid cache copy.

Algorithm 2: DCCA at the data source

Notation:

- *G[n][n]*: the adjacency matrix of the network topology.
- *CC[n]*: the set of contention coefficients for all the nodes.
- *C[n]*: the set of cache nodes.
- *NL[i]*: the neighboring list of a node.

```
(A) Receives a data update.
    Send CacheUpdate(C, d, version) to each cache node in C by unicast.
(B) Receives CacheRequest(i)
    if C[i]!=1, then include $N_i$ into C and send CacheReply(ACK, d, version) to $N_i$.
(C) Receive CacheLeave(i)
    Remove $N_i$ from the cache node set C.
(D) Receive TopologyBroadcast(NL[i], C[i], VF[i])
    Update G[n][n] by NL[i].
(E) Receive AccessRequst(j)
    Send AccessReply(d) to node $N_j$.
```

Figure 5.4: DCCA at the data source

Therefore, there are four operations in the **Algorithm 2** at the data source. First, the data source *S* sends a *CacheUpdate* message to all the cache nodes once the data item changes. Second, if the data source receives a *CacheRequest* message from a node, it sends a *CacheReply* message to answer the request. Third, if the data source receives a *CacheLeave* message, it removes the node from the cache set. Fourth, the data source also can update the topology information according to the received *TopologyBroadcast* message in the **Algorithm 3**. The details are describes in the **Algorithm 2**.

5.3.4.2. DCCA at the client node

In this subsection, we describe DCCA at the client node. Each node in the network is responsible for detecting the changes of the network topology and its contention level, in order to decide whether it is should be a cache node by local information. There are several key operations for each client node.

Firstly, each node collects its neighbour set every T_{top} seconds. Then each node broadcasts a *TopologyBroadcast* message to its one-hop neighbours. The *TopologyBroadcast* message for N_i contains the list of its neighbouring node *NL(i)*, the flag of cache nodes *C[i]* (1 means N_i is one of the cache nodes, otherwise 0), and its own access frequency $f_a(i)$ to the data item *d*.

Secondly, each node collects its contention coefficient every T_{con} seconds. If the contention coefficient *CC[i]* at N_i is smaller than Θ_3, N_i calculates its hedging flow *HF(i)*. If *HF(i)* is greater than Θ_4, N_i broadcasts a *ContentionBroadcast* message to its one-hop neighbours. The *ContentionBroadcast* message consists of its contention coefficient *CC[i]* and its hedging flow *HF(i)*. The *ContentionBroadcast* message is broadcast with a timeout T_{cb}. If N_i receives the *ContentionBroadcast* from all its neighbours within T_{cb}, N_i compares its own *HF(i)* with that of its neighbours. If *HF(i)* is the maximum one, N_i sends a *CacheRequest* message to the data source *S* with a timeout T_{cr}. If the data source *S* sends back a *CacheReply* message, which includes an acknowledgment and the latest update of the data item *d*, N_i updates its state to open a cache node and updates the cache copy with the latest version. Then N_i broadcasts an *OpenCacheNode* message to its neighbors. If N_i does not receive a *CacheReply message within* T_{cr}, N_i waits for the next period of contention detection.

Thirdly, if any node forwards a *CacheUpdate* message, it can select a specified number of cache nodes as its backup cache nodes. If a client node knows that its current closest cache node holds an invalid copy of the data item *d*, it can send the query to one of the backup cache nodes. This operation is also carried out at each cache node for forwarding a query when its cache copy is invalid.

Fourthly, each client node updates its nearest cache node when it receives an *OpenCacheNode* message or a *CloseCacheNode* message, whether the message is sent to it or not.

```
Algorithm 3: DCCA at the client node $N_i$
Notation:
  • $T_{top}$: the timer for topology detection
  • $T_{con}$: the timer for contention detection
  • $T_{cb}$: the timeout for receiving ACK messages from the data source $S$.
  • $T$: the time-to-live value for each copy
  • $\Theta_3$: the threshold of contention coefficient
  • $\Theta_4$: the threshold of hedging flow
  • $C[n]$: the set of cache nodes
  • $NL[i]$: the neighboring list of $N_i$
  • $NC[i]$: the nearest cache node for $N_i$
  • $VF[i]$: the valid flag of the cache copy at $N_i$.
  • backup_num: the number of backup cache nodes in $BL[i]$
  • $BL[i]$: an ids list of backup cache nodes when the local copy is invalid

(A) Procedure TopologyDectection()
    Broadcast TopologyBroadcast(NL[i], C[i], VF[i], $f_a(i)$) every $T_{top}$ seconds.
(B) Procedure ContentionDectection()
    Calculate its contention coefficient CC[i] and hedging flow HF(i) every $T_{con}$ seconds.
    if $CC[i]<\Theta_3$, & $HF(i)>\Theta_4$, then broadcasts ContentionBroadcast(CC[i], HF(i)) with a timeout $T_{cb}$.
    else if $C[i]==1$, then send CloseCacheNode(j) to the data source s.
(C) Receive TopolgoyBroadcast(NL(j), C[j], VF[j])
    if $VF[j]==true$ & $C[j]==1$, then update NC[i].
(D) Receive ContentionBroadcast(CC[j], HF(j))
    if $HF(i)<HF(j)$, then send AckContention(true)
    else send AckContention(false)
(E) Receive AckContention(T)
    if $T_{cb}>0$ & receive AckContention(true) from all of its neighbours, then send CacheRequest(i) to the data source s and set a waiting timeout $T_{cr}$.
    else wait for the next ContentionDectection().
(F) Receive CacheReply(C, d, version)
    if $T_{cr}>0$, then update its cache copy and broadcasts OpenCacheNode(i) to its one-hop neighbours.
(G) Receive OpenCacheNode(j)
    if $w(i,j)<w(i,NC[i])$, then $NC[i]=j$.
(H) Receive AccessRequst(j)
    if $C[i]==1$ & $VF[i]==True$ , then send AccessReply(d) to node $N_j$.
    else if $BL[i] \;!= \emptyset$, then forward AccessRequest(j) to BL[i].
        else send AccessRequest(j) to S.
(I) Receive CacheUpdate(C, d, version)
    if $C[i]==1$, then update data d and set the TTL value T.
    select backup_num closer cache nodes from C into BL[i].
(J) Receive CloseCacheNode(j)
    if $NC[i]==j$, then select the nearest cache node from BL[i].
```

Figure 5.5: DCCA at the client node

5.4. Performance Analysis

In this section, we develop a simple analytical model to analyze the performance of the proposed cache management strategy. Specifically, we want to estimate average query delay and caching overheads for the proposed scheme. We focus on the performance of the proposed algorithms evaluated in terms of the above two metrics. All the notations used in our analytical model are showed in Table 5.3.

Table 5.3: Notations used in the analytical model

t_u	time instant of a data update
μ	The mean of the exponential distribution in the updating process (i.e.,f_u)
T	timeout in the TTL consistency model
λ_i	node N_i's query arrival rate (i.e.,$f_a(i)$)
$\bar{h}_1$	average path length between the data source and the cache nodes
$\bar{h}_2$	average path length between the querying node and the closest cache node
r_0	the maximum node transmission range
$E[d]$	the average access delay for all the mobile nodes accessing all the data items
C_i	the flag to show whether the node N_i is a cache node.
E_1	the event that the contention coefficient *CC[i]* at the node N_i is smaller than the threshold Θ_3
E_2	the hedging flow *HF[i]* at the node N_i is greater than the threshold Θ_4.

There are some assumptions to be used in our analytical model. They are described as follows:

- The MAC protocol used in our model is based on a contention-based protocol.
- The time between updates to the data item *d* is assumed to follow a Poisson distribution with the mean μ.
- Node N_i generates a query according to a Poisson distribution with the mean λ_i.
- The process of querying at each node is a Poisson process and they are independent with each other.

5.4.1. Average Access Delay

There are two steps in the process of evaluating average access delay. In the first step, we will evaluate the average hops between two mobile nodes in a rectangular topology. In the second step, we consider the average access delay in two cases. Under the first case, the timeout *T* in our TTL consistency model is greater than the mean μ in the update model. Under the second case, the timeout *T* in our TTL consistency model is smaller than the mean μ in the update model.

5.4.1.1. Average Path Length

In our model, we assume a rectangular topology with area a×b and the uniform distribution of mobile nodes like the setting in [BE03]. Two nodes can form a link if the distance *s* between them is smaller than r_0, where r_0 is the maximum node transmission range. We want to evaluate the expected number of hops between any two nodes. Using the stochastic geometry, the probability function of *s* is given in [BE03] as

$$f(s) = \frac{4s}{a^2b^2}\left(\frac{\pi}{2}ab - as - bs + 0.5s^2\right) \tag{5.13}$$

for 0≤s<b<a. Let *E[H]* be the expected number minimum number of hops between any two nodes in the network. Let $SL=hr_0$. Then *E[SL]* can be obtained by the following equation [BE03]:

$$\begin{aligned} E[SL] = {} & \frac{1}{15}\left[\frac{a^3}{b^2} + \frac{b^3}{a^2} + \sqrt{a^2+b^2}\left(3 - \frac{a^2}{b^2} - \frac{b^2}{a^2}\right)\right] + \\ & \frac{1}{6}\left[\frac{b^2}{a} ar\cosh\frac{\sqrt{a^2+b^2}}{b} + \frac{a^2}{b} ar\cosh\frac{\sqrt{a^2+b^2}}{a}\right] \end{aligned} \tag{5.14}$$

And the value of *E[H]* can be evaluated by the following equation

$$E[H] \geq E[SL]/r_0 \tag{5.15}$$

Therefore, we use *E[H]* to evaluate the average path $\overline{h}_1$ length between the data source and the cache nodes. In the same way, the average path length $\overline{h}_2$ between the querying node and the closest cache node is also can be evaluated.

5.4.1.2. When *T>μ*

In this case, each node can access the most updated data item *d* because our TTL consistency model guarantees the valid copy of the data item at each cache node. Let *E[d]* be the average access delay for all the mobile node accessing all the data items.

$$E[d] = \frac{1}{n}\int_0^t \sum_{i=0}^{n} (C_i \times \lambda_i \times \overline{h}_1 \times E[d_i^c])dt \tag{5.16}$$

In Eq. (5.16), the contention delay d_i^c is defined in Eq. (5.5).

5.4.1.3. When *T≤μ*

In this case, it is possible that a cache copy at one of the cache nodes may be invalid in our TTL consistency model before an update comes. In our proposed algorithm, if the query cannot be responded when the cache copy is invalid, the receiver should forward the query to one of its backup cache nodes or the data source. Therefore, the average access delay for all the mobile nodes accessing all the data items can be evaluated by the following equation:

$$E[d] = \frac{1}{n}\int_0^t \sum_{i=0}^{n} (C_i \times \lambda_i \times (\overline{h}_1 + \overline{h}_2) \times E[d_i^c])dt \tag{5.17}$$

5.4.2. Caching Overheads

In general, there are seven kinds of messages in our caching system, including *TopologyDectection*, *ContentionBroadcast*, *AckContention*, *CacheRequest*, *OpenCacheNode*, *CloseCacheNode* and *CacheReply*.

Since we detect the topology information periodically, the total number O_{top} of the *TopologyDectection* messages can be evaluated by the following equation:

$$O_{top} = n \times \int_0^t \frac{1}{t_{top}} dt \tag{5.18}$$

To evaluate the total number O_{con} of the *ContenttionBroadcast* messages, we define two events, E_1 and E_2. E_1 is the event that the contention coefficient *CC[i]* at node N_i is smaller than the threshold Θ_3. E_2 is the event that the hedging flow *HF[i]* at node N_i is greater than the threshold Θ_4. Let $P(E_1)$ be the probability of the event E_1, and $P(E_2)$ be the probability of the event E_2. Then we can obtain the following equation:

$$O_{con} = n \times \int_0^t \frac{1}{t_{con}} \times P(E_1) \times P(E_2) dt \tag{5.19}$$

To evaluate the total number O_{ack} of the *AckContention* messages, we define another event E_3, which is the event that a node receives all the *AckContention* within the timeout T_{cb}. Let $\overline{N}$ be the average number of a node's neighbours. Then we can obtain the following equation:

$$O_{ack} = n \times \int_0^t (P(E_1) \times P(E_2) \times P(E_3) \times \overline{N}) dt \tag{5.20}$$

With respect to other messages, the total caching overheads mainly consists of the three messages we have discussed. Thus, the total caching overhead O can be evaluated by the following equation:

$$O \approx O_{top} + O_{con} + O_{ack} \tag{5.21}$$

5.5. Simulations

In this section, we demonstrate the performance of our proposed algorithms over randomly generated network topologies through simulations. We first compare the performance of CCCA with that of POACH in a wireless network with static topology, in term of average query delay and caching overheads. Then we compare the performance of the solutions returned by NOCACHE, POACH and DCCA in mobile networks. Here NOCACHE is a simple strategy that has no cache node in the network.

Our simulations are carried out by using the NS2 simulator. The NS2 [FV97] simulator contains many models for common ad hoc network routing protocols, IEEE Standard 802.11 MAC layer protocol, and two-way ground reflection propagation models [BMJ$^+$98]. The DSDV routing protocol [CB94] is used in our work to provide routing services.

We simulated our algorithm in a network of randomly placed 100 nodes in an area of 2000x500m^2. Note that the normal radio range for two directly communicating nodes in the NS2 is about 250 meters. In our simulations, we set node 0 as the data source S and the size of data item d is 1500 bytes.

Each network node is a client node. Each client node in the network sends out a single stream of read-only queries for a data item maintained by a data source. Each query is essentially an http request for the data item d with a timeout to guarantee a valid reply.

We also apply a simple TTL-based caching consistency strategy in our simulation. The value of the timeout T is 8 seconds.

5.5.1. Static Networks

In this subsection, we evaluate the relative performance of CCCA and the previous work POACH [NSC03].

In our simulations, the time interval between two consecutive queries is known as the *query generation time* and follows a Poisson distribution with the mean value T_q which we vary from 2 to 12 seconds. In our update model, the time between two consecutive updates is defined as the *update generation time* and follows a Poisson distribution with the mean value T_u, which we vary from 2 to 12 seconds. We set the timeout for each request as 20 seconds.

We measure two performance metrics for comparison of various cache placement strategies, viz., average query delay and caching overheads. Query delay is defined as the interval between the time when a client node sends its data requests out and the time when the client node receives the reply from its closest cache node. Average query delay is the average of query delays over all queries sent by all the nodes. Caching overheads includes all of the data packets in our caching system, viz., data requests, data replies, data updates and other messages for caching systems. Note that these packets do not include routing packets because the two strategies use the same routing protocol DSDV. In our CCCA, we set the hedging flow threshold Θ_1 as 100.0 and set the contention threshold Θ_2 as 0.04.

5.5.1.1. Average Query Delay

Figure 5.6 and Figure 5.7 show the results of comparison between POACH and CCCA in terms of average query delay. In Figure 5.6, we vary the mean query generation time while keeping the mean update generation time T_u constant at 2 seconds. We observe that our CCCA outperforms POACH constantly for all mean query generation time. The difference is much more significant for lower mean generation time, which suggests that CCCA is more suitable for higher frequency of data access. In Figure 5.7, we vary the mean update generation time while keeping the mean query generation time constant at 4 seconds. The results shows that CCCA also outperforms the POACH.

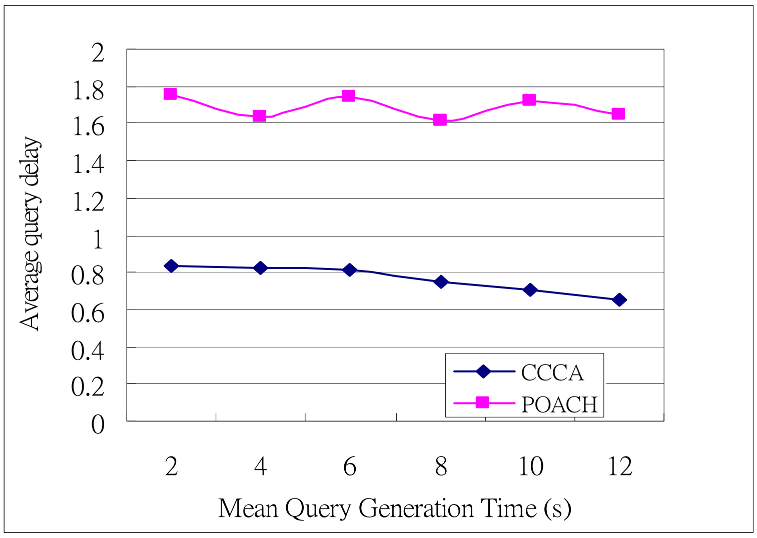


Figure 5.6: Average query delay vs. Mean query generation time.

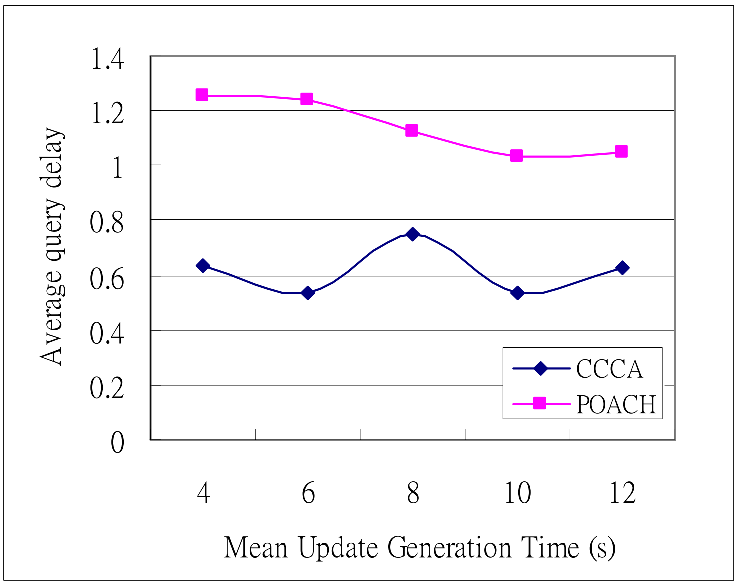


Figure 5.7: Average query delay vs. Mean update generation time.

5.5.1.2. Caching Overheads

Figure 5.8 and Figure 5.9 are presented to show the performance comparison between POACH and our CCCA in terms of caching overheads. In Figure 5.8, we vary the mean query generation time while keeping the mean update generation time T_u constant at 2 seconds. We observe that CCCA has a better performance than POACH constantly for all mean query generation time. In Figure 5.9, we vary the mean update generation time while keeping the mean query generation time constant at 4 seconds. Figure 5.8 and Figure 5.9 show CCCA outperforms POACH, especially in the case with higher update and access frequency.

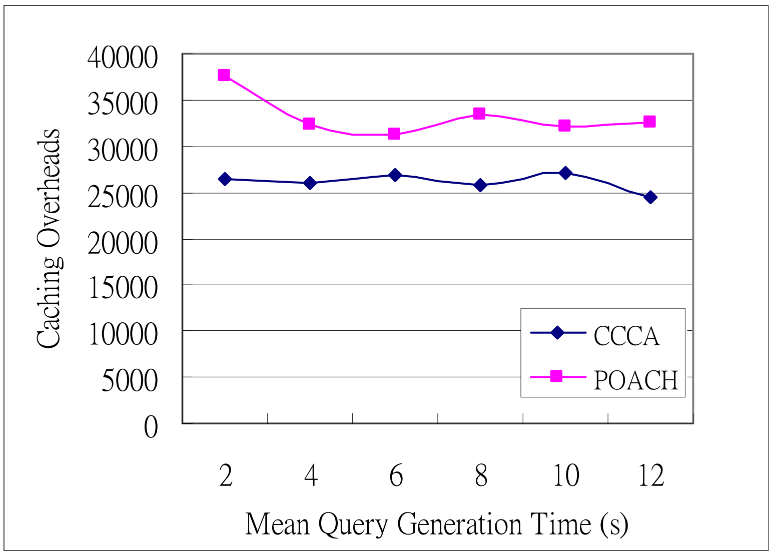


Figure 5.8: Caching Overheads vs. Mean query generation time.

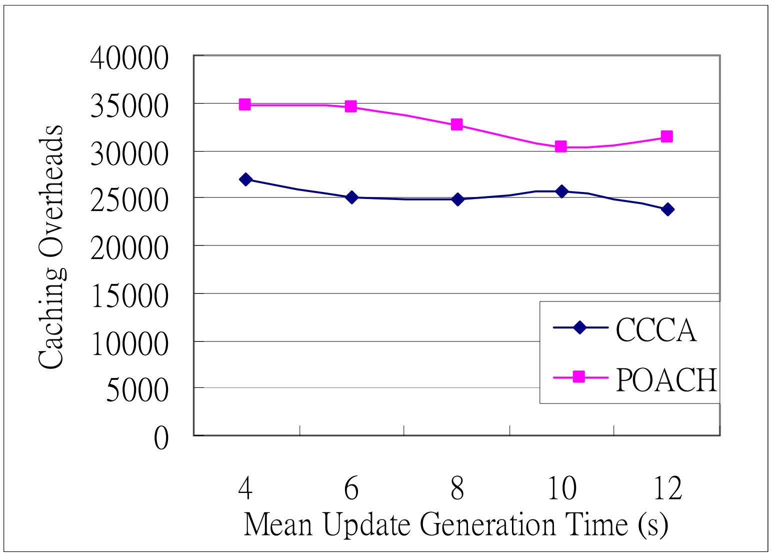


Figure 5.9: Caching Overheads vs. Mean update generation time.

