Analysis of Behaviors of Mobile Agents in Agent-Based Network Applications

by

Wenyu Qu

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

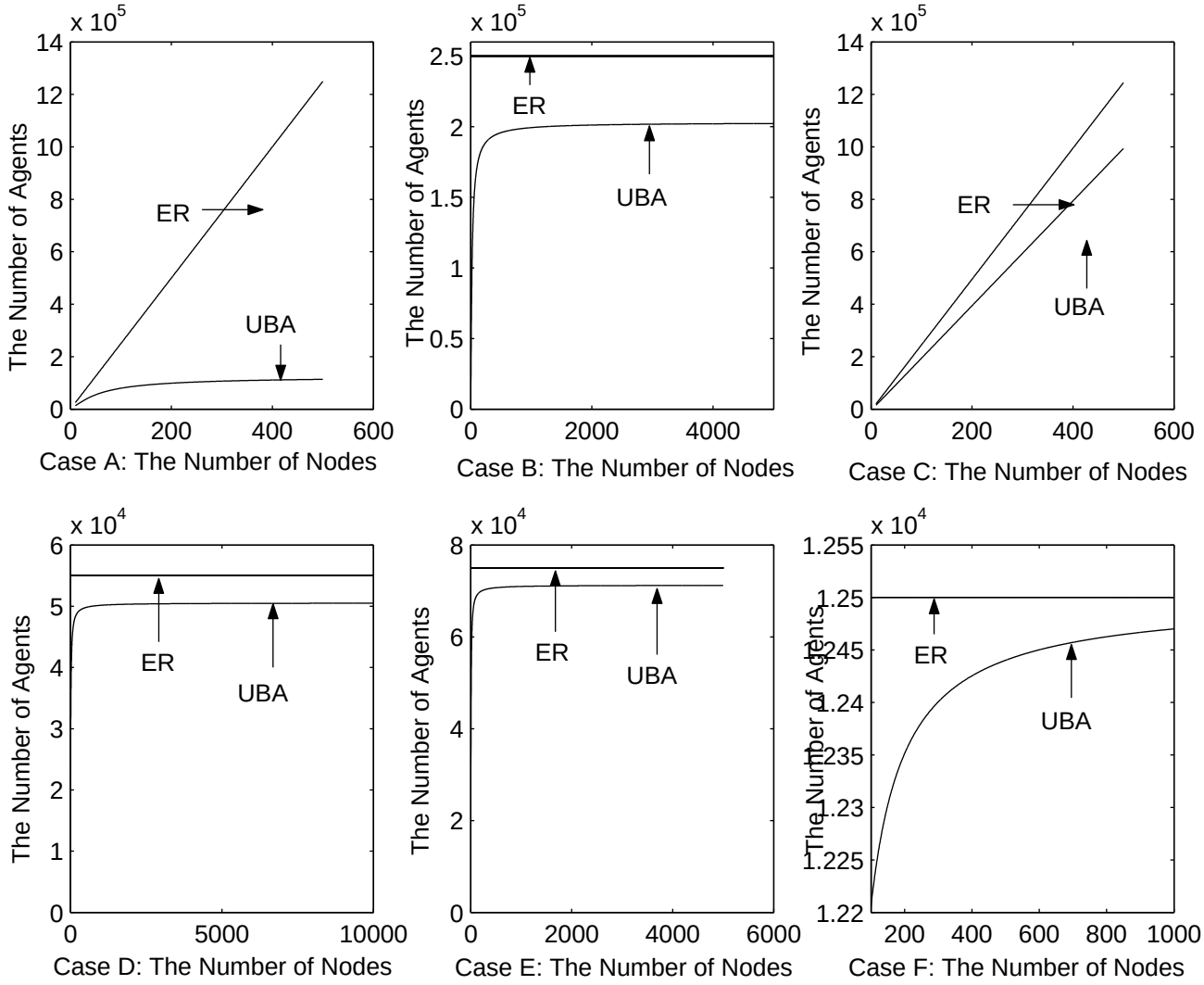
Supervisor: Professor Hong Shen

School of Information Science
Japan Advanced Institute of Science and Technology

February 10, 2006

Abstract

Mobile agents technology has been a popular topic in the area of information engineering for several years. Lots of expectations have been laid on them, but large-scale usage of agents is still waiting. One of the main reasons for the immaturity of agent technology is that we have not been able to make it effective enough to meet the strict requirements put on it. This dissertation addresses the problems of what affects the performance of agent-driven networks and how to improve the network performance. In detail, we analyze some key parameters for characterizing the behaviors of mobile agents, such as population distribution, probability of success, migration time, life expectancy, and propose new effective models based on the analytical results. The main contributions of this dissertation are outlined as follows:



- We address the problem of mobile agent-based network routing. Several routing models and stochastic analysis are presented. We first focus on the traditional ant-like routing algorithm, analyze both the population distribution and the probability of success as two important parameters for measuring the network performance. Based on our analytical results, we proposed several agent-based routing strategies for improving the network performance and presented relevant analysis. We also extend our solution to solve the problem of mobile agent-based e-shopping. Extensive simulation experiments are conducted to evaluate our proposed solutions over a wide range of performance metrics in comparison with existing solutions proposed in the literature.

- We address the problem of fault-tolerant execution of mobile agents. we propose a new fault-tolerant execution model of mobile agents, which effectively combines the replication approach and the checkpointing approach. The behaviors of mobile agents are statistically analyzed through several key parameters, including the migration time from node to node, the life expectancy of mobile agents, and the population distribution of mobile agents, to evaluate the performance of our model. We compare the performance of our solutions with other solutions over various performance metrics through extensive simulation experiments. The simulation results show that our solutions outperform existing solutions proposed in the literature.

- We address the problem of agent-based peer-to-peer networks. We consider the problem of traffic congestion when agents roaming in the network and propose a new agent-based model which can balance the network traffic. We also prove the convergence of the approximating function to the optimal solution and analyze the properties of the approximate function to show the rationality of our algorithm.

Key words: Mobile agents, Internet, routing, fault tolerance, peer-to-peer networks, performance evaluation.

Acknowledgments

I am full of gratitude to Prof. Hong Shen, my supervisor, for guiding me through this work. His keenness of insight and judgement, promptness of reactivity, and constant confidence and persistence have inspired me much and will encourage me for the future.

I deeply appreciate Associate Professor Xavier Defago, my sub-theme supervisor, for his valuable instruction on my research. I also appreciate to all members in Shen Lab, Dr. Xiaohong Jiang, Dr. Haibin Kan, Gui Xie, Zonghua Zhang, Hui Tian, Chao Peng, Yingpeng Sang, Haibo Zhang, Yawen Chen, Bo Wang, Zhibo Zhou, Qing Gong, Yongguang Jin, and all the people who either directly or indirectly provide me knowledge, experience, and support.

I would like to make a special grateful acknowledgement to my family, including my husband, my daughter, my parents and parents-in-law. They have sacrificed many things for me and given me their fullest support. Especially, I am grateful to my parents' taking care of my daughter. Without their backing, it is impossible for me to concentrate on my research and finish my Ph.D study.

I am deeply indebted to the Japan Society for the Promotion of Science (JSPS) Grant-in-Aid for Scientific Research under its grant for Research in Special Areas, the FUNAI Foundation for information technology and the 21st Century COE Program by Ministry of Education, Culture, Sports, Science and Technology for granting me, which makes my study in Japan feasible. Thanks also go to Foundation for C&C Promotion, Telecommunications Advancement Foundation, Inoue Foundation for Science, Tateisi Science and Technology Foundation, and JAIST Foundation Research Grant for Students for supporting me to attend and present our work at some international conferences.

Finally, I dedicate this dissertation to my husband and daughter, Keqiu Li and Mengning Li.

Contents

Chapter 1

Introduction

1.1 Background

The World Wide Web has become the most successful application on the Internet since it provides a simple way to access a wide range of information and services. Mobile agents have been a popular topic in the area of information engineering for several years. Lots of expectations have been laid on them, but large-scale usage of agents is still waiting. One of the main reasons for the immaturity of agent technology is that we have not been able to make it effective enough to meet the strict requirements put on it.

Mobile agents are programs with persistent identity which move around a network on their own volition and can communicate with their environment and with other agents. These systems use specialized servers to interpret the agent's behavior and communicate with other servers. Mobile agents may execute on any machine in a network without the necessity of having the agent code pre-installed on every machine the agent could visit. Mobile agents represent a new model in the evolution of executable content on the Internet, introducing program code that can be transported along with state information.

Mobile agents appeared in the scene the last decade of the previous millennium. They have been embraced by researchers and practitioners as a potential technology that could revolutionize the way we perform computations, develop applications and systems. They were, and still are, viewed as a unique way to approach mobile and wireless computing. Indeed, this is not an exaggeration; mobile agents have been used in a variety of applications and computing areas. The driving force motivating mobile agent-based computation is multifold: Mobile agents provide an efficient, asynchronous method for searching for information or services in rapidly evolving network; mobile agents may be launched into the unstructured network and roam around to gather information. Second, mobile agents support intermittent connectivity, slow networks, and light-weight devices. Thus, mobile agents provide many benefits in Internet system programming (otherwise networking programming) [48, 68, 137], in which there is a need for different kinds of integrated information, monitoring and notification, encapsulating artificial intelligence techniques, security and robustness [30, 66, 67, 105]. Also the mobile agent paradigm has demonstrated satisfactory performance when deployed for distributed access to Internet databases, distributed retrieving and filtering of information and minimizing network workload [67, 88, 95, 104, 129]. Finally, mobile agents have proved very effective in supporting asynchronous execution of client requests, weak connectivity and disconnected operations and the dynamic adaptation to the various types of user connectivity common

in wireless environments [67, 103, 113].

Various mobile agent platforms have been developed. They can be broadly categorized as Java and non-Java based ones, and they can be further split into experimental and commercial ones. The Java-based mobile agents platforms include IBM'Aglets Workbench [4], Recursion Software's Voyager [131], Mitsubishi's Concordia [140], and General Magic's Odyssey [85]. The non-Java-based systems include, for example, TACOMA [58] and Agent Tcl [45]. There is an increasing interest in Java-based platforms due to the inherent advantages of Java, namely, platform independence support, highly secure program execution, and small size of compiled code. These features of Java combined with its simple database connectivity interface (JDBC) that facilitates application access to multiple relational databases over the Web, make the Java approaches very attractive.

Mobile agent technology has already been applied successfully in a wide range of application domains. At the head we have e-commerce, which has for a long time shown potential benefits of agents in theory, and lately proven many agent solutions to be functional also in practice. Another creditable participant in the research of mobile agent technology is the telecommunication industry, which has successfully adapted mobile agents in their applications.

1.2 A Closer Look at the Mobile Agents

Mobile agents technology originally emerged from advances made in distributed systems research [18]. Mobile agents are computational software processes capable of roaming wide area networks (WANs) such as the WWW, interacting with foreign hosts, gathering information on behalf of its owner and coming "back home" having performed the duties set by its user (owner).

But for a mobile agent to work on the different host environments on a network, each host must have a software environment in which mobile agent can exist. We call this the mobile agent environment, which is built on top of a host system. This environment supports services that relate to the mobile agent environment itself, services pertaining to the environments on which the mobile agent environment is built, services to support access to other mobile agent systems, and finally support for openness when accessing non-agent-based software environments.

There are some reasons that mobile agents are highlighted in recent years. Firstly, although much of the earlier work done on mobile agents has been important, the benefit of these agents did not scale up well when they were placed alongside large real-time applications, including manufacturing, network management, and distance learning. In addition, previous research has failed to comprehensively address the behavior of agents serving the telecommunications and applications infrastructure.

Secondly, mobile agents can work against the problems of latency and bandwidth of client-server applications and reduce vulnerability of network disconnection.

Thirdly, WWW and the Internet, which are growing and becoming ubiquitous, dynamic and mobile, are driving mobile agents. People believe that current trends in Internet technology and usage lead to the use of mobile agents. Mobility (users, devices, and programs) becomes important paradigm, in particular e-commerce and mobile commerce. Agent system must be integrated with the Internet and WWW.

Fourthly, mobile agents also provide a single, general framework in which distributed,

information-oriented applications can be implemented efficiently and easily, with the programming burden spread evenly across information, middleware, and client providers.

The idea about agent, autonomous entities able to solve problems by themselves has been fascinating people for a long time. Nowadays, various kinds of agents can be found around us, we just might not realize all of them. Classification of the agents at the main level can be divided into three categories: biological, robotic and computational agents. Our interest focuses on computational agents.

Although the definition of a mobile agent varies largely between different interest groups, there are some common characteristics that fit with most definitions of mobile agents. At minimum, a mobile agent is defined to be an autonomous, self-contained, and identifiable computer program that is able to move within a network, interact with its environment, and act on behalf of the user or another entity. Thus, it is able to react independently without user intervention to stimulus from outside, for example, from humans, platforms or other agents. A few other properties of mobile agents are listed below:

- Mobility – Agent's ability to transport itself from one execution environment to another.
- Adaptive – Being able to learn and improve with experience.
- Character/Personality – An agent has personality and emotional state.
- Co-operative/Collaborative – An agent can act together with other agents for a common purpose.
- Intelligence – Ability to sense its environment, reflect upon it, and take actions to achieve a goal. State is formalized by knowledge.
- Proactive – Agent's actions are goal-orientated. It is able to choose if certain action will bring it closer to its goal and acts accordingly.
- Reactive – Responds in a timely fashion to changes in its environment.
- Social – Can communicate with other agents or people.
- Temporally continuous – Is a continuously running process.

One of the most interesting additional properties of software agents form our point of view is mobility, which means that the agent is able to transport itself from one execution environment to another. This moving mechanism is also referred to as migration. A mobile agent is a specific form of software agent, which is a self-contained and identifiable computer program that can move within a network and act on behalf of the user or another entity.

Most mobile agent systems base their agent mobility on code mobility, where both the data and the code of the agent are transferred during migration. This kind of agent mobility can be classified as either strong or weak. The difference between these is that strong mobility allows the code and entire execution state of the agent to be transferred, while weak mobility only makes the code to transfer. Strong code mobility is more complex to achieve, it negatively affects performance and introduces more security risks

than weaker mobility, however, it allows an agent to restart execution from the exact point where it was before migration. Weak code mobility also has some unsolved weaknesses in security when applying it to mobile agent systems.

It is evidently clear that there are some of deficiencies in agent technology. First of all, the development of multi-agent systems is still in the hands of top experts in the domain. We are missing efficient tools that would make it possible for non-specialists to unlock the power of agents, which would make it easier to spread mobile agent technology. Secondly, among others, there are also some unsolved problems with interoperability, scalability and security. Most of the developed multi-agent systems are built from scratch for specific purposes in an ad hoc manner, thus thy are on quite a small scale, without good reusability possibilities, and unable to co-operate with other multi-agent systems. Development towards large-scale multi-agent systems and connecting them to each other thus forming large interconnected networks of multi-agent systems has begun lately, but it has raised complexity issues that we are still not able to cope with. Quantitative analysis, the issue in which this thesis is concentrating on, has also proved to be very complex. Some have even claimed that mobile agents will never be analyzed quantitatively enough to be controlled, but this is what we will discuss later on.

1.2.1 Applications of Mobile Agents

Recent years mobile agent systems have become sufficiently well developed to be considered as using them as tools in other areas, such as gathering information, network management, electronic commerce, and mobile user and devices. Applications of mobile agents include, but are not limited to:

- Management and maintenance: In large networks, monitoring, fault detection and remote installation of software components is very difficult and involves large amounts of logging data. It is not possible to produce out-of-the-box diagnostic programs for every purpose, but it would be possible to use mobile agents to monitor the system and bring possible trouble spots to the attention of the system maintainer. There are some telecommunications service providers, such as Lucent Technologies Inc. and Bell Atlantic Corp., who have tried developing products by using mobile agents to facilitate their network management. They believe that mobile agent technology can detect and react to network faults, and enable installable/extensibel/modifiable services, and so on. The system, such as Perpetuum Mobile Procura, uses mobile agents - Netlets to enable advanced network management [94].

- Electronic commerce: Electronic commerce is becoming a reality as standards for electronic payment are deployed. Mobile agents can help to locate the most appropriate offerings, negotiate deals or even conclude business transactions on behalf of their owners. In InAMos project [53], electronic commerce consists of sellers, buyers, traders, and brokers. There are mobile agents to represent consumers in the market environment and also other ordinary mobile agents and services. They attempt to bring electronic market places to the mobile user.

- Mobile devices, mobile users, and mobile computing: Mobile devices are one of the hottest areas of growth in the computer industry. Portable computers get smaller and smaller. Devices from laptops to palmtops to electronic books, from cars to

telephones to pagers, will access Internet services to accomplish user tasks. Yet the wireless access to the services is likely to stay slow. Users will not want to stay online while some complicated query is handled on behalf of the user (power consumption, connection fees). Users can submit a mobile agent which embodies their queries and log off, waiting for the agents and the results to be picked up at a later time.

- Information search and retrieval: The nature of today's work enhances mobile agents' attractions because they offer better ways to deal with information and manipulating it. Agents can act as sophisticated assistants for exposing information systems, move to the place where the data are actually stored rather than moving all of the data through the network and search for the desired information, freeing people to interact with information indirectly and efficiently. For example, the Stormcast system [115] uses mobile gents created at the client side and sent to servers to retrieve and process huge amounts data. Another scenario is to use mobile agent approach instead of traditional client-server approach in search engines. The mobile agent approach processes data and data source.

- Secure brokering. An interesting application of mobile agents is in collaborations in which not all the collaborators are trusted. The parties could let their mobile agents meet on a mutually agreed secure host where collaboration takes place without risk of the host taking the side of one of the visiting agents.

- Telecommunication networks services. Support and management of advanced telecommunication services are characterized by dynamic network reconfiguration and user customization. The physical size of these networks and the strict requirements under which they operate call for mobile agent technology to function as the glue keeping the system flexible yet effective.

- Workflow applications and groupware. The nature of workflow applications includes support for the flow of information among coworkers. Mobile agents can be useful here because, in addition to mobility, they provide a degree of autonomy to the workflow item. Individual workflow items fully embody the information and behavior they need to move through the organization-independent of any particular application.

- Parallel processing. Mobile agents are capable of cloning themselves in the network, this makes them suitable for parallel processing tasks. If a computation requires so much processor power that it must be distributed among multiple processors, an infrastructure of mobile agent hosts can be a plausible way to allocate the related processes.

1.2.2 Standardizations of agent technology

Today there are numerous different research groups and commercial actors researching agent technologies and implementing various agents and platforms for therm. Because of the diversity of the agent technology research, a common standardization organization was needed. the Foundation for Intelligent Physical Agents (FIPA) was formed in 1996 to fill this gap and since then it has contributed a lot to agent technology. FIPA is a non-profit organization that produces software standards for heterogeneous and interacting agents

and agent-based systems. In the production of these standards, FIPA requires input and collaboration from its memberships and from the agent's field in general to build specifications that can be used to achieve interoperability between agent-based systems developed by different companies and organizations. Also these days, FIPA is the main driving force aiming to produce standards for the interoperation of heterogeneous software agents.

FIPA's core standardization activities are mainly handled by Technical Committees, which concentrate on creating and maintaining specifications. There are currently many active Technical Committees including, for example, ontology, interaction protocols, semantics and security, which is particularly interesting to us. The Security Technical Committee's objective is to lead the research and development of multi-agent systems. Its area of responsibility covers the understanding of security requirements and properties, defining FIPA-based abstract security specification and to concretizes security specifications to form an agent-based security service for deployment in application domains. The Security Technical Committee's work started in the beginning of 2003 and is planned to be ready at the end of 2004.

Another major standardization organization and contributor in the field of mobile agent technology is the object Management Group (OMG), which has been particularly active in making initiatives and standards for interoperability and security. OMG's core infrastructure for mobile agent technology is specified in MASIF (Mobile Agent System Interoperability Facility) with its main objective to enable interoperability between agents systems. Although MASIF addresses security quite extensively, concentrating on authentication and authorization, it has a few deficiencies and limitations. For example, MASIF does not address multi-hop authentication, thus agents are only allowed to migrate one-hop from the source system. An agent's itinerary is also restricted to only trusted nodes, which limits autonomous agent activity. MASIF's security solutions conform extensively to CORBA security services. Agent communication is also dependent on CORBA standards, since communication goes through the ORB communication channel. Despite the promising standardization work OMG has carried out mainly in the late 1990's, a number of its contributions seem to have diminished noticeably in recent years. It remains to be seen how important a part OMG will have in the future of mobile agent technology.

In addition to FIPA and OMG, there are also a few other standardization organizations that are working towards initiations and standards for mobile agent technology. IETF (Internet Engineering Task Force) has paid special interest to Internet-agent technologies. Another standard body, W3C has addressed Web languages and ontology for mobile agents. Some other research groups and formal forums that have promoted mobile agent technology are the Agent Society, NIST and the Agent InteropWorking Group.