5.5.2. Mobile Networks

In this subsection, we evaluate the performance of NOCACHE, POACH and DCCA in mobile networks. Note that we still use POACH in a centralized manner due to the unpractical factors to implement a distributed version in a mobile ad hoc network. To implement POACH in a distributed manner, each node needs to maintain some progeny and distance information, as mentioned in [NSC$^+$06]. However, these information are dynamic in a mobile network so that any error can result in the failure of the execution of POACH. Moreover, the distributed version of POACH requires global topological information and disseminating such information in a low-cost manner. It is not a good choice in a mobile ad hoc network.

In our simulations, mobile nodes move based on the random waypoint model in NS2. In this model, each node selects a random destination and moves towards the destination with a speed selected randomly from ($0m/s,\ v_{max}m/s$). After the node reaches its destination, it pauses for a period of time (chosen to be 20 seconds) in our simulation and repeat the movement pattern. To focus on the impact of mobile speed on the performance of the cache placement algorithms, we set the mean query generation time as 2 seconds and set the mean update generation time as 30 seconds. We set the contention threshold Θ_3 as 0.04 and set the hedging flow threshold Θ_4 as 100.

Due to the mobility of wireless nodes, data queries may be lost, or the data reply may be lost, or the query delay is beyond the timeout that mobile nodes can endure. Therefore, we introduce another metric named query success ratio besides average query delay and caching overheads in mobile networks. The query success ratio is defined as the percentage of the queries that receive the requested data item within the query success timeout period. We set the timeout of the request as 20 seconds.

5.5.2.1. Average Query Delay

Figure 5.10 shows simulation results of comparing three caching algorithms, viz. NOCACHE, POACH and DCCA in terms of average query delay. The NOCACHE strategy means that there is no data caching in the system. In Figure 5.10, we vary the maximum speed of mobile nodes from 2m/s to 16m/s while keeping the mean update generation time and the mean update generation time. We observe that DCCA outperforms POACH and NOCACHE constantly for all the maximum speed. The results also show that DCCA has a much better performance with higher speed.

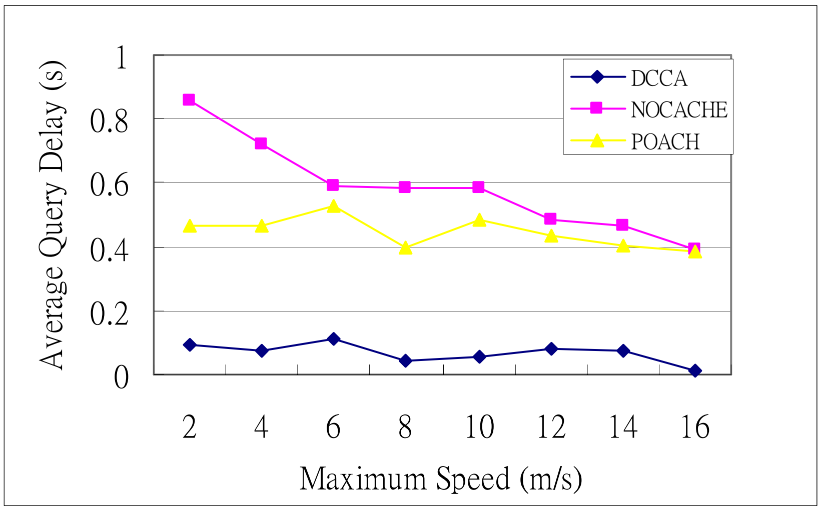


Figure 5.10: Average Query Delay vs. Maximum Speed

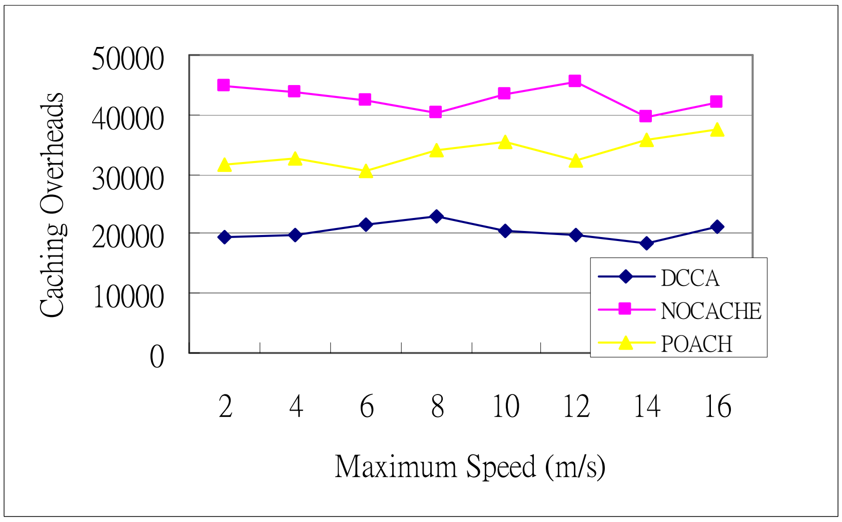


Figure 5.11: Caching Overheads vs. Maximum Speed.

5.5.2.2. Caching Overhead

Figure 5.11 shows simulation results in terms of caching overheads. In Figure 5.11, we vary the maximum speed of mobile nodes from 2 m/s to 16 m/s while keeping the mean update generation time and the mean update generation time. We observe in Figure 5.11 that DCCA outperforms POACH and NOCACHE constantly for all the maximum speed and the performance of DCCA is stable.

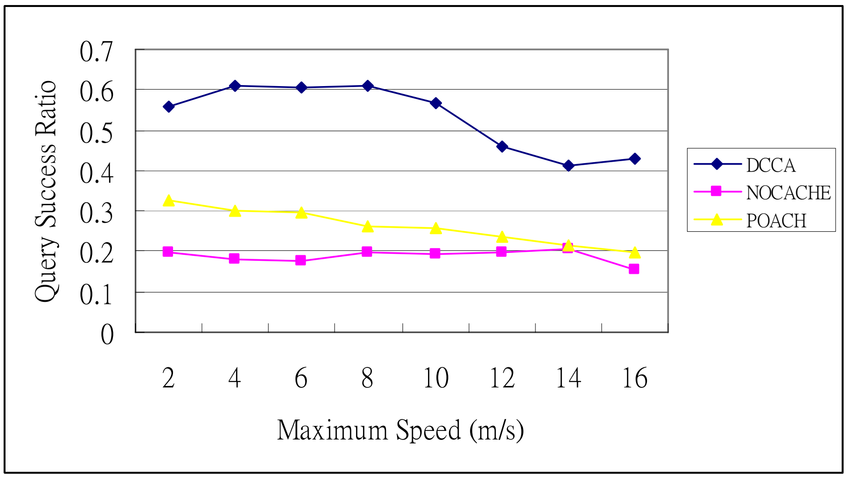


Figure 5.12: Query Success Ratio vs. Maximum Speed

5.5.2.3. Query Success Ratio

Figure 5.12 shows simulation results in terms of query success ratio. We vary the maximum speed of mobile nodes from 2 m/s to 16 m/s while keeping the mean update generation time and the mean update generation time. Although the query success ratio is decreased with higher speed, we still observe that DCCA outperforms POACH and NOCACHE constantly for all the maximum speed.

In summary, simulation results show that CCCA and DCCA both achieves better performance than other alternatives in terms of average query delay, caching overheads, and query success ratio, especially in a mobile environment. Additionally, the performance of CCCA and DCCA are much stable than other alternatives. Here we should clarity the point that there are still some ups and downs from Fig.5.6 to Fig. 5.9 as the mean query or update generation time increases. This is because we now evaluate each point by using the average value of simulations for 5 times. We can reduce this undulation by carrying out our simulations with more times.

5.6. Summary

In this chapter, we addressed the cache placement problem in wireless multi-hop ad hoc networks. We investigate the impact of contentions on the performance of data caching. To the best of our knowledge, this is the first time to consider the contribution of contentions to the total cost in a caching system in wireless ad hoc networks. Since most of the previous work aim to minimize total cost evaluated by the hops of messages, these strategies cannot achieve their expected performance in real applications in wireless multihop ad hoc networks. We present two heuristic rules to minimize the total cost as follows. The first rule aims to reduce the access traffic flows as many as possible and increase the update traffic flows as few as possible. The second rule intends to select cache nodes with fewer contentions from the candidates selected from the first heuristic rule. Based on the two heuristic rules, we present CCCA (Centralized Contention-aware Caching Algorithm) and DCCA (Distributed Contention-aware Caching Algorithm). In our simulations, we evaluate the proposed algorithms in static networks and mobile networks respectively. We take caching overheads, average query delay, and success query ratio as our performance metrics. The results demonstrate that our algorithms outperform the previous work, particularly in more challenging scenarios, such as higher query frequency, higher update frequency and higher mobility.

Chapter 6. Divide-and-Rule Cooperative Caching in IMANETs

In this chapter, we introduce the proposed Divide-and-Rule Cooperative Caching algorithm, DRCC. This chapter is organized as follows. Firstly, Section 6.1 is the overview to this work. Section 6.2 describes the network model and the problem formulation. In Section 6.3, we present the design of our algorithms and mathematically analyze the performance. In Section 6.4, we make an analysis on the key parameters in DRCC in order to obtain the optimal solution. Simulation results are reported in Section 6.5. Finally, Section 6.6 concludes this chapter.

6.1. Overview

In Internet-based Mobile Ad Hoc Networks (IMANETs) [CMC99], mobile nodes may cache data items in a cooperative way in order to reduce total access cost. Cooperative caching [YC06, TGD06] has bee proved as an efficient way to improve the performance of data access in wireless networks. A typical strategy in cooperative caching works as follows. Data sources transfer some data copies to some nodes called *cache nodes*. Other nodes may access data items from the corresponding closest cache nodes, instead of data sources. Consequently, total access cost can be reduced because of the service provided by these cache nodes. However, due to the limited cache space, not all the data items can be cached locally. It is required to consider what data items should be selected at which mobile node to minimize total access cost.

Therefore, “to be selfish, or to be altruistic, that is the question in cooperative caching.” Each mobile node should make caching decision on whether to cache a data item or not. If each mobile node caches data items only according to its own preference, some data items will be only accessed from their data sources. This definitely results in longer access delay. If each mobile node caches data items only for others, this brings more message costs for exchanging the information about access frequencies and the corresponding distances for accessing data items. Consequently, there should be a trade-off between selfishness and altruism in cooperative caching.

In fact, the major problem in cooperative caching is the cache placement problem [TG07, NSC03, TCC07], i.e., how can each mobile node select **a subset of data items** to cache cooperatively in its limited cache so that the total access cost of all the nodes is minimized? Clearly, total access cost mainly depends on two important factors. One is the access frequency for each mobile node accessing each data item. The other is the distance between each mobile node and the corresponding closest cache node for each data item. As we mentioned, there are two important drawbacks in the existing cache placement strategies. Firstly, it is costly for one node collecting the information about other nodes' access frequencies to various data items, and letting others know its own access frequencies. Secondly, the distances between mobile nodes and the corresponding closest cache nodes for multiple data items are dynamic in an IMANET, due to the dynamic topology in an ad hoc network. Additionally, it is also costly to update the information about the distances to the closest cache nodes in time.

In this chapter, we aim at improving the performance of cooperative caching by taking the impact of data access pattern into consideration. There are three key points on the novelty of DRCC. First, we consider the cache placement problem from a new view. Our solution is based on the observation on data access pattern. We find that most of the simulations in previous works apply the Zipf-like distribution as data access pattern and the Zipf-like distribution has been validated in web page requests in practice [BCF$^+$99]. Since data access pattern describes the distribution of access frequency, it is beyond the level of access frequency. Second, the objective of cache placement is to minimize total access delay. DRCC changes the original objective by trying to provide an optimal trade-off between nodes' selfishness and altruism, and make the cache placement problem much easier to be addressed in mobile environments. How to divide the cache space at one node makes the granularity of cache placement upgrade from data item to the whole cache. Third, the difficulty of detecting dynamic distances between client nodes and the closest cache nodes is circumvented by reducing average access hops. DRCC provides a new probabilistic way to evaluate total access delay.

In this chapter, we propose a novel caching strategy for the cache placement problem, in which different data items are treated with different schemes with respect to selfishness and altruism. We aim to provide an optimal trade-off between selfishness and altruism by dividing the limited cache space into two components, including the selfish component and the altruistic component. In the selfish component, each mobile node selects the most frequently accessed T data items according to its own data access frequencies, showing the side of selfishness of a node. On the other hand, each mobile node randomly selects y_i data items from the rest of data items and guarantee y_i follows the Poisson distribution $Po(R)$. The reason why we choose the Poisson distribution will be explained in Section 6.3.2. These data items selected are cached into the altruistic component. Since it is costly to find the accurate hops from a mobile user to the closest cache node due to the dynamic topology, we define *average access hops* as average hops for any mobile node accessing any data item in IMANETs. Therefore, our objective is to minimize total access cost by minimizing average access hops due to the intractability of the original NP-hard problem. We model the relationship between mobile nodes and data items into a *bipartite random graph* [NSW01]. Thus we can evaluate the impact of average access hops on total access cost.

We treat the selection of data items in the selfish component and the altruistic component with different schemes, which enlightens us on the name of our solution, i.e., Divide-and-Rule Cooperative Caching (DRCC). We must clarify that the term "Divide-and-Rule" is different from the one "Divide-and-Conquer" in the way that the latter is to divide a problem into smaller and similar ones and solve each with the *same* way, while the first is to divide a problem into smaller ones and solve each with different ways.

Our new cooperative caching scheme, DRCC, focuses on the problem of how to make full use of limited cache space efficiently at each mobile node so that total access cost can be minimized. Compared with other alternative solutions, DRCC has the following four advantages: (1) DRCC reduces caching overheads to a great degree. Since each mobile node makes caching decision based on its own data access frequencies, there are fewer communication messages among mobile nodes; (2) DRCC is adaptive in the way that it can adjust the spaces allocated for the selfish component and the altruistic component according to the distribution of the data access frequencies. Once the distribution is changed, the allocation strategy is also in tune with the parameter of the distribution; (3) DRCC is with higher scalability. The altruistic component is designed to reduce average access hops between mobile nodes and cache nodes without sending communication messages for exchanging the information about mobile nodes' cache contents. The data items in the altruistic component are randomly selected, but the number of the selected data items at each node follows a specified distribution in order to guarantee the probability of reducing average access hops. To the best of our knowledge, this is a novel way to deal with the relationship between mobile nodes and data items; and (4) DRCC is independent to the mobility models. As we all know, it is not easy to find a mobility model to describe various cases in the real world. DRCC focuses on the relationship between mobile nodes and data items so that it can adapt itself to any mobility model.

6.2. Problem Formulation

In this section, we first introduce the system model for our work, and then we formulate the cache placement problem with multiple data items in IMANETs.

6.2.1. System Model

Let $G(V, E)$ be a graph representing an IMANET with n ($n=|V|$) mobile nodes, $V=\{MN_1, MN_2,..., MN_n\}$. Two mobile nodes communicate directly with each other, which is represented by an edge on the graph. Mobile nodes access m data items in the set $D=\{d_1, d_2, ..., d_m\}$, such as web pages. The size of data item d_i is denoted by SD. Each data item is maintained by only one data source, i.e., a gateway node. We define SR_i as the data source of data item d_i. Moreover, mobile node MN_i accesses the data item d_j with the frequency $p(j)$.

Let $d_{min}(i,j)$ denote the minimum hops for MN_i accessing data item d_j from the closest cache node. The closest cache node for MN_i accessing d_j can be one of the nodes that have the copy of data item d_j or the source node SR_j. The size of MN_i's cache is denoted by S.

To model the cache placement problem in a MANET, we have the following assumptions:

- Gateway nodes serve as the sources of the data items that they obtain from the Internet.
- Links are bidirectional.
- Mobile nodes never fail.
- Each hop delay is the same.

6.2.2. Problem Formulation

The objective of the cache placement problem is to determine which data items should be cached at which mobile nodes so that the total access cost of all the nodes is minimized. In our problem, cache placement is subjected to limited cache sizes at individual nodes. Therefore, the cache placement problem is to select a set of sets of cache nodes $M=\{M_1, M_2, M_3, ..., M_m\}$, where each mobile node in M_j stores a copy of d_j, in order to minimize the total access cost as follows:

$$\tau(G,M)=\sum_{i=1}^{n}\sum_{j=1}^{m}(p(j)\times d_{\min}(i,j)) \qquad \textbf{(6.1)}$$

under the cache space constraint that

$$|\{M_j|i\in M_j\}|\leqq\lfloor S/SD\rfloor, \text{ for all } i\in V,$$

which means that MN_i appears at most $\lfloor S/SD \rfloor$ sets of M.

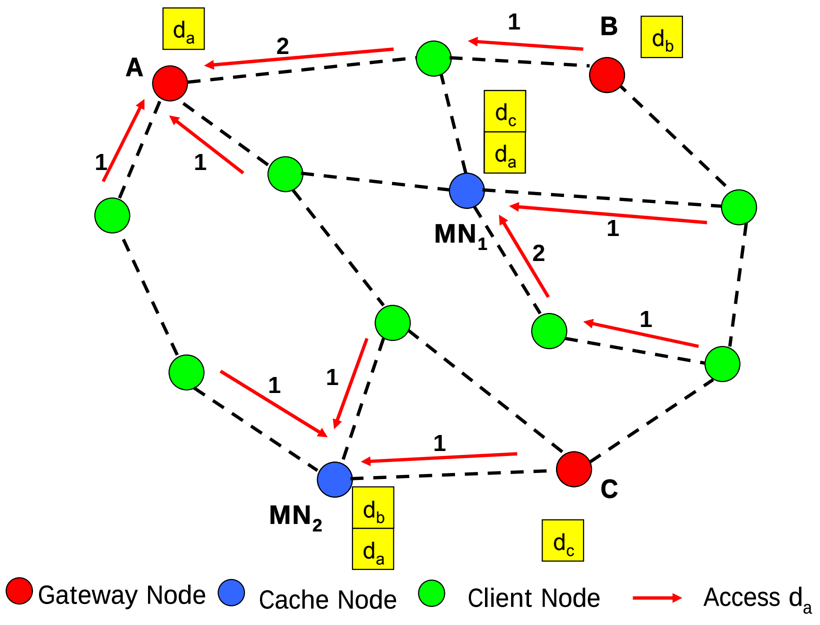


Figure 6.1: An example to explain how to calculate average access hoops.

The cache placement problem has been proved NP-hard [BR01], and can be viewed as a generalization of the facility location problem with multiple types of facilities and constraints on the number of facilities located at a point. Eq. (6.1) shows that the key contributor of the total access cost is the access hops for mobile nodes accessing all the data items, since the access frequencies to data items are normally stable for a mobile node. As we mentioned in the introduction, it is costly and difficult to know the minimum access hops from the requestor to the closest cache node in a dynamic topology.

In this chapter, we use *average access hops*, denoted by h, to replace the minimum hops d_{min} in the objective function, in order to circumvent the intractability of the NP-hard problem. We will provide a probabilistic way to evaluate the impact of average access hops on the total access cost. The proposed solution DRCC is also designed to achieve the optimal trade-off between selfishness and altruism by minimizing average access hops.

We provide an example in Figure 6.1 to explain the meaning of average access hops further. At time t, the topology of an IMANET is described in Figure 6.1. There are 13 mobile nodes in the network. Three gateway nodes, denoted by A, B, and C, maintain three data items d_a, d_b, and d_c. Let h_M be the average access hops for the strategy M. In Figure 6.1, there are two cache nodes MN_1 and MN_2, at which the cache size is equal to 2. In the strategy M, MN_1 caches data items d_a and d_c, while MN_2 caches data items d_a and d_b. Other mobile nodes access the three data items d_a, d_b, and $\mathrm{d_c}$ from their corresponding nearest cache nodes or the corresponding gateway nodes. In Figure 6.1, we just describe the hops for accessing d_a by using the red lines with arrows. Let D_a be the total hops for all the nodes accessing d_a and we find that D_a is equal to 12. In the same way, we obtain D_b and D_c, 18 hops and 16 hops respectively. Then h_M can be obtained by the following equation:

$$h_M = (D_a + D_b + D_c)/(n \times m) = 46/(13 \times 3) \approx 1.18 \tag{6.2}$$

From Eq. (6.2), we know that the strategy M can make every mobile node access every data item with about 1.18 hops in an IMANET. The objective function in Eq. (6.1) can be revised as follow:

$$\tau(G,M)=\sum_{i=1}^{n}\sum_{j=1}^{m}(p(j)\times h_M) \qquad (6.3)$$

6.3. A Divide-and-Rule Solution

In this section, we present our Divide-and-Rule Cooperative Caching (DRCC) algorithm in detail. First, we introduce the basic idea in DRCC. Second, we describe the details of our objective function for evaluating the performance of cooperative caching by DRCC. Third, we present the detailed design of the two components in DRCC with mathematical models. Here we take the Zipf-like distribution [BCF+99] as an example to describe the data access frequencies, and investigate the characteristics of the Zipf-like distribution to find some interesting phenomena for our design. Finally, we present the distributed implementation of DRCC, including the key data structures and the proposed algorithm.