Standardization is always very important from interoperability and reusability points of view, having thus also an important effect on extended use of any new technology. On the other hand, standardization is a compromise, and the standardization process itself can be very slow. Therefore, there is also a possibility that the standardization process itself prevents the broad use of new technology. In the area of multi-agent systems, the main problem of standardization is the lack of consistent theory and definitions, although agent communities are working all the time to bridge these problems.

1.2.3 Existing Mobile Agents Platforms

Enabling Technologies

Java and Obliq [18] are the two main technologies, which enable the building of mobile agent frameworks.

- Obliq is a lexically-scoped un-typed interpreted language that supports distributed object-oriented computation. It is one of the oldest mobile agent systems around. It is impractical as an implementation system.
- Java environment is a new approach to distributed computing. It is an interpreted, object-oriented language and library set. Java and its run-time system have combined them to produce a flexible and powerful programming system which supports distributed computing. It has recently received a lot of attention, which has resulted in significant revisions of the important RMI and OS facilities.

Existing Mobile Agents Systems

Today, there are many mobile agents systems, some of them are still in the research phase, some are commercial systems. Five of the most important mobile agents systems in use of research today are reviewed in this section. They are Bee-agent [126], Aglets [52], Hive [81], AgentTcl [44], and Telescript [135].

- **Ara**

  The Agent for Remote Action (ARA) [93] system is a platform for mobile agents developed at the University of Kaiserslautern. Mobile agents in Ara are programmed in Tcl, Java, and C/C++. For all three languages, Ara provides a *go* instruction, which automatically captures the complete state of the agent, transfers the state to the target machine, and resumes agent execution at the exact point of the *go*. Ara also allows the agent to *checkpoint* its current internal state at any time during its execution. Unlike other multiple-language systems, the entire Ara system is multi-threaded; the agent server and the language interpreters run inside a single Unix process. Although this approach complicates the implementation, it has significant performance advantages, since there is little interpreter startup or communication overhead. When a new agent arrives, it simply begins execution in a new thread, and when one agent wants to communicate with another, it simply transfers the message structure to the target agent, rather than having to use inter-process communication.

  The Ara group is finishing implementation work on their initial security mechanisms. An agent's code is cryptographically signed by its manufacturer (programmer); its arguments and its overall resource allowance are signed by its owner (user). Each machine has one or more virtual places, which are created by agents and have agent-specified admission functions. A migrating agent must enter a particular place. When it enters the place, the admission function rejects the agent or assigns it a set of allowances based on its cryptographic credentials. These allowances, which include such things as filesystem access and total memory, are then enforced in simple wrappers around resource-access functions.

- **D'Agents**:

  D'Agents, formerly known as Agent Tcl [46], is a mobile-agent system developed in the Darthmouth College in Hanover, New Hampshire. It is based on the Tcl scripting language, to run agents, and the Safe Tcl extension, to ensure security. Like Ara, D'Agents provides a *go* instruction for each language, and automatically captures and restores the complete state of a migrating agent. Unlike Ara, only the D'Agent server is multi-threaded; each agent is executed in a separate process, which simplifies the implementation considerably, but adds the overhead of inter-process communication.

  The D'Agent server uses public-key cryptography to authenticate the identity of an incoming agent's owner; stationary resource-manager agents assign access rights to the agent based on the authentication and the administrator's preferences; and language-specific enforcement modules enforce the access rights, either preventing a violation from occurring (e.g., filesystem access) or terminating the agent when a violation occurs (e.g., total CPU time exceeded). Each resource manager is associated with a specific resource such as the filesystem. The resource managers can be as complex as desired, but the default managers simply associate a list of access rights with each owner. Unlike Ara, most resource managers are not consulted when the agent arrives, but instead only when the agent (1) attempts to access the corresponding resource or (2) explicitly requests a specific access right. At that point, however, the resource manager forwards all relevant access rights to the enforcement module, and D'Agents behaves in the same way as Ara, enforcing the access rights in short wrapper functions around the resource-access functions. D'Agents has been used in several information-retrieval applications.

- **Aglet**:

  Aglets [4,67] is developed at the IBM Japan. Agents in Aglets are Java objects that can move from one host on the Internet to another. That is, an agent that executes on one host can suddenly halt execution, dispatch to a remote host, and resume execution there. When the agent moves, it brings along its program code as well as its data.

  Rather than providing the *go* primitive of Ara and D'Agents, Aglets uses variants of Tacoma model where agent execution is restarted from a known entry point after each migration. In particular, Aglets uses an event-driven model. When an agent wants to migrate, it calls the *dispatch* method. The Aglets system calls the agent's *onDispatching* method, which performs application-specific cheanup, kills the agent's threads, serializes the agent's code and object state, and sends the code and object state to the new machine. On the new machine, the system calls the agent's *onArrival* method, which performs application-specific initialization, and then calls the agent's run method to restart agent execution.

  Aglets includes a simple persistence facility, which allows an agent to write its code and object state to secondary storage and temporarily "deactivate" itself; proxies, which act as representatives for Aglets, and among other things, provide location transparency; a lookup service for finding moving Aglets; and a range of message-passing facilities for inter-agent communication.

- **Telescript**:

  Telescript [136], developed by General Magic in the early 1990s as the first system designed expressly to support mobile agents in commercial applications, includes an object-oriented, type-safe language for agent programming. Telescript servers, which are called *places*, offer services, usually by installing stationary agents to interact with visiting agents. Agents use the *go* primitive for absolute migration to places, specified using DNS-based hostnames. The system captures execution state at the thread level, so the agent resumes operation immediately after the *go* statement. Relative migration is also possible using the *meet* primitive. collocated agents can invoke each other's methods for communication. An event-signaling facility is also available.

  Telescript was not commercially successful, primarily because it required programmers to learn a completely new language. In 1997, the supporting company, General Magic Inc, has now shelved the Telescript project and embarked on a similar, Java-based system called Odyssey [85] that uses the same design framework.

- **Ajanta**:

  Ajanta (Mobile Agent Research Project) is a mobile agent programming system developed at the university of Minnesota. The first evaluation version of Ajanta was made publicly available in 1999 and few years later in 2003, Ajanta's second commercial version was released. The Ajanta agent system is implemented using Java language and is built from agent platforms called agent servers and mobile agents.

  In Ajanta, an agent executes in an isolated protection domain, to prevent any interference by other agents. A server protects its resources by encapsulating them in proxy objects, which are created dynamically and customized for specific client agents. The same mechanism can allow secure interagent communication via method invocation. Communication across the network is also possible using remote-method invocation. Authenticated control functions allow applications to recall or terminate their remote agents at any time. Ajanta also addresses the problem of protecting agent state from malicious servers. It provides cryptographic mechanisms that let an agent's owner secure parts of the agent's state and detect any subsequent tampering. Agents can also keep parts of their state private and selectively reveal certain objects to specific servers.

- **Voyager**

  Voyager, a Java-based agent system developed by Object Space, provides a convenient way to interact, somewhat transparently, with remote objects (via proxy object called "virtual" references), and for objects to move from host to host [84, 131]. When an object moves, it leaves behind a forwarder object that redirects any messages to the new location. "Agent" objects, unlike other objects, may move themselves autonomously by applying the *moveTo()* method on themselves. Voyager moves the code and data of the agent, but not thread state, to the new location and invokes the desired method there. Voyager also allows objects and agents to be made persistent through explicit *saveNow()* calls, supports group communication

(multicase), and a provides federated directory service. Voyager provides the same basic security mechanisms as other Java-based systems.

- **Tacoma**

  Tacoma is a joint project of Norway's University of Tromso and Cornell University [56]. The project focuses on OS support for MA and how MA can help solve problems traditionally addressed by operating systems. TACOMA proposes abstractions such as briefcase, folder, and file cabinet as extensions to the OS to provide mechanisms for data transfer and broker agent for scheduling agents. It supports both synchronous and asynchronous communication. An alternative communication mechanism is the use of cabinets, which are immobile repositories for shared state. Agents can store application-specific data in cabinets, which other agents then can access. Unlike Ara and D'Agents, Tacoma does not provide automatic state-capture facilities. Instead, when an agent wants to migrate to a new machine, it creates a folder into which it packs its code and any desired state information . The folder is sent to the new machine, which starts up the necessary execution environment and then calls a known entry point within the agent's code to resume agent execution. For fault tolerance, Tacoma uses checkpointing and provides rearguard agents for tracking mobile agents as they migrate.

- **Concordia**

  Concordia is developed by Mitsubishi Electric [132, 140]. It is a Java-based mobile-agent system that has a strong focus on security and reliability. Like most other Java-based systems, it moves an agent's object code and data, but not thread state, from one machine to another. Like many other systems, Concordia agents are bundled with an itinerary of places to visit, which can be adjusted by the agent while en route. Agents, events, and messages can be queued if the remote site is not currently reachable. Agents are carefully saved to a persistent store, before departing a site and after arriving at a new site, to avoid agent loss in the event of a machine crash. Agents are protected from tampering through encryption while they are in transmission or stored on disk; agent hosts are protected from malicious agents through cryptographic authentication of the agent's owner, and access control lists that guard each resource.

- **Jumping Beans**

  Computers wishing to host mobile agents run a Jumping Beans [60] agency, which is associated with some Jumping Beans domain. Each domain has a central server, which authenticates the agencies joining the domain. Mobile agents move from agency to agency within the domain, and agents can send messages to other agents in the domain. Both mechanisms are implemented by passing through the domain server. Thus the server becomes a central point for tracking, managing, and authenticating agents. It also becomes a central point of failure or a performance bottleneck, although the company plans to develop scalable servers to run on parallel machines. Another approach to scalability is to create many small domains, each with its own server. In the current version, agents cannot migrate between domains, but the company plans to support cross-domain migration in future versions. Security and reliability are key aspects of Jumping Beans. Public-key cryptography

is used to authenticate agencies to the server and vice versa, and access-control lists are used to control an agent's access to resources, based on the permissions given to the agent's owner.

- **Obliq**

  An Obliq object is a collection of named fields that contain methods, aliases and values [17,18]. An object can be created at a remote site, cloned onto a remote site, or migrated with a combination of cloning and redirection. Implementing mobile agents on top of these mobile objects is straightforward. An agent consists of a user-defined procedure that takes a briefcase as its argument; the briefcase contains the Obliq objects that the procedure needs to perform its task. The agent migrates by sending its procedure and current briefcase to the target machine, which invokes the procedure to resume agent execution.

- **Bee-Agent**:

  Bee-agent [54], which is developed by Computer and Network Systems Laboratory Corporate Research and Development Center, TOSHIBA Corporation in 1999, is a new type of development framework in that it is a 100% pure agent system. As opposed to other following systems, which make only some use of agents, Bee-agent completely "Agentifies" the communication, that takes place between software applications. The applications become agents, and all messages are carried by agents. Bee-agent allows developers to build flexible open distributed systems that make optimal use of existing applications, and is able to work with Java 2 environment. In this research, Bee-Agent is used to implement the mobile agents within the Web-based prototype developed by using Java 2.

- **Hive System**:

  Hive system is a decentralized system for building applications by networking local system resources. It is applied to the problem of networking "Things That Think" [125] embedding computation and communication in everyday objects and appliances. Hive consists of three components: cells-nodes in the decentralized network, shadows-local resources, and agents-active computation. Its mobility is in many ways a reimplementation of systems such as Telescript, AgentTcl, Aglets, and Mole. But the research intends to create a simple version of mobility to the hard problems.

1.3 Dissertation Outline

For easy understanding and reading, we organize each chapter in a self-contained format as follows:

Chapter 2 addresses the problem of network routing by using of mobile agents. First, an ant-like mobile agent-based routing algorithm is described and the behavior of mobile agents is analyzed by two parameters: the population distribution and the probability of success. A smaller upper bound of the number of mobile agents was provided compared with existing results and for the first time the probability of success was considered. Second, we extend this solution to two more complex cases, namely settleable case and

non-settleable case, and faulty networks. Finally, extensive simulation experiments are conducted to evaluate our solutions.

Chapter 3 concentrates on the problem of fault-tolerant execution of mobile agents. In this chapter, we first surveyed existing fault-tolerant schemes. Based on the existing literature, s solution, we present a new fault-tolerant execution model of mobile agents by effectively combining existing techniques. We then theoretically analyzed the migration time, the life expectancy, and the population distribution of mobile agents to evaluate the performance of our model. Finally, we compare the performance of our solutions with existing solutions proposed in the literature with simulation experiments.

Chapter 4 focuses on agent-based peer-to-peer networks. We propose an effective agent-based execution algorithm for use in peer-to-peer networks and present an optimal solution for agents deciding their routes.

Chapter 5 summarizes our work and discusses some future work.

Chapter 2

Mobile Agent-Based Network Routing

Mobile agent-based network routing is a new technique for routing in large-scale networks. An analysis of the searching activity and population growth of mobile agents is important for improving performance in agent-driven networks.

Many network routing techniques have been developed to deal with the enormous amount of data available on the Internet. These techniques have created exciting opportunities for both business and science [16, 69, 74, 114]. The deployment of mobile agents, which are small decision-making programs capable of migrating autonomously from node to node in a computer network, is an important representative of these new techniques and is an effective way to reduce network load and latency [68]. In [78], Milojicic described that mobile agents are autonomous, adaptive, reactive, mobile, cooperative, interactive, and delegated software entities. The application of mobile agents in network routing has attracted significant attention [24,98,143]. Successful examples of mobile agent applications can be found in [67,68]. The use of mobile agents in applications ranging from electronic commerce to distributed computation has also been studied extensively [11, 109, 137].

Routing is a key factor for network performance. It is the process of moving a packet of data from source to destination. As searching for the optimal path in a stationary network is already a difficult problem, the searching for the optimal path in a dynamic network or mobile network will be much more difficult. Mobile agent-based routing algorithm is a promising option for use in these environments [19, 21, 122]. In a mobile agent-based routing algorithm [31,50], once a request is received from the server, the server will generate a number of mobile agents, which will then begin to search for the corresponding destination host. These agents roam around the network and gather relevant information. When a mobile agent finds the destination, the collected information will be sent back to the source host along the same path and marks will be left on the host along the path. Once all the mobile agents come back, the server will determine the best path to satisfy the request. Since little communication is needed between the agents and the server during the process of searching, the burden on the network generated by mobile agents is very light.

In order to conserve network resources and achieve a good probability of success, it is desirable to dispatch a small number of mobile agents. In a mobile agent-based routing algorithm for large-scale networks, mobile agents will be generated frequently. If there are too many agents running in the network, they will consume too many network resources;

thus, affecting the network performance and ultimately blocking the entire network; on the other hand, if there are too few agents running in the network, there is no guarantee that the destination will be found quickly. Therefore, for network management, it is necessary to analyze both the probability of success and the population growth of mobile agents. Furthermore, the probability of success directly affects the searching process. Unfortunately, there are few work done on the analysis for mobile agents, therefore, analysis on mobile agents is necessary and important for network management. In this chapter, we analyze both the probability of success and the population distribution of mobile agents. From the population distribution formula, we estimate the number of mobile agents running in the total network and in one host. The probability of success serves as an important measure for monitoring network performance, and the analysis of population distribution provides a useful tool for reducing the computational resource consumption.

2.1 Related Work

With the exploitation of the Internet, mobile agent technology has attracted an increasing attention. In [143], the authors used a system net, agent nets, and a connector net to model the environment, agents, and the connector, respectively. They also presented a case study of modelling and analyzing an information retrieval system with mobile agents. In [24], an overview of the security issues of mobile agents was given and a state-of-the-art survey of mobile agent-based secure electronic transactions was presented. In [98], the authors identified two important properties (i.e., nonblocking and exactly-once) for fault-tolerant mobile agent execution and showed that fault-tolerant mobile agent execution could be modelled as a sequence of agreement problems.

Routing is a key feature of the Internet because it enables messages to pass from one computer to another and eventually reach the target machine [134]. Existing routing algorithms can be broadly classified into two classes: static or dynamic. In static routing, packets are sent to the destination following a predetermined path, without considering the current network state. Static routing are appropriate for small networks and some dedicated links. In a large distributed network with irregular topology, routing decisions can only be made on the basis of local and approximate information about the current and the future network states. Dynamic routing can discover the changes of network states, automatically adjust its routing tables, and inform other routers of the changes.

RIP (Routing Information Protocol) is a popular dynamic routing protocol for small private networks [111]. With RIP, routers periodically exchange entire routing tables. OSPF (Open Shortest Path First) is now the most important protocol on large networks that provide Internet service [111]. With OSPF, routers send routing information to all nodes by calculating the shortest path to each node based on a topology of the Internet constructed by each node. Each router sends that portion of the routing table that describes the state of its own links. Both RIP and OSPF choose the path with minimum cost (generally the shortest path) between the pair of nodes. In [27, 102], the authors studied the behavior of routers and announced that specific routers are the cause of bottlenecks in the Internet. This is because the path with minimum cost could congest, in spite of other paths, possibly expensive, but less congested. In [55], the authors provided necessary and sufficient conditions for deadlock-free unicast and multicast routing with

the path-based routing model in interconnection networks.

Real ants are capable of finding the shortest path from a food source to the nest based only on local information [11, 31, 43, 51]. Inspired from the research on ants, Caro and Dorigo [19, 21] firstly proposed mobile agent-based routing algorithm in 1996. In a mobile agent-based routing algorithm, a group of mobile agents build paths between pair of nodes, exploring the network concurrently and exchanging data to update routing tables. In [19, 21], the authors showed that this mobile agent-based routing algorithm is very encouraging with many experiments over real and artificial IP datagram networks by comparing it with both static and adaptive state-of the-art routing algorithms, such as RIP and OSPF. A number of experiments have been conducted to compare the performance between the mobile agent-based routing algorithm and other routing algorithms. The results showed both favorable performances and robustness to classic routing algorithms like RIP and OSPF.

Although the efficiency of applying the mobile agent technique has been demonstrated and reported in the literature, the mathematical modelling and analysis of mobile agents' behaviors are still in its infancy. In [16], Brewington et al. formulated a method of mobile agent planning, which is analogous to the travelling salesman problem [39] to decide the sequence of nodes to be visited by minimizing the total execution time until the desired information is found. In [64], several statistical analysis of mobile agents were proposed, including dwell time on nodes, average life-span, and the reports arrival process. Besides the execution time for a task, theoretical analysis on the number of agents can be found in [6–8, 118].

In [118], an ant-like routing model was studied and the number of mobile agents was estimated. In [99], a smaller upper bound of the number of mobile agents was provided based on the same model, and for the first time the probability of success was considered. Mobile agents in the model were assumed to be blind; they could only know information about the node they were in and know nothing about the surrounding nodes. This made the action of mobile agents for each step static and reduced the analytical difficulty. In the model proposed in this chapter, mobile agents know information not only about their host nodes, but also about the neighboring nodes of their host nodes. We analyze both the probability of success and the population distribution of mobile agents.

A mobile agent is an autonomous object that possesses the ability for migrating autonomously from node to node in a computer network. Usually, the main task of a mobile agent is determined by specified applications of users, which can range from E-shopping and distributed computation to real-time device control. In recent years, a number of research institutions and industrial entities have been engaged in the development of elaborating supporting systems for this technology [66, 141]. Different results have been published describing mobile agents perform with different performance when compared with the other technology. In [66], several merits for mobile agents are described, including network load and latency reduction, protocol encapsulation, adaption, heterogeneity, robustness and fault-tolerance. Successful examples using mobile agents can be found in [62, 67].

Network routing is a problem in network management. Ant routing is a recently proposed mobile agent based network routing algorithm for use in these environments [137, 138]. The continuing investigation and research of naturally occurring social systems [43] offer the prospect of creating artificial systems that are controlled by emergent behavior and promise to generate engineering solutions to distributed systems manage-

ment problems such as those in communication networks [19, 109].

Real ants are able to find the shortest path from a food source to the nest without using visual cues. Also, they can adapt to changes in the environment, for example finding a new shortest path once the old one is no longer feasible due to a new obstacle [11, 43]. In ant routing algorithm described in [31], artificial ants are agents which concurrently explore the network from node to node and exchange collected information when they meet each other. They irrespectively choose the node to move to using a probabilistic function which was proposed here to be a function of the connecting situation of each node. Artificial ants probabilistically prefer nodes that are connected immediately. Initially, a number of artificial ants are placed on randomly selected nodes. At each time step they move to new nodes and select useful information. When an ant has completed it's task, it will send a message back to the source host.

[118,119] analyzes the population growth in one host and the number of agents move out from one host to any of the neighboring hosts. The probability that the jumping hops exceed a given number is also estimated.

2.2 Preliminaries

In this section, we first introduce some notations and definitions used in this chapter, then describe the network model used in this chapter.

As we know, the topology of a network can be decided uniquely by its connectivity matrix. Assume that there are n nodes in the network, matrix $C = (c_1, c_2, \cdots, c_n) = (c_{ij})_{n\times n}$ is the connectivity matrix which describes the connectivity of the network, i.e., if there is a direct link between node i and node j, then $c_{ij} = c_{ji} = 1$; otherwise, $c_{ij} = c_{ji} = 0$. Let $d_j = \|c_j\|_1 = \sum_{i=1}^{n} |c_{ij}|$, $\sigma_1 = \max_{1\le j\le n} d_j$, $\sigma_n = \min_{1\le j\le n} d_j$. Clearly, $D = \mathrm{diag}(d_1, d_2, \cdots, d_n)$ is a diagonal matrix. It is easy to see that d_j is the number of neighboring nodes of the j-th node, i.e., the degree of the j-th node, and $\|C\|_1 = \max\limits_{1\le j\le n} \|c_j\|_1 = \sigma_1$.

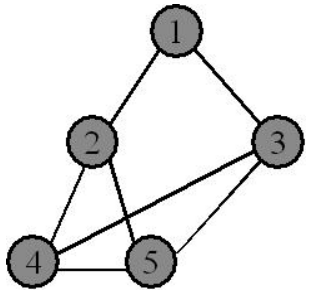


Figure 2.1: An example of a small network.

For the small network shown in Figure 2.1, we have $n = 5$, $\sigma_1 = 3$, and $\sigma_5 = 2$. Matrices C and D are as follows:

$$C = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

In our model, mobile agents are the medium for carrying routing information. The behavior of agents corresponds to the transport of routing information among nodes. For a large network, suppose that agents can be generated from every node in the network, and each node in the network provides to mobile agents an execution environment. Routing table on each node only details its neighboring nodes. A node which generates mobile agents is called the server of these agents. At any time, requests may be keyed in the network. Once a request for sending a packet is received from a server, the server will generate a number of mobile agents. Each agent carries the addresses of its server, its destination, the previous node it jumped from, and some control information for routings such as life-span limit and hop counter. All these data can be contained in several lines of Java code, thus the size of a mobile agent is very small, resulting in great reduction on network load and latency.These agents will then move out from the server and roam in the network. Once an agent reaches a node, it will search for the destination in the set of neighboring nodes of the host node. If the agent finds its destination, it will go back to the server along the path searched, update the routing tables on the nodes along the path, and submit its report about the searched path to the server. Otherwise, it will select a neighboring node and move on. To eliminate unnecessary searching in the network, a life-span limit is assigned to each agent. An agent will die if it cannot find its destination in its life-span limit. Moreover, if an agent cannot return to its server in two times the life-span limit (e.g., its return route is interrupted due to a link/node failure), the agent also will die.When a certain number of those agents have come back, the server selects the optimal path by certain criterion and send the packet to the destination along the new path. At the same time, the server updates its routing table by the information of the new path.

There are also some assumptions convenient for our analysis:

1. There are n nodes in the network, and each node has the same probability of $1/n$ to be the destination. The assumption of uniform probability of 1/n has been widely applied in the open literature [19, 21, 137].

2. The expected number of requests received from a server once is m. Once a request arrives, k agents are created and sent out into the network.

3. The life-span limit of agents is set to be d.

2.3 Ant-like Routing

In this section, we consider the problem of ant-like mobile agent based network routing. We analyze the probability of success (the probability that an agent can find the destination) and the population growth of mobile agents. First, We give an estimation on the probability of success, $P(d)$, than an agent can find the destination in d jumps as $P(d) \leq \frac{1}{n}\left(1-\frac{1}{n}\right)^d\left(\frac{\sigma_1-1}{\sigma_1}\right)^{d-1}$, n is the number of nodes in the network, and σ_1, σ_n are the largest and smallest degrees of nodes in the network respectively. The probability of success that k agents can find the destination in d jumps is estimated as $P^*(d) \leq 1-\left(\frac{n-1}{n+\sigma_1-1}\right)^k$, where k is the number of agents generated per request. Second, the distribution of mobile agents in the network is analyzed, $\overrightarrow{p}(t) = (I + A + \cdots + A^{t-1})km\overrightarrow{e}$ when $0 < t \leq d$ and

$\overrightarrow{p} = (I + A + \cdots + A^{d-1})km\overrightarrow{e}$ when $t > d$, where A is a matrix derived from the connectivity matrix. We further estimate that the number of agents running in the network is less than $\frac{n^2\sigma_1 km}{n+\sigma_1-1}$, and the population of mobile agents running in each host, $p_j(t)$, satisfies: $km + (1-\xi^{d-1})[1 - \frac{1}{n(1-\xi)}](D_j - 1)km \leq p_j(t) \leq km + (1-\zeta^{d-1})[1 - \frac{1}{n(1-\zeta)}](D_j - 1)km$, where $\xi = \|A\|_1 = \max_{1\leq j\leq n} \|a_j\|_1$, $\zeta = \min_{1\leq j\leq n} \|a_j\|_1$, a_j is the jth column of matrix A, and D_j is the degree of the jth host.

The rest of this section is organized as follows. Subsection 2.3.1 describes the network model used in this section and introduces the idea of ant routing. Subsection 2.3.2 presents some analysis results for mobile agents, including the probability of success and the population of agents. Subsection 2.3.4 concludes this section.

2.3.1 Execution Model

Based on the one presented in [118], the main idea of our algorithm as follows:

1. Assume that at any time t, the expected number of requests keyed in one host is m. Once a request arrives, k agents (the explorer agents) are created and sent out into the network.

2. The explorer agents traverse the network from the source to search for the destination. At each intermediate node, if the node is the destination of the agent, the agent will die and a new message agent is generated and send the message back to the source node. Otherwise, it will randomly select a neighbor node to move on.

3. In the source node, each message agent returned will compare its path explored with other explorer agents returned. If the cost of the corresponding traverse path is acceptable, then an allocator agent will thus be sent out immediately, and allocate network resources on the notes and links used in the path.

4. When the path is no longer required, a de-allocator agent traverses the path, and de-allocates the network resources used on the nodes and links.

5. To reduce unnecessary searching in the network, we further assume that if an agent cannot find its destination in d jumps, it will die.

2.3.2 Analysis

The Probability of Success

First, we assume that the length of each link between two nodes are the same, and the probability that an agent can find its destination at current host is $\frac{1}{n}$, then the searching time is determined by the number of jumping. Assume that the probability of jumping to any neighboring host or die is the same, then we have the following theorem on the probability that an agent can find the destination in d jumps as follows:

Theorem 1 *The probability, $P^*(d)$, that at least one agent among the k agents can find the destination in d jumps satisfies the following inequality:*

$$P^*(d) \leq 1 - \left(\frac{n-1}{n+\sigma_1-1}\right)^k \tag{2.1}$$

Proof Denote the sequence number of node that the agent entered at dth jump by J_d and the probability that an agent can find its destination at the d th jump by $P(d)$. The

probability that an agent can find its destination at the first jump is $P(1) = \frac{1}{n}$ and the probability it can't find the destination is $1 - \frac{1}{n}$. If the agent can't find its destination, the probability that it can jump out and search on is $(1 - \frac{1}{n}) \cdot \frac{D_{J_1}-1}{D_{J_1}}$, and the probability that it can find its destination at the second jump is $P(2) = \frac{1}{n}(1 - \frac{1}{n}) \cdot \frac{D_{J_1}-1}{D_{J_1}}$, and the probability it can't find its object at the second jump is $(1 - \frac{1}{n})^2 \cdot \frac{D_{J_1}-1}{D_{J_1}}$. If the agent can't find its destination at the second jump, the probability that it takes the third jump is $(1 - \frac{1}{n})^2 \cdot \frac{D_{J_1}-1}{D_{J_1}} \cdot \frac{D_{J_2}-1}{D_{J_2}}$, and $P(3) = \frac{1}{n}(1 - \frac{1}{n})^2 \cdot \frac{D_{J_1}-1}{D_{J_1}} \cdot \frac{D_{J_2}-1}{D_{J_2}}$. Similarly, the probability that an agent can find its destination host at the dth jump is

$$P(d) = \frac{1}{n}\left(1 - \frac{1}{n}\right)^{d-1} \prod_{i=1}^{d-1} \frac{D_{J_i} - 1}{D_{J_i}} \leq \frac{1}{n}\left(1 - \frac{1}{n}\right)^{d} \left(\frac{\sigma_1 - 1}{\sigma_1}\right)^{d-1}$$

So the probability, $P^*(d)$, that at least one agent among k agents can find the destination host in d jumps satisfies the following:

$$P^*(d) = \sum_{s=1}^{k} C_k^s \left[\sum_{t=1}^{d} P(t)\right]^s \left[1 - \sum_{t=1}^{d} P(t)\right]^{k-s} = 1 - \left[1 - \sum_{t=1}^{d} P(t)\right]^k$$

Due to

$$\sum_{t=1}^{d} P(t) \leq \sum_{t=1}^{d} \frac{1}{n}\left(1 - \frac{1}{n}\right)^{t-1} \left(\frac{\sigma_1 - 1}{\sigma_1}\right)^{t-1} \leq \frac{1}{n} \cdot \frac{1}{1 - \left(1 - \frac{1}{n}\right)\left(1 - \frac{1}{\sigma_1}\right)} = \frac{\sigma_1}{n + \sigma_1 - 1}$$

we have

$$P^*(d) \leq 1 - \left[1 - \frac{\sigma_1}{n + \sigma_1 - 1}\right]^k = 1 - \left(\frac{n - 1}{n + \sigma_1 - 1}\right)^k$$

Hence the theorem is proven. □

The probability that an agent can find its destination is decided by the topology of network and parameters k and d, which coincides with practice. From the theorem above, we can easily get that the probability that none of those k agents can find the destination is equal to $\left(\frac{n-1}{n+\sigma_1-1}\right)^k$, and the probability that all the k agents can find the destination is equal to $\left(\frac{\sigma_1}{n+\sigma_1-1}\right)^k$.

The Population of Agents

Assume that at time $t-1$, there are $p_i(t-1)$ agents in the ith host, these agents search for the destination locally, and the expected number of agents cannot find the destination is equal to $\left(1 - \frac{1}{n}\right) p_i(t-1)$, and the expected number of mobile agents move from the ith host to the jth host is equal to $\left(1 - \frac{1}{n}\right) \frac{1}{D_i} p_i(t-1)$. The total number of agents move to the jth host at time t is $\sum_{i \in NB(j)} \left(1 - \frac{1}{n}\right) \frac{1}{D_i} p_i(t-1)$, where $NB(j)$ denotes the host set make up of the neighboring hosts of the jth host. Consider about the new generated agents in the jth host at time t, we have the following equation:

$$p_j(t) = km + \sum_{i \in NB(j)} \left(1 - \frac{1}{n}\right) \frac{1}{D_i} p_i(t-1) \tag{2.2}$$

where m is the average number of requests initiated at time t at a host, and k is the number of agents generated per request. Let $\overrightarrow{p}(t) = (p_1(t), p_2(t), \cdots, p_n(t))^T$, and $\overrightarrow{e}$ be a vector that all the elements of it are 1, we can express the population distribution of mobile agents in vector-matrix format:

$$\overrightarrow{p}(t) = A\overrightarrow{p}(t-1) + km\overrightarrow{e} \tag{2.3}$$

where $A = (1 - \frac{1}{n})(C - I)D^{-1} = (a_1, a_2, \cdots, a_n)$ is a matrix, a_j is the jth column vector of A. Obviously, we have $\|a_j\|_1 = (1 - \frac{1}{n})\frac{D_j - 1}{D_j}$.

Theorem 2 *The distribution of mobile agents can be expressed as follows:*

$$\overrightarrow{p}(t) = \begin{cases} 0 & t = 0 \\ (I + A + A^2 + \cdots + A^{t-1})km\overrightarrow{e} & 0 < t \leq d \\ (I + A + A^2 + \cdots + A^{d-1})km\overrightarrow{e} & t > d \end{cases} \tag{2.4}$$

Proof First If the distribution of mobile agents initialled at time 0 is $\overrightarrow{p}(0)$, then after time t, the distribution of the survival among these agents is $A^d\overrightarrow{p}(0)$.
Second, by the recursive formula of the distribution of mobile agents, $\overrightarrow{p}(t) = km\overrightarrow{e} + A\overrightarrow{p}(t-1)$, and the assumption that if an agent cannot find the destination in d steps, it will die, we have
(1) If $t \leq d$,

$$\begin{aligned} \overrightarrow{p}(t) &= km\overrightarrow{e} + A\overrightarrow{p}(t-1) \\ &= km\overrightarrow{e} + A(km\overrightarrow{e} + A\overrightarrow{p}(t-2)) \\ &= (I + A)km\overrightarrow{e} + A^2\overrightarrow{p}(t-2) \\ &= \cdots \\ &= (I + A + \cdots + A^{t-1})\overrightarrow{e} + A^t\overrightarrow{p}(0) \end{aligned}$$

As we assumed that the initial population of mobile agents $\overrightarrow{p}(0) = 0$, the result for $t \leq d$ is proven.
(2) If $t > d$, then at time t, all the survival agents generated at time $t - d$ die, so the distribution of agents under this case can be illustrated as

$$\begin{aligned} \overrightarrow{p}(t) &= km\overrightarrow{e} + A\overrightarrow{p}(t-1) - A^d km\overrightarrow{e} \\ &= (I + A + \cdots + A^{t-d-1})km\overrightarrow{e} + A^{t-d}\overrightarrow{p}(d) - (I + A + \cdots + A^{t-d-1})A^d km\overrightarrow{e} \\ &= (I + A + \cdots + A^{t-d-1})km\overrightarrow{e} + A^{t-d}[(I + A + \cdots + A^{d-1})km\overrightarrow{e} + A^d\overrightarrow{p}(0)] \\ &\quad -(I + A + \cdots + A^{t-d-1})A^d km\overrightarrow{e} \\ &= (I + A + \cdots + A^{d-1})km\overrightarrow{e} \end{aligned}$$

Hence the theorem is proven. □

From the theorem above, we can easily see that the distribution of mobile agents will not exceed $(I + A + \cdots + A^{d-1})km\overrightarrow{e}$, that is, the number of mobile agents in our model has a upper limitation.

Since mobile agents are generated frequently and dispatched to the network, it is important to estimate the maximum number of mobile agents running in the network and in each host. When there are too many agents in the network, they will introduce too much computational overhead to host machines, which will eventually become very busy and indirectly block the network traffic.

Regarding the number of agents running in the network, we have the following theorem.

Theorem 3 *The number of agents running in the network can be estimated as follows:*

$$\sum_{j=1}^{n} p_j(t) \leq \frac{n^2 \sigma_1 km}{n + \sigma_1 - 1} \tag{2.5}$$

Proof By the definition of matrix 1−norm,we have

$$\begin{aligned} \sum_{j=1}^{n} P_j(t) &= \|\overrightarrow{p}(t)\|_1 \leq \|I + \cdots + \cdots + A^{d-1}\|_1 \cdot \|km\overrightarrow{e}\|_1 \\ &\leq nkm \cdot \sum_{i=0}^{d-1} (\|A\|_1)^i \leq \frac{nkm}{1 - \|A\|_1} \end{aligned}$$

Due to $\|A\|_1 = \left(1 - \frac{1}{n}\right) \frac{\sigma_1 - 1}{\sigma_1}$, it is easy to prove that

$$\|\overrightarrow{p}(t)\|_1 \leq \frac{nkm}{1 - \left(1 - \frac{1}{n}\right)\left(1 - \frac{1}{\sigma_1}\right)}$$

hence the theorem is proven. □

Denote that $\xi = \|A\|_1 = \max\limits_{1 \leq j \leq n} \|a_j\|_1 = \left(1 - \frac{1}{n}\right) \frac{\sigma_1 - 1}{\sigma_1}$, and $\zeta = \min\limits_{1 \leq j \leq n} \|a_j\|_1$ $= \left(1 - \frac{1}{n}\right) \frac{\sigma_n - 1}{\sigma_n}$. For the number of agents running in a host, we have the following theorem.

Theorem 4 *The number of agents running in the jth host can be estimated as follows:*

$$\begin{aligned} km + (1 - \xi^{t-1}) \quad [1- \quad & \frac{1}{n(1-\xi)}](D_j - 1)km \\ \leq \quad & p_j(t) \leq km + (1 - \zeta^{t-1})[1 - \frac{1}{n(1-\zeta)}](D_j - 1)km \end{aligned}$$

where $0 < t \leq d$*, and when* $t \geq d$*,*

$$\begin{aligned} km + (1 - \xi^{d-1}) \quad [1- \quad & \frac{1}{n(1-\xi)}](D_j - 1)km \\ \leq \quad & p_j(t) \leq km + (1 - \zeta^{d-1})[1 - \frac{1}{n(1-\zeta)}](D_j - 1)km \end{aligned}$$

Proof Assume that $NB(j)$ is a set constituted by all the neighboring hosts of the jth host. By recursion, we have

$$\begin{aligned}
p_j(0) &= 0 \\
p_j(1) &= km \\
p_j(2) &= km + \sum_{i \in NB(j)} \frac{(1-\frac{1}{n})p_i(1)}{D_i} \\
&\leq km + \sum_{i \in NB(j)} [km - \frac{1}{n}km - \zeta km] \\
&= D_j km - \frac{1}{n}[D_j - 1]km - \zeta[D_j - 1]km \\
p_j(3) &= km + \sum_{i \in NB(j)} \frac{(1-\frac{1}{n})p_i(2)}{D_i} \\
&\leq km + \sum_{i \in NB(j)} [km(1-\frac{1}{n}) - \frac{1}{n}\zeta km - \zeta^2 km] \\
&= D_j km - \frac{1}{n}[D_j - 1]km - \frac{1}{n}\zeta[D_j - 1]km - \zeta^2[D_j - 1]km \\
&= D_j km - \frac{1}{n}[D_j - 1]km(1+\zeta) - \zeta^2[D_j - 1]km \\
\cdots &\quad \cdots \quad \cdots \\
p_j(t) &\leq D_j km - \frac{1}{n}[D_j - 1]km(\sum_{s=0}^{t-2}\zeta^s) - \zeta^{t-1}[D_j - 1]km \\
&= km + (1-\zeta^{t-1})[1 - \frac{1}{n(1-\zeta)}](D_j - 1)km
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
p_j(t) &\geq D_j km - \frac{1}{n}[D_j - 1]km[\sum_{s=0}^{t-2}\xi^s] - \xi^{t-1}[D_j - 1]km \\
&= km + (1-\xi^{t-1})[1 - \frac{1}{n(1-\xi)}](D_j - 1)km
\end{aligned}$$

When $t \geq d$, since $\overrightarrow{p}(t) = (I + A + A^2 + \cdots + A^{d-1})km\overrightarrow{e}$, the bounds are fixed and irrelevant with t. Hence, the theorem is proven. □

It is easy to see that $p_j(t)$ is related with the values of k, m, t and the topology of the network which is coherent with our intuition. When t is big enough, both ζ and ξ approximate to 0, then we have the following corollary:

Corollary 1 *If t is big enough, then we have*

$$p_j(t) \approx km + (1 - \frac{1}{n})[D_j - 1]km \tag{2.6}$$

Furthermore, since both $(1-\zeta^{d-1})$ and $[1 - \frac{1}{n(1-\zeta)}]$ are less than 1, we have $p_j(t) \leq D_j km$, which is a result given in [118].

2.3.3 Experimental Results

In this section, we present some experimental results to explain the analytical results given in Section 2.3.2. The following cases of network topologies are included in the experiments:

1. Case A: σ_1 large, σ_n large;
2. Case B: σ_1 large, σ_n middle;
3. Case C: σ_1 large, σ_n small;
4. Case D: σ_1 middle, σ_n middle;
5. Case E: σ_1 middle, σ_n small;
6. Case F: σ_1 small, σ_n small.

Figure 2.2 gives some simple examples for these cases.