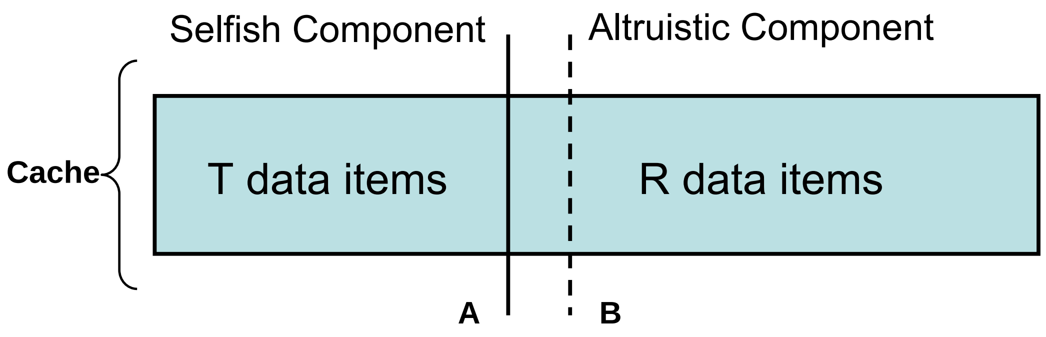


Figure 6.2: Basic idea in DRCC with two strategies A and B

6.3.1. Divide and Rule

To improve the performance of data access, there are normally two kinds of strategies in cooperative caching to reduce the total access cost. In the first strategy, a mobile node caches the data items that it accesses with higher frequencies. The total access cost will be reduced because the cached data items will be obtained without any delay and message cost. Therefore, a mobile node in the first strategy is selfish. In the second strategy, a mobile node caches the data items that are requested by other mobile nodes. This depends on the number of these mobile nodes and the distances from them. If the number of these requesting nodes is great enough, and the distances are small enough, there will be enough benefits for a mobile node to cache a data item at the cost of a part of its cache size. In such a case, a mobile node is altruistic and the total access cost can also be reduced.

In DRCC, we aim to propose a strategy that treats the data items with different caching schemes in terms of selfishness and altruism. We divide the whole cache at each node into two components: the selfish component and the altruistic component. In the selfish component, each mobile node caches the most frequently accessed T data items so that it can access them with the minimum distance, i.e. zero. In the altruistic component, each mobile node cache a group of data items randomly selected from the set of the rest of data items. However, the number of data items in the altruistic component should follow the Poisson distribution $Po(R)$ in order to guarantee the probability that each mobile node can access (m-T) data items. Note that the total number of data items in the cache is constrained by the cache sizes of mobile nodes. We try to make an optimal trade-off between selfishness and altruism.

In Figure 6.2, there are two strategies A and B. The only difference between them is that the sizes for the selfish component and the altruistic component are different. Therefore, the key problem in DRCC is how to design the parameter T and R to find the optimal allocation strategy according to the data access frequencies.

6.3.2. Objective Function

As we mentioned in Section 6.2, we aim to minimize total access cost by minimizing average access hops and considering the constraint of limited cache space. Obviously, reducing average access hops can reduce total access hops if mobile users access all the data items with the same access pattern. In our design, we consider the benefit of caching some data items into the selfish component in order to make mobile nodes access these data items with zero cost. Moreover, we consider the effect of caching some data items randomly into the altruistic component in order to reduce average access hops as few as possible under the constraint of cache size allocated for the altruistic component.

In Figure 6.2, there are two caching strategies of DRCC, including A and B. There are more spaces allocated for the selfish component in the strategy *B* than that in the strategy A. Therefore, the strategy *A* makes mobile nodes access more data items with zero cost but its average access hops is greater than that in the strategy B. There is a trade-off between the selfish component and the altruistic component with respect of selfishness and altruism.

First, we take the extreme case, the strategy *X*, as the opponent to other strategies. In the strategy *X*, all the cache spaces are allocated for the altruistic component. Let h_X be the average access hops. Therefore total access cost can be obtained by the following equation:

$$\tau(G,X)=\sum_{i=1}^{n}\sum_{j=1}^{m}p(j)\times h_X=n\times h_X \tag{6.4}$$

Second, let h_A be the average access hops in the strategy A. Thus total access cost can be obtained by the following equation:

$$\tau(G,A)=\sum_{i=1}^{n}\sum_{j=1}^{T}p(j)\times 0+\sum_{i=1}^{n}\sum_{j=T+1}^{m}p(j)\times h_A \tag{6.5}$$

Obviously, the result of designing the selfish component is that the delay of accessing these data items in the selfish component is zero. Therefore, we can obtain Eq. (6.6) from Eq. (6.5).

$$\tau(G, A) = n \times h_A \times (1 - \sum_{i=1}^{T} p(i)) \tag{6.6}$$

In Section 6.3.3, we will discuss how to obtain average access hops h_A by our mathematical model. In Section 6.4, we will discuss how to find the optimal parameters for the strategy A.

6.3.3. Two Components in DRCC

In this subsection, we first take the Zipf-like distribution as an example to describe data access frequencies. Then we explain the detailed design of two components respectively. Additionally, we also provide the mathematical models for both components.

6.3.3.1. The Zipf-like Distribution

To show the design of DRCC, we take the Zipf-like distribution as an example. The reason why we select it is that the Zipf-like distribution is considered as the most famous mathematical model of web requests [BCF+99]. Moreover, there are many simulation works in cooperative caching which use the Zipf-like distribution as the data access pattern. However, note that DRCC does not depend on the Zipf-like distribution. Given any distribution, we can analyze the performance of DRCC mathematically.

Here, each mobile node accesses all the data items by following the Zipf-like distribution $Z(\theta)$ as follows:

$$p(k) = \frac{1}{k^{\theta} \sum_{i=1}^{m} \frac{1}{i^{\theta}}} \tag{6.7}$$

where the probability $p(k)$ of a request for the k-th popular data item is proportional to $1/k^{\theta}$ and $0 \le \theta \le 1$.

Firstly, let us observe some interesting phenomena in the Zipf-like distribution. In Figure 6.3, there are three instances of the Zipf-like distribution. If θ=1, the distribution follows the strict Zipf distribution, while for θ=0, the corresponding distribution follows the uniform distribution. We are interested in the impact of the Zipf-like distribution on our proposed DRCC. Figure 6.3 shows that three Zipf-like distributions with different values of θ have different impacts on the access probability in terms of the preference ranking of data items. The curve with θ=1 is the most skewed one compared with the other two. It shows that caching the most frequently accessed data items in the selfish component can bring much more benefits to the performance of data caching because the accumulated probability of accessing the top T data items almost accounts for the whole probabilistic space. On the other hand, the straight line with $\theta=0$ shows that the probability for accessing each data item is with the same benefit such that the advantage of caching any data item should be determined by other factors, such as the hops to other mobile nodes.

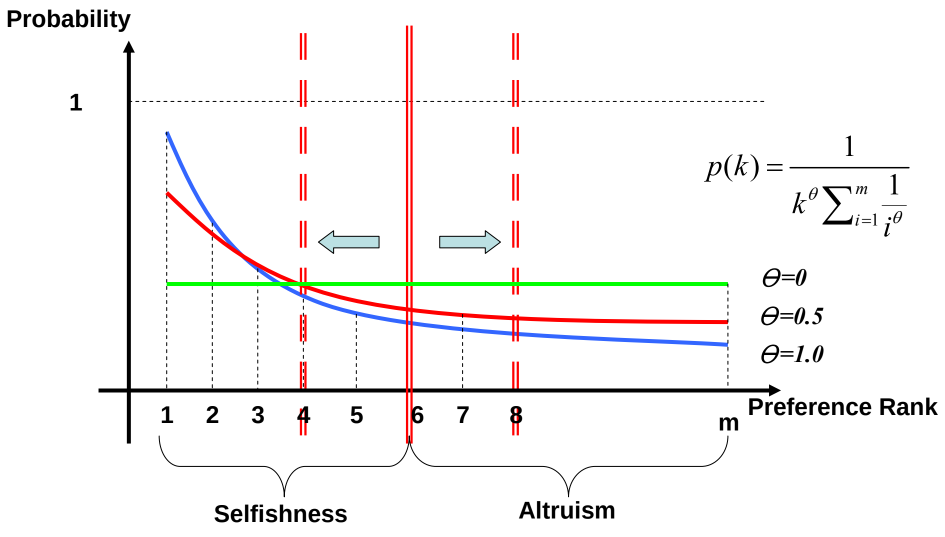


Figure 6.3: Three Zipf-like distributions to show the impact of θ on the strategies of cache placement

Secondly, as Figure 6.3 shows, the selfish component is designed for node itself, which shows the side of selfishness of a mobile node. However, the altruistic component is designed for other nodes, which shows the side of altruism of a mobile node. Therefore, when the parameter θ is smaller enough, we should cache more data items on the altruistic component. On the other hand, when the parameter is greater enough, we should cache more data items in the selfish component.

6.3.3.2. The Selfish Component

In the selfish component, we select the most frequently accessed *T* data items from the set *D* of data items, which means *T* data items are the most popular data items for one node.

Let S_{STA} be the set of data items that are located at the selfish component of MN_i's cache. Thus, S_{STA} is the set of data items $\{d_1, d_2, ..., d_T\}$. Let *AP(k)* be the accumulated access probability for the top *k* data items. *AP(T)* can be obtained by the following equation:

$$AP(T) = \sum_{i=1}^{T} p(i) \tag{6.8}$$

where *p(i)* can be determined by Eq. (6.7).

It is obvious that a mobile node with higher *AP(T)* can reduce its own access delay more. However, this does not consider others' requirements. Some data items will be accessed only at the data sources. The total access cost will be increased by the retransmissions of remote requests due to the long delay or the network partitions.

Another important point is the factor $\sum 1/i^{\theta}$ in Eq. (6.7) has the same style of the *Riemann Zeta-function* ζ(s) [RZF].

$$\varsigma(s) = \sum_{i=1}^{\infty} \frac{1}{i^s} = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + ... \qquad \textbf{(6.9)}$$

where s is a complex variable. Eq. (6.9) is a *Dirichlet series* which converges absolutely to an analytic function on the open half-plane of s such that $Real(s)>1$ and diverges elsewhere. It is clear we cannot obtain an analytical function in Eq. (6.8) when $0 \le \theta \le 1$. Thus our objective function can not be expressed with an analytical style.

6.3.3.3. The Altruistic Component

As we mentioned in the introduction, it is costly for each mobile node to detect the variation of the dynamic distances to the closest cache nodes. Our method is to reduce average access hops with a high probability, not to find the path with the minimum access hops for data items. This is much different from the existing cache placement strategies. According to the definition, average access hops depend on the topology and the data items in the altruistic component. As we all know, in an IMANET, we cannot predict the changes of the topology. Therefore, we try to reduce average access hops by increasing the number of data items in the altruistic component, so that the probability that a mobile node can access a data item can be increased.

The data items in the altruistic component are randomly selected in order that other mobile nodes can access these data items with high probability. This is the side of mobile nodes' altruism. Obviously, the more data items we select in the altruistic component are, the more possibly mobile nodes can access any data item within fewer hops. However, the size of the altruistic component is still subjected to the total cache space.

In the altruistic component, we select y_i data items from the set (D-S_{STA}) in order to minimize average access hops, where y_i follows the Poisson distribution $Po(R)$. We investigate the expectation R with its impact on the probability that each mobile node can access any data item.

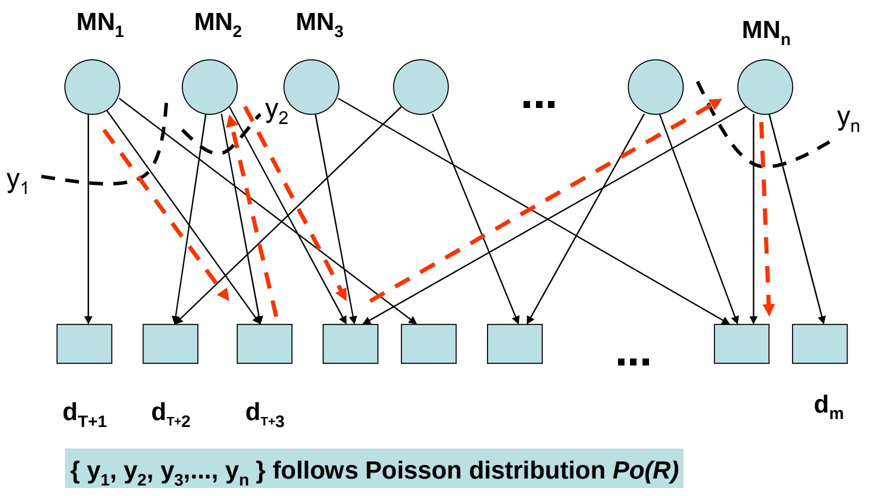


Figure 6.4: A Bipartite Random Graph to model the relation between mobile nodes and data items

We propose a *Bipartite Random Graph* [NSW01] to model the result of randomly selecting data items from a data set. We select average R data items from (m-T) data items by following the Poisson distribution [NSW01] $Po(R)$. The reason why we choose the Poisson distribution is as follows: 1) the results of randomly selecting data items are under the full control of the parameters we select; 2) the average length of shortest paths between mobile nodes and data items can be obtained easily and well-formedly; and 3) the total access hops in the altruistic component can be balanced because of the even nature of the Poisson distribution. One of our future works is to investigate the impact of other distributions.

As Figure 6.4 shows, we construct a bipartite random graph with n mobile nodes and (m-T) data items. Mobile node MN_i selects y_i data items randomly from the set of data items (D-S_{STA}). Thus we obtain a degree sequence $\{ y_1, y_2, y_3, \ldots, y_n \}$ for all the mobile nodes, and y_i follows the Poisson distribution *Po(R)* with the expectation *R*. Therefore, each mobile node caches a subset of the set ($D\text{-}S_{STA}$) of data items with the average size *R* data items. Hence it can be deduced that each data item is also cached by a subset of mobile nodes *V* with the average size *v*. The relationship among the parameters: *m-T*, *n*, *R* and *v* can be described by the following equation:

$$\frac{R}{m-T} = \frac{v}{n} \tag{6.10}$$

As Figure 6.4 shows, we have two sets of nodes in each part in the bipartite random graph. One is the set of mobile nodes, and the other is the set of data items. By following this way, a bipartite random graph consists of the two parts. Let p_j be the probability distribution of the degree of the mobile node MN_i. We also define q_k as the probability distribution of the degree of the data item d_k. Therefore, according to the theory of bipartite random graph, we can define two generating functions which generate the above two probability distributions:

$$f_0(x) = \sum_j p_j x^j \tag{6.11}$$

$$g_0(x) = \sum_k q_k x^k \tag{6.12}$$

Since the two distributions are two Poisson distributions, it is trivial that we can obtain the following equations:

$$f_0(x) = e^{R(x-1)} \tag{6.13}$$

$$g_0(x) = e^{v(x-1)} \tag{6.14}$$

If we choose an edge randomly on the bipartite random graph and follow it in both ways to reach the data item and the mobile node, the distribution of the number of other edges leaving those two nodes is generated by the following equations:

$$f_1(x) = \frac{1}{R} f_0'(x) \tag{6.15}$$

$$g_1(x) = \frac{1}{\nu} f_0'(x) \tag{6.16}$$

Now we can define the generating function *$G_0(x)$* for the distribution of the number of ***co-items*** (i.e., the data items that are cached by two mobile nodes) of a randomly selected mobile node as:

$$G_0(x) = f_0(g_1(x)) = e^{R(e^{\nu(x-1)}-1)} \tag{6.17}$$

If we randomly select an edge on the bipartite random graph, the distribution of the number of co-items of the mobile node to which it leads can be generated by the follow function:

$$G_1(x) = f_1(g_1(x)) = e^{R(e^{\nu(x-1)}-1)} \tag{6.18}$$

Let z_k be the average number of the neighbours who are k hops away from a randomly selected mobile node. The generating function for z_m can be found as follows:

$$G^k(x) = \begin{cases} G_0(x) & for\ k = 1 \\ G^{(k-1)}(G_1(x)) & for\ k \geq 2 \end{cases} \tag{6.19}$$

Then the average number z_m of the k^{th} nearest neighbours can be obtained by the following equation:

$$z_k = \frac{dG^{(k)}}{dx}\Big|_{x=1} = \left[G_1'(1)\right]^{k-1} G_0'(1) = \left[\frac{z_2}{z_1}\right]^{k-1} z_1 \tag{6.20}$$

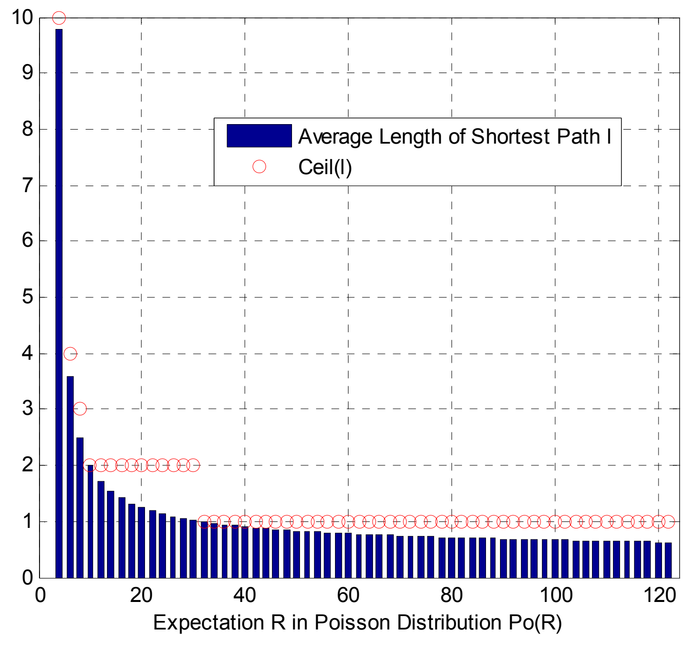


Figure 6.5: Analytical results for average length of the shortest path l, m=1000, n =100, T=0.

Here we are mainly interested in the average length *l* of the shortest path between two randomly selected mobile nodes on our bipartite random graph. The path length can be estimated approximately when the total number of neighbours of a mobile node out to that distance is equal to the total number of the mobile nodes on the bipartite random graph, i.e., when

$$1+\sum_{k=1}^{l} z_k = n \tag{6.21}$$

From Eq. (6.15) and Eq. (6.16), we know that $z_1=nR^2(m-T)$ and $z_2=z_1^2$. Therefore, the average length *l* can be obtained as follows:

$$l=\frac{\log n}{\log(nR^2)-\log(m-T)} \tag{6.22}$$

To investigate the relation between average length *l* and the expectation *R* in *Po(R)*, we present the analytical results in Figure 6.5. To make the results more clear, we also describe the relation among $\lceil l \rceil$ and *R*. As Figure 6.5 shows, average length *l* drops sharply at the beginning of increasing *R*. However, from some point, average length *l* is increased slowly, which means that adding more data items to the altruistic component cannot make more benefits any longer.

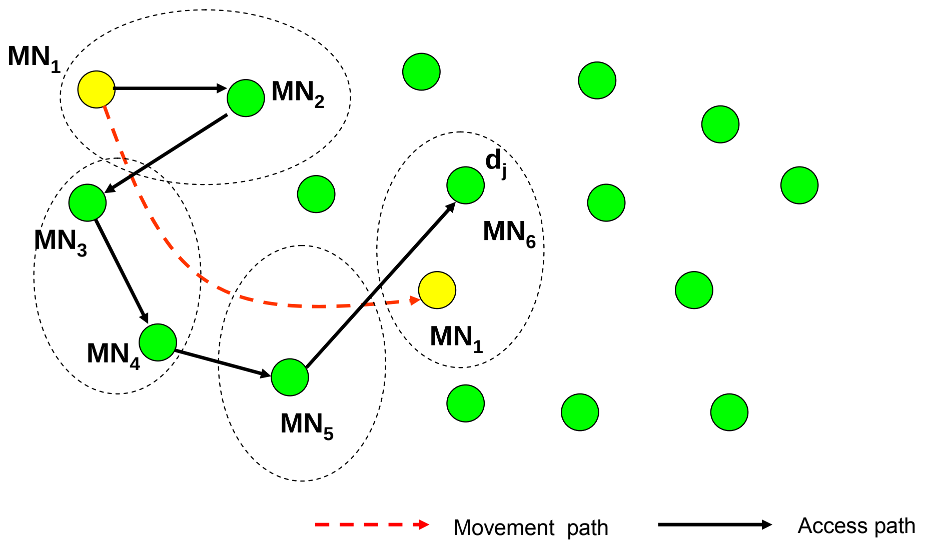


Figure 6.6: A sample process for a mobile node accessing a data item in movement

In DRCC, average length *l* is used to evaluate the probability that each mobile node can access any data item by encountering a specified number of other mobile nodes, i.e., *if we randomly select one mobile node and one data item, we find that the shorter the shortest path is, the more possibly the mobile node accesses the data item.* In Figure 6.6, we present an example that describes the process for the mobile node MN_i accessing the data item d_j. MN_1 firstly encounters MN_2 and d_j may be in the cache of MN_2. Then MN_1 encounters MN_3 and MN_4 at the same time. Then MN_1 comes across MN_5. Finally, MN_1 meets with MN_6 and finds the requested data item d_j. In an IMANET, it cannot be predicted which mobile node will come across. However, given enough conditions, we can know the probability that a mobile node encounters how many mobile nodes during a time period when moving. This is beyond the scope of this chapter. In this chapter, we mainly focus on the impact of the number of mobile nodes encountered on the length of path for a mobile node accessing a data item.