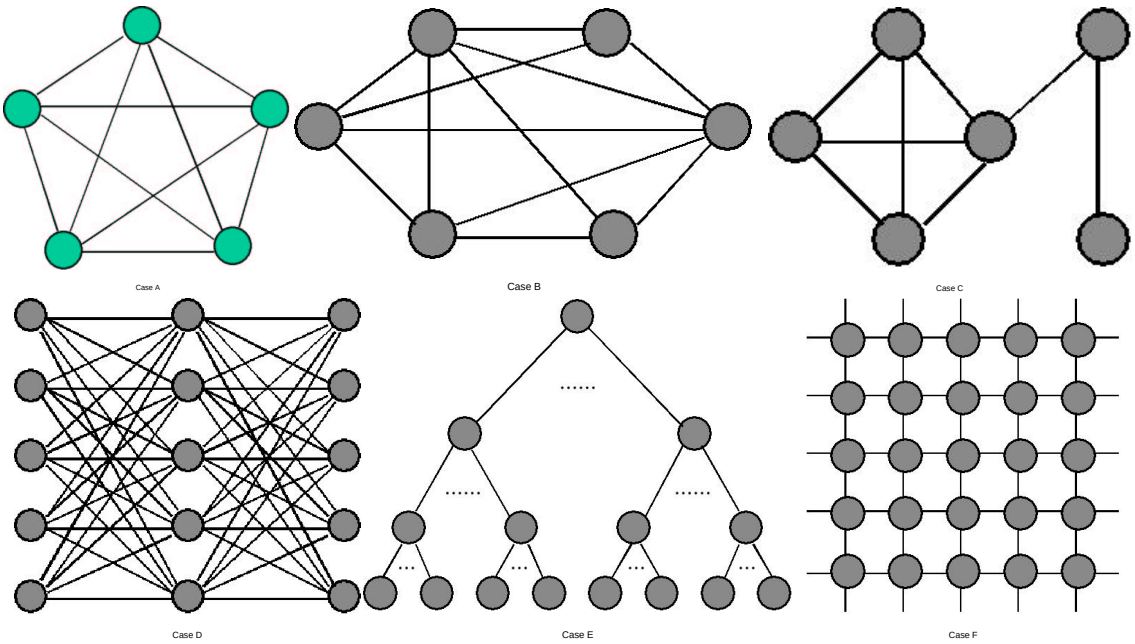


Figure 2.2: Simple Examples for Each Case

The Searching Time Per Quest

First, we present the experimental results on the upper bound of the survival probability (UBSP) and the lower bound of the successful probability (LBSP) in Table 2.1, where the survival probability is defined as the probability that an agent can survive after d jumps and the successful probability is defines as the probability that the destination can be found in d jumps. Here, k is set to be 20. From Table 2.1, we can see that when d increases, USBP will decrease and LBSP will increase. When the number of n increases, USBP will increase and LBSP will decrease. Furthermore, if UBSP is very large, LBUP will increase rapidly with the growth of d, and if USBP is very small, LBUP will increase

Table 2.1: The Experiment Results on UBSP and LBSP

n	case	20		50		100	
		UBSP	LBSP	UBSP	LBSP	UBSP	LBSP
200	A	0.8266	0.8519	0.6119	0.9878	0.3707	0.9995
	B	0.7511	0.7700	0.4780	0.9076	0.2251	0.9359
	C	0.8258	0.1810	0.6104	0.1813	0.3688	0.1813
	D	0.3756	0.7273	0.0801	0.8449	0.0061	0.8642
	E	0.4774	0.3276	0.1486	0.3284	0.0212	0.3284
	F	0.0131	0.2586	1.3955e-05	0.2586	1.5502e-10	0.2586
500	A	0.9267	0.5441	0.8219	0.8510	0.6727	0.9729
	B	0.7953	0.4455	0.5540	0.6205	0.3033	0.6784
	C	0.8744	0.0769	0.7074	0.0769	0.4969	0.0769
	D	0.3978	0.4067	0.0928	0.5315	0.0082	0.5592
	E	0.5055	0.1472	0.1722	0.1476	0.0286	0.1476
	F	0.0139	0.1129	1.6174e-05	0.1130	2.0885e-10	0.1130
1000	A	0.9267	0.3273	0.9066	0.6232	0.8203	0.8507
	B	0.8106	0.2556	0.5819	0.3856	0.3349	0.4364
	C	0.8912	0.0392	0.7430	0.0392	0.5486	0.0392
	D	0.4054	0.2300	0.0974	0.3170	0.0091	0.3385
	E	0.5152	0.0766	0.1808	0.0768	0.0316	0.0768
	F	0.0140	0.0582	1.6987e-05	0.0582	2.3062e-10	0.0582
10000	A	0.9962	0.0.0392	0.0.9902	0.0948	0.9804	0.1805
	B	0.8246	0.0291	0.6081	0.0478	0.3661	0.0563
	C	0.9066	0.0040	0.7765	0.0040	0.5998	0.0040
	D	0.4124	0.0258	0.1018	0.0376	0.0099	0.0408
	E	0.5241	0.0080	0.1890	0.0080	0.0345	0.0080
	F	0.0144	0.0060	1.7753e-05	0.0060	2.5212e-10	0.0060

slowly. If LBUP increases too slowly, the process of searching should be shut down. All of these are coherent with Theorem 1 and its conclusions.

Table 2.2 shows the experimental results of the lower bound on k when d approximates to infinity. We can see that the larger the degrees of nodes (the number of neighboring hosts) are, the fewer the agents are needed, which is also coincided with our analytical results.

		The desired probability of success								
n	*case*	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
200	A	1	1	1	1	1	2	2	3	4
	B	1	1	1	2	2	3	4	5	7
	C	1	1	1	1	2	2	3	3	5
	D	2	3	4	5	7	9	12	16	22
	E	1	2	3	4	5	7	9	12	17
	F	5	9	15	21	28	37	49	65	93
500	A	1	1	1	1	1	2	2	3	4
	B	1	2	2	3	4	6	7	9	13
	C	1	1	2	2	3	3	4	5	8
	D	3	6	9	12	17	22	28	38	54
	E	2	4	7	9	12	16	21	28	40
	F	11	23	36	52	70	92	121	162	231
1000	A	1	1	1	1	1	2	2	3	4
	B	2	3	4	6	8	10	13	17	25
	C	1	2	3	3	4	6	7	9	13
	D	5	11	17	24	32	43	56	74	106
	E	4	8	13	18	24	31	41	55	78
	F	22	45	72	103	139	184	242	323	462
10000	A	1	1	1	1	1	2	2	3	4
	B	11	23	36	52	70	93	121	162	232
	C	6	12	19	27	35	47	62	83	118
	D	48	102	163	233	316	417	548	733	1048
	E	36	75	120	171	232	306	402	538	769
	F	211	447	714	1022	1387	1833	2409	3220	4606

Table 2.2: The lower bound on k when d approximates to infinity

The Population of Agents

Second, we present some experiment results on the population of mobile agents by comparing our results with existing results [118]. We use UBA to denote our results, i.e., the upper bound on the population of agents running in a host, while ER to denote the existing results. Figures 2.3, 2.4, 2.5, 2.6, 2.7, and 2.8 show the results on the population of agents running in a host. From the experimental results we can obviously see that our results can outperform the existing results greatly when the number of neighboring hosts is not very small.

2.3.4 Concluding Remarks

In this section, we addressed the problem of ant-like mobile agents-based network routing and management by deploying mobile agents. We analyzed the growth of the population of mobile agents under a modified ant routing algorithm. We obtained the fol-

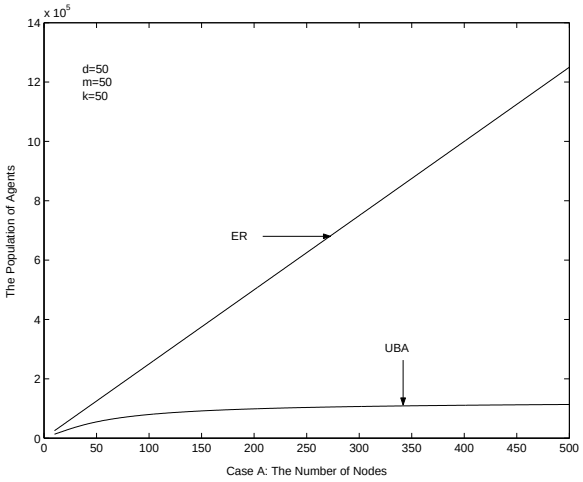


Figure 2.3: Results for Case A

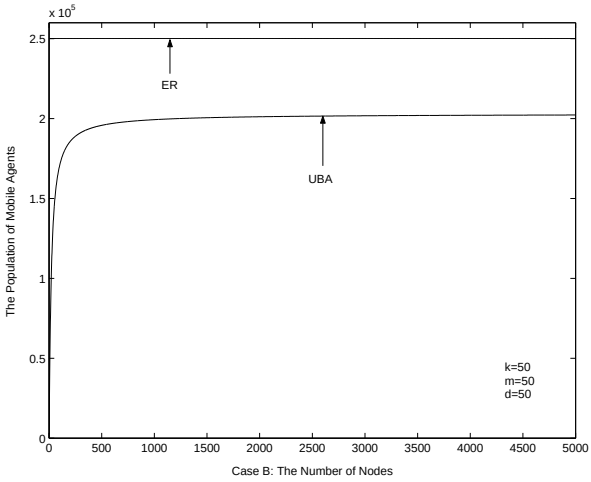


Figure 2.4: Results for Case B

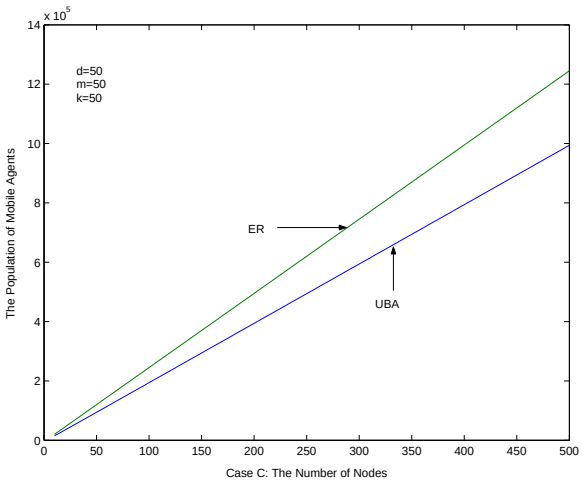


Figure 2.5: Results for Case C

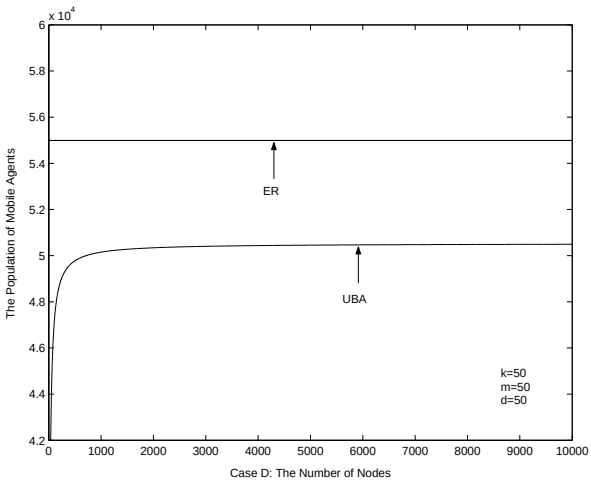


Figure 2.6: Results for Case D

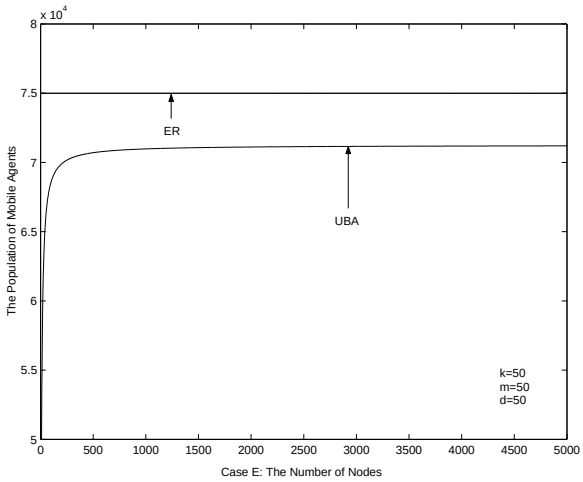


Figure 2.7: Results for Case E

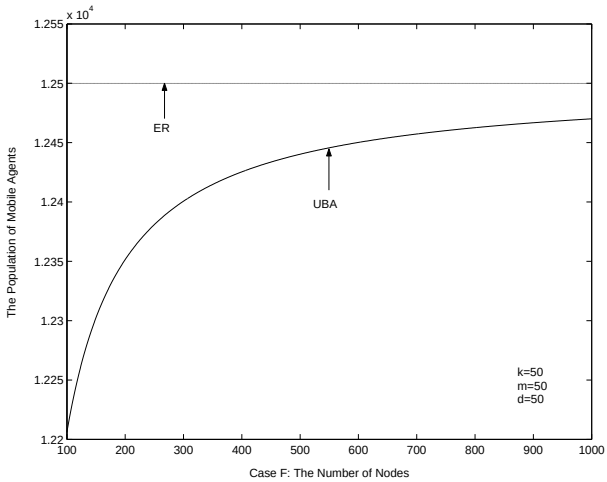


Figure 2.8: Results for Case F

lowing analytical results: (1) The probability of success, $P(d)$, that an agent can find the destination in d jumps is less than $\frac{1}{n}\left[\left(1-\frac{1}{n}\right)\left(1-\frac{1}{\sigma_1}\right)\right]^{d-1}$, where n is the number of hosts in the network, σ_1 is the maximum degree of hosts in the network, and d is the number of jumping hops. (2) The probability of success that k agents can find the destination in d jumps is estimated as $P^*(d) \leq 1-\left(\frac{n-1}{n+\sigma_1-1}\right)^k$, where k is the number of agents generated per request. (3) The population distribution can be express as $\overrightarrow{p}(t) = km\overrightarrow{e} + A\overrightarrow{p}(t-1)$ or $\overrightarrow{p}(t) = (I+A+\cdots+A^{t-1})km\overrightarrow{e}$ when $0 < t \leq d$ and $\overrightarrow{p}(t) = (I+A+\cdots+A^{d-1})km\overrightarrow{e}$, where m is the average number of request keyed in one host once, $\overrightarrow{e}$ is a vector that all its elements are 1, and A is a matrix can be derived from the connectivity matrix. (4) The total number of agents running in the network is less than $\frac{n\sigma_1}{n+\sigma_1-1}\cdot nkm$. (5) The population of mobile agents running in each host, $p_j(t)$, satisfies $km+(1-\xi^{d-1})[1-\frac{1}{n(1-\xi)}](D_j-1)km \leq p_j(t) \leq km+(1-\zeta^{d-1})[1-\frac{1}{n(1-\zeta)}](D_j-1)km$, where $\xi = \max\|a_j\|_1$, $\zeta = \min\|a_j\|_1$, a_j is the jth column of matrix A, and D_j is the degree of the jth host.

2.4 Two Extended Routing Algorithms

In this section, considering the fact that the neighborhood information of a host node is readily available (e.g. it is built in the routing table in TCP/IP), we propose a general agent-based routing model, in which a mobile agent knows information not only about its host node, but also about the neighboring nodes of its host node. Since the connectivity of different nodes may change dynamically over time, the agents have to dynamically adapt themselves to the environment, which increases the difficulty for theoretical analysis. We further classify the model into two cases, namely settleable and nonsettleable (to the host node):

- *Settleable* case: If the host node is not the agent's destination, the agent can either stay in the host node or move to a neighboring node.
- *Nonsettleable* case: An agent will not die unless it finds its destination or it is out of its life-span limit. If an agent cannot find its destination on the host node, it must to jump out from the host node and go on searching.

, and analyze the probability of success and the population distribution of mobile agents for each case. Both our theoretical and experimental results show that we can control the probability of success and the number of mobile agents by tuning the number of agents generated per request and the number of jumps each mobile agent can make. The main contributions of this section are summarized as follows:

- A general and more practical agent-based routing model is described and is further classified into settleable and nonsettleable cases.
- The probability of success is analyzed, which serves as an important measure for monitoring network performance.
- An analysis on population growth of mobile agents is presented, providing a useful tool to reduce computational resource consumption by adjusting the number of agents to be generated at individual nodes.

- Extensive experiments validating our analysis have been performed and compared with previous work.

The rest of this section is organized as follows: Subsection 1 and Subsection 2 present mathematical analysis on the probability of success and population distribution for both cases. Subsection 3 describes our experimental results, and subsection 4 concludes our section.

2.4.1 Probability of Success

Analysis of the probability of success is important because the probability of success directly affects the searching process, and influences the network performance as a result. However, it has not been sufficiently taken into account in existing work. In this section, the probabilities of success for both single agent and multiple agents are analyzed. Our results show that the probability of success is affected by the connectivity of the network, the life-span limit, and the number of mobile agents.

In a mobile agent-based network routing model, a mobile agent will visit a sequence of nodes. The sequence of nodes between the server and the destination is called the itinerary of the mobile agent. A static itinerary is entirely defined at the server and does not change during the agent's travels, whereas a dynamic itinerary is subject to modification by the agent itself. In this context, the mobile agents traverse the network with a dynamic itinerary. Suppose that the w-th node in the network is the destination. The sequence of nodes in the itinerary of an agent is denoted by $J^{(0)}$, $J^{(1)}$, $\cdots$, w or $J^{(d)}$, where $J^{(0)}$ denotes the server node and $J^{(i)}$ is the i-th node visited by the agent.

Lemma 1 *Let $i \rightarrow j$ indicates the event that an agent jumps from the i-th node to the j-th node (in one jump), the probability of success that an agent can find its destination at the t-th jump, denoted by $p(t)$, satisfies*

$$p(t) = \sum_{l \in NB(J^{(t-1)})} Pr\left\{J^{(t-1)} \rightarrow l\right\} \frac{d_l - 1}{n} \left[1 - \sum_{k=0}^{t-1} p(k)\right] \tag{2.7}$$

where $NB(j)$ is the set of node j's neighboring nodes and node j itself, $p(0) = (d_{J^{(0)}}+1)/n$, and $t \leq d$.

Proof After being generated by the server, $J^{(0)}$, mobile agents move out from the server and search for their destination in the network. The probability that an agent can find its destination at birth, $p(0)$, i.e., the probability that the destination is in $NB(J^{(0)})$, equals to $(d_{J^{(0)}} + 1)/n$. If the agent cannot find its destination in $NB(J^{(0)})$,it will jump to one of the server's neighboring node, $J^{(1)}$, and continue searching.Since both $J^{(0)}$ and $J^{(1)}$ are not the destination, the probability that the agent can find its destination at the first jump equals to $p(1) = \sum_{i \in NB(J^{(0)})} \frac{d_i - 1}{n} Pr\{J^{(0)} \rightarrow i\}[1 - p(0)]$. If the agent cannot find its destination in $NB(J^{(1)})$, it will jump to a neighboring node, $J^{(2)}$, and continue searching.The probability that an agent can find its destination at the second jump is $p(2) = \sum_{j \in NB(J^{(1)})} \frac{d_j - 1}{n} Pr\{J^{(1)} \rightarrow j\}[1 - p(0) - p(1)]$. By recursion, it is easy to prove that the probability, $p(t)$, that an agent can find its destination at the t-th jump for $t > 1$ satisfies:

$$p(t) = \sum_{l \in NB(J^{(t-1)})} \frac{d_l - 1}{n} Pr\{J^{(t-1)} \rightarrow l\} \left[1 - \sum_{k=0}^{t-1} p(k)\right]$$

□

It can be seen that the probability $p(t)$ is relevant to the connectivity of the network and the number of jumps. From Lemma 1, it is easy to estimate the probability of success that an agent can find its destination in d jumps, denoted by $P(d)$, by the relationship

$$\begin{aligned} P(d) &= p(0) + \sum_{t=1}^{d} p(t) \\ &= \frac{d_{J^{(0)}}+1}{n} + \sum_{t=1}^{d} \sum_{l \in NB(J^{(t-1)})} Pr\{J^{(t-1)} \to l\} \\ &\quad \cdot \frac{d_l - 1}{n} \left[1 - \sum_{k=0}^{t-1} p(k)\right] \end{aligned} \tag{2.8}$$

From Lemma 1, it can be seen that $p(t)$ is a decreasing function on d. Therefore, based on Eq. (2.8), the increase of $P(d)$ is slower when d becomes larger.Obviously, if the increase of $P(d)$ is too small, it is not necessary to go on searching. Thus, the life-span limit of mobile agents, d, can be defined by a given threshold $\epsilon \geq 0$ such that $P(d) - P(d-1) \leq \epsilon$. Based on Eq. (2.7) and (2.8), we have the following theorem:

Theorem 5 *The probability of success, P_s, that at least s agents from k agents can find the destination in d jumps satisfies the following inequality:*

$$\frac{\triangle^s}{\sqrt{2\pi s}} < P_s < \frac{\triangle^s}{\sqrt{2\pi s}} \cdot \frac{1}{1 - \lambda/(s+1)} \tag{2.9}$$

where $\lambda = kP(d)$ and $\triangle = \frac{\lambda}{s} e^{1-\lambda/s}$.

Proof As we know, this probability satisfies binomial distribution, so we have

$$P_s = \sum_{i=s}^{k} C_k^s P(d)^s [1 - P(d)]^{k-s}$$

Since it can be approximated by a Poisson distribution, we have

$$P_s \approx \sum_{i=s}^{k} \frac{\lambda^s e^{-\lambda}}{s!}$$

where $\lambda = kP(d)$. Therefore, we have

$$\begin{aligned} e^{-\lambda}\frac{\lambda^s}{s!} &< \sum_{i=s}^{k} \frac{\lambda^s e^{-\lambda}}{s!} \\ &= e^{-\lambda}\frac{\lambda^s}{s!} \cdot \left[1 + \frac{\lambda}{s+1} + \frac{\lambda^2}{(s+1)(s+2)}\right] \\ &\quad + \cdots + e^{-\lambda}\frac{\lambda^s}{s!} \cdot \left[\frac{\lambda^{k-s}}{(s+1)\cdots k}\right] \\ &\leq e^{-\lambda}\frac{\lambda^s}{s!} \cdot \left[1 + \frac{\lambda}{s+1} + \left(\frac{\lambda}{s+1}\right)^2\right] \\ &\quad + \cdots + e^{-\lambda}\frac{\lambda^s}{s!} \cdot \left[\left(\frac{\lambda}{s+1}\right)^{k-s}\right] \\ &\leq e^{-\lambda}\frac{\lambda^s}{s!} \cdot \frac{1}{1 - \lambda/(s+1)} \end{aligned}$$

Applying Stirling's formula

$$n! \approx \sqrt{2\pi n} n^n e^{-n}, \text{for large } n.$$

we have

$$\frac{\triangle^s}{\sqrt{2\pi s}} < P_s < \frac{\triangle^s}{\sqrt{2\pi s}} \frac{1}{1-\lambda/(s+1)}$$

□

In particular, P_1, the probability that at least one agent among k agents can find the destination, is $1-[1-P(d)]^k$.

Settleable Case

In the settleable case, both the host node and its neighboring nodes have the same possibility to be selected since each direction is equally likely to connect with the destination. Thus, the probability $Pr\{i \to j\}$ equals to $1/(d_i+1)$ where $j \in NB(i)$ and d_i is the number of neighboring nodes of node i. Therefore, the probability $p(t)$ in Lemma 1 satisfies

$$p(t) = \sum_{l \in NB(J^{(t-1)})} \frac{d_l - 1}{n} \cdot \frac{1}{d_{J^{(t-1)}}+1} \cdot \left[1 - \sum_{k=0}^{t-1} p(k)\right] \tag{2.10}$$

where $p(0) = (d_{J^{(0)}}+1)/n$ and $0 < t \leq d$. In particular, when the average connectivity of the network , denoted by θ, is available, we can further estimate $p(t)$ based on Eq. (2.10) as follows:

$$\begin{aligned} p(t) &= \sum_{l \in NB(J^{(t-1)})} \frac{\theta - 1}{n(\theta+1)} \cdot \left[1 - \sum_{k=0}^{t-1} p(k)\right] \\ &= \frac{\theta(\theta-1)}{n(\theta+1)} \cdot \left[1 - \sum_{k=0}^{t-1} p(k)\right] \end{aligned} \tag{2.11}$$

$$p(t-1) = \frac{\theta(\theta-1)}{n(\theta+1)} \cdot \left[1 - \sum_{k=0}^{t-2} p(k)\right] \tag{2.12}$$

where $\theta = E[d_i]$. Denote $\theta(\theta-1)/[n(\theta+1)]$ by a, then we have

$$p(t) - p(t-1) = a \cdot [-p(t-1)] \tag{2.13}$$

Therefore

$$\begin{aligned} p(t) &= (1-a)p(t-1) = (1-a)^{t-1}p(1) \\ &= a(1-a)^{t-1}\left(1 - \frac{\theta+1}{n}\right) \end{aligned} \tag{2.14}$$

since $p(1) = a[1 - p(0)]$ and $p(0) = (\theta + 1)/n$. Furthermore, the probability of success, $P(d)$, that an agent can find its destination in d jumps satisfies

$$\begin{aligned} P(d) &= \sum_{t=0}^{d} p(t) = p(0) + \sum_{t=1}^{d} p(t) \\ &= \frac{\theta+1}{n} + \sum_{t=1}^{d} a(1-a)^{t-1}\left(1 - \frac{\theta+1}{n}\right) \\ &= \frac{\theta+1}{n} + a\left(1 - \frac{\theta+1}{n}\right)\frac{1-(1-a)^d}{a} \\ &= 1 - \left(1 - \frac{\theta+1}{n}\right)(1-a)^d \end{aligned} \tag{2.15}$$

Nonsettleable Case

In the nonsettleable case, the probability $Pr\{J^{(t)} \rightarrow l\}$ equals to $1/d_{J^{(t)}}$ since all neighboring nodes have the same possibility to be selected. Substituting this value in Eq. (2.7), the probability of success that an agent can find its destination at the t-th jump can be estimated as follows:

$$p(t) = \sum_{l \in NB(J^{(t-1)})} \frac{d_l - 1}{n \cdot d_{J^{(t-1)}}} \left[1 - \sum_{k=0}^{t-1} p(k)\right] \tag{2.16}$$

where $p(0) = (d_{J^{(0)}} + 1)/n$ and $0 < t \leq d$. Similar to the analysis for the settleable case, when the average connectivity θ is available, the probability can be estimated as follows

$$p(t) = b(1-b)^{t-1}\left(1 - \frac{\theta+1}{n}\right) \tag{2.17}$$

where $b = (\theta - 1)/n$. The probability of success, $P(d)$, that an agent can find its destination in d jumps satisfies

$$P(d) = 1 - (1-b)^d\left(1 - \frac{\theta+1}{n}\right) \tag{2.18}$$

2.4.2 Population Distribution

As we mentioned in sections 2.2 and 2.3,mobile agents are frequently generated and dispatched to the network. If the number of agents generated per request is small, there is no guarantee that the destination will be found quickly. On the other hand, if there are too many agents running in the network, they will introduce too much computational overhead; the host nodes will eventually become very busy and indirectly block the network traffic. Therefore, analysis of the number of mobile agents running in the network and on each node is another important task. Since mobile agents will search for their destinations node by node in the network, the agents in one node can be divided into two parts, as shown in Figure 2.9.

We claim that the population distribution of mobile agents running in the network at time t can be expressed as follows:

$$\begin{aligned} \overrightarrow{p}(t) = k\overrightarrow{r}(t-1) + kA\overrightarrow{r}(t-2) \\ + B\overrightarrow{p}(t-1) - B^d k\overrightarrow{r}(t-d-1) \end{aligned} \tag{2.19}$$

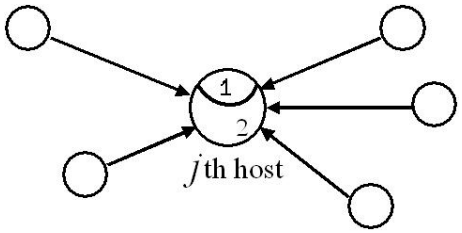


Figure 2.9: Mobile agents in one node can be divided into two parts: Part 1 – the set of mobile agents generated by the j-th node, Part 2 – the set of mobile agents coming from neighboring nodes.

where

$$\overrightarrow{p}(t) = \begin{bmatrix} p_1(t) \\ p_2(t) \\ \cdots \\ p_n(t) \end{bmatrix}, \quad \overrightarrow{r}(t) = \begin{bmatrix} r_1(t) \\ r_2(t) \\ \cdots \\ r_n(t) \end{bmatrix},$$

$p_j(t)$ is the number of mobile agents running on the j-th node, while $r_j(t)$ indicates the number of requests received by the j-th node at time t and $r_j(t) = 0$ when $t < 0$. k is the number of mobile agents generated per request, d is the life-span limit of mobile agents. $A = (a_{ji})$ and $B = (b_{ji})$ are $n \times n$ coefficient matrices with elements

$$a_{ji} = \begin{cases} Pr^{(1)}\{i \to j\} \\ \quad -Pr^{(2)}\{i \to j\} \quad \text{if } j \in NB(i) \\ \quad 0 \qquad\qquad \text{otherwise} \end{cases} \tag{2.20}$$

and

$$b_{ji} = \begin{cases} Pr^{(2)}\{i \to j\} \quad \text{if } j \in NB(i) \\ \quad 0 \qquad\qquad \text{otherwise} \end{cases} \tag{2.21}$$

$Pr^{(l)}\{i \to j\}$ (l=1,2) is the probability that a mobile agent in the l-th part at node i will select the j-th node to move when $j \in NB(i)$.

From Eq. (2.19), we can see that the population distribution of mobile agents can be modelled as a stochastic process. In the following, we explain the validity of Eq. (2.19) in detail. It is easy to see that the number of mobile agents newly generated at time t (part 1 in Figure 2.9) equals to $k \cdot r_j(t-1)$. Denote the number of agents in node j at time t that come from the i-th node by $p_{ji}(t)$, then the number of mobile agents in part 2 equals to $\sum_{i \in NB(j)} p_{ji}(t)$. Therefore, the total number of mobile agents running on the j-th node at time t, denoted by $p_j(t)$, satisfies

$$p_j(t) = k \cdot r_j(t-1) + \sum_{i \in NB(j)} p_{ji}(t) \tag{2.22}$$

The analysis of $p_{ji}(t)$ is much more complex. To simplify the analysis, we first consider the situation that mobile agents have an infinite life-span. Regarding to the population distribution of mobile agents with infinite life-span in the network, we have the following lemma.

Lemma 2 *The number of mobile agents with infinite life-span running on the j-th node with infinite life-span satisfies*

$$\overrightarrow{p}(t) = k\overrightarrow{r}(t-1) + kA\overrightarrow{r}(t-2) + B\overrightarrow{p}(t-1) \tag{2.23}$$

Proof It is no surprise that each neighboring node of the i-th node has the same possibility to be selected by mobile agents in the i-th node since each direction is equally likely to link with the destination. Thus, the average number of mobile agents that traverse from the i-th node to the j-th node at time t, denoted by $p_{ji}(t)$, equals to $\sum_{l=1}^{2} p_i^{(l)}(t-1)Pr^{(l)}\{i \to j\}$, where $p_i^{(l)}(t-1)$ $(l = 1, 2)$ is the number of mobile agents belong to the l-th part on the i-th node at time $t-1$(see Figure 2.9). Thus, the number of mobile agents on the j-th node with infinite life-span satisfies

$$p_j(t) = kr_j(t-1) + \sum_{i \in NB(j)} \sum_{l=1}^{2} p_i^{(l)}(t-1)Pr^{(l)}\{i \to j\}$$

Since $p_i^{(1)}(t-1)$ equals to $kr_i(t-2)$, and $p_i^{(2)}(t-1)$ equals to $p_i(t-1) - kr_i(t-2)$, we have

$$\begin{aligned} p_j(t) &= kr_j(t-1) + \sum_{i \in NB(j)} \big[kr_i(t-2)Pr^{(1)}\{i \to j\} \\ &\quad + (p_i(t-1) - kr_i(t-2))\, Pr^{(2)}\{i \to j\}\big] \\ &= kr_j(t-1) + \sum_{i \in NB(j)} \big[Pr^{(2)}\{i \to j\}p_i(t-1) \\ &\quad + \big(Pr^{(1)}\{i \to j\} - Pr^{(2)}\{i \to j\}\big)\, kr_i(t-2)\big] \end{aligned}$$

Let $[\cdot]_j$ denotes the j-th entry of a vector. From Eq. (2.20) and (2.21), we have

$$\begin{aligned} &\sum_{i \in NB(j)} Pr^2\{i \to j\}p_i(t-1) \\ &\qquad = \sum_{i} Pr^2\{i \to j\}p_i(t-1) \\ &\qquad = \sum_{i} b_{ji} \cdot p_i(t-1) = [B\overrightarrow{p}(t-1)]_j \end{aligned}$$

and

$$\begin{aligned} &\sum_{i \in NB(j)} \big(Pr^{(1)}\{i \to j\} - Pr^{(2)}\{i \to j\}\big)\, kr_i(t-2) \\ &\quad = \sum_{i} \big(Pr^{(1)}\{i \to j\} - Pr^{(2)}\{i \to j\}\big)\, kr_i(t-2) \\ &\quad = \sum_{i} a_{ji} \cdot r_i(t-2) = [A\overrightarrow{r}(t-2)]_j \end{aligned}$$

Therefore, we have

$$\overrightarrow{p}(t) = k\overrightarrow{r}(t-1) + kA\overrightarrow{r}(t-2) + B\overrightarrow{p}(t-1)$$

□

Lemma 3 *Suppose that the distribution of mobile agents generated at $t = 0$ in the network is $\overrightarrow{p}(0)$, then the population distribution of these agents at time t is $B^t\overrightarrow{p}(0)$.*

Proof Since we only consider the distribution of mobile agents that generated at $t = 0$ and do not pay attention to other agents, $\overrightarrow{r}(t)$ in Eq. (2.23) equals to 0. Therefore,

$$\overrightarrow{p}(t) = B \cdot \overrightarrow{p}(t-1) = B^t\overrightarrow{p}(0)$$

□

From Lemma 3, it is straightforward that at time d, the distribution of mobile agents that are generated at time $t = 0$ is $B^d\overrightarrow{p}(0)$. Since these agents will die at time d, they should be removed from the distribution in Eq. (2.23). Thus, based on Lemma 2 and Lemma 3, the population distribution of mobile agents with life-span limit at time t can be expressed as Eq. (2.19).

In particular, take expectation on both side of Eq. (2.19) and denote the average number of requests received from a node at any time by r, then when $t \leq d$, the population distribution of mobile agents satisfies

$$\begin{aligned}\overrightarrow{p}(t) &= kr(I+A)\overrightarrow{e} + B\overrightarrow{p}(t-1) \\ &= \sum_{i=0}^{t-1} B^i kr(I+A)\overrightarrow{e} + B^t\overrightarrow{p}(0)\end{aligned} \tag{2.24}$$

where $\overrightarrow{e} = (1, \cdots, 1)^T$. Here we still use the notation $\overrightarrow{p}(t)$ as $E[\overrightarrow{p}(t)]$. When $t > d$, the population distribution of mobile agents satisfies

$$\begin{aligned}\overrightarrow{p}(t) &= (I+A)kr\overrightarrow{e} + B\overrightarrow{p}(t-1) - B^d kr\overrightarrow{e} \\ &= \sum_{i=0}^{t-d-1} B^i kr(I+A)\overrightarrow{e} - \sum_{j=0}^{t-d-1} B^j krB^d\overrightarrow{e} \\ &\quad + B^{t-d}\overrightarrow{p}(d) \\ &= \sum_{i=0}^{t-d-1} B^i kr(I+A)\overrightarrow{e} - \sum_{j=0}^{t-d-1} B^j krB^d\overrightarrow{e} \\ &\quad + B^{t-d}\left[\sum_{l=0}^{d-1} B^l kr(I+A)\overrightarrow{e} + B^d\overrightarrow{p}(0)\right] \\ &= \sum_{i=0}^{t-1} B^i kr(I+A)\overrightarrow{e} + B^t\overrightarrow{p}(0) - \sum_{j=d}^{t-1} B^j kr\overrightarrow{e}\end{aligned}$$

Thus, the population distribution of mobile agents in Eq. (2.19) can be described as follows

$$\overrightarrow{p}(t) = \begin{cases} \sum_{i=0}^{t-1} B^i kr(I+A)\overrightarrow{e} \\ \qquad + B^t\overrightarrow{p}(0) & 0 \leq t \leq d \\ \sum_{i=0}^{t-1} B^i kr(I+A)\overrightarrow{e} & \\ \quad + B^t\overrightarrow{p}(0) - \sum_{j=d}^{t-1} B^j kr\overrightarrow{e} & t > d \end{cases} \tag{2.25}$$

Settleable Case

As shown in Figure 2.9, mobile agents in the i-th node can be subdivided into two kinds: newly generated in the i-th node or coming from its neighboring nodes. If an agent is newly generated, it can find its destination at birth with probability $(d_i+1)/n$. Otherwise, the agent will select a neighboring node with probability $1/(d_i+1)$. Therefore, we have

$$Pr^{(1)}\{i \to j\} = \frac{1}{d_i+1}\left(1 - \frac{d_i+1}{n}\right)$$

For those agents came from neighboring nodes, since the i-th node and the node they came from have been checked, their chance of finding their destination has the probability $(d_i-1)/n$. If they cannot find their destination, they will select a neighboring node with probability $1/d_i$. Therefore, we have

$$Pr^{(2)}\{i \to j\} = \frac{1}{d_i}\left(1 - \frac{d_i-1}{n}\right)$$

Substitute these two parameters into Eq. (2.19), the corresponding matrix A and B can be expressed as $A = C(D+I)^{-1} - (1+\frac{1}{n})CD^{-1}$ and $B = (1+\frac{1}{n})CD^{-1} + C/n$, respectively.

Lemma 4 *If there are $\|\overrightarrow{p}(0)\|_1 = \delta$ agents initially in the network (at time 0), then the total number of the survival agents running in the network at time t, is less than $\delta\left(1 - \frac{\sigma_n - 1}{n}\right)^t$.*

Proof From the definition of matrix B, it is easy to see that

$$\|B\|_1 = \max\left|1 - \frac{d_i - 1}{n}\right| = 1 - \frac{\sigma_n - 1}{n}$$

Therefore, from Lemma 3, we have

$$\begin{aligned}\sum_{1\le j\le n} p_j(t) = \|\overrightarrow{p}(t)\|_1 &= \|B^t\overrightarrow{p}(0)\|_1 \\ &\le (\|B\|_1)^t \|\overrightarrow{p}(0)\|_1\end{aligned}$$

□

Theorem 6 *If $\overrightarrow{p}(0) = 0$, the total number of agents running in the network at any time is no more than $\alpha(\sigma_1, \sigma_n, d)nkr$ where $\alpha(\sigma_1, \sigma_n, d) = \frac{n\sigma_1 - \sigma_1 - 1}{(\sigma_1+1)(\sigma_n-1)}\left[1 - \left(1 - \frac{\sigma_n - 1}{n}\right)^d\right]$.*

Proof According to Eq. (2.25), when $0 \le t \le d$, the total number of mobile agents in the network can be estimated as follows

$$\begin{aligned}\sum_{j=1}^{n} p_j(t) &= \|\overrightarrow{p}(t)\|_1 = \left\|\sum_{i=0}^{t-1} B^i kr(I+A)\overrightarrow{e}\right\|_1 \\ &\le nkr \cdot \|I+A\|_1 \cdot \sum_{i=0}^{t-1} (\|B\|_1)^i\end{aligned}$$

From the definitions of matrices A and B, it is easy to see that $\|I+A\|_1 = 1 - \frac{1}{n} - \frac{1}{\sigma_1+1}$ and $\|B\|_1 = 1 - \frac{\sigma_n - 1}{n}$. Substitute these two values into the above, we have

$$\begin{aligned}\sum_{j=1}^{n} p_j(t) &\leq \frac{(n\sigma_1 - \sigma_1 - 1)nkr}{(\sigma_1+1)(\sigma_n-1)}\left[1 - \left(1 - \frac{\sigma_n - 1}{n}\right)^t\right] \\ &= \alpha(\sigma_1, \sigma_n, t)nkr\end{aligned}$$

where $\alpha(\sigma_1, \sigma_n, t)$ is $\frac{n\sigma_1 - \sigma_1 - 1}{(\sigma_1+1)(\sigma_n-1)}\left[1 - \left(1 - \frac{\sigma_n-1}{n}\right)^t\right]$ for short[1]. When $t \geq d$, from Eq. (2.25), we have

$$\overrightarrow{p}(t) = \sum_{i=0}^{d-1} B^i kr(I+A)\overrightarrow{e} + \sum_{i=d}^{t-1} B^i krA\overrightarrow{e}$$

Due to the fact that

$$a_{ji} = \frac{1}{d_i+1}\left(1 - \frac{d_i+1}{n}\right) - \frac{1}{d_i}\left(1 - \frac{d_i - 1}{n}\right) < 0$$

when $j \in NB(i)$ and $a_{ji} = 0$ otherwise, we have

$$\begin{aligned}\|\overrightarrow{p}(t)\|_1 &\leq \left\|\sum_{i=0}^{d-1} B^i kr(I+A)\overrightarrow{e}\right\|_1 \\ &\leq nkr \cdot \|I+A\|_1 \cdot \sum_{i=0}^{d-1} (\|B\|_1)^i \\ &\leq \alpha(\sigma_1, \sigma_n, d)nkr\end{aligned}$$

□

For the number of agents running on each node, we have the following theorem.

Theorem 7 *The number of agents running on the j-th node, denoted by $p_j(t)$, is no more than $\frac{nd_jkr}{(\sigma_n-1)^2+1}$.*

Proof See Appendix. □

We now estimate the number of agents in each link by the following theorem.

Theorem 8 *The number of agents moving out from the j-th node to each of its neighboring nodes at time t, denoted by $f_j(t)$, satisfies:*

$$f_j(t) = \left(1 - \frac{d_j - 1}{n}\right)\frac{p_j(t)}{d_j} \leq \frac{n - \sigma_n}{n - \sigma_n + \sigma_n^2}kr \tag{2.26}$$

[1] $\alpha(\sigma_1, \sigma_n, t)$is analyzed in Section 6.