Moreover, in Figure 6.4, we observe that the access path on a bipartite random graph follows the style that two terminal nodes attached to each edge are located in two parts, like a zigzag. In an IMANET, a normal path for a mobile node accessing a data item consists of all the intermediate nodes from the sender to the receiver. Therefore, half of the edges of the path on a bipartite random graph can be removed. Average access hops h can be evaluated by the average length l of the shortest paths. We assume that the underlying routing protocols can provide the shortest path for all the mobile nodes. The evaluation is with a very high probability, which converges to 1 if the numbers of mobile nodes and data items are infinite. We present average access hops h by the following equation:

$$h = \frac{1}{2} \times \frac{\log n}{\log(nR^2) - \log(m - T)} \tag{6.23}$$

From Eq. (6.23), we know that each mobile node can access every data item that are not cached locally within h other mobile nodes with a very high probability. We can increase the value of R in order to reduce the value of h. However, the cache sizes at each mobile node are limited, we should increase the number of data items in caches under the constraint of the cache sizes. The selection of the parameters in DRCC will be discussed in Section 6.4.

6.3.4. Divide-and-Rule Cooperative Caching

In this subsection, we present a distributed algorithm named Divide-and-Rule Cooperative Caching (DRCC) to solve the cache placement problem with multiple data items in IMANETs.

6.3.4.1. Key Data Structures

Closest Cache Tables are the most important data structure in DRCC. Mobile node MN_i maintains the closest cache node MN_j that has a copy of the data item d_k for each data item in the network. MN_i also maintains a closest_cache table, where an entry in the closest_cache table is of the form (d_k, MN_j) where MN_j is the closest cache node that has the copy of d_k, known by MN_i. Note that if MN_i is the data source SR_k or has cached d_k, MN_i is the closest cache node for d_k.

Waiting Node Lists are another key data structure in DRCC. Each data source SR_i maintains a waiting list for the data item d_i. In the waiting list for data item d_i, there is the set of nodes that have sent the cache requests for requesting to be as a cache node for data item d_i. Data source SR_i does not send the cache copies of d_i to the waiting nodes until the number of waiting nodes meets with the requirement of the parameter ($nR/(m-T)$), which is determined by Eq. (6.10) in DRCC.

Cache Request Lists are maintained by all the mobile nodes. The objective of this data structure is to improve the efficiency of sending cache requests. Since one mobile node maybe act as more than one data source in the network, we call such mobile node as multiple data source. For each multiple data source SR_i, MN_j maintains a list that records the cache requests sent to SR_i. MN_i combines these cache requests into one message. When SR_i sends the corresponding cache replies to these waiting nodes, each of the intermediary nodes can know which node will be its own closest cache node for data item d_i.

```
Algorithm 1: Divide-and-Rule Cooperative Caching Algorithm
SETTING:
    A Network Graph G(V,E) and a data set D with m data items.
BEGIN
    $MN_i$ selects the top T data items from D and sends CacheRequest to the data sources.
    $MN_i$ selects $y_i$ data items randomly ($y_i$ follows Poisson distribution Po(R)) from $D\text{-}S_{sta}$ and sends the CacheRequest to the data sources.
    $MN_i$ sends DataRequest if there is an entry in ClosestCacheTable, else broadcast BcastDataRequest.
    On receiving CacheRequest
        If $MN_i$ is the data source, wait until collets $nR/(m-T)$ CacheRequest and then sends CacheCopy to the requestors.
    On receiving DataReply
        If $MN_i$ is the requestor, checks the timeout $T_{req}$ and if valid, then serves the application and counts the successful access, else discards it.
        Else If local memory has available space, then caches it.
    On receiving CacheCopy
        If $MN_i$ is the requestor, saves the copy.
        Else updates its ClosestCacheTable from CacheNodeList. If local memory has available space, then caches it.
    On receiving DataRequest($d_j$)
        If $MN_i$ holds $d_j$, sends DataReply($d_j$) to the requestors.
        Else If $MN_i$ knows the closest cache node for $d_j$, forwards the request to the node.
    On receiving BcastDataRequest($d_j$)
        If $MN_i$ holds $d_j$, sends DataReply($d_j$) to the requestors.
        Else if $MN_i$ knows the closest cache node for $d_j$, forwards the request to the node and updates ReceivedNodeList.
        Else if Node_Sum in ReceivedNodeList is greater than h, discards the BcastDataRequest and forwards DataRequest to the data source.
        Else broadcasts BcastDataRequest and updates ReceivedNodeList.
END
```

Figure 6.7: Divide-and-Rule Cooperative Caching

6.3.4.2. Caching Strategy

In this subsection, we describe the caching strategy and the corresponding cache replacement strategy in the two components in DRCC. The first one is about how to minimize total access hops. The second is about how to replace data items from cache contents once the cache space is full.

The caching policy of DRCC is as follows. To minimize the total access cost in the selfish component, each mobile node is selfish such that they should cache the most frequently accessed data items for self. Each mobile node selects the most frequently accessed T data items and caches them into the selfish component, which is determined by Eq. (6.10) and Eq. (6.27). Each data item is with different weight by the Zipf-like distribution. We aim at accessing the most popular data items locally so that total access cost in the selfish component is minimized.

Next, each mobile node generates a random number y_i by following the Poisson distribution $Po(R)$, where the parameter R is determined by Eq. (6.26) after we find the near-optimal T in Eq. (6.27). Then each mobile node randomly selects y_i data items from the set of data items, $D\text{-}S_{sta}$.

Then the mobile node MN_i holds a small data set CR_i that including $(T+y_i)$ data items and the total data sizes in CR_i should be smaller than its cache size S. MN_i sends the *CacheRequest* message to the corresponding data source SR_i for data item d_i that is one of the members in CR_i. After the data source SR_i collects ($nR/(m-T)$) *CacheRequest* messages from mobiles nodes. Then SR_i sends the *CacheCopy* message to the requestors. The *CacheCopy* message is composed of two parts: (1) the copy of data item d_i, and (2) the *CacheNodeList* that includes all of the cache nodes which have sent *CacheRequest* messages for data item d_i until now. Consequently, mobile node can know other nodes that cache the same data items. If

MN_i receives any *CacheCopy* message, it updates its own *ClosestCacheTable* to store the closest one.

If MN_i sends a *DataRequest* message for data item d_j with a timeout T_{req}, MN_i first searches the content of its own cache. If d_j is in MN_i's cache, MN_i accesses the data item immediately. Otherwise, MN_i looks up its *ClosestCacheTable* to find the closest cache node that holds the copy of d_j. If MN_i finds the closest cache nodes, it sends the *DataRequest* message to the closest cache node. Otherwise, if there is no any entry in the *ClosestCacheTable*, MN_i sends the *DataRequest* message to the data source SR_j and broadcasts the *BcastDataRequest* message to its neighbours. The *BcastDataRequest* message includes a field named *ReceviedNodeList*, which records the new receivers. Mobile node MN_k receives the *BcastDataRequest* message from MN_i. If MN_k holds the copy of d_j, it sends a *DataReply* message to MN_i. If MN_k knows the closest cache node, it forwards the *DataRequest* message to the node and updates the *ReceivedNodeList* in the *BcastDataRequest* message. Otherwise, MN_i counts the total number *Node_Sum* of mobile nodes in the *ReceivedNodeList*. If the number *Node_Sum* is greater than h in Eq. (6.23), MN_k discards the *BcastDataRequest* message. Otherwise, MN_k updates the *RecivedNodeList* and broadcasts the *BcastDataRequest* message to its neighbours.

If MN_i does not receive the *DataReply* within the timeout T_{req}, it sends the *DataRequest* message again. If MN_i does not receive the *DataReply* for two times, it sends the *DataRequest* message to the corresponding data source.

When the local cache of a mobile node is not full, each mobile node can cache the passing-by data items that it does not cache in its own cache before. When the local cache is full, mobile node MN_i is required to replace a data item randomly selected from its altruistic component. Remember that the data items in the selfish component should not be replaced. MN_i only replaces the data items in the altruistic component.

6.4. Parameter Analysis

In this section, we develop a simple mathematical model to analyze the key parameters in DRCC. According to our analysis, we find a near-optimal solution to allocate cache sizes for the two components, which can minimize the total access cost.

To analyze the performance of DRCC, we first provide how to calculate the total access cost for the strategy A by Eq. (6.6) and Eq. (6.23) as follows:

$$\tau(G,A)=\frac{n}{2}\times\frac{\log n}{\log(nR^2)-\log(m-T)}\times(1-\sum_{i=1}^{T}p(i)) \tag{6.24}$$

Then the cache placement problem in cooperative caching can be mapped to an optimal problem as follows:

$$Minimize \quad \tau(G,A) \tag{6.25}$$

$$s.t. \quad T+R\le\left\lfloor\frac{S}{SD}\right\rfloor \tag{6.26}$$

As we mentioned in Eq. (6.9), we cannot find an analytical function in the form of the sum for the *Riemann Zeta-function*. Therefore, we must make an investigation of the value of the total access cost. To investigate the relation between the total access cost and the two parameters *T* and *θ*, we present the analytical results in Figure 6.8. It shows that the total access cost is increased with increasing the number of data items in the selfish component, or reducing the parameter *θ* in the Zipf-like distribution *Z(θ)*. Given the parameter *θ*, the cache size *S* and the data item size *SD*, the optimal value of *T* lies on the middle of the range from 1 to 100. When *T* is much smaller or bigger, the total access cost will be much greater than the ones when *T* is in the middle of the range [*0, S/SD*].

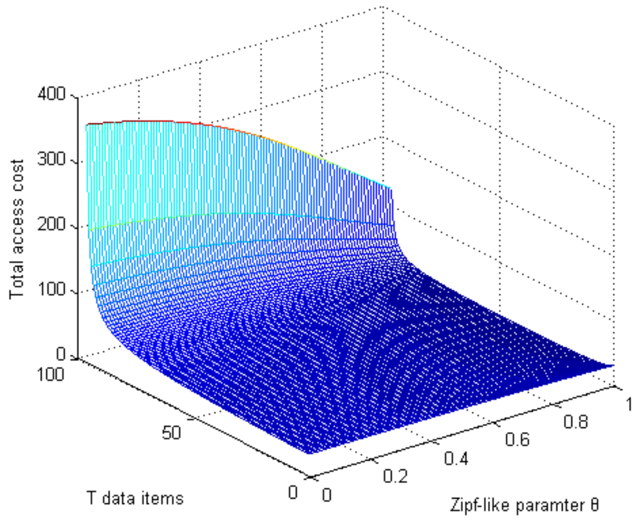


Figure 6.8: Analytical results on total access delay with T data items in the random component and the parameter θ in the Zipf-like distribution, n =100, m=1000.

To make the relation more clear, we find the number of data items *T* when we achieve the minimum total access cost for a specific Zipf-like parameter *θ*. The analytical results are presented in Figure 6.9. It shows that the number of data items, *T*, in the selfish component is nearly linearly proportional to the parameter θ in the Zipf-like distribution.

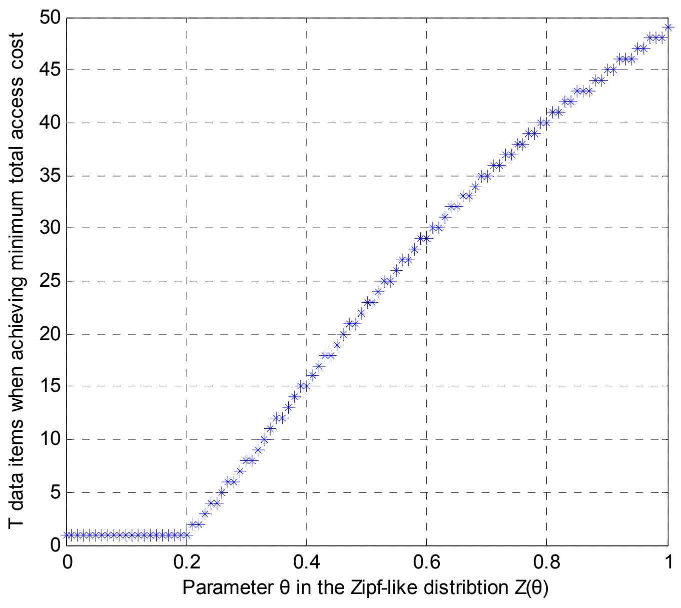


Figure 6.9: Analytical results on the relation between *T* data items in the altruistic component and the parameter θ when achieving the minimum total access cost, n=100, m=1000, S/SD=100

We take our simulation setting as a case study. The number of mobile node n is equal to 100 and the number of data items m is equal to 1000. The cache size S is equal to 75000 bytes and the size of each data item is equal to 750 bytes. We present an approximate result for describing the relation between the optimal T and the Zipf-like parameter θ as follows:

$$T_{opt} \approx 62.5 \times \theta - 10 \tag{6.27}$$

We take one of the settings in our simulations as an example. We take the value of the parameter θ of the Zipf-like distribution as 0.8. We find that the number of data items in the selfish component T is equal to 40, which means that there should be the most frequently accessed **40** data items in the selfish component at the cache of each mobile node, while 60 data items in the altruistic component.

6.5. Simulations

In this section, we demonstrate the performance of DRCC compared with Benefit-Based Data Caching (BDC) [TGD06] through simulations over randomly generated network topologies. To the best of our knowledge, BDC is with the best performance in cooperative caching for multiple data items in mobile environments. We will show the performance of two strategies by presenting the results in terms of the same metrics under the same setting.

6.5.1. Simulation Settings

Our simulations are carried out by using the NS2 simulator [FV97]. The NS2 simulator contains models for common ad hoc network routing protocols, IEEE Standard 802.11 MAC layer protocol, and two-way ground reflection propagation models [BMJ$^+$98]. The DSDV routing protocol [PB94] is used in our work to provide routing services.

6.5.1.1. Network Setup

We simulated DRCC and BDC in a wireless network with randomly placing 100 mobile nodes in an area of 2000x500m^2. Mobile nodes move based on the random waypoint model [BMJ$^+$98] in NS2. In this model, each node selects a random destination and moves towards the destination with a speed selected randomly from ($0m/s$, v_{max} m/s). After the node reaches its destination, it pauses for a period of time (chosen to be 100 seconds) in our simulation and repeat the movement pattern. Note that the normal radio range for two directly communicating nodes in the NS2 is about 250 meters. We assume that there are 1000 data items with the same size 750 bytes. We setup two randomly placed data sources (servers) S_0 and S_1 as the gateway nodes, where S_0 maintains the data items with even IDs and S_1 maintains the data items with odd IDs.

6.5.1.2. Client Query Model

In our simulation, each mobile node is a client node. Each client node sends out a single stream of read-only queries and each query is a request for a data item. We use the http requests as these queries in the simulations. The time interval between two consecutive queries is defined as the query generate time, and follows the exponential distribution with the mean value T_{query}, which we vary from 3 to 40 seconds. The success of data access means that the data reply responds to the data request within a timeout. We set the timeout as 40 seconds, same as BDC.

6.5.1.3. Data Access Pattern

Data access pattern means the distribution for mobile node accessing a group of data items. As we mentioned at the beginning of this chapter, we assume that mobile users in our simulations are with the same access pattern. In our simulation, each mobile node accesses these 1000 data items by following the Zipf-like distribution

Z(θ). Additionally, we investigate the impact from the parameter θ on the performance of both caching strategies.

6.5.1.4. Performance Metrics

We measure three performance metrics for comparison of various cache placement strategies, viz., *average query delay*, *caching overheads* and *query success ratio*. Query delay is defined as the interval between the time when a client node sends its data request out and the time when the client node obtains the reply from its nearest cache node. *Average query delay* is the average of query delays over all queries. *Caching overheads* includes all of the packets in our caching system, viz., data requests, data replies, and other messages for caching systems. Note that these packets do not include routing packets because the two strategies use the same routing protocol DSDV. *Query success ratio* is referred to as the percentage of the queries that receive the reply data item within the period of the query success timeout.

6.5.2. Simulation Results

In this subsection, we present the simulation results comparing the two caching strategies, viz. BDC and DRCC, under the data access pattern and study the effect of various values on the performance metrics.

6.5.2.1. Varying Zipf-like Parameter θ

In Figure 6.10, Figure 6.11 and Figure 6.12, we vary the Zipf-like distribution parameter θ from 0.0 to 1.0 while keeping the cache size as a constant, the mean query generate time T_q as 10 seconds, and the maximum speed V_{max} as 2.0m/s. We choose the cache size to be big enough to fit 100 data items, i.e., 75Kbytes. We observe that DRCC outperforms BDC, especially in terms of caching overheads. The results show that DRCC can reduce at least 20% and up to 50% message cost, while it obtains much better performance in terms of average access delay. When the Zipf-like parameter θ is smaller than 0.2, the performance of average query delay and query success ratio are much similar between BDC and DRCC.

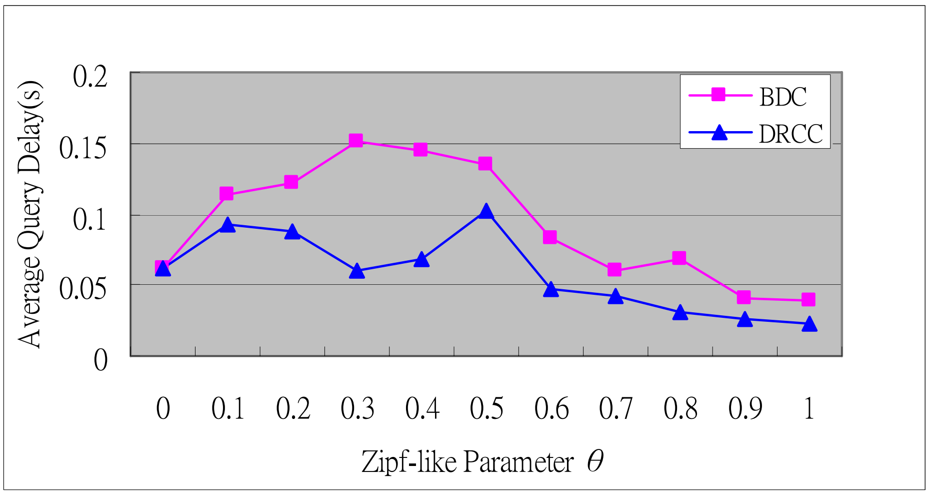


Figure 6.10: Average query delay vs. Zipf-like Parameter

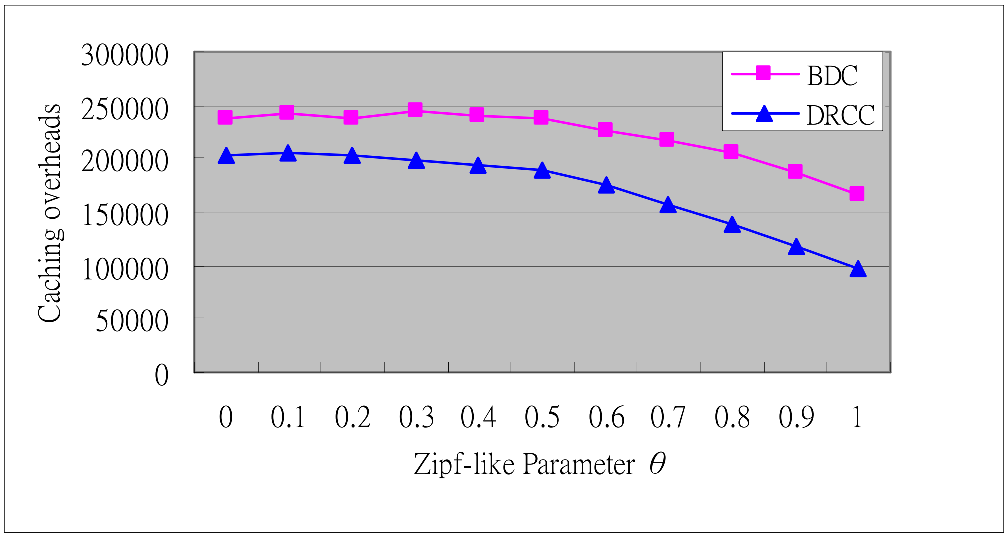


Figure 6.11: Caching Overheads vs. Zipf-like Parameter

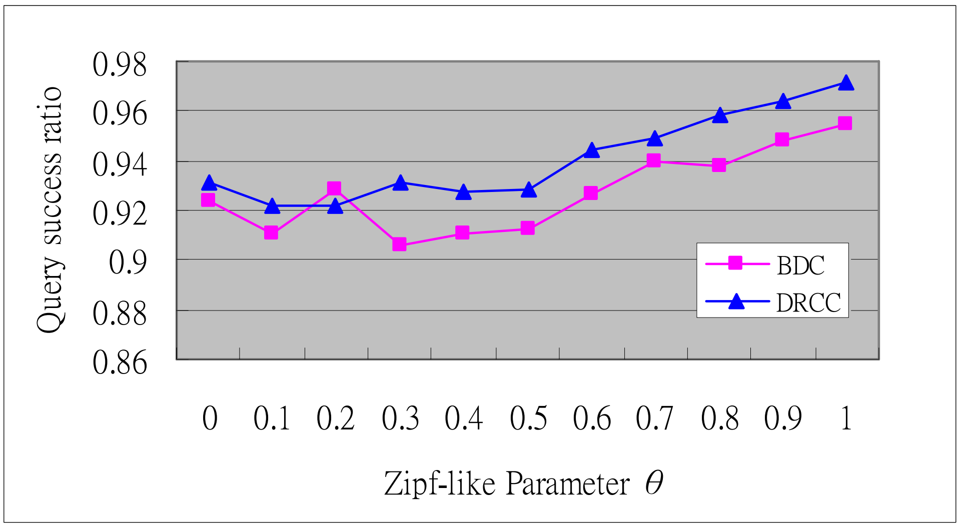


Figure 6.12: Query Success Ratio vs. Zipf-like Parameter.

6.5.2.2. Varying Mean Query Generate Time Tq

In Figure 6.13, Figure 6.14 and Figure 6.15, we vary the mean query generate time T_q from 3 to 40 seconds in our data access pattern while keeping the Zipf-like distribution parameter θ as 0.8 (this selection is based on the real web trace study [BCF$^+$99]), and the maximum speed V_{max} as 2.0m/s. We observe that DRCC outperforms BDC but the differences mainly focus on the setting when the mean query generate time is small, which means that the performance are similar when the query arrival rate is low. For a system with sparse data requests, there are few differences among all the caching strategies. The results show that DRCC reduces at least 25% and up to 50% message cost while obtain better performance than that of BDC.

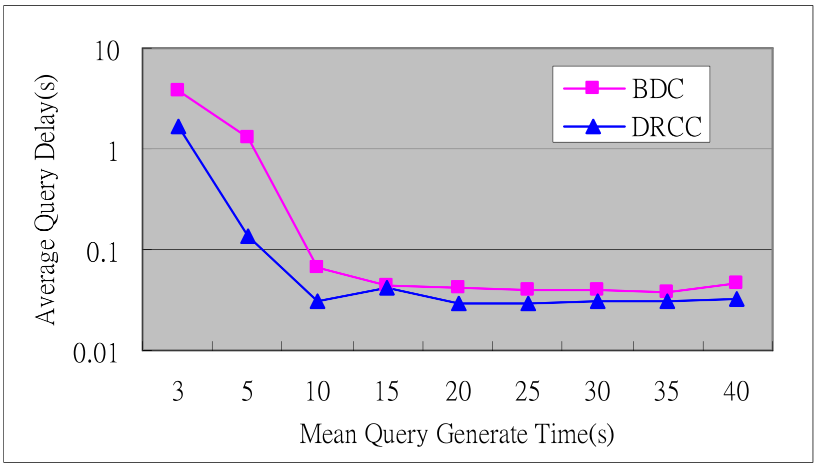


Figure 6.13: Average query delay vs. Mean Query Generate Time.

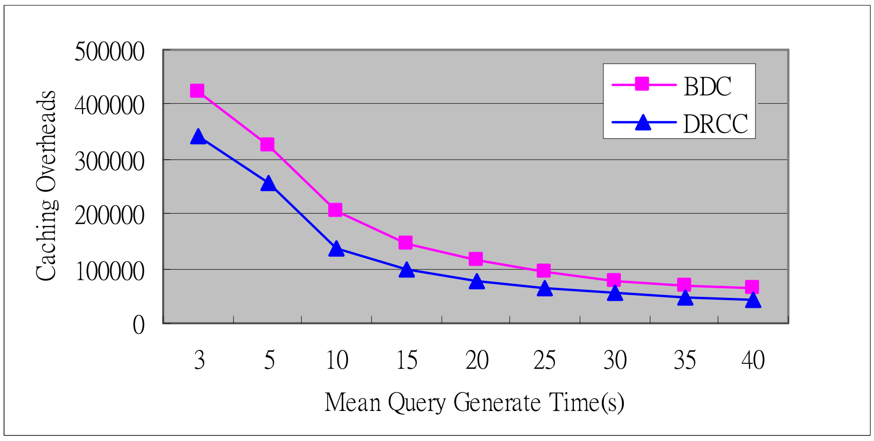


Figure 6.14: Caching Overheads vs. Mean Query Generate Time.

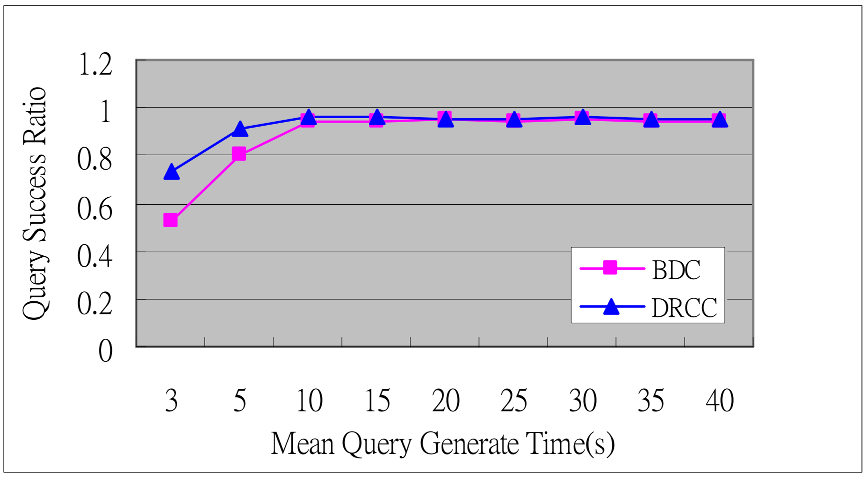


Figure 6.15: Query Success Ratio vs. Mean Query Generate Time.

6.5.2.3. Varying Maximum Speed Vmax

In Figure 6.16, Figure 6.17 and Figure 6.18, we vary the maximum speed V_{max} from 2 to 20m/s in our data access pattern while keeping the Zipf-like distribution parameter θ as 0.8, and the mean query generate time T_q as 8. We observe that DRCC outperforms BDC in most cases, especially in terms of caching overheads, which means DRCC uses much less messages but obtains better performance metrics compared with BDC. The results show that DRCC reduces at least 16% and at most 71% message cost. Moreover, the results of BDC are not stable when the speed of mobile nodes is increasing while the performance of DRCC is much stable. Therefore, the results also show that DRCC is much suitable for mobile networks, which is also one of our motivations for designing such a strategy of cooperative caching in mobile networks.

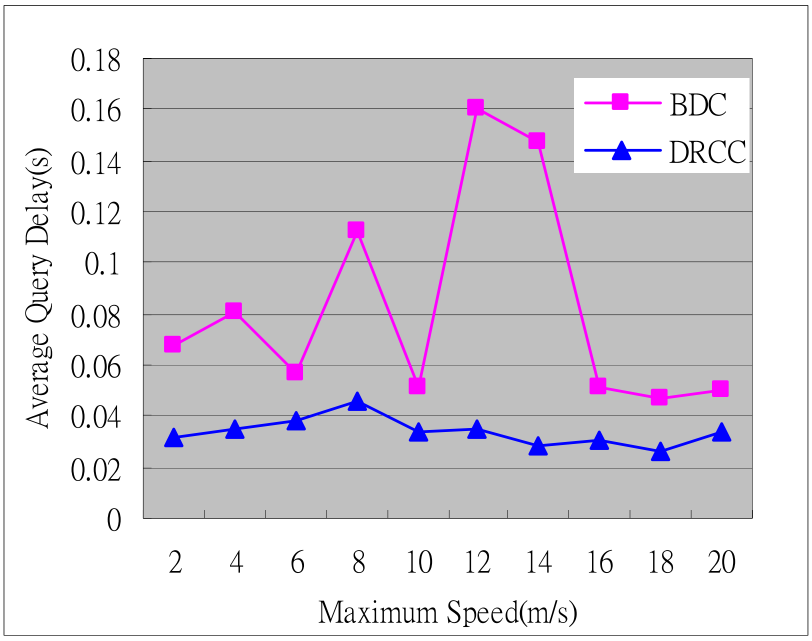


Figure 6.16: Average query delay vs. Maximum speed

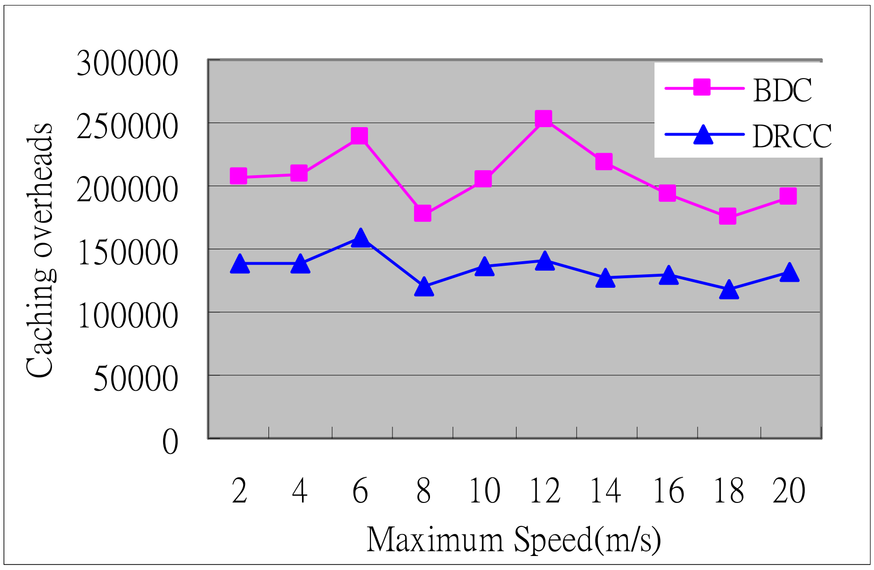


Figure 6.17: Caching overheads vs. Maximum speed.

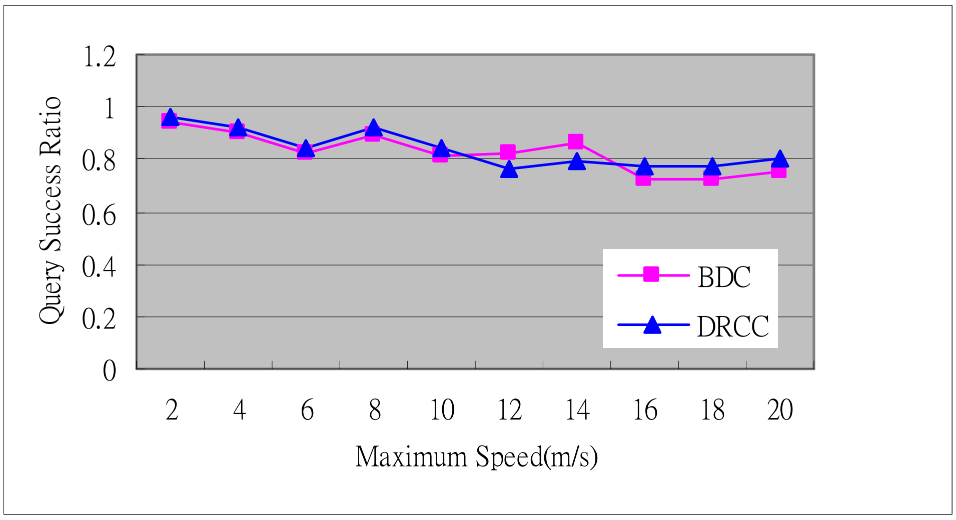


Figure 6.18: Query success ratio vs. Maximum speed.

In summary, simulation results show that DRCC achieves better performance than other alternatives in terms of average query delay, caching overheads, and query success ratio, especially reducing overheads by 40% in average. The performance of DRCC is much stable than that of BDC, especially in mobile networks.

6.6. Summary

In this chapter, we address the cache placement problem with multiple data items in IMANETs. We consider the impact of the distribution of data access frequencies on the performance of cooperative caching. We first formulate the cache placement problem in IMANETs as an optimization problem under the constraint of limited cache size. To deal with the dynamic topology of IMANETs, average access hops is proposed to replace the length of the shortest paths in a mobile ad hoc network. We present a novel caching strategy named Divide-and-Rule Cooperative Caching (DRCC) that caches different data items with different strategies to minimize average access hops. We also present the analytical results for the problem above. In our simulations, we evaluate DRCC in IMANETs with different settings. We take average query delay, caching overheads and query success ratio as our performance metrics. The results demonstrate that our algorithms outperform the previous work.

Chapter 7. Gossip-based Cooperative Caching for Data with Sequential Relation in IMANETs

In this chapter, we introduce the proposed Gossip-based Cooperative Caching algorithm, GosCC. This chapter is organized as follows: Firstly, Section 7.1 is the overview to this work. Section 7.2 describes the network model and the problem formulation. In Section 7.3, we present the design of our algorithms. In Section 7.4, we make an analysis on the parameter in GosCC. Simulation results are reported in Section 7.5. Finally, Section 7.6 concludes this chapter.

7.1. Overview

Before we discuss the cache placement problem in this chapter, we make a brief introduction on data relations. In general, there are two kinds of basic relations among data items. Firstly, the relation is inherent and we name it as *Inherent Relation (IR)*. For example, if a video file can be divided into many segments as different data items due to the big size of a video, there is an inherent sequential relation among these data items for a mobile user to access. Secondly, the relation among data items is extrinsic and we name it as *Extrinsic Relation (ER)*. For example, when a mobile user is driving in a city and the traffic information [WCL$^+$08] to the destination should be cached on the mobile nodes in advance along the path to the destination. If we consider the traffic information as data items, the relation among these data item comes from the movement of mobile user. To clarify the existing data relations in detail, we can categorize them into three categories, including time-based relationship, spatial-based relationship, and semantic relationship. The first one means that the relations among data items can be described in a time-related way. For example, a video file is composed of a series of segments that are used by following the sequence of time. This is one of the most typical relations in the real world so that we take the sequential relation as our target. The spatial-based relationship means that the relation results from the spatial information of data items. For example, a driver plans to drive to a destination. The traffic condition on each crossing along the path from the source to the destination is related by the location of the user. The third one means that the relations among data items are defined in a semantic way. The semantics can be defined by applications.

Additionally, we consider data relations in cooperative caching but our topic is much different from several other related topics on caching. Firstly, context-aware caching considers the behaviors of mobile users and the relation between data items. For example, the authors [KP07] propose the relation among files by investigating the users' file access experience. In [SUE00], the authors propose a general automatic hoarding algorithm by establishing the association rules of data items. It is clear that the relation in context-aware caching is mainly from the investigation of user experience, rather than the inherent relations among data items. Secondly, semantic caching [DFJ$^+$96, RDK03, LLS99] is derived from query reasoning in query optimization in database systems. Tuples in a distributed database system are mapped into a multidimensional semantic space. Each cluster of tuples within a semantic subspace is cached and described by a restrict condition. Comparing restrict conditions between a query and the cache, clients are able to determine what tuples are available and what tuples are missing. However, there is no directly cooperation among users. Semantic caching just plays the role of describing the content of caches and the queries at each client node.

In a cooperative caching strategy, as we mentioned, it is important that how mobile nodes can communicate with each other to exchange the information about caching. In our work, mobile nodes mainly exchange two kinds of information, including the progress report that describes the current IDs of data items used by mobile nodes, and the cache contents in mobile nodes' caches. As we all know, nodes in a mobile ad hoc network are not connected by any fixed infrastructure. Gossiping [DGH$^+$87, RMH98, BHO$^+$99, KMG03] is well matched to the needs of ad hoc networks because it is a controlled form of flooding, i.e., messages are slowly propagated through the whole network without congesting the wireless medium. Gossiping is also independent of the network topology. Gossip-based multicast protocols rely on a peer-to-peer interaction model for multicasting a message, and they are scalable since the load is distributed among all participating nodes. Redundant messages are used to achieve reliability and fault tolerance. In wireless networks, gossip-based protocols have been proposed for multicast in mobile ad hoc networks (MANETs).

In this chapter, we propose an adaptive strategy for cache placement to make mobile users access all the data items in a sequential order. Our objective is to minimize total interruption intervals for all the mobile users across all the data items. We name the proposed strategy as Gossip-based Cooperative Caching (GosCC). The information about the progress reports and the cache content is exchanged by gossiping, which is a suitable way to improve the reliability of communication in such a mobile and wireless environment like IMANETs. Our caching decision is made based on the information above and constrained by the limited cache sizes of mobile nodes. GosCC have the three advantages: 1) the proposed strategy is reliable in a mobile wireless environment because of reliable communication by gossiping; 2) GosCC implements cooperative caching in a randomized way; and 3) GosCC is independent to mobility models.

There are four points on the novelty of GosCC. Firstly, we introduce the relation among data items into cooperative caching. We improve the performance of caching by making use of these relations. Secondly, we apply gossiping as a reliable dissemination way to exchange the information of the progresses of mobile user using data items, and the cache content among mobile nodes. Thirdly, GosCC makes total interruption intervals as the objective of cooperative caching, rather than total query delay. It is much different from the existing works because we introduce data relation into cooperative caching so that we are mainly interested in the efficiency of using data items according to user experience. Lastly, the difficulty of detecting dynamic distances between client nodes and the corresponding closest cache nodes is circumvented by providing a probabilistic way for mobile nodes to communication with each other. Caching decisions are made based on more accurate and timely information. GosCC can also be analyzed in a mathematical way to evaluate the performance of cooperative caching for data relation.

7.2. Problem Formulation

In this section, we first introduce the system model for our work. Next, we formulate the cache placement problem with multiple data items in IMANETs.

7.2.1. System Model

Let $G(V, E)$ be a graph representing an Internet-based Mobile Ad Hoc Networks with n $(n=|V|)$ nodes, $V=\{MN_1, MN_2,..., MN_n\}$. Two mobile nodes communicate directly with each other, which is represented by an edge on the graph. Mobile users require to access m data items $D=\{d_1, d_2, ..., d_m\}$ by following the sequential order from d_1 to d_m, such as the segmentations in a video file. The size of data item d_i is the same one denoted by SD. Each data item is maintained only by one data source, such as a gateway node (GW). We define SR_i as the data source of the data item d_i. The size of MN_i's cache is the same and denoted by S.

To model our problem, we have the following assumptions:

- Gateway nodes always maintain the copies of data items they cached from the Internet.
- Links are bidirectional.
- Mobile nodes never fail.

- Each hop delay is the same.

7.2.2. Problem Formulation

In our problem, cache placement is to determine what data items should be cached at which mobile nodes so that the total interruption intervals are minimized. We name the proposed problem as the *Sequential Relation Cache Placement* problem (SRCP). Our objective is much different from the traditional cache placement problem in the way that we aim to minimize the total interruption intervals rather than the total access delay. The interruption interval for data item d_j at the mobile node MN_i is defined as the interval $t(i,j)$ between the time when MN_i finishes using the data item d_{j-1}, and the time when MN_i begins to use the data item d_j. Note that the interruption interval is much different from the query delay, which is defined as the interval between the time when a client node sends its data request out and the time when the client node obtains the reply from its nearest cache node. We consider user experience as our major concern after we introduce the sequential data relation into cooperative caching.

In SRCP, cache placement is subjected to limited cache sizes at individual nodes. The cache placement problem is to select a set of sets of cache nodes $M=\{M_1, M_2, M_3, ..., M_m\}$, where each mobile node in M_j stores a copy of d_j, to minimize the total interruption intervals as follows:

$$T(G,M)=\sum_{i=1}^{n}\sum_{j=1}^{m}t(i,j) \qquad (7.1)$$

$$\text{s.t. } |\{M_j | i \in M_j\}| \leqslant S,$$

where MN_i appears at most S sets of M. The traditional cache placement problem has been proved NP-hard [RDK03]. In this chapter, the optimal solution for SRCP is that the total interruption intervals are zero. However, it is clear that the optimal solution is hard to be achieved due to the access delay for each mobile node to access each data item in such a mobile environment. We aim to minimize the total interruption intervals by making mobile nodes access all the data items at local cache as more as possible. But mobile nodes also need to consider the progresses of other nodes using data items and cooperate with each other, in order to reduce the probability that one mobile node cannot access the data items continuously, i.e., the process of using data items is interrupted.

7.3. Gossip-based Cooperative Caching

In this section, we firstly present our design rationality for our proposed caching strategy, Gossip-based Cooperative Caching. Next, we describe GosCC in detail with the key data structures and the detailed algorithm.

7.3.1. Design Rationale

In this subsection, we present our design rationale for our proposed Gossip-based Cooperative Caching (GosCC). The whole subsection is divided into two parts: 1) two important characteristics in SRCP that are much different from the traditional cache placement problem; and 2) three heuristic rules in our design.

7.3.1.1. Characteristics in SRCP

Normally, the traditional cache placement strategy balances a tradeoff between access delay and message cost. In SRCP, we also concern access delay, but it is not the most important objective. The reasons are presented as follows.

Firstly, total access delay may be the most important metrics in the previous works for measuring the performance of cooperative caching. As we know, access delay is defined as the interval between the time when a mobile node sends its data request out, and the time when the client node obtains the reply from its nearest cache node. However, in SRCP, we are concerned about the interruption interval much more than access delay. This is because we know the access model of each node, i.e., each mobile user accesses all the data items in a sequential order from data item d_1 to d_m. If a mobile node fetches the next data item in its own cache, there will not be any delay when using the next data item. We do not care about how long the data item is fetched from other nodes and where the data item comes from, i.e., access delay on the data item is not the most important one after we consider the sequential relation among data items. We further define a new event named as *Use-in-Time, which means the interruption interval for using one data item is zero, i.e., the data item is cached locally before using it.* Therefore, we consider the most important performance metric in SRCP as *Average Interruption Interval (AII)*, which is referred to as the average of interruption intervals over all nodes. *AII* is applied to describe the performance of a cooperative caching system for SRCP from a global view.