Proof This theorem is proved by induction as follows. Based on Eq. (2.25), we have

$$\begin{aligned} p_j(0) &= 0 \\ p_j(1) &= kr \\ &\vdots \\ p_j(t) &= kr + \sum_{i \in NB(j)} \left(1 - \frac{d_i - 1}{n}\right) \frac{p_i(t-1)}{d_i} \end{aligned}$$

Then, by the definition of $f_j(t)$, we have:

$$\begin{aligned} f_j(0) &= 0 \\ f_j(1) &= \frac{1}{d_j}\left(1 - \frac{d_j - 1}{n}\right) kr \leq \frac{n - \sigma_n + 1}{\sigma_n(\sigma_n - 1)} kr \end{aligned}$$

If the theorem holds for time $t-1$, then at time t, we have

$$\begin{aligned} f_j(t) &\leq \frac{1}{d_j}\left(1 - \frac{d_j - 1}{n}\right)\left[kr + \sum_{i \in NB(j)} f_i(t-1)\right] \\ &\leq \frac{1}{d_j}\left(1 - \frac{d_j - 1}{n}\right)\left[kr + \sum_{i \in NB(j)} \frac{n - \sigma_n + 1}{\sigma_n(\sigma_n - 1)} kr\right] \\ &\leq \frac{1}{\sigma_n}\left(1 - \frac{\sigma_n - 1}{n}\right) kr \\ &\quad + \left(1 - \frac{\sigma_n - 1}{n}\right) \frac{n - \sigma_n + 1}{\sigma_n(\sigma_n - 1)} kr \\ &= \frac{kr}{\sigma_n}\left[\left(1 - \frac{\sigma_n - 1}{n}\right)\left(1 + \frac{n - \sigma_n + 1}{\sigma_n - 1}\right)\right] kr \\ &= \frac{kr}{\sigma_n}\left[\frac{n - \sigma_n + 1}{n} \cdot \frac{n}{\sigma_n - 1}\right] \\ &= \frac{n - \sigma_n + 1}{\sigma_n(\sigma_n - 1)} kr \end{aligned}$$

□

Nonsettleable Case

In this case, the corresponding probabilities are

$$Pr^{(1)}\{i \to j\} = \frac{1}{d_i}\left(1 - \frac{d_i + 1}{n}\right) \tag{2.27}$$

$$Pr^{(2)}\{i \to j\} = \frac{1}{d_i - 1}\left(1 - \frac{d_i - 1}{n}\right) \tag{2.28}$$

and the corresponding coefficient matrix in Eq. (2.19) are $A = C[(1 - \frac{1}{n})D^{-1} - (D - I)^{-1}]$ (where $a_{ji} \leq 0$) and $B = C[(1 + \frac{1}{n})(D - I)^{-1} - \frac{I}{n}]$. Furthermore, based on the definition of matrix 1-norm, we have $\|A\|_1 = \frac{1}{n} + \frac{1}{\sigma_n - 1}$, $\|A + I\|_1 = 1 - \frac{1}{n} - \frac{1}{\sigma_1 - 1}$, and $\|B\|_1 = 1 + \frac{1}{n} - \frac{\sigma_n}{n}$. Similar to that in settleable case, we have the following theorem.

Theorem 9 *If $\overrightarrow{p}(0) = 0$, the total number of mobile agents running in the network is no more than $\beta(\sigma_1, \sigma_n, d)nkr$ where $\beta(\sigma_1, \sigma_n, d) = \frac{n(\sigma_1-2)-(\sigma_1-1)}{(\sigma_1-1)(\sigma_n-1)}[1-(1+\frac{1}{n}-\frac{\sigma_n}{n})^d]$.*

Regarding to the number of agents running at one node, we have the following theorem.

Theorem 10 *The number of agents running on the j-th node is no more than $kr+\frac{nd_jkr}{(\sigma_n-1)^2}$.*

Proof See Appendix. □

Regarding to the number of agents moving out from the j-th node at time t, $f_j(t) = (1-\frac{d_j-1}{n})\frac{p_j(t-1)}{d_j-1}$, we have the following theorem.

Theorem 11 *The number of agents moving out from the j-th node at time t, denoted by $f_j(t)$, satisfies:*

$$f_j(t) \leq \frac{n-\sigma_n+1}{(\sigma_n-1)^2}kr \tag{2.29}$$

Proof By the definition of $f_j(t)$, we have

$$f_j(t) = \left(1-\frac{d_j-1}{n}\right)\frac{1}{d_j-1}\left[kr+\sum_{i\in NB(j)} f_i(t-1)\right]$$

and $f_j(0) = 0$ for $j = 1, \cdots, n$. Then, by induction, the theorem can be easily proven. □

It can be seen that the number of agents in the settleable case is smaller than that in the nonsettleable case, but with a lower probability of success. Therefore, each case has its own strength and weakness. Thus we can apply the different cases to control the number of mobile agents running in the network and improve network performance for differing application requirements and network characteristics.

2.4.3 Experimental Results

We have conducted extensive simulation experiments to validate our analytical results for many network topologies with different parameters and found that our results were well suited for various network topologies.

The experimental results for the probabilities of success are depicted in Figure 2.10. The figures in the upper row show the results of the probability of success that an agent can find its destination at the t-th jump, denoted by $p(t)$, and those in the lower row show the results of the probability of success that an agent can find its destination in t jumps, denoted by $P(t)$. The figures in the left column are the probabilities of success in the settleable case, and those in the right are for probabilities of success in the nonsettleable case. In each figure, there are three curves, which indicate the following three cases respectively:

- Case 1: $n = 10000, \theta = 100$;
- Case 2: $n = 10000, \theta = 50$;
- Case 3: $n = 10000, \theta = 20$.

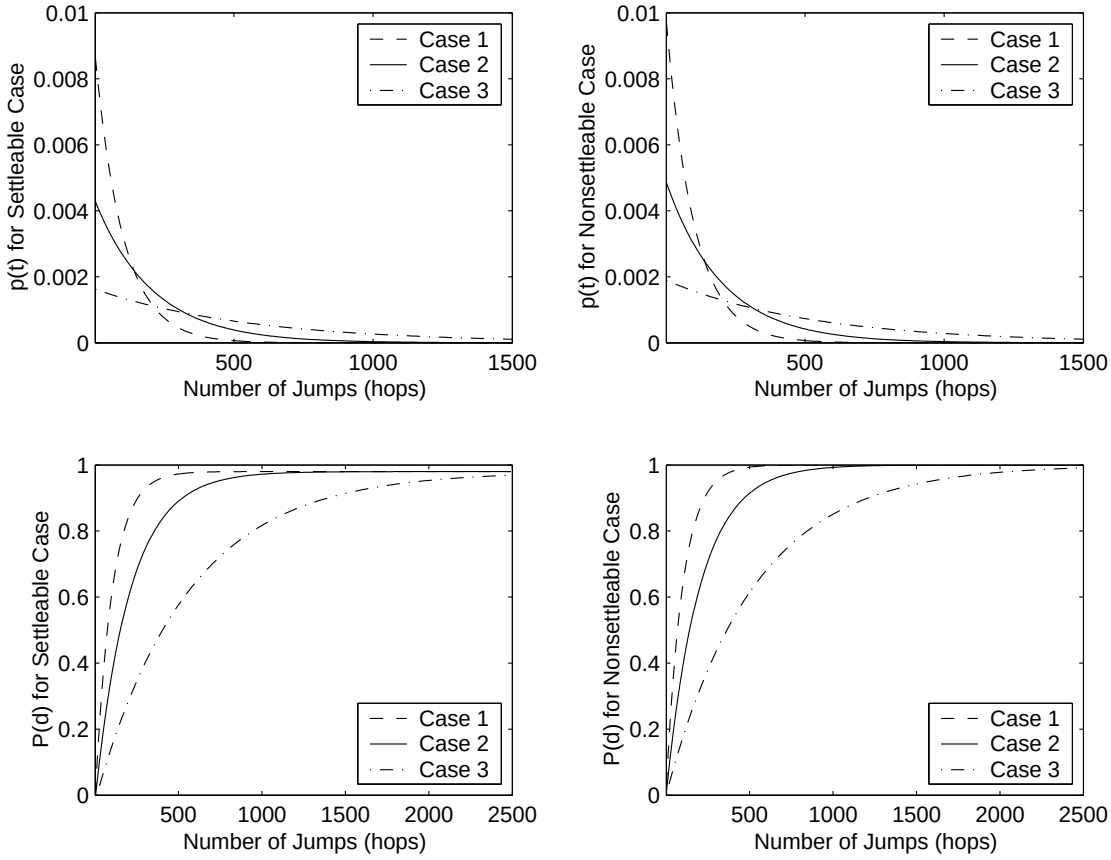


Figure 2.10: Probabilities of Success.

where θ is the average connectivity degree of the network.

From the figures on the upper row we can see that $p(t)$ is a monotonically decreasing function of t and decreases more quickly for large θ. We can also see that $p(t)$ for the nonsettleable case is larger than that for the settleable case, which conforms to the analysis presented previously for each case. From the figures on the lower row we can see that $P(t)$ is a monotonically increasing function of t. It can also be seen that for a network which has larger θ, a higher probability of success, $P(t)$, can be gained in a shorter time. $P(t)$ for the nonsettleable case is larger than that for the settleable case, which also conforms to our analysis for each case. We also compare the probability of success between settleable case and nonsettleable case in Table 2.3.

			$t = 100$	$t = 200$	$t = 500$	$t = 1500$	$t = 2500$
1	$p(t)$	S	0.0033	0.0012	6.4040e-005	3.3768e-009	1.7806e-013
		N	0.0037	0.0014	6.8403e-005	3.2669e-009	1.5603e-013
	$P(t)$	S	0.6067	0.8406	0.9727	0.9800	0.9800
		N	0.6340	0.8647	0.9932	1.0000	1.0000
2	$p(t)$	S	0.0027	0.0016	3.8908e-004	3.1528e-006	2.5548e-008
		N	0.0030	0.0018	4.2022e-004	3.0917e-006	2.2747e-008
	$P(t)$	S	0.3624	0.5984	0.8900	0.9793	0.9800
		N	0.3882	0.6257	0.9142	0.9994	1.0000
3	$p(t)$	S	0.0014	0.0011	6.5825e-004	1.0760e-004	1.7589e-005
		N	0.0016	0.0013	7.3400e-004	1.0958e-004	1.6361e-005
	$P(t)$	S	0.1459	0.2841	0.5758	0.9139	0.9692
		N	0.1734	0.3165	0.6137	0.9423	0.9914

Table 2.3: The comparison of the probability of success between settleable case (denoted by "S") and nonsettleable case (denoted by "N").

Figure 2.11 shows the results of $\alpha(\sigma_1, \sigma_n, d)$ and $\beta(\sigma_1, \sigma_n, d)$ as functions on σ_1 and σ_n, respectively. It can be seen that both of them are increasing functions on σ_1 and decreasing functions on σ_n. Meanwhile, $\beta(\sigma_1, \sigma_n, t)$ is greater than $\alpha(\sigma_1, \sigma_n, t)$ as a function on the connectivity of the network.

Figure 2.12 shows the number of mobile agents running on each node as a function on the number of nodes n where $n = 10000$, $r = 100$, and $k = 50$. The figures on the upper row show the number of mobile agents running on a node at time t, denoted by $p_j(t)$, and the figures on the lower row show the number of mobile agents running in the network at time t, denoted by $\sum_{j=1}^{n} p_j(t)$. The figures on the left column are for the number of mobile agents for the settleable case, and the figures on the right column are for the number of mobile agents for the nonsettleable case. In each graph, there are two curves, which indicate the following two cases respectively:

- Case 1: $\sigma_1 = 50, \sigma_n = 20$;
- Case 2: $\sigma_1 = 200, \sigma_n = 20$.

From the figures on the upper row we can see that $p_j(t)$ is a monotonically increasing function on t. For both cases, $p_j(t)$ increases more quickly for large connectivity. We

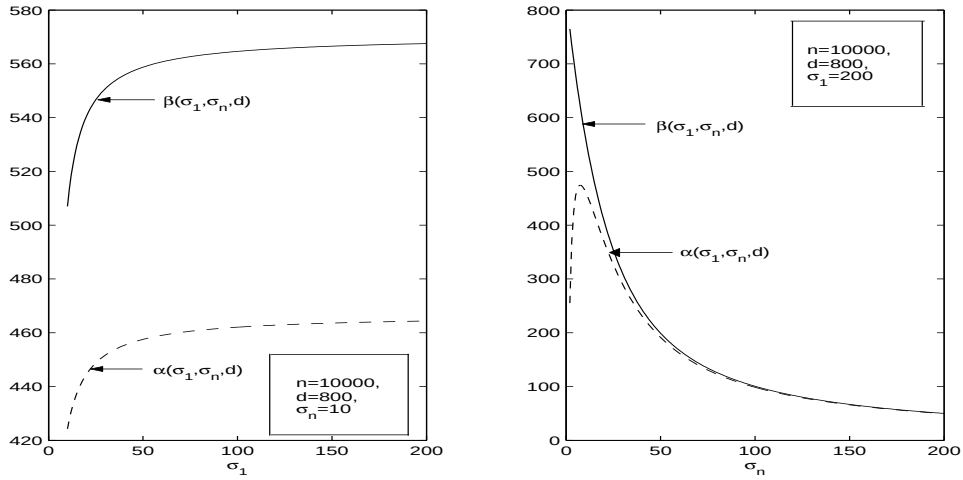


Figure 2.11: The changes of $\alpha(\sigma_1, \sigma_n, d)$ and $\beta(\sigma_1, \sigma_n, d)$.

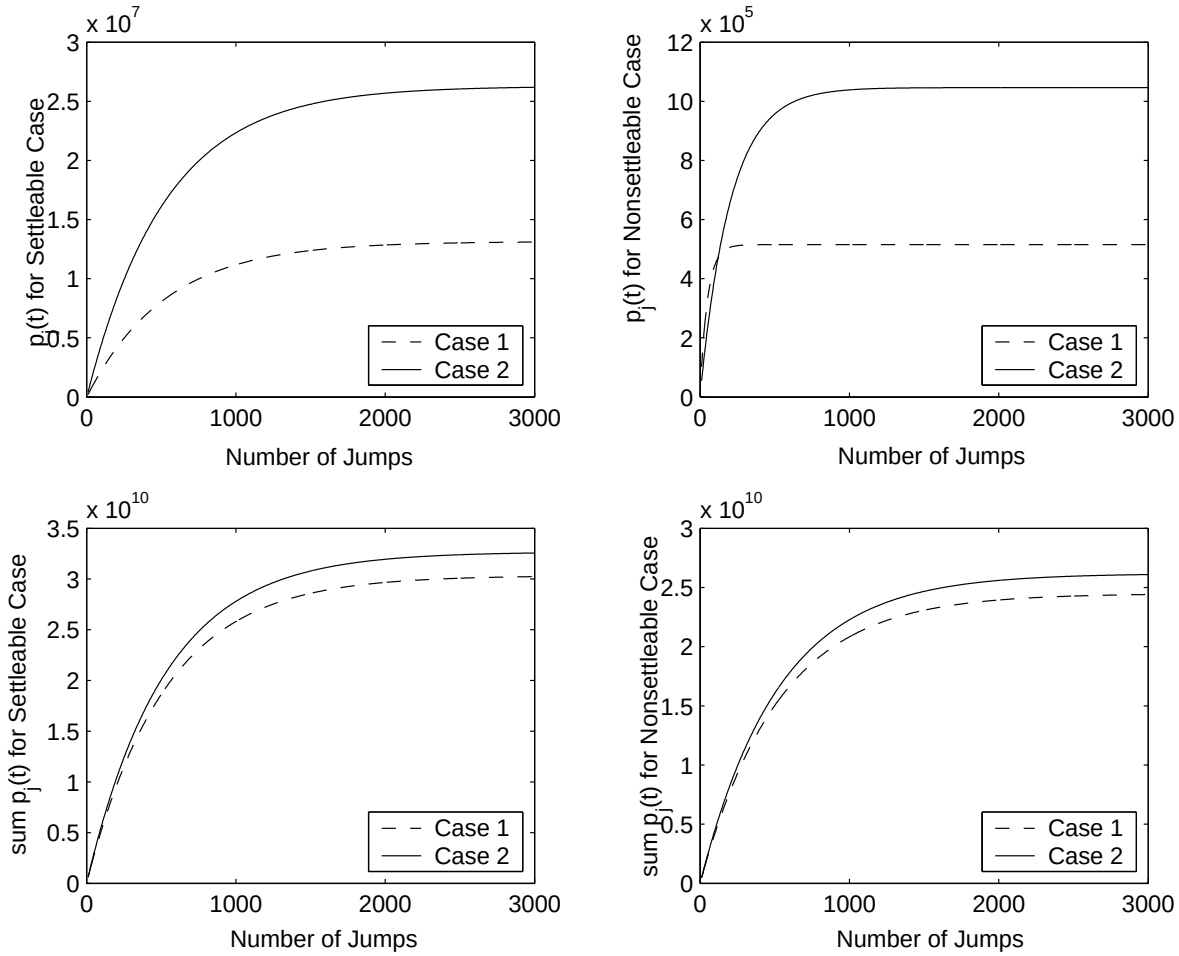


Figure 2.12: Variations of $p_j(t)$ with n.

can also see that $p_j(t)$ for the settleable case is less than that for the nonsettleable case, which conforms to our analysis proposed previously for each case. From the figures on the lower row we can see that $\sum_{j=1}^{n} p_j(t)$ is a monotonically increasing function on t. It can also be seen that for a network which has larger connectivity, the number of mobile agents is larger. $\sum_{j=1}^{n} p(t)$ for settleable case is less than that for nonsettleable case, which conforms to our analysis for each case. We also compare the number of agents between settleable and nonsettleable cases in Table 2.4.

			$d = 100$	$d = 500$	$d = 1000$	$d = 2000$	$d = 3000$
1	$p_j(t)$	S	4.0911e+005	9.5692e+005	1.0386e+006	1.0462e+006	1.0462e+006
		N	2.2431e+006	8.0491e+006	1.1175e+007	1.2849e+007	1.3099e+007
	$\sum_{j=1}^{n} p_j(t)$	S	4.2410e+009	1.5026e+010	2.0831e+010	2.3941e+010	2.4406e+010
		N	5.2550e+009	1.8618e+010	2.5812e+010	2.9666e+010	3.0241e+010
2	$p_j(t)$	S	4.4737e+005	5.1502e+005	5.1504e+005	5.1504e+005	5.1504e+005
		N	4.4711e+006	1.6083e+007	2.2334e+007	2.2583e+007	2.6183e+007
	$\sum_{j=1}^{n} p_j(t)$	S	4.5343e+009	1.6065e+010	2.2272e+010	2.5597e+010	2.6094e+010
		N	5.6592e+009	2.005e+010	2.7798e+010	3.1948e+010	3.2568e+010

Table 2.4: The comparison of the number of agents between settleable case (denoted by "S") and nonsettleable case (denoted by "N").

2.4.4 Concluding Remarks

In this section, we described a general agent-based routing model in which the neighborhood information of each node is known in the node's routing table.We further classify the model into settleable and nonsettleable cases according to whether an agent may stay at the host node during the next step of the search. For each case, we analyzed both the probability of success and the population distribution of mobile agents. We also presented experiments to validate our theoretical results. Both the theoretical and experimental results showed that the behaviors of mobile agents can be characterized by the population of mobile agents and the probability of success, and both two parameters can be controlled by tuning the number of agents generated per request and the number of jumps each mobile agent can make. Our results reveal new theoretical insights into the statistical behaviors of mobile agents and provide useful tools for effectively managing mobile agents in large networks.

2.4.5 Appendix: Proof of Theorem 7 and Theorem 10

Assume that $\overrightarrow{p}(0) = 0$, the number of mobile agents running on a node can be estimated as follows based on Eq. 2.19 and the fact $a_{ji} \leq 0$:

- When $0 \leq t \leq d$,

$$\begin{aligned}\overrightarrow{p}(t) &= \sum_{i=0}^{t-1} B^i kr(I+A)\overrightarrow{e} + B^t \overrightarrow{p}(0) \\ &\leq \sum_{i=0}^{t-1} B^i kr\,\overrightarrow{e} = kr\,\overrightarrow{e} + B\overrightarrow{p}(t-1)\end{aligned}$$

- When $t \geq d$,

$$\begin{aligned}\overrightarrow{p}(t) &= \sum_{i=0}^{t-1} B^i kr(I+A)\overrightarrow{e} \\ &\qquad + B^t \overrightarrow{p}(0) - \sum_{j=d}^{t-1} B^j kr\,\overrightarrow{e} \\ &= \sum_{i=0}^{d-1} B^i kr(I+A)\overrightarrow{e} + B^t \overrightarrow{p}(0) + \sum_{j=0}^{t-1} Akr\,\overrightarrow{e} \\ &\leq \sum_{i=0}^{d-1} B^i kr\,\overrightarrow{e}\end{aligned}$$

From the analysis, it is easy to see that

$$p_j(t) \leq kr + \sum_{i \in NB(j)} Pr^{(2)}\{i \to j\} p_j(t-1) \tag{2.30}$$

In the following, we will present analysis on the number of mobile agents running on a node for each cases respectively.