Secondly, user experience in cooperative caching becomes more and more important than before. It is not desirable for a mobile user to see a video in his mobile device with many interruptions. Therefore, we consider another metrics, *Average Interruption Times (AIT)*, which is defined as the average of interruption times when mobile users access all the data items in a sequential order. In fact, the final objective of our work is to provide a QoS-based data access service for all the mobile users. Our work is a pioneering work on combining cooperative caching with QoS.

7.3.1.2. Three Heuristic Rules in SRCC

Since we have explained the two characteristics of SRCP above, we next consider three heuristic rules in our proposed GosCC strategy. Before we explain the details of the three heuristic rules, there are three terms to be defined clearly. The first term is the parameter "*Buffer Window*" (*BW*), which is defined as the number of the cached data items that consists of an array of data items continuously following the current data item using by mobile users.

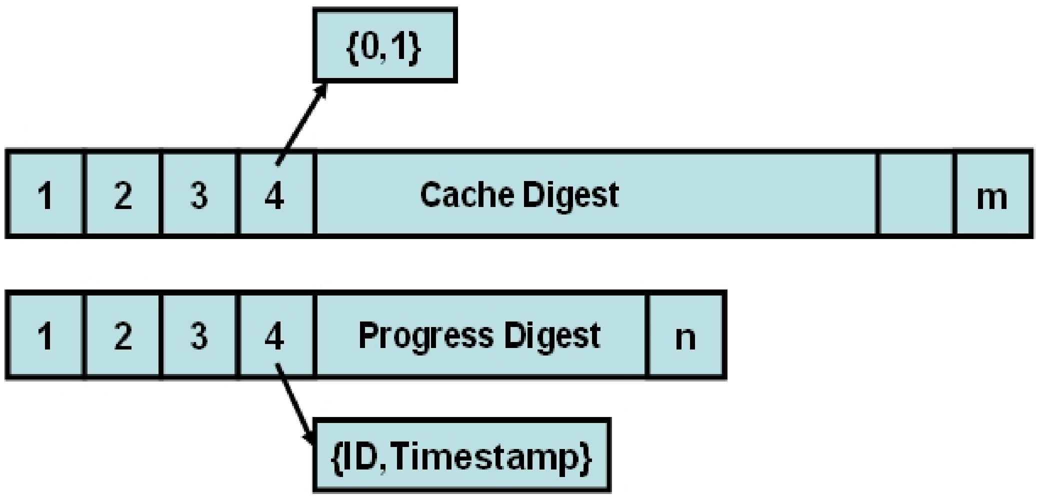


Figure 7.1: The structure of gossip report

The second term is "*Gossip Report*" (*GR*) described in Figure 7.1. In GosCC, we apply gossiping as the style of exchanging the caching information among mobile nodes. There are two kinds of the digests in one GR, including the cache digest and the progress digest. The cache digest is designed to describe the cache content in a mobile node. Let *C_DIGEST(i, t)* be the cache digest of the mobile node MN_i at time t. *C_DIGEST(i,t)* is composed of a vector with m bits. The j^{th} bit is used to denote whether the j^{th} data item is cached at MN_i or not. The progress digest is reported as follows. Let *P_DIGEST(i,t)* be the progress digest of MN_i at time t in one GR. *P_DIGEST(i,t)* consists of n data IDs and the corresponding timestamps. The k^{th} data ID is the current data ID used by mobile node MN_k, known by MN_i at time t. The detailed structure of a gossip report is described in Figure 7.1 as follows.

In GosCC, each mobile node periodically gossips its own gossip reports to other nodes randomly selected from n mobile nodes. In *C_DIGEST*, MN_i describes its own cache content to other nodes so that other mobile nodes can send data requests to the closest cache node for data items. In *P_DIGEST*, MN_i just stores the IDs of the data item that is used now by all the nodes and each ID is with a timestamp to show the latest updated version. Each mobile node is only allowed to updates its own current data item in *P_DIGEST*. To control redundant gossip messages, each gossip report should not be forwarded beyond R rounds. It means that each gossip report is gossiped only R times. We will discuss the determination of the parameter R in Section 7.4.

Rule 1: The next data item to be used by a mobile node is always the most important one.

In GosCC, the first important point is how to make the interruption interval as less as possible. As we mentioned, mobile nodes know the sequential order of data access. Thus the interruption interval mainly depends on the interval between the time when a mobile user finishes using a data item and the time when the mobile user begins to use the next data item. Therefore, GosCC applies a scheme named the *Buffering-Next Scheme* (BNS). BNS is designed for collecting the set of data items in the buffer window of a mobile node in a timely way.

Before we explain the details of BNS, we categorize the data request into 3 levels to establish the order of priority. Data item d_j at mobile node MN_i is assigned with a priority index, $p(i,j)$. If the data item d_j is the next data item in the buffer window of MN_i, we set the priority index $p(i,j)$ as 1; If the data item d_j is not the next one but in the buffer window of MN_i, we set the priority index $p(i,j)$ as 2. If the data item d_j is not in the buffer window of MN_i, we set the priority index $p(i,j)$ as 3.

In BNS, each mobile node detects the progress of its using data item periodically and sends the "active" cache requests with the predefined priorities. The maximum number of cache requests in one round cannot reach the maximum number of data items that can be cached in the remainder of the MN_i cache. The maximum number *Max_Num* of data items that can be cached in one mobile node is determined by $\lfloor S/SD \rfloor$.

Rule 2: When there are still available spaces to cache more data items, mobile nodes investigate the progress digest and sends cache requests for the most popular data item in a global view.

In GosCC, we mentioned that the progress digest can provide mobile nodes with the global information on the progresses of other mobile users using data items. Mobile node MN_i checks the progress digest in the latest gossip report received. If the data item d_j is with the maximum times in appearance and there are still available space at MN_i, MN_i sends a cache request for data item d_{j+BW} to its closest cache node.

Rule 3: When cache replacement is required, each data item with its ID smaller than the minimum data ID in the progress digest should be replaced.

Once mobile node MN_i receives a gossip report from MN_j, MN_i updates the progress digest and forwards the report to randomly selected mobile nodes if the gossip report is still in its life time. MN_i also checks its own cache content. If MN_i finds that there are some data items that are smaller than the minimum data ID in the received progress report, MN_i replace these data items from its own cache. It means that there is no use for these data items because we know the progress of the "slowest" mobile node.

In a gossip report, there are two important data item IDs. One is the maximum one *MAX_ID*, and the other is the minimum one, *MIN_ID*. A mobile node with higher current data ID means that the node consumes the data items with a higher speed. On the other hand, a mobile node with lower current data ID means the node consumes the data items with a lower speed. Let *VAR_RANGE* be the variation range of a progress digest. It can be obtained by Eq. (7.2).

$$VAR\_RANGE = MAX\_ID - MIN\_ID \qquad \textbf{(7.2)}$$

It is clear that the best case for *VAR_RANGE* is zero. This means that the only data item that all mobile nodes should cache is the next one. On the other hand, if the variation range is very large (the maximum value is m-1), the range of data items for all the mobile nodes to cache is from 1 to m. Therefore, we propose a threshold named the range threshold TH_r. If mobile node MN_i detects that the variation range *VAR_RANGE* is greater than TH_r, MN_i actively sends the data items that will be used by the mobile node with *MIN_ID* to the node.

7.3.2. Gossip-based Cooperative Caching

In this subsection, we present a distributed algorithm named Gossip-based Cooperative Caching (GosCC) to solve the cache placement problem with accessing multiple data items in a sequential order. GosCC is formed of two important components, including the key data structures and the detailed algorithm.

7.3.2.1. Key Data Structures

Closest Cache Tables are the most important data structure in GosCC. Mobile node MN_i maintains the closest cache node MN_j that has a copy of the data item d_k for each data item d_k in the network. MN_i also maintains a closest_cache table, where an entry in the closest_cache table is of the form (d_k, MN_j) where MN_j is the closest cache node that has the copy of d_k, known by MN_i. Note that if MN_i is the data source or has cached d_k, MN_i is the closest cache node for d_k.

Progress Tables are another major data structure maintained by each mobile node. In the progress table, mobile node MN_i stores the progress information for other mobile nodes when MN_i receives a progress digest from other nodes. An entry in the progress table is of the form (MN_i, *Current_ID, Timestamp*).

7.3.2.2. GosCC Algorithm

GosCC is mainly composed of three components, including gossiping, sending cache requests, and cache replacement. The details of GosCC are described in Figure 7.2.

Every T_g seconds, mobile node MN_i builds up a *GossipReport(MN_i,TS,GR)* message, where *TS* is the timestamp and *GR* is the current gossip report at mobile node MN_i. The *GossipReport(MN_i,TS,GR)* message mainly includes two components: *C_DIGEST* and *P_DIGEST*. Let f_i be the fanout of MN_i and the fanout f_i follows the Poisson distribution [FCW+08, NSW01]. MN_i selects the f_i mobile nodes as its gossiping targets and send the *GossipReport(MN_i,TS,GR)* message to these targets. If MN_j is one of the MN_i's gossiping targets, MN_j receives the *GossipReport(MN_i,TS,GR)* message, and updates its *ClosestCacheTable* and *ProgressTable*. If the attached round counter in *GR* is smaller than the maximum round *R*, MN_j forwards the *GossipReport(MN_i,TS,GR)* message to f_j gossiping targets randomly selected. The gossip operation does not finish until the round counter in the *GR* is equal or greater than *R*.

Every T_r seconds, mobile node MN_i carries out the BNS scheme as explained in the above subsection. MN_i builds up several *CacheRequest* messages and sends them to the corresponding closest cache nodes from MN_i's *ClosestCacheTable*. Note that the maximum number of cache requests should be smaller than the maximum number of data items that can be cached in the current cache size.

Once MN_i receives a *GossipReport(MN_j,TS,GR)* message from MN_j, two operations will be followed. Firstly, MN_i calculates the variation range and checks whether *VAR_RANGE* is greater than TH_r or not. If the answer is yes, MN_i selects the mobile node MN_j with *MIN_ID* as the target to help. If MN_i holds the copy of the next data item that MN_j maybe requires, MN_i sends a *CacheReply* message with the data item d_k to MN_j. Secondly, MN_j checks the progress digest in $GossipReport(MN_j,TS,GR)$. If the data item d_j is with the maximum times in appearance and there are still available spaces at MN_j, MN_j sends a *CacheRequest* message for data item d_{i+BW} to its closest cache node.

When the cache space is full and there are still another data item needs to be cached at MN_i, MN_i removes all the data items with their IDs smaller than *MIN_ID* in its progress table. If there are no such data items, MN_i just remove the data items with the oldest timestamp.

```
Algorithm 1: Gossip-based Cooperative Caching Algorithm
SETTING:
  A Network Graph $G(V,E)$ and a data set $D$ with $m$ data items.
BEGIN
    Every $T_g$ second, $MN_i$ sends $GossipReport(MN_i,TS,GR)$ to $f_i$ gossiping targets
randomly selected from all the mobile nodes.
    Every $T_r$ seconds, $MN_i$ sends $CacheRequest$ to the corresponding mobile
nodes by following the priority order.

  On receiving $GossipReport(MN_j,TS,GR)$
    If $MN_i$ is the destination and receives $GossipReport(MN_i,TS,GR)$ for the first
time, $MN_i$ updates ClosestCacheTable from $C\_DIGEST$ and updates its
ProgressTable from $P\_DIGEST$. $MN_i$ increases the round counter $RC$ in $GR$ by
one.
        If the $RC < R$, $MN_j$ sends $GossipReport(MN_i,TS,GR)$ to $f_j$ gossiping targets.
        Else $MN_j$ stops gossiping.
    If $MN_i$ is not the destination, $MN_i$ updates its ClosestCacheTable from
$C\_DIGEST$. $MN_i$ updates the $i^{th}$ entry in $P\_DIGEST$ in $GossipReport(MN_i,TS,GR)$.
$MN_i$ compares $P\_DIGEST$ with its own ProgressTable and makes revisions on
both tables based on the timestamps. Then $MN_i$ forwards $GossipReport$ to the next
hop.
        If $VAR\_RANGE > TH_r$ and $MN_i$ holds any copy of the data items in the
buffer window of the mobile node with $MIN\_ID$ in $P\_DIGEST$, $MN_i$ sends a
$CacheReply(MN_i, d_k)$ to the node.
        If $x$ is the data ID that with the maximum times in appearance in
$P\_DIGEST$ and there is still available space, $MN_i$ sends a $CacheRequest$ message
to its closest cache node.

  On receiving $CacheRequest(MN_j, ID)$
    If $MN_i$ is the destination and the copy of ID is in its cache, $MN_i$ sends
$CacheReply$ to $MN_j$.
    If $MN_i$ is not the destination but the copy of ID is in its cache, $MN_i$ sends
$CacheReply$ to $MN_j$ and discards $CacheRequest$.
    Else $MN_i$ forwards $CacheRequest$ to the next hop.

  On receiving $CacheReply(MN_j, d_k)$
    If $MN_i$ is the requestor, $MN_i$ updates its ClosestCacheTable and stores $d_k$
locally if there are available spaces. If there is no space, $MN_i$ replaces data items
by following our cache placement strategy.
    If $MN_i$ is not the requestor but $d_k$ is in request, $MN_i$ caches it locally and
forwards $DataReply$ to the next hop.
END
```

Figure 7.2: Gossip-based Cooperative Caching Algorithm.

7.4. Parameter Analysis

In this section, we provide the analysis work for the parameter design in our proposed GosCC. Firstly, we make a brief introduction to the Gossiping-based Algorithms (GAs) and find the most important factors on the performance of these algorithms. Secondly, we mainly focus on two problems in GosCC, including 1) how much percentage of mobile nodes can eventually receive a gossip message; and 2) how long a gossip message can be propagated to all the mobile nodes. Thirdly, we evaluate the probability that a *Use-in-Time* event happens, i.e., the probability that a data item can be accessed from the local cache when it is in need.

As we mentioned in Chapter 2 and Chapter 3, gossip-based Algorithms, i.e., epidemic algorithms, are well examined in the recent years [EGK$^+$04]. Such an algorithm is considered as a potentially effective solution for disseminating information in large-scale systems [BHO$^+$99, KMG03]. GAs mimic the spread of a contagious disease. In addition to their inherent scalability, gossip-based algorithms are easy to deploy, robust, and resilient to failures. It is possible to adjust the parameters of a gossip-based algorithm to achieve higher reliability, despite process crashes and disconnections, packet losses, and a dynamic network topology [CRB01, LEH03].

In our analysis, we apply the infect-and-die model [NSW01, M93]. Each node gossips the same gossip message exactly once, i.e., each node forwards the gossip message to its gossiping targets only for the first time when receiving the gossip message, even if the node maybe receives the copies of the message from other nodes again.

The fact that a healthy person is infected by an infectious means a mobile node receives a gossip message from another mobile node successfully. Basically, each node forwards the gossip message with a limited number of times *R*. Each mobile node forwards the gossip message each time to a randomly selected set of mobile nodes of limited size *f*, the fanout of the dissemination. To simplify our analysis, the fanout *f* is the mean value of a Poisson distribution *Po(f)* [FCW+08, EGK+04]. Here we will use the mathematical models for gossiping in Chapter 2.

In this model, we mainly focus on two problems. The first one is what the final proportion of mobile nodes eventually receives a gossip message is, i.e., what the reliability of gossiping is. The second one is how long a gossip message can be propagated to all the mobile nodes, i.e., how many rounds *R* is required to make all nodes receive the gossip message.

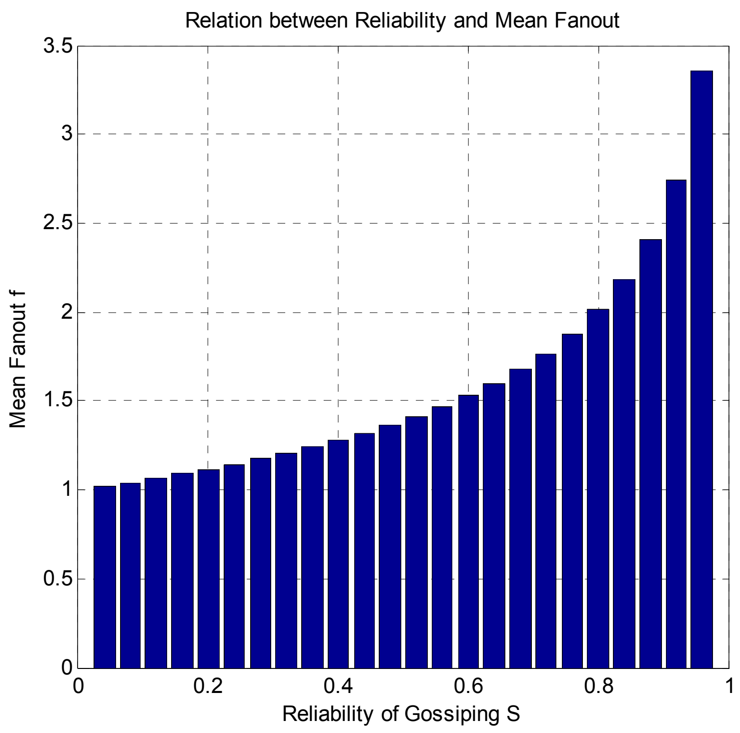


Figure 7.3: Relation between gossiping reliability S and fanout f.

With respect to the first problem, let *m* be a gossip message. Let *S* be the final proportion of mobile nodes that received *m*, i.e., the probability that the message *m* can be received by all the mobile nodes successfully. Note that *S* is also named as the reliability of gossiping. It has been proved that S can be calculated from the following equation [FCW+08, EGK+04].

$$S = 1 - e^{-fS} \tag{7.3}$$

To explain the relation between S and f, we can transform Eq. (7.3) to Eq. (7.4).

$$f = \ln(1/(1-S))/S \tag{7.4}$$

As Figure 7.3 shows, we increase the value of the mean fanout f so that we can improve the reliability of gossiping S dramatically. It is necessary to increase the mean fanout to a much greater value, like 10, so that we can make the reliability to reach 99.99% [FCW+08]. Therefore, there is no need to use the reliability of gossiping to 100% because we need also consider the message cost in GosCC.

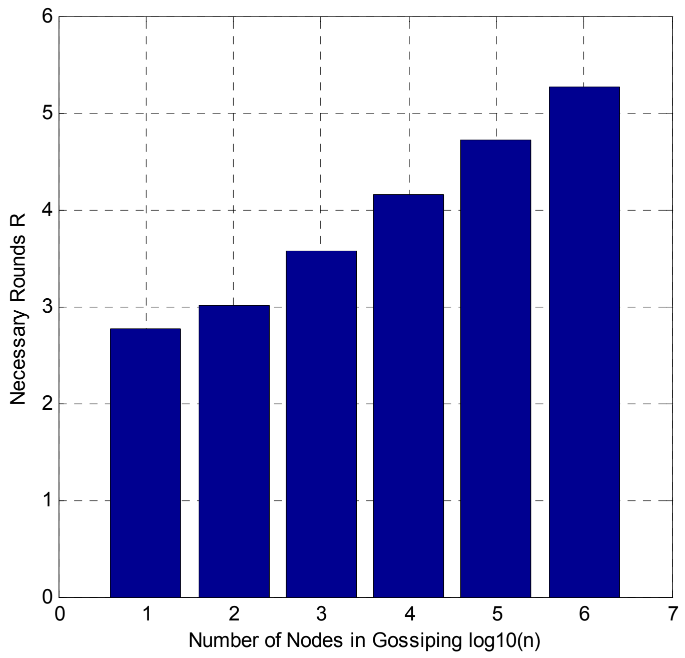


Figure 7.4: Relation between rounds R and number of nodes n

The second problem is about the latency of infection, which can be used to evaluate the speed of gossiping. Bella [B01] shows that the number of necessary rounds R to infect all of the nodes can be obtained as Eq. (7.5) shows.

$$R = \ln(n)/\ln(\ln(n)) + O(1) \tag{7.5}$$

However, we know that the result from Eq. (7.5) is only the theoretical result. The reason is that any intermediate node on the path from the sender to the receiver in an IMANET can know the content of the gossip message, as we make use of this point in our GosCC. Therefore, the speed of the convergence of GosCC will be much quicker than that of the theoretical case. We only consider R in Eq. (7.5) as an upper bound. As Figure 7.4 shows, the necessary round R for 100 nodes is almost 3.