Proof of Theorem 7

Proof Substitute the value of $Pr^{(2)}\{i \to j\}$ for settleable case in inequality (2.30), we have

$$p_j(t) \leq kr + \sum_{i \in NB(j)} \frac{1}{d_i}\left(1 - \frac{d_i - 1}{n}\right) p_j(t-1)$$

Since $p_j(0) = 0$ and $p_j(t) = 1$, we have

$$\begin{aligned}p_j(t) &\leq (d_j+1)kr \\ &\quad - \left[\frac{\sigma_n - 1}{n} - \frac{\sigma_n - 1}{\sigma_n}\left(1 - \frac{\sigma_n - 1}{n}\right)\right] d_j kr \\ &= (d_j+1)kr - \mu d_j kr\end{aligned}$$

where $\mu = \frac{\sigma_n - 1}{n} + \frac{\sigma_n - 1}{\sigma_n}\left(1 - \frac{\sigma_n - 1}{n}\right)$. If the following inequality is hold,

$$\begin{aligned}p_j(t) &\leq (d_j+1)kr - \left[\frac{\sigma_n - 1}{n} - \frac{1}{\sigma_n}\left(1 - \frac{\sigma_n - 1}{n}\right)\right] \\ &\cdot \sum_{i=0}^{t-2}\left(1 - \frac{\sigma_n - 1}{n}\right)^i d_j kr - \mu\left(1 - \frac{\sigma_n - 1}{n}\right)^{t-1} d_j kr\end{aligned}$$

then, it can be easily proved that the inequality is also hold for $t+1$. Therefore, we have

$$\begin{aligned} p_j(t) &\leq (d_j+1)kr - \mu\left(1-\frac{\sigma_n-1}{n}\right)^{t-1} d_j kr \\ &\quad -\frac{\sigma_n^2-n-1}{n\sigma_n}\sum_{i=0}^{t-2}\left(1-\frac{\sigma_n-1}{n}\right)^{i} d_j kr \\ &= (d_j+1)kr - \frac{\sigma_n^2-n-1}{\sigma_n(\sigma_n-1)} d_j kr \\ &\quad +\left[\frac{\sigma_n^2-n-1}{\sigma_n(\sigma_n-1)}-\mu\right]\left(1-\frac{\sigma_n-1}{n}\right)^{t-1} d_j kr \\ &= kr + \xi d_j kr + \zeta\left(1-\frac{\sigma_n-1}{n}\right)^{t-1} d_j kr \end{aligned}$$

where $\xi = \frac{n-\sigma_n+1}{\sigma_n(\sigma_n-1)}$ and $\zeta = \frac{\sigma_n^2-n-1}{\sigma_n(\sigma_n-1)} - \mu = \frac{2}{\sigma_n} - \frac{n^2+(\sigma_n-1)^2}{n\sigma_n(\sigma_n-1)}$. Therefore, a convenient upper bound of the number of mobile agents is gain as follows

$$p_j(t) \leq \left[\frac{nd_j}{(\sigma_n-1)^2+1}\right] kr$$

□

Proof of Theorem 10

Proof Similar to that in the proof of theorem 7, substitute the value of $Pr^{(2)}\{i \to j\}$ for unsettleable case in inequality (2.30), we have

$$p_j(t) \leq kr + \sum_{i\in NB(j)} \frac{1}{d_i-1}\left(1-\frac{d_i-1}{n}\right) p_j(t-1)$$

By induction, it can be proved that

$$\begin{aligned} p_j(t) &\leq kr + \frac{n}{(\sigma_n-1)^2} d_j kr \left[1-\left(1-\frac{\sigma_n-1}{n}\right)^{t}\right] \\ &\leq kr + \frac{n}{(\sigma_n-1)^2} d_j kr \end{aligned}$$

□

2.5 Routing in Faulty Networks

This section focuses on behavior analysis on mobile agents used in vulnerable network routing. We describe a general agent-based routing model and classify it into two cases based on the reaction of mobile agents to a system failure, namely MWRC (mobile agents with weak reaction capability) and MSRC (mobile agents with strong reaction capability). For each case, we analyze the probability of success (the probability that an agent can find the destination) and the population distribution (the number of mobile agents) of mobile agents.

Since any component of the network (machine, link, or agent) may fail at any time, we classify mobile agents into two kinds based on their reaction to a failure: weak and strong. A mobile agent with weak reaction capacity (MWRC) will die if it subjects to a failure, while one with strong reaction capacity (MSRC) will go back to the previous node, reselect another node, and go on its trip. In this section, we analyze both the probability of success and the population distribution for each case, respectively.

The rest of this section is organized as follows. Subsection 2.5.1 presents the analytical results for mobile agents on the probability of success. Subsection 2.5.2 presents the analytical results for mobile agents on the the population of agents. Section 2.5.3 concludes the section.

2.5.1 The Probability of Success

The Probability of Success for MWRC

For a network with n nodes (i.e., n_1, n_2, $\cdots$, n_n), every node can be the destination of a request, and each node has an independent error rate. Let X_i be a binary valued variable defined as follows:

$$X_i = \begin{cases} 1 & \text{agent dies in the } i\text{-th node due to a failure} \\ 0 & \text{otherwise} \end{cases}$$

with a probability $Pr\{X_i = 1\} = p$. Then, the parameter p measures the incidence of failure in the network. We say a node is down if it is out of work; otherwise, it is up. Once a point-to-point request [2] is made, a number of agents are generated and dispatched into the network. Once an agent reaches an up node, it will find its destination locally with a probability $\frac{1}{n}$. If the agent cannot find its destination here, it will select a neighboring node and move on. Assume that the probability of jumping to any neighboring nodes or die in the current node is same. Regarding to the probability that an agent can find the destination in d jumps, we have the following theorem:

Theorem 12 *The probability, $P^*(n, d, p, k)$, that at least one agent among the k agents can find the destination in d jumps satisfies the following equality:*

$$P^*(n, d, p, k) = 1 - \left[1 - \frac{a(1 - \tau^d)}{1 - \tau}\right]^k, \tag{2.31}$$

where $a = (1 - p)/n$, $b = E[1/c_i]$, and $\tau = (1 - a)(1 - b)$.

Proof Denote the sequence number of node that the agent entered at i-th jump by J_i and the probability that an agent can find its destination at the i-th jump by $P(i)$. The probability that an agent can find its destination at the first jump is $P(1) = \frac{1}{n}(1 - p)$ and the probability that it can't find the destination is $1 - \frac{1}{n}(1 - p)$. If the agent can't find its destination, the probability that it can jump out and search on is $(1 - \frac{1}{n}(1 - p)) \cdot \frac{c_{J_1} - 1}{c_{J_1}}$, and the probability that it can find its destination at the second jump is $P(2) = \frac{1-p}{n}[1 - \frac{1}{n}(1 - p)] \cdot \frac{c_{J_1} - 1}{c_{J_1}}$, and the probability it can't find the destination at the second jump is $(1 - \frac{1-p}{n})^2 \cdot \frac{c_{J_1} - 1}{c_{J_1}}$. If the agent can't find its destination at the second

[2] For point-to-multiple-point requests, the idea is intrinsic same.

jump, the probability that it takes the third jump is $(1-\frac{1-p}{n})^2 \cdot \frac{c_{J_1}-1}{c_{J_1}} \cdot \frac{c_{J_2}-1}{c_{J_2}}$, and $P(3) = (1-p)\frac{1}{n}(1-\frac{1-p}{n})^2 \cdot \frac{c_{J_1}-1}{c_{J_1}} \cdot \frac{c_{J_2}-1}{c_{J_2}}$. Similarly, the probability that an agent can find its destination node at the i-th jump is

$$P(i) = (1-p)\frac{1}{n}\left[1-\frac{1}{n}(1-p)\right]^{d-1}\prod_{i=1}^{d-1}\frac{c_{J_i}-1}{c_{J_i}}.$$

Assume that the number of neighboring nodes of each node in the network is independent with each other with the same distribution, then take expectation on both side of the above equation, and denote $(1-p)/n$ by a, we have

$$\hat{P}(i) = a(1-a)^{i-1}\prod_{j=1}^{i-1}\left[1-E\left(\frac{1}{c_{J_i}}\right)\right] = a(1-a)^{i-1}\left[1-E\left(\frac{1}{c_{J_i}}\right)\right]^{i-1}.$$

Denote $E(1/c_{J_i})$ by b and $(1-a)(1-b)$ by τ, we have

$$\hat{P}(i) = a\tau^{i-1}.$$

So the probability, $P^*(n,d,p,k)$, that at least one agent among k agents can find the destination node in d jumps satisfies the following:

$$P^*(n,d,p,k) = \sum_{s=1}^{k} C_k^s \left[\sum_{i=1}^{d} P(i)\right]^s \left[1-\sum_{i=1}^{d} P(i)\right]^{k-s} = 1-\left[1-\sum_{i=1}^{d} P(i)\right]^k.$$

Due to

$$\sum_{i=1}^{d} P(i) = \sum_{i=1}^{d} a\tau^{i-1} = a\cdot\frac{1-\tau^d}{1-\tau},$$

we have

$$P^*(n,d,p,k) = 1-\left[1-\frac{a(1-\tau^d)}{1-\tau}\right]^k.$$

Hence the theorem is proven. □

The value of b is depended on the probability distribution of c_i. For example, if c_i $(1 \le i \le n)$ are independent and satisfy the uniform distribution, we have

$$b = E[1/c_i] = \int_1^n \frac{1}{c_i}\cdot\frac{1}{n-1}dc_i = \frac{\ln n}{n-1}.$$

From theorem 12, it is easy to estimate $P^*(n,d,p,k)$ as follows.

Corollary 2 *The probability $P^*(n,d,p,k)$ satisfies the following inequalities:*

$$1-\left(\frac{1-a}{1+a\sigma_n-a}\right)^k \le \lim_{d\to\infty} P^*(n,d,p,k) \le 1-\left(\frac{1-a}{1+a\sigma_1-a}\right)^k,$$

where $a = (1-p)/n$, σ_1 and σ_n are the maximum and minimum number of c_i.

Proof Since

$$P(d) = a(1-a)^{d-1}\prod_{i=1}^{d-1}\frac{c_{J_i}-1}{c_{J_i}} \le a(1-a)^{d-1}\prod_{i=1}^{d-1}\frac{\sigma_1-1}{\sigma_1} = a(1-a)^{d-1}\left(\frac{\sigma_1-1}{\sigma_1}\right)^{d-1},$$

we have

$$\sum_{t=1}^{d} P(t) \le \sum_{t=1}^{d} a(1-a)^{t-1}\left(\frac{\sigma_1-1}{\sigma_1}\right)^{t-1} \le a\cdot\frac{1}{1-(1-a)\left(1-\frac{1}{\sigma_i}\right)} = \frac{a\sigma_1}{1+a\sigma_1-a}.$$

Therefore,

$$P^*(n,d,p,k) = 1-\left[1-\sum_{t=1}^{d}P(t)\right]^k \le 1-\left(\frac{1-a}{1+a\sigma_1-a}\right)^k.$$

Similarly, the second inequality can be proved. □

The probability that an agent can find its destination is decided by the connectivity of network and parameters k and d, which coincides with practice. From the theorem above, we can easily get that the probability that none of those k agents can find the destination is less than $\left(\frac{1-a}{1+a\sigma_1-a}\right)^k$ and the probability that all the k agents can find the destination is less than $\left(\frac{a\cdot\sigma_1}{1+a\sigma_1-a}\right)^k$.

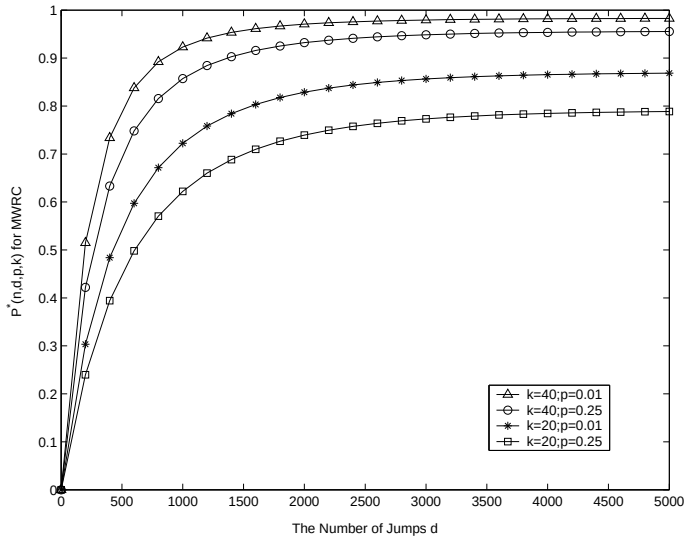


Figure 2.13: The changes of $P^*(n,d,p,k)$ over d for MWRC where c_i satisfies uniform distribution. It is easy to see that $P^*(n,d,p,k)$ is an increasing function on d with a loose upper bound 1. When $p \neq 0$, $P^*(n,d,p,k)$ will not reach 1 no matter how long time the agent can search. The reason is that there is a possibility that the agent will die before it finds its destination. From the figure, it also can be seen that $P^*(n,d,p,k)$ is an increasing function on k and a decreasing function on p.

The Probability of Success for MSRC

Since for MSRC an agent will not die if it has not reached its destination within its life-span, the probability of success for MSRC equals to r/n where r is the number of nodes that the agent has entered and checked. Denote the i-th node that an agent enters by h_i, the number of neighboring nodes of the i-th node c_i, and the number of neighboring nodes that the agent has selected by v_i (i.e., the agent fails to enter the first $v_i - 1$ nodes and can only enter the v_i-th selected node). Regarding to the average number of nodes selected, we have the following result.

Lemma 5 *The average number of neighboring nodes selected by an agent at each node* $E(v_i) = \frac{1-E[p^{c_i}]}{1-p} - E\left[c_i p^{c_i}\right]$.

Proof The probability that an agent can enter the first selected node, h_i^1, in $NB(i)$, equals to $1-p$, and the probability that the agent can enter the second selected node equals to $p(1-p)$. By recursion, the probability that the agent enters the v_i-th node equals to $p^{v_i-1}(1-p)$. Therefore, the average number of nodes the agent selected in $NB(i)$ satisfies

$$E(v_i|NB(i)) = \sum_{v_i=1}^{c_i} v_i p^{v_i-1}(1-p) = \frac{1-p}{p} \cdot \frac{c_i p^{c_i+2} - (c_i+1)p^{c_i+1} + p}{(1-p)^2} = \frac{1-p^{c_i}}{1-p} - c_i p^{c_i}.$$

Thus, the average number of nodes the agent selected at each node during the agent's trip satisfies

$$E(v_i) = E[E(v_i|NB(i))] = \frac{1-E\left[p^{c_i}\right]}{1-p} - E\left[c_i p^{c_i}\right].$$

Hence, the lemma is proven. □

Regarding the estimation of r, we have the following result.

Lemma 6 *Let r be the number of nodes that the agent visits, then the average number of nodes that an agent enters satisfies*

$$E(r) = \left\lfloor \frac{d}{2E(v_i) - 1} \right\rfloor,$$

where $\lfloor x \rfloor$ indicates the greatest integer less then or equal to x (i.e., $x - 1 < \lfloor x \rfloor \leq x$).

Proof Denote the j-th selected node from the neighboring nodes of node h_i by h_i^j, the path the agent traverse from h_i to h_{i+1} can be expressed as $h_i, h_i^1, h_i, h_i^2, \cdots, h_i, h_i^{v_i}$. The v_i-th selected node is the node h_{i+1}. Inside this process, there are $2(v_i - 1) + 1$ jumps the agent takes. Since an agent will die if it cannot find its destination in d jumps, we have

$$r = \max\left\{ l : \sum_{i=1}^{l} (2v_i - 1) \leq d \right\}.$$

Taking expectation on the inequality, we have

$$d \geq E\left[\sum_{i=1}^{l} (2v_i - 1)\right] = E(l) \cdot [2E(v_i) - 1],$$

Table 2.5: The comparison of the probability of success between MWRC and MSRC

k			1	2	5	10
$n = 6000, p = 0.001$	d=500	MWRC	0.0571	0.1110	0.2548	0.4447
		MSRC	0.0832	0.1594	0.3522	0.5803
	d=1000	MWRC	0.0826	0.1583	0.3501	0.5776
		MSRC	0.1663	0.3050	0.5973	0.8378
$n = 10000, p = 0.001$	d=500	MWRC	0.0391	0.0767	0.1809	0.3292
		MSRC	0.0499	0.0973	0.2258	0.4006
	d=1000	MWRC	0.0626	0.1213	0.2763	0.4762
		MSRC	0.0998	0.1896	0.4089	0.6505

since l and v_i are independent to each other, and the distributions of v_i are same for $1 \leq i \leq l$. Let $r = \max\{l\}$, then the lemma is proven. □

From Lemma 1 and Lemma 2, it is readily to get the following theorem.

Theorem 13 *The probability, $P^*(n, d, p, k)$, that at least one agent among k agents can find the destination in d jumps equals to $1 - [1 - E(r)/n]^k$, where $E(r) = \left\lfloor \frac{d}{2E(v_i)-1} \right\rfloor$ and $E(v_i) = E[E(v_i|NB(i))] = \frac{1-E[p^{c_i}]}{1-p} - E\left[c_i p^{c_i}\right]$.*

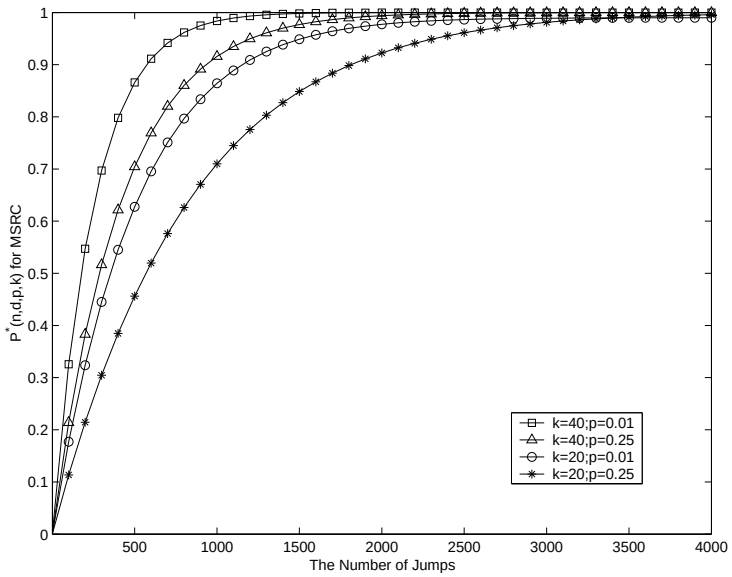


Figure 2.14: The changes of $P^*(n, d, p, k)$ over d for MSRC where c_i satisfies uniform distribution. From the figure, it can be seen that $P^*(n, d, p, k)$ is an increasing function on k and a decreasing function on p.

Table 1 compares the probability of success $P^*(n, d, p, k)$ between MWRC and MSRC with different n, d, and k. Since a node failure is a rare event, we set $p = 0.001$ in this simulation. From the table, it can be seen that $P^*(n, d, p, k)$ for MSRC is greater than that for MWRC with the same parameters n, d, p, and k.

2.5.2 The Population Distribution of Mobile Agents

The Population Distribution of Mobile Agents for MWRC

First, we consider the situation that the agents run in the network with infinite life-span. Assume that at time $t-1$, there are $p_i(t-1)$ agents in the i-th node, these agents search for the destination locally, and the expected number of agents that cannot find the destination is equal to $\left(1-\frac{1}{n}\right)p_i(t-1)$, and the expected number of mobile agents that move from the i-th node to the j-th node is equal to $\left(1-\frac{1}{n}\right)\frac{1-p}{c_i}p_i(t-1)$. The total number of agents that move to the j-th node at time t is $\sum_{i\in NB(j)}\left(1-\frac{1}{n}\right)\frac{1-p}{c_i}p_i(t-1)$, where $NB(j)$ denotes the set of the neighboring nodes of the j-th node. Consider the new generated agents in the j-th node at time t, we have the following equation:

$$p_j(t) = km - \Omega_j(t) + \sum_{i\in NB(j)}\left(1-\frac{1}{n}\right)\frac{1-p}{c_i}p_i(t-1),$$

where m is the average number of requests initiated at time t at a node, k is the number of agents generated per request, and $\Omega_j(t)$ indicates the number of mobile agents on the j-th node at time t that are generated at time $t-d$. We eliminate the number of these agents from $p_j(t)$ because these agents will die at time t.