Next, we try to evaluate the probability that a *Use-in-Time* event happens. As we mentioned in Section 7.2, a *Use-in-Time* event means a data item is cached locally before a mobile user use it. There is no any interruption interval for this data item. Let d_k be one of the data items. We define the event $UIT(d_k, MN_i)$ as the *Use-in-Time* event that MN_i uses the data item d_k without any interruption interval. Let t_{SD} be *the consuming time* for a mobile node uses a data item since we assume that the sizes of all the data items are the same. Let v_u be the speed of mobile user consuming data items, such as the speed of mobile users playing a movie. Therefore the consuming time t_{sd} can be obtained as follows:

$$t_{SD} = SD/v_u \tag{7.6}$$

We mainly concern the total interruption intervals. As we mentioned in Section 7.2, the interruption interval $t(i,j)$ is zero if mobile node MN_i accesses the data item d_j and the copy of d_j is in MN_i's cache. If d_j is not in the local cache, MN_i is required to obtain the copy from other nodes, including the data source SR_j. Let h be the average length of the shortest path from MN_i to the closest cache node that holds the copy of d_j. We assume that each hop delay is the same and it can be denoted by t_{hop}. But the precondition for a successful retrieval is that each mobile node knows the cache content of other nodes. This is guaranteed by gossiping these contents in a probabilistic way. A gossip message is propagated to all the nodes with the success probability *S*. Therefore, total interruption intervals *T(G,M)* can be calculated as follows:

$$T(G,M)=\sum_{i=1}^{n}\sum_{j=1}^{m}(1-P(UIT(d_j,MN_i)))\times t_{hop}\times h\times S \tag{7.7}$$

Our final objective is to minimize the total interruption intervals under the constraint of the cache sizes. If the event $UIT(d_k, MN_i)$ happens, the data item d_k should be cached before all the data items in the buffer window *BW* are consumed completely. However, the occurrence of the event $UIT(d_k, MN_i)$ should be under the premise that the gossip message *m* is received by all the mobile nodes within *R* rounds. If a gossip message is gossiping only by one round, we consider that the gossiping algorithm is executed one time. In the repeated executions, each execution can be viewed as one independent Bernoulli trial [FCW+08]. So *R* rounds of executions can be considered as *R* times Bernoulli trials. We define *X* as the number of executions in which a mobile node receives the gossiping message *m* during *R* rounds. We do not consider how many times for each node to receive the message *m* in one execution. It is obvious that the reliability of gossiping *S* is the probability that a mobile node receives the gossiping message in one execution. Thus *X* follows a Binomial distribution $B(R, P_R)$. The distribution of *X* is in the following:

$$P(X=k)=C_R^k(S)^k(1-S)^{R-k} \tag{7.8}$$

where $k = 0, 1, 2, \ldots, R$. We denote P_R as the probability that a mobile node receives the gossiping message *m* at lease one time within *R* rounds. Therefore, P_R can be obtained from the following equation:

$$P_R=P(X\geq 1)=1-(1-S)^R \tag{7.9}$$

Finally, the following condition described in Eq. (7.10) should be satisfied when we calculate the total interruption intervals in Eq. (7.7).

$$\frac{S}{SD} \geq BW \geq \frac{2h \times t_{hop} \times (1-(1-S)^R)}{t_{SD}} \tag{7.10}$$

However, it does not mean that the greater the buffer window *BW* is, the better the performance is. Since the number of cache requests sent by a mobile node is based on the size of the buffer window, we should also consider the message cost. There is a tradeoff between the total interruption intervals and the total message cost. Therefore, we set the timer T_r for sending cache requests as follows:

$$T_r = \frac{BW \times SD}{v_u} \tag{7.11}$$

Another timer is the gossip timer T_g. To determine the value of T_g, we should consider the number of messages in gossiping. Let *T* be the total time of running our proposed GosCC. Let M_g be the total number of messages generated by gossiping. We can obtain M_g from Eq. (7.12).

$$M_g = \frac{T}{T_g} \times n \times f^R \times h \tag{7.12}$$

As Eq. (7.12) shows, the gossip timer T_g is related to not only the parameters of gossiping algorithm, but also the network condition. In real application, GosCC is required to detect the average length of the shortest path from the source to gossiping targets.

7.5. Simulations

In this section, we demonstrate the performance of GosCC compared with Benefit-Based Data Caching (BDC) [TGD06] through simulations over randomly generated network topologies.

7.5.1. Simulation Settings

Our simulations are carried out by making use of the NS2 simulator [FV97]. The NS2 simulator contains different models for routing protocols in common ad hoc networks, IEEE Standard 802.11 MAC layer protocol, and two-way ground reflection propagation models [BMJ+98]. The DSDV routing protocol [PB94] is used in our work to provide routing services.

7.5.1.1. Network Setup

We simulated GosCC and BDC in a network with 100 mobile nodes randomly placed in an area of 2000x500m^2. Mobile nodes move based on the random waypoint model [BMJ+98] in NS2. In this model, each node selects a random destination and moves towards the destination with a speed selected randomly from ($0m/s$, $v_{max}m/s$). After the node reaches its destination, it pauses for a period of time (chosen to be 100 seconds) in our simulation and repeat the movement pattern. Note that the normal radio range for two directly communicating nodes in the NS2 is about 250 meters. We assume that there are 1000 data items with the same size as 1000 bytes. We setup two randomly placed data sources (servers) S_0 and S_1 as the gateway nodes, where S_0 maintains the data items with even IDs and S_1 maintains the data items with odd IDs.

7.5.1.2. Client Query Model

In our simulation, each mobile node is a client node. Each client node sends out a stream of the http requests by following the sequential order of data items from d_1 to d_{1000}. Once a mobile node receives an http reply or obtains the requested data from the passing-by reply messages, the mobile node makes the schedule for requesting the next data item, according to the size of data item and the consuming data speed of mobile nodes. We count the interruption interval for each data item and sum them up as the total interruption intervals.

7.5.1.3. Data Access Pattern

As we mentioned at the beginning of this chapter, we consider the impact of the sequential order among data items on the performance of cooperative caching. Each data item should be accessed successfully by local cache or from other nodes' reply. Otherwise, a mobile node will wait for the requested data item until it obtains the data item. In our simulation, each mobile node accesses the 1000 data items by following the sequential order from 1 to 1000.

7.5.1.4. Performance Metrics

We measure three performance metrics for comparison of various cache placement strategies, viz., *average interruption interval*, *average interruption times*, and *caching overheads*. As we mentioned in Section 7.2, *Average Interruption Interval (AII)* is referred to as the average of interruption intervals over all nodes. *Average Interruption Times (AIT)* is defined as the average of interruption times when mobile users access all the data items in a sequential order. *Caching overheads* includes all of the related packets in our caching system, viz., data requests, data replies, and other messages for caching systems. Note that these packets do not include routing packets because the two strategies use the same routing protocol DSDV.

7.5.2. Simulation Results

In this subsection, we present the simulation results comparing the two caching strategies, viz. BDC and GosCC, under the data access pattern and study the effect of various values on the performance metrics.

7.5.2.1. Varying Data Item Size SD

In Figure 7.5, Figure 7.6 and Figure 7.7, we vary the data size from 1000 bytes to 8000 bytes while keeping the total cache size as 750k bytes, the maximum speed V_{max} as 2.0m/s and the consuming data speed v_u as 400 bytes per second. We observe that GosCC outperforms BDC in terms of average interruption interval and average interruption times. However, the message cost is higher than BDC. The simulation results show the reliability of gossiping depends on the redundant messages.

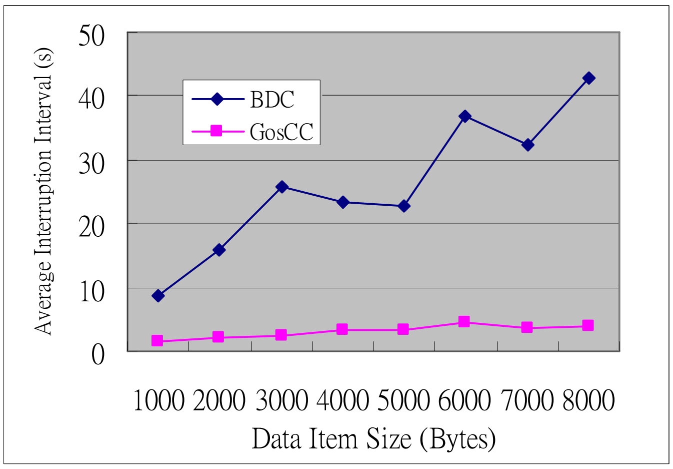


Figure 7.5: Average interruption interval vs. Data item size.

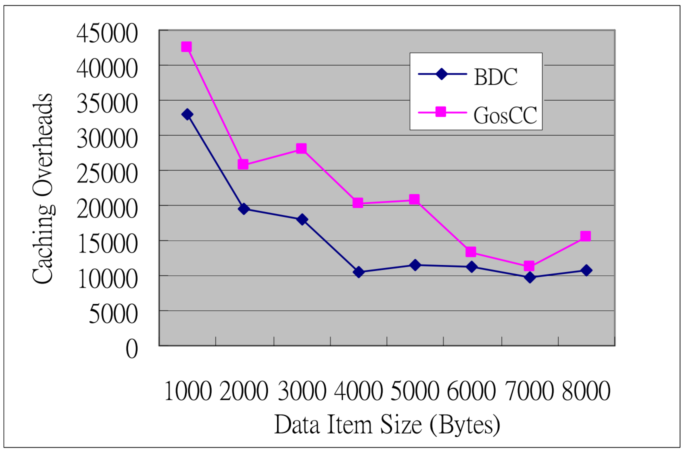


Figure 7.6: Caching overheads vs. Data item size.

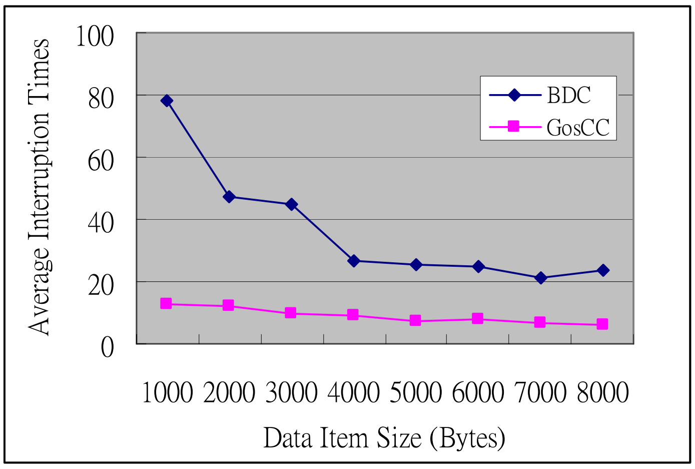


Figure 7.7: Average interruption times vs. Data item size.

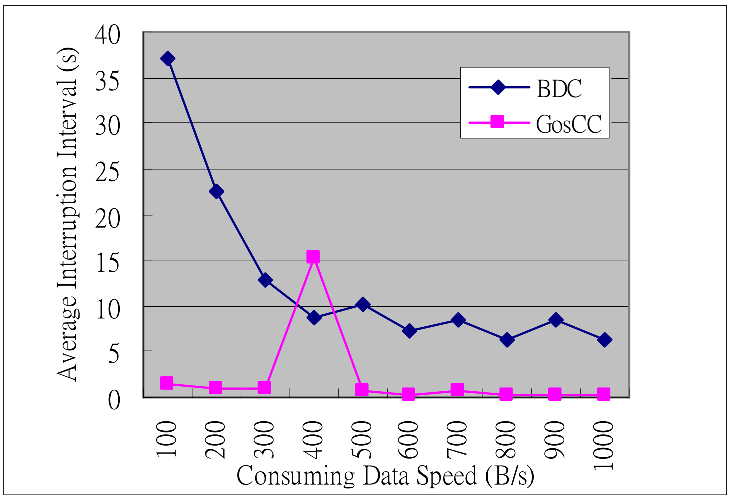


Figure 7.8: Average interruption interval vs. Consuming data speed.

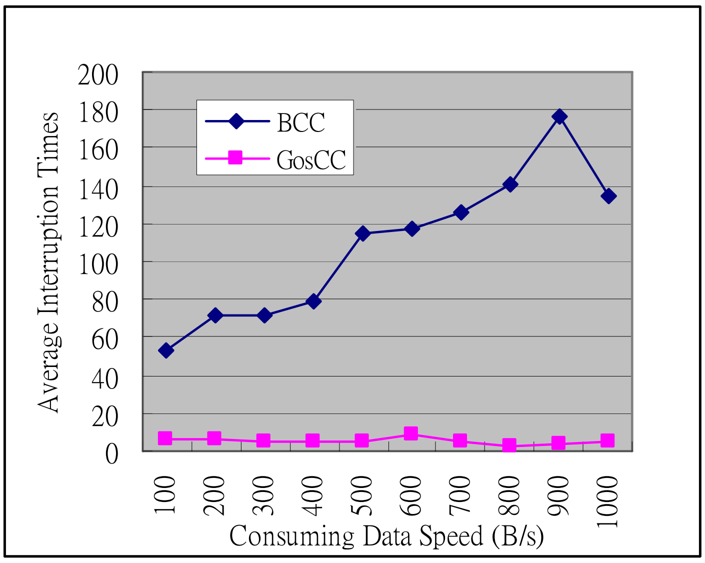


Figure 7.9: Caching overheads vs. Consuming data speed.

7.5.2.2. Varying Consuming Data Speed v_u

In Figure 7.8, Figure 7.9 and Figure 7.10, we vary the consuming data speed V_u from 100 to 1000 bytes per second in our data sequential access pattern while keeping the cache size as 75k bytes, and the maximum speed V_{max} as 2.0m/s. We observe that GosCC outperforms BDC again in terms of average interruption interval, average interruption times and caching overheads. The simulation results show the performance of GosCC is stable and almost independent to the consuming data speed.

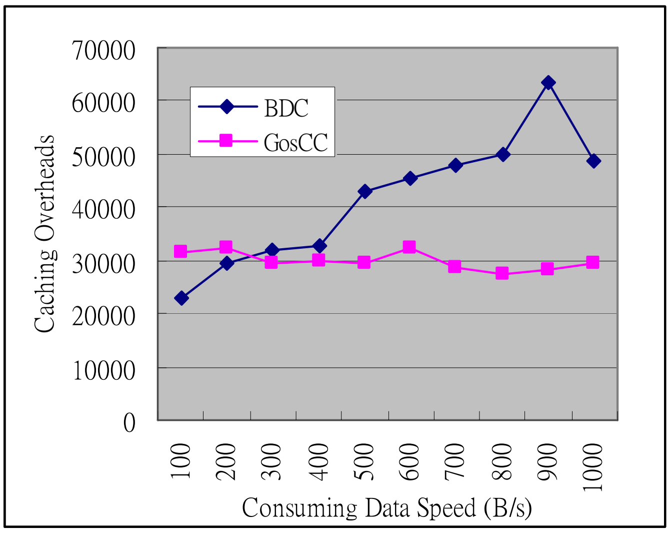


Figure 7.10: Average interruption times vs. Consuming data speed

7.5.2.3. Varying Maximum Speed Vmax

In Figure 7.11, Figure 7.12 and Figure 7.13, we vary the maximum speed V_{max} from 2 to 20m/s in our data access pattern while keeping the cache size as 75k bytes, and the consuming data speed as 400 bytes per second. We observe that GosCC outperforms BDC in terms of average interruption interval and average interruption times. However, caching overheads of GosCC are worse than that of BDC to a certain degree.

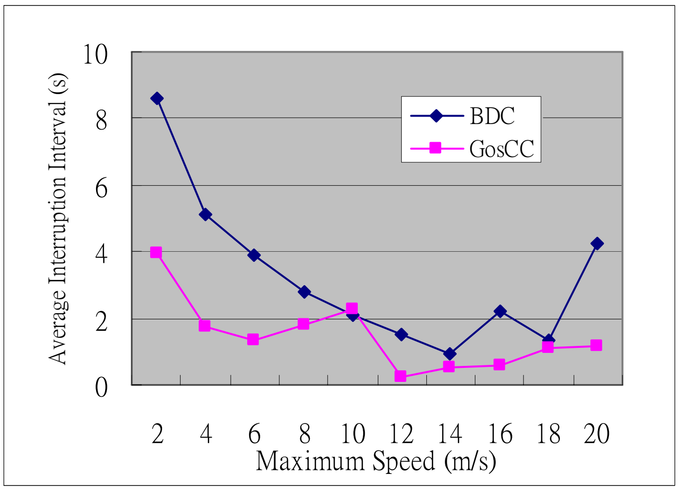


Figure 7.11: Average interruption interval vs. Maximum speed.

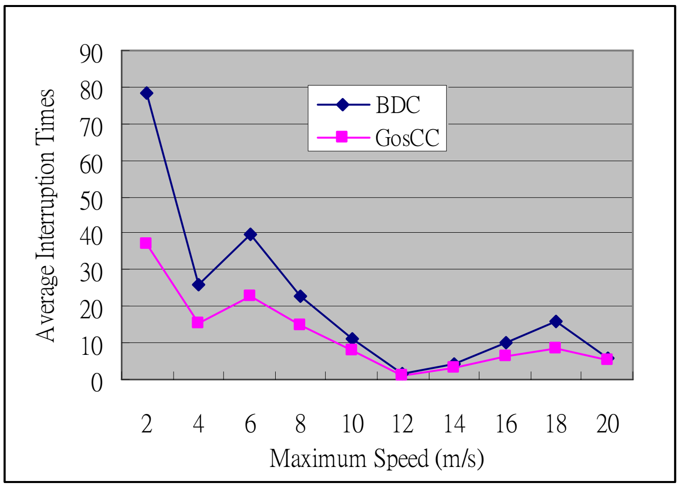


Figure 7.12: Caching overheads vs. Maximum speed.

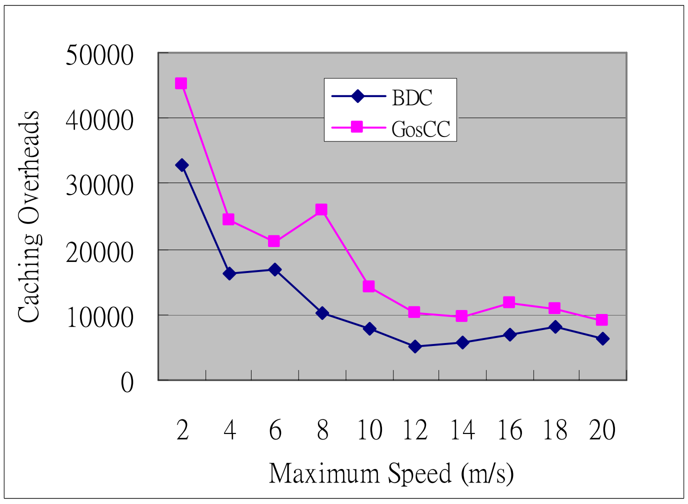


Figure 7.13: Average interruption times vs. Maximum speed.

Simulation results show that GosCC achieves much better performance than BDC in terms of average interruption interval, average interruption times. However, the message cost is normally greater than that of BDC. The reason mainly includes: 1) we consider the sequential relation among data items such that GosCC is much “smarter” than other cooperative caching strategies. The progresses of mobile users using data items play an important role in making caching decisions. The global view of the progress information is propagated throughout the whole network in a timely way; and 2) redundant messages in gossiping cannot be avoided so that message cost is still worse than other cooperative caching strategies. This will be one of our future works to control the redundant message in a more grained way.

7.6. Summary

In this chapter, we first present the Sequential Relation Cache Placement problem (SRCP) and aim at minimizing total interruption intervals. SRCP is with the major difference from the traditional cache placement problem, in the way that we consider user experience as our major concern after we introduce the sequential data relation into cooperative caching. To increase the probability that mobile nodes access data item in their local caches, we present a cooperative caching strategy named Gossip-based Cooperative Caching (GosCC) that helps mobile nodes exchanges the information on cache content and the progress of using data reliably and timely. We introduce three heuristic rules to minimize total interruption intervals. In our simulations, we evaluate the proposed algorithms in IMANETs with different settings. We take average interruption interval, average interruption times, and caching overheads as our performance metrics. The results demonstrate that our algorithms outperform the previous work in terms of average interruption intervals and average interruption times, while sacrificing message cost to a certain degree.

Chapter 8. Conclusions and Future Works

In this chapter, we briefly summarize our works and outline the directions for future research.

8.1. Conclusions

Data dissemination and sharing are two important issues on the topic of mobile data management. In this thesis, we mainly study how to disseminate and share one data item or multiple data items in a mobile ad hoc network, from the point of view of theoretical algorithm research and practical protocol design. The major work of the thesis can be summarized as follows.

In Chapter 3, considering the weak capacity of mobile devices, we investigate the problem, to what extent gossip-based protocols can tolerate node failures, yet guarantee the specified message delivery. We propose a generalized gossiping algorithm and develop a mathematical model based on generalized random graphs for evaluating the fault-tolerance of gossiping. We analytically derive the maximum ratio of failed nodes that can be tolerated without reducing the required degree of reliability. We also investigate the impact of the parameters, namely the fanout distribution and the non-failed member ratio, on the reliability of gossip-based reliable multicast protocols. Simulations have been carried out to validate the effectiveness of our analytical model in terms of the reliability of gossiping and the success of gossiping. The results obtained can be used to guide the design of fault tolerant gossip-based protocols.

In Chapter 4, considering the typical hierarchical structure in an infrastructure-based wireless network, we propose a generalized hierarchical gossiping algorithm. To evaluate the reliability of the proposed algorithm, we develop a mathematical model based on generalized random graphs. We investigate the impact of the parameters, the fanout distributions at the two levels of hierarchy, on the reliability of hierarchical gossiping. We also discover the critical condition for guaranteeing the gossiping messages to be propagated from local subgroups to the whole group. Our simulation works have been carried out to validate the effectiveness of our analytical model, in terms of the reliability of gossiping and the success of gossiping.

In Chapter 5, considering the contentions when wireless nodes communicate with each other, we define the cache placement problem on a dynamic network topology as Dynamic Cache Placement (DCP), in which mobile nodes aim to share single data item in mobile computing environments. There is only one data source that maintains one data item and other client nodes require accessing the data item. Due to the impact of contentions in wireless networks, one hop delay is different from each other and it varies with the traffic load. Thus the previous algorithms cannot achieve their expected performance. We define three data flows in DCP, including access flows, reply flows, and update flows. Then we present the idea of hedging flow to evaluate the benefits of select a node as a candidate of a cache node. We propose two heuristic cache placement algorithms, Centralized Contention-Aware Caching Algorithm (CCCA) and Distributed Contention-aware Caching Algorithm (DCCA), both of which detect the variation of contentions to evaluate the benefits of selecting a node as cache node. Simulation results show that the proposed algorithms achieve much better performance than other alternative ones in terms of average query delay, caching overheads, and query success ratio.

In Chapter 6, we consider the cache placement problem in cooperative caching for sharing multiple data items among a group of mobile nodes in an Internet-based Mobile Ad hoc Network (IMANET). A typical cache placement strategy deals with the problem that what data items should be cached at which mobile nodes, in order to minimize total access delay for mobile users accessing a set of data items.

However, most of the existing solutions rely on a node's knowledge about data access frequencies. In an IMANET, due to the dynamic topology changes and scarcity of resources on mobile nodes, it is difficult and costly for mobile nodes to exchange the information about data access frequencies and the hops to their corresponding closest cache nodes. To address these challenging issues, we propose a solution named Divide-and-Rule Cooperative Caching (DRCC), which divides the cache space of each node into two components: selfish and altruistic. In the selfish component, mobile nodes cache the most frequently accessed data items according to its own preference, showing the side of selfishness of a node. In the altruistic component, mobile nodes select data items in a randomized way, showing the side of altruism of a node. Given a specific distribution of data access frequencies, we can find a near-optimal allocation solution to allocate cache sizes for the two components, aiming at minimizing the total access cost. Simulation results show that DRCC achieves much better performance than the existing best cooperative caching strategy in MANETs in terms of average query delay, caching overheads, and query success ratio. In particular, DRCC reduces caching overheads by 40% in average.

In Chapter 7, considering the relation among a set of data items, we reconsider the cache placement problem in cooperative caching for sharing multiple data items in Internet-based Mobile Ad Hoc Networks (IMANETs). With more demands on sharing a video or other media contents, the relations among data segments (items) becomes much more important to improve the efficiency of data access. We review the existing relations among data items and select the sequential relation as the most typical one. We present a novel solution named Gossip-based Cooperative Caching (GosCC) to address the cache placement problem, considering the sequential relation among data items. Each mobile node accesses all the data items in a sequential order and the current ID of the data item accessed by a mobile node is stored into a progress report. GosCC makes use of the information about the progress reports of mobile nodes and the contents in mobile nodes' caches, in order to determine whether a data item should be cached at a mobile node. To obtain the aforementioned information reliably and in time, GosCC applies the gossiping scheme to guarantee that mobile nodes receive the accurate and timely information for making caching decisions. Simulation results show that GosCC achieves better performance than other alternatives in terms of average interruption intervals and average interruption times, while sacrificing message cost to a certain degree.

8.2. Future Research Works

In Chapter 3, we propose a generalized gossiping algorithm, which can cover all the distributions of the fanouts. We take the Poisson distribution as an example in the thesis. However, it is still challenging to consider other distributions in order to discover some new characteristics of gossiping, such as the Power-law distribution or other specific distributions. Another possible issue is to consider how to select gossiping targets more reasonable. In the existing works, there are few works considering the relationship between the neighbors of a mobile node in view of membership and the physical neighbors in real network settings. As we all know, it is relatively easier to construct an overlay in a fixed network. However, how to build up an overlay in a mobile network will have a great impact on the performance of gossiping. This can be one of the possible directions in this topic.

In Chapter 4, we propose a general hierarchical gossiping algorithm in order to take advantage of the inherent hierarchical structure of the infrastructure-based wireless networks. However, our work is a preliminary work for protocol designs in wireless networks. We should take a typical hierarchical wireless network as an example to provide reliable multicast protocols. We consider how to provide a gossip-based reliable multicast protocol for Wireless Mesh Networks (WMNs), which have emerged as a significant technology for the next-generation networking. WMNs are dynamically self-organized and self-configured, with the nodes in the network automatically establishing an ad hoc network and maintaining the mesh connectivity. There are two kinds of nodes in WMNs, mesh routers and mesh clients. Mesh routers plays the role of building the backbone of WMNs due to their minimal mobility. WMNs can be integrated with other networks such as the Internet, cellular, wireless LANs, wireless MAN, etc. through the gateway and bridging functions in the mesh routers. According to these characteristic of WMNs, building up a hierarchical gossip-based reliable multicast protocol is a promising work.

In Chapter 5, 6, and 7, we mainly discuss the cache placement problem in mobile computing environments. We take wireless interferences, user mobility, limited cache size, data access pattern, and data relation into consideration, and find some new problems in our research. Although we consider a TTL-based consistency model in CCCA and DCCA, cache consistency is still another important problem in data caching. Few works consider both the cost of data access and the cost of maintaining cache consistency. For example, Tang et al. [TCC07] considers the access cost with the maintaining cost for a TTL-based consistency model. This can be one of pioneering works on considering both two costs. Nevertheless, this work is considered in the Interned. Tang et al. [TG07] and Hara [H01] consider the update cost with the access cost together. However, there is no any consistency model in the two papers. As we all know, there are many consistency models for maintaining different levels of cache consistency. How to consider the maintaining cost with the access cost under different consistency models will be one of the possible directions on the topic of cooperative caching.

In Chapter 7, we introduce data relation into the topic of cooperative caching. However, we only take the sequential relation among data items as our research subject, although we categorize these relations into time-based relationship, spatial-based relationship, and semantic relationship, as we mentioned in the Section 7.1. What are the impacts of other relationships on the performance of cooperative caching? Is there any more formal model on how to describe the relationship among data items? To answer these problems, there should be some novel works to be done in the future.

References

[AAF+95] S. Acharya, R. Alonso, M. Franklin and S. Zdonik. Broadcast Disk: Data Management for Asymmetric Communication Environments. In *Proc. ACM SIGMOD*, pp. 199-210, May 1995.

[ADH05] A. Allavena, A. Demers and J. Hopcroft. Correctness of a Gossip-based Membership Protocol. In *Proc. 24th ACM Symposium on the Principle of Distributed Computing (PODC)*, pp. 292–301, 2005.

[AM98] S. A. Ahson, and I. Mahgoub. Research Issues in Mobile Computing. In *Proc. of IEEE Int'l Performance, Computing, and Communications Conference (IPCCC'98)*, pp. 209-215, 1998.

[ASM+08] H. Artail, H. Safa, K. Mershad, Z. Abou-Atme, N. Sulieman. COACS: A Cooperative and Adaptive Caching System for MANETS. *IEEE Transactions on Mobile Computing*, Vol. 7, No. 8, pp. 961-977, 2008.

[AWW05] I.F. Akyildiz, X. Wang, and W. Wang, Wireless Mesh Networks: A Survey, *Computer Networks Journal (Elsevier)*, March 2005.

[B01] B. Bollobás. *Random Graphs*, Cambridge University Press, U.K, pp. 130-153, 2001.

[BBK+00] S. Bespamyatnikh, B. Bhattacharya, D. Kirkpatrick, and M. Segal. Mobile facility location. In *Proc. of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing & Communications*, 2000

[BCF+99] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. of INFOCOM*, 1999.

[BE03] C. Bettstetter and J. Eberspacher, Hop Distances in Homogeneous Ad Hoc Networks, In *Proc. 57th IEEE Vehicular Technology, Conf. (VTC-Spring '03)*, vol. 4, pp. 2286-2290, Apr. 2003.

[BHO+99] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal

Multicast. *ACM Transactions on Computer Systems*, Vol. 17, No. 2, pp 41-88, May, 1999.

[BI94] D. Barbara and T. Imielinski. Sleepers and Workaholics: Caching Strategies in Mobile Environments. In *Proc. ACM SIGMOD*, pp. 1-12, 1994.

[BMJ+98] J. Broch, D. A. Maltz, D. B. Johnson, Y-C. Hu, and J. Jetcheva. A Performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols. In *Proc. of MOBICOM*, 1998.

[BO00] G. Barish and K. Obraczka. World Wide Web Caching: Trends and Technologies. *IEEE Communications Magazine*, Internet Technology Series, 2000.

[BPC+07] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Hu. Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee standards. *Computer Communications*, 30(7): 1655-1695 (2007).

[BR01] I. Baev and R. Rajaraman. Approximation Algorithms for Data Placement in Arbitrary Networks. In *Proc. of SODA*, 2001.

[C03] G.. Cao. A Scalable Low-Latency Caching Invalidation Strategy for Mobile Environments. *IEEE Tran. Knowledge and Data Engineering*, 15, 1, 2003.

[CG99] M. Charikar and S. Guha. Improved Combinatorial Algorithms for the Facility Location and k-median Problems. In *Proc. of FOCS*, 1999.

[CLC07] C. Chow, H. Leong, A. Chan. GroCoca: Group-based Peer-to-Peer Cooperative Caching in Mobile Environment. *IEEE Journal on Selected Areas in Communications*, Vol. 25, No. 1, January 2007.

[CMC99] M.S. Corson, J.P. Macker, G.H. Cirincione. Internet-Based Mobile Ad Hoc Networking. *IEEE Internet Computing*, July–August 1999, pp. 63–70.

[CNS+00] D. Callaway, M. Newman, S. Strogatz and D. Watts. Network Robustness and Fragility: Percolation on Random Graphs. *Phys. Rev. Lett.* 85, pp. 5468-5471. 2000

[CS02] E. Cohen and S. Shenkar. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of the ACM Conference of the Special Interest Group on Data*

Communication (SIGCOMM), 2002.

[CRB01] R. Chandra, V. Ramasubramanian, and K. Birman. Anonymous gossip: improving multicast reliability in mobile ad hoc networks. In *Proc. 21st Int. Conf. Distributed Computing Systems (ICDCS)*, 2001, pp. 275–283.

[CZX$^+$07] J. Cao, Y. Zhang, L. Xie and G. Cao, Data Consistency for Cooperative Caching in Mobile Environments, *IEEE Computer*, Apr. 2007.

[DFJ$^+$06] S. Dar, M. Franklin, B. Jónsson, and D. Srivastava, and M. Tan. Semantic Data Caching and Replacement. In *Proceedings of the 22th International Conference on VLDB*, 1996.

[DGH$^+$87] A.J. Demers, D.H. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic Algorithms for Replicated Database Maintenance. In *Proc. Sixth Ann. ACM Symp. Principles of Distributed Computing (PODC)*, pp. 1-12, Aug. 1987.

[DGV09] Y. Du, A, Gupta, G. Varsamopoulos. Improving On-demand Data Access Efficiency in MANETs with Cooperative Caching. Elsevier Ad Hoc Networks, pp. 579-598, 2009.

[EGH$^+$03] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A. Kermarrec. Lightweight probabilistic broadcast. *ACM Tran. Computer*, (4): pp. 341-374 (2003).

[EGK$^+$04] P. Euqster, R. Guerraoui, A. Kermarrec and L. Massoulieacute. Epedimic Information Dissemination in Distributed System. *IEEE Computer*, Vol. 37, No 5, May, 2004.

[EGR$^+$08] Friedrich Eisenbrand, Farizio Grandoni, Thomas Rothvob, Guido Schafer. Approximating Connected Facility Location Problems via Random Facility Sampling and Core Detouring. In *Proc. 19th Annu. ACM-SIAM Symp. on Discrete Algorithms (SODA'08)*, San Francisco, California, 20-22.01.2008.

[FanCW$^+$08] X. Fan, J. Cao, W. Wu and H. Cheng. Modeling Hierarchical Gossiping in Reliable Multicast Protocols. In Proc. of the *2nd International Conference on Future*

Generation Communication and Networking (FGCN'2008), December 2008, Sanya, Hainan Island, China, (Invited Paper).

[FCW09] X. Fan, J. Cao, and W. Wu. Contention-Aware Data Caching in Wireless Multi-hop Ad Hoc Networks. In Proc. of the *sixth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'2009),* October 2009, Macau SAR, China.

[FCW$^+$08] X. Fan, J. Cao, W. Wu, and M. Raynal. On Modeling Fault Tolerance of Gossip-Based Reliable Multicast Protocols. In *Proc. of ICPP*, 2008.

[FV97] Kevin Fall and Kannan Varadhan. NS notes and documentation. in The VINT Project, UC Berkely, LBL, USC/ISI, and Xerox PARC, 1997.

[FZ94] G. Forman and J. Zahorjan. The Challenges of Mobile Computing. *IEEE Computers*, 1994, pp 38-47.

[GKM01] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie´. SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication. In *Proc. of the Third Int'l Workshop Networked Group Comm. (NGC)*, Nov. 2001.

[GT92] R. Golding and K. Taylor. Group Membership in the Epidemic Style. *Technical Report UCSC-CRL-92-13*, Dept. of Computer Science, Univ. of California, Santa Cruz, 1992.

[H01] T. Hara. Effective Replica Allocation in Ad Hoc Networks for Improving Data Accessibility. In *Proc. of INFOCOM,* 2001.

[HCW$^+$07] Yu Huang, Jiannong Cao, Zhijun Wang, Beihong Jin and Yulin Feng, Achieving Flexible Cache Consistency for Pervasive Internet Access, in *Proc. of the 5th Annual IEEE Intl. Conf. on Pervasive Computing and Communications (PerCom)*, pp.239-250, 2007

[IB93] T. Imielinski, and B. Badrinath. Data Management for Mobile Computing. ACM SIGMOD Record, 1993.

[IK96] T. Imielinski, and H. F. Korth. *Mobile Computing*. Kluwer Acdemic Publishers, 1996.

[JLW$^+$07] W. Jia, D. Lu, G. Wang, L. Zhang and W. Lu. Local Retransmission-based Gossip

Protocol in Mobile Ad Hoc Networks. In *Proc. of IEEE WCNC 2007*, pp. 4244-4249.

[JV01] K. Jain and V. Vazirani. Approximation Algorithms for Metric Facility Location and k-median Problems using the Primal-dual Schema and Lagrangian Relaxation. *Journal of the ACM*, 48(2), 2001.

[KD02] M.R. Korupolu and M. Dahlin, Coordinated Placement and Replacement for Large-Scale Distributed Caches, *IEEE Trans. Knowledge and Data Eng.*, vol. 14, no. 6, pp. 1317-1329, Nov./Dec. 2002.

[KMG03] A.-M. Kermarrec, L. Massoulie´, and A.J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Trans. Parallel and Distributed Systems*, VOL. 14, NO. 3, March, 2003.

[KP97] G. Kuenning and G. Popek. Automated Hoarding for Mobile Computers. In *Proc. of SOSP*, 1997.

[KRS00] P. Krishnan, D. Raz, and Y. Shavitt, The Cache Location Problem, *IEEE/ACM Trans. Networking*, vol. 8, no. 5, pp. 568-582, Oct. 2000.

[KRW03] C. Krick, H. Ra¨cke, and M. Westermann, Approximation Algorithms for Data Management in Networks, *Theory of Computing Systems*, vol. 36, no. 5, pp. 497-519, Sept. 2003.

[LEH03] J. Luo, P. T. Eugster, and J. P. Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *Proc. of IEEE INFOCOM*, Apr. 2003, pp. 2229–2239.

[LGI$^+$99] B. Li, M. Golin, G. Ialiano, and X. Deng. On the Optimal Placement of Web Proxies in the Internet. In *Proc. of IEEE INFOCOM*, March 1999.

[LLS99] K. Lee, H. Leong, and A. Si. Semantic Query Caching in a Mobile Environment. In *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 2, Issue 2, April, 1999.

[LWY93] A. Leff, J.L. Wolf, and P.S. Yu, Replication Algorithms in a Remote Caching Architecture, *IEEE Trans. Parallel and Distributed Systems*, vol. 4, no. 11, pp. 1185-1204, Nov. 1993.

[M01] V. Murthy. Mobile Computing: Operational Models, Programming Modes and Software Tools. In *Proc. of the 15th IEEE International Parallel and Distributed Processing Symposium (IPDPS'01)*, pp. 2016-2025, 2001.

[M03] M. E. J.Newman. The Structure and Function of Complex Networks. *SIAM Review* 45, pp. 167-256, 2003.

[M93] J.D. Murray. *Mathematical Biology*. Springer, Berlin, 2nd edition, 1993.

[M99] A. Murphy. Algorithm Development in the Mobile Environment, In *Proc. of Int'l Conf. on Software Engineering*, pp. 728-729, 1999.

[MHV$^+$97] B.M. Maggs, F.M. Heide, B. Vo¨cking, and M. Westermann, Exploiting Locality for Data Management in Systems of Limited Bandwidth, *Proc. IEEE Symp. Foundations of Computer Science (FOCS '97)*, pp. 284-293, Oct. 1997.

[MM03] M. Mallick. *Mobile and Wireless Design Essentials*. Weley Publishing, Inc., Indianapolis, Indianan, 2003.

[N03] M. E. J.Newman. The Structure and Function of Complex Networks. *SIAM Review* 45, pp. 167-256, 2003.

[NSC03] P. Nuggehalli, V. Srinivasan, and C. Chiasserini. Energy-efficient Caching Strategies in Ad Hoc Wireless Networks. In *Proc. of MobiHoc*, 2003.

[NSC$^+$06] P.Nuggehalli, V.Srinivasan, C. F. Chiasserini and R. R. Rao. Efficient Cache Placement in Multi-hop Wireless Networks. In *ACM/IEEE Transactions on Networking*, Volume 14 , Issue 5 (October 2006).

[NSW01] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E* 64, 026118 (2001).

[P97] C. E. Perkins. Mobile IP. *IEEE Communications Magazine*, vol. 35 no. 5, pp. 84-99, 1997.

[PB94] C. Perkins and P. Bhagwat. Highly Dynamic Dsdv Routing for Mobile Computers.

In *Proc. of SIGCOMM*, 1994.

[QPV01] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the placement of web server replicas. In *Proc. of the IEEE Conference on Computer Communications (INFOCOM)*, 2001.

[RDK03] Q. Ren, M. Dunham, and V. Kumar. Semantic Caching and Query Processing. In *IEEE Transaction on Konwledge and Data Engineering*, Vol 15, No.1, pp. 192-210, Januarey/Febauary, 2003.

[RMH98] R. Renesse, Y. Minsky, and M. Hayden. A Gossip-Style Failure Detection Service. In *Proc. of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing, Middleware'98*.

[RS02] M. Rabinovich and O. Spatscheck, Web Caching and Replication. *Addison-Wesley*, 2002.

[RZF] Http://en.wikipedia.org/wiki/Riemann_zeta_function.

[SK02] C. Swamy and A. Kumar. Primal-dual Algorithms for Connected Facility Location Problems. In *Proc. of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX 2002)*, 2002.

[SUE00] Y. Saygin, O. Ulusoy, and A. Elmagarmid. Association Rules for Supporting Hoarding in Mobile Computing Environments. In *Proc. of the 10th International Workshop on Research Issues in Data Engineering*, 2000.

[TCC07] X. Tang, H. Chi, and S. T. Chanson. Optimal Replica Placement under TTL-Based Consistency. *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, pp. 351-363, March 2007.

[TCO01] K. Tan, J. Cai, and B. Ooi. An Evaluation of Caching Invalidation Strategies in Wireless Environments. *IEEE Tran. Parallel and Distributed System*, 12, 789, 2001.

[TG07] B. Tang and H. Gupta. Caching Placement in Sensor Networks Under Update Cost Constraint. *Journal of Discrete Algorithms*, Volume 5, Issue 3, pp. 422-435, September 2007.

[TGD06] B. Tang, H. Gupta and S. Das. Benefit-based Data Caching in Ad Hoc Networks.

In *Proc. of ICNP*, Santa Barbara, California, November 2006.

[V99] U. Varshney. Networking Support for Mobile Computing. *Communications of AIS*, vol. 1 article. 1, pp. 1-30, 1999.

[W99] J. Wang. A survey of web caching schemes for the Internet. *ACM SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 5, pp. 35–46, Oct. 1999.

[WCL+08] Y. Wang, E. Chan, W. Li and S. Lu. Location Dependent Cooperative Caching in MANETs. *In Proc. of ICPP*, 2008.

[WCF09] W. Wu, J. Cao, and X. Fan. Overhearing-aided Data Caching in Wireless Ad Hoc Networks. In Proc. of the *6th IEEE ICDCS International Workshop on Wireless Ad hoc and Sensor Networks (WWASN'09)*, June, 2009, Montreal, Canada.

[WFC07] Z. Wang, X. Fan, and J. Cao. Design a Hierarchical Cache System for Effective Loss Recovery in Reliable Multicast, In Proc. the 7th International Symposium on Advanced Parallel Processing Technologies (APPT'2007), November, 2007, Guangzhou, China.

[YC04] L. Yin and G. Cao. Balancing the Tradeoffs between Data Accessibility and Query Delay in Ad Hoc Network. In *Proc. of SRDS*, 2004.

[YC06] L. Yin and G. Cao. Supporting Cooperative Caching in Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, Vol. 5, No. 1, pp. 77- 89, January, 2006.

[YK06] Y. Yang and R. Kravets. Achieving Delay Guarantees in Ad Hoc Networks through Dynamic Contention Window Adaptation. In *Proc. of IEEE INFOCOM*, 2006.