Let $A=(1-p)(1-\frac{1}{n})(\Phi-I)C^{-1}=(a_1,a_2,\cdots,a_n)$ be an $n\times n$ matrix, where a_j is the j-th column vector of A. Obviously, we have $\|a_j\|_1=(1-p)(1-\frac{1}{n})\frac{c_j-1}{c_j}$. Let $\overrightarrow{p}(t)=(p_1(t),p_2(t),\cdots,p_n(t))^T$, and $\overrightarrow{e}=(1,\cdots,1)^T$. At any time, the distribution of newly generated mobile agents is $km\overrightarrow{e}$ based on the assumption that the average number of requests received by a node is m. After searching d nodes, the distribution of survival agents among these agents is $A^dkm\overrightarrow{e}$. Therefore, we have $\overrightarrow{\Omega}(t)=(\Omega_1(t),\cdots,\Omega_n(t))^T=A^dkm\overrightarrow{e}$. Thus, the population distribution of mobile agents can be expressed in vector-matrix format as follows:

$$\overrightarrow{p}(t)=A\overrightarrow{p}(t-1)+km\overrightarrow{e}-A^dkm\overrightarrow{e} \tag{2.32}$$

.

Regarding to the population distribution of mobile agents with limited life-span, we have the following theorem based on the above analysis.

Theorem 14 *The distribution of mobile agents can be expressed as follows:*

$$\overrightarrow{p}(t)=\begin{cases} 0 & t=0; \\ \sum_{i=0}^{t-1}A^{i-1}km\overrightarrow{e} & 0<t\le d; \\ \sum_{i=0}^{d-1}A^{i-1}km\overrightarrow{e} & t>d. \end{cases} \tag{2.33}$$

Proof If the distribution of mobile agents generated at time 0 is $\overrightarrow{p}(0)$, then after time t, the distribution of the agents is $A^d\overrightarrow{p}(0)$. Thus, according to the assumption that an agent will die if it cannot find the destination in d steps, we have
(1) When $t\le d$,

$$\begin{aligned}\overrightarrow{p}(t) &= km\overrightarrow{e}+A\overrightarrow{p}(t-1)=km\overrightarrow{e}+A\left[km\overrightarrow{e}+A\overrightarrow{p}(t-2)\right] \\ &= (I+A)km\overrightarrow{e}+A^2\overrightarrow{p}(t-2)=\cdots=\sum_{i=0}^{t-1}A^ikm\overrightarrow{e}+A^t\overrightarrow{p}(0).\end{aligned}$$

Since the initial population of mobile agents $\overrightarrow{p}(0) = 0$, the result for $t \leq d$ is proven.
(2) When $t > d$, then at time t, all the survival agents generated at time $t - d$ will die, so the distribution of agents under this case can be illustrated as

$$\begin{aligned}\overrightarrow{p}(t) &= km\overrightarrow{e} + A\overrightarrow{p}(t-1) - A^d km\overrightarrow{e} = \sum_{i=0}^{t-d-1} A^i km\overrightarrow{e} + A^{t-d}\overrightarrow{p}(d) - A^d \cdot \sum_{i=0}^{t-d-1} A^i km\overrightarrow{e} \\ &= \sum_{i=0}^{t-d-1} A^i km\overrightarrow{e} + A^{t-d} \cdot \left[\sum_{i=0}^{d-1} km\overrightarrow{e} + A^d\overrightarrow{p}(0)\right] - A^d \cdot \sum_{i=0}^{t-d-1} A^i km\overrightarrow{e} = \sum_{i=0}^{d-1} A^i km\overrightarrow{e}.\end{aligned}$$

Hence the theorem is proven. □

From the theorem above, we can easily see that the distribution of mobile agents will not exceed $(I + A + \cdots + A^{d-1})km\overrightarrow{e}$, that is, the number of mobile agents in our model will not increase infinitely.

Since mobile agents are generated frequently and dispatched to the network, it is important to estimate the maximum number of mobile agents running in the network and in each node. When there are too many agents in the network, they will introduce too much computational overhead to node machines, which will eventually become very busy and indirectly block the network traffic.

Regarding to the number of agents running in the network, we have the following theorem.

Theorem 15 *The number of agents running in the network can be estimated as follows:*

$$\sum_{j=1}^{n} p_j(t) \leq \frac{n^2\sigma_1 km}{n + \sigma_1 - 1}. \tag{2.34}$$

Proof By the definition of matrix 1−norm,we have

$$\sum_{j=1}^{n} p_j(t) = \|\overrightarrow{p}(t)\|_1 \leq \|km\overrightarrow{e}\|_1 \cdot \|\sum_{i=0}^{d-1} A^i\|_1 \leq nkm \cdot \sum_{i=0}^{d-1} (\|A\|_1)^i \leq \frac{nkm}{1 - \|A\|_1}.$$

Due to $\|A\|_1 = (1-p)\left(1 - \frac{1}{n}\right)\frac{\sigma_1 - 1}{\sigma_1}$, it is easy to prove that

$$\|\overrightarrow{p}(t)\|_1 \leq \frac{nkm}{1 - (1-p)\left(1 - \frac{1}{n}\right)\left(1 - \frac{1}{\sigma_1}\right)} = \frac{n^2\sigma_1 km}{pn\sigma_1 + (1-p)(n + \sigma_1 - 1)}.$$

Since $n\sigma_1 \geq n + \sigma_1 - 1$, the theorem is proven. □

Regarding to the number of agents running in a node, we have the following theorem.

Theorem 16 *The number of agents running in the j-th node can be estimated as follows:*

$$p_j(t) \leq km + \frac{nc_j km}{(np + 1 - p)\sigma_n}(1 - \alpha^t) \leq km + \frac{nc_j km}{(np + 1 - p)\sigma_n},$$

where $\alpha = (1 - \frac{1}{n})(1 - p)$.

Proof The theorem can be proved by induction. First, for $t = 0$, it is easy to see that the theorem is hold. Assume that for any t, the theorem is hold, that is,

$$p_j(t) \leq km + \frac{nc_jkm}{(np+1-p)\sigma_n}(1-\alpha^t) = km + \frac{c_jkm}{\sigma_n}km\sum_{l=1}^{t-1}\alpha^l,$$

then for $t+1$, we have

$$\begin{aligned} p_j(t+1) &= km + \sum_{i\in NB(j)}\left(1-\frac{1}{n}\right)\frac{1-p}{c_i}p_i(t-1) = km + \alpha\sum_{i\in NB(j)}\frac{p_i(t-1)}{c_i} \\ &\leq km + \alpha\cdot\sum_{i\in NB(j)}\left(\frac{km}{c_i} + \frac{km}{\sigma_n}\sum_{l=1}^{t-1}\alpha^l\right) \leq km + \frac{c_jkm}{\sigma_n}\sum_{l=1}^{t}\alpha^l \\ &\leq km + \frac{nc_jkm}{(np+1-p)\sigma_n}(1-\alpha^{t+1}) \leq km + \frac{nc_jkm}{(np+1-p)\sigma_n}. \end{aligned}$$

Hence, the theorem is proven. □

From the above analytical results, we can claim that both the number of mobile agents in the network and the number of mobile agents on each node will not increase infinitely over time t. The upper bound of these two numbers can be controlled by tuning the number of mobile agents generated per request.

The Population Distribution of Mobile Agents for MSRC

For MSRC, a mobile agent will not die when it moves to a failing node. It will return back to the previous node and reselect another neighboring node to move. Thus, similar to the analysis for MWRC, the population distribution of mobile agents can be expressed as follows.

$$\overrightarrow{p}(t) = \begin{cases} 0 & t = 0; \\ km\overrightarrow{e} & t = 1; \\ A\overrightarrow{p}(t-1) + p\cdot\overrightarrow{p}(t-2) + km\overrightarrow{e} & 2 \leq t \leq d; \\ A\overrightarrow{p}(t-1) + p\cdot\overrightarrow{p}(t-2) + km\overrightarrow{e} - A^dkm\overrightarrow{e} & t \geq d. \end{cases} \tag{2.35}$$

From Eq. (2.35), we can estimate the number of mobile agents running in the network as follows:

Theorem 17 *The total number of mobile agents running in the network is no more than* $\frac{n^2\sigma_1km}{(n+\sigma_1-1)(1-p)}$.

Proof By the definition of vector norm, the total number of mobile agents running in the network can be expressed as $\sum_{j=1}^{n}p_j(t) = \|\overrightarrow{p}(t)\|_1$. Therefore, from Eq. (2.35), it is easy to see that

$$\|\overrightarrow{p}(t)\|_1 \leq \|A\|_1\|\overrightarrow{p}(t-1)\|_1 + p\|\overrightarrow{p}(t-2)\|_1 + \|km\overrightarrow{e}\|_1,$$

since all the parameters a_{ji}, k, m are positive. For $t = 0$ and $t = 1$, we have

$$\left.\begin{array}{l}\|\overrightarrow{p}(0)\|_1 = 0 \\ \|\overrightarrow{p}(1)\|_1 = nkm\end{array}\right\} \leq \frac{n^2\sigma_1km}{(n+\sigma_1-1)(1-p)}.$$

If the theorem is hold for $t-1$ and $t-2$, then for t, we have

$$\|\overrightarrow{p}(t)\|_1 \leq \left[(1-p)\left(1-\frac{1}{n}\right)\left(1-\frac{1}{\sigma_1}\right)+p\right]\cdot\frac{n^2\sigma_1 km}{(n+\sigma_1-1)(1-p)}+nkm=\frac{n^2\sigma_1 km}{(n+\sigma_1-1)(1-p)},$$

where $\|A\|_1=(1-p)\left(1-\frac{1}{n}\right)\frac{\sigma_1-1}{\sigma_1}$. Hence, the theorem is proven. □

Regarding to the maximum number of mobile agents running on a node, we have the following theorem.

Theorem 18 *The number of mobile agents running on a node is no more than* $\frac{nc_j km}{(1-p)\sigma_n}$.

Proof From Eq. 2.35, we know that when $t \leq d$,

$$p_j(t)=km+p\cdot p_j(t-2)+\sum_{i\in NB(j)}\left(1-\frac{1}{n}\right)\frac{1-p}{c_i}p_i(t-1).$$

Define $f_j(t)=(1-\frac{1}{n})\frac{1-p}{c_j}p_j(t)$ and substitute it in the above function, we have

$$f_j(t) = \left(1-\frac{1}{n}\right)\frac{1-p}{c_j}km+p\cdot f_j(t-2)+\left(1-\frac{1}{n}\right)\frac{1-p}{c_j}\sum_{i\in NB(j)}f_i(t-1).$$

By induction, we can prove that

$$f_j(t)\leq\frac{n-1}{\sigma_n}km.$$

Therefore, it can be easily prove that for all $0\leq t\leq d$,

$$p_j\leq\frac{\frac{n-1}{\sigma_n}km}{\left(1-\frac{1}{n}\right)\frac{1-p}{c_j}}=\frac{nc_j km}{\sigma_n(1-p)}.$$

When $t\geq d$, since

$$\overrightarrow{p}(t)=A\overrightarrow{p}(t-1)+p\cdot\overrightarrow{p}(t-2)+km\overrightarrow{e}-A^d km\overrightarrow{e},$$

we have

$$p_j(t)\leq km+p\cdot p_j(t-2)+\sum_{i\in NB(j)}\left(1-\frac{1}{n}\right)\frac{1-p}{c_i}p_i(t-1).$$

Similar to the analysis for $0\leq t\leq d$, the upper bound $p_j(t)\leq\frac{nc_j km}{\sigma_n(1-p)}$ is also hold. Hence, the theorem is proven. □

It is easy to see that both the total number of mobile agents running in the network and the number of mobile agents running on a node are greater than that for MWRC. The reason is because mobile agents in MWRC case have a higher death rate than in MSRC case. It also can be seen that the number of mobile agents can be justified by tuning the number of mobile agents generated per request.

2.5.3 Concluding Remarks

In this section, we addressed the problem of network routing and management by deploying mobile agents. We analyzed the probability of success and the population growth of mobile agents under our agent-based routing model. For mobile agents with weak reaction capability (MWRC), we obtained the following analytical results: (1) The probability of success, $P^*(n, d, p, k)$, that at least one agent among k agents can find the destination in d jumps equals to $1 - [1 - \frac{a(1-c^d)}{1-c}]^k$, where $a = (1-p)/n$, $b = E[1/c_i]$, $c = (1-a)(1-b)$, d is the maximum number of jumps an agent can make, k is the number of agents generated per request, p is the probability that a node may fail, n is the number of nodes in the network, and c_i is the connectivity of the i-th node. (2) The total number of agents running in the network is less than $\frac{n^2\sigma_1 km}{n+\sigma_1-1}$, where $\sigma_1 = \max_{1\le j\le n} c_j$, and m is the average number of requests keyed in one node once. (3) The number of mobile agents running in each node, $p_j(t)$, is less then $km + \frac{nc_jkm}{(np+1-p)\sigma_n}$. For mobile agents with strong reaction capability (MSRC), we obtained the following analytical results: (1) $P^*(n, d, p, k) = 1 - (1 - E(r)/n)^k$, where $E(r) = \lfloor d/[2E(v_i - 1)] \rfloor$, and v_i is the number of neighboring nodes of the i-th node an agent selected. (2) $\sum_{j=1}^{n} p_j(t) \le \frac{n^2\sigma_1 km}{(n+\sigma_1-1)(1-p)}$, where $\sigma_1 = \max_{1\le j\le n} c_j$. (3) $p_j(t) \le \frac{nc_jkm}{(1-p)\sigma_n}$, where $\sigma_n = \min_{1\le j\le n} c_j$.

We can see that the probability of success $P^*(n, d, p, k)$ is a monotonically increasing function on k, d, and a monotonically decreasing function on p, n; while the number of agents is a monotonically increasing function on k, n, d and a monotonically decreasing function on p. For the same p, routing in a smaller network may get a greater probability of success. For a network with a lot of nodes, this probability can be enlarged by increasing k and/or d. We can also see that for the same k and p, both the probability of success and the number of agents for MWRC are less than those for MSRC. Based on these results, we can dispatch a small number of mobile agents and achieve a good probability of success by selecting an optimal number of mobile agents generated per request and giving them an optimal life-span limit.

Chapter 3

Mobile Agent-Based Fault-Tolerant Execution

In the context of the mobile agent-based technology, fault tolerant is an important issue to guarantee mobile agents' consistent execution. In this chapter, we propose a new fault-tolerant execution model of mobile agents, which effectively combines the replication approach and the checkpointing approach. Failures are classified into two classes based on their intrinsic different effects on mobile agents. For each kind of failures, a specified exceptional-event handling method is adopted. The introduction of exceptional-event handling allows performance improvements during mobile agents' execution. The behaviors of mobile agents are statistically analyzed through several key parameters, including the migration time from node to node, the life expectancy of mobile agents, and the population distribution of mobile agents, to evaluate the performance of our model.

3.1 Introduction

As we know, any component (node, link, or agent) in a network may fail, thus preventing agents from proceeding with their executions. To integrate agent-driven systems into today's e-society, issues such as reliability, i.e., the ability of an agent to consistently perform a required function under stated conditions for a specified period of time, are of paramount importance [116]. Fault tolerance is one of the most important issues on reliability, to guarantee consistent execution even if an agent is subjected to a system failure.

Existing mobile agent-based execution mechanisms can be broadly categorized into two kinds: replication versus checkpointing. Replication approach [75] uses replicated servers to mask the failures. When one server is down, the computation still continues by using the results from other servers. However, this approach requires multiple servers at each stage, even when no failure occur; these redundant servers add an overhead to the communication between servers to guarantee the consistent execution, especially when they are widely separated. Checkpointing [89], on the other hand, saves the intermediate states of an agent into a stable storage so that the agent's execution can be resumed in case of a failure. Although the communication cost for replication is avoided, the agents may be blocked even by a single failure. Besides, the occupied storage spaces are locked until the agent entirely completes execution at its destination. Unlocking storages requires additional messages sent to all nodes of the itinerary. Furthermore, there is a

lack of theoretical analysis on assessing the effectiveness of using mobile agents for both approaches.

In this chapter, we propose a new fault-tolerant mobile agent execution model based on a combination of the replication approach and the checkpointing approach. In our model, the total overhead can be greatly reduced by eliminating redundant communication between nodes where there is no failure occurred. Theoretical analysis of mobile agent's migration time which is the time period that a mobile agent migrates from one node to another, life expectancy which is the average life-span of mobile agents, and population distribution which is the number of mobile agents running in the network are given for our model. Our approach exploits a new way to design cost-effective fault-tolerant agent-driven systems. Our analysis reveals new theoretical insights into the statistical behaviors of mobile agents and provides useful tools for effectively estimating the performance of agent-driven systems.

The main contributions of this chapter are summarized as follows:

- We proposed a new fault-tolerant execution model which effectively combines available techniques. In our model, failures are classified into two classes based on their intrinsic different effects on mobile agents. For each kind of failure, an exceptional-event handling method is adopted. Our model greatly decreases network resources consumption and improves the overall network performance. It achieves better cost-effectiveness than existing models.

- We applied stochastic process to analyze the behaviors of mobile agents in fault-tolerant execution, which exploits a new approach to assessing the performance of agent-based systems. For the first time, to the best of our knowledge, the behaviors of mobile agents in networks that may contain faults are statistically analyzed in a quantitative way.

- We analyzed several key parameters which dominate the behaviors of mobile agents, including the migration time, the life expectancy, and the population distribution, in great theoretical depth. Our analysis provides a useful way for controlling mobile agents' behaviors by tuning relevant parameters according to various system characteristics.

The remainder of this chapter is structured as follows. Section 2 reviews fault-tolerant approaches that utilize either replication or checkpointing. Section 3 presents our model. Section 4 analyzes the migration time, the life expectancy, and the population distribution. presents some discussion on the case that the network is reliable. Finally, Section 5 presents our experimental results and Section 6 gives our conclusion remarks.

3.2 Related Work

Mobile agents' ability to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems [66]. The characters of mobile agents make agent-driven system different from those traditional systems. Many mobile agent-based fault-tolerant approaches have been proposed so far. These existing approaches can be mainly divided into two classes: replication or checkpointing.

In a replication scheme, an agent is replicated and migrated to several sites. The replicas must agree on one execution site for each stage and the redundant execution of other sites must be undone. For the agreement procedure, a distributed migration proposed in [130] injects an agent replica into stable storage upon arriving at an agent server. However, if an agent server crashes, the replica remains unavailable for an unknown time period. In [57, 101, 116], a protocol was presented that provides the exactly-once property in the migration of mobile agents. Every time an agent wants to migrate, it is replicated to a set of nodes. In every group of replicas there is one worker node, which is responsible for the execution of the agent. The other replica nodes receive a copy of the agent and act as observers of the worker node. When the worker node fails, the observers will detect it and elect a new worker by running an election protocol. The elected worker will try to provide the same service or, if that is not possible, execute an exception handling mechanism. In [112] a protocol was proposed which can be seen as a variation of that proposed in [101]. This scheme includes the distributed storage of recovery information and it used a three-phase commit protocol. However, the execution of a 3PC protocol is more expensive and introduces a higher latency. In [96, 97], the consensus protocol is used; among the replicas, one, called primary, is initially responsible for the execution. The priorities of the replicas are predetermined, hence, a replica takes over the execution, only when all of the higher priority replicas fail. To detect a possible failure, the time-out is used in [87]. When a primary does not respond within a certain time-out period, the first replica broadcasts the failure of the primary and starts up the agent execution. However, it is possible that the primary was too slow to respond within the time-out period, in which case, two agents may have performed the same execution stage. In [98], a replication model is modelled as a sequence of agreement problems. This model ensures both nonblocking and exactly-once properties, but it generates multiple replica agents and introduces a large amount of communication for each hop in the network, which will certainly consume a certain amount of network resources.

In the checkpointing scheme, the intermediate states of an agent are saved into a stable storage periodically, so that the agent can resume the execution from the most recent saved state after a crash [35, 83]. In [132], checkpointing is provided to deal with node failures, but a mobile agent in this system cannot proceed with its computation until the failed node recovers and restores the state of the mobile agent from persistent storage. In [28], a checkpoint manager monitors all the agents inside a cluster of machines and responds to restart the agents when there is a node failure. In addition to communication cost, the checkpoint manager can also be a point of failure. In [57], a rear-guard agent is associated with each agent, to respond by lunching a new agent when a failure causes the first agent to vanish. Again, along with the communication cost, the checkpoint manager is also a potential point of failure. In addition, when the rear agent loses track of the work agent, in an asynchronous system it can not correctly determine whether the work agent has failed or is just obstructed by the network latency. In the approach proposed by [91], the server that generates the agent has to track is during the whole execution process, which results in high communication costs. In [92], a replica is generated at each server the agent visits and responds by generating a new agent in case of a failure; this approach has the same problems as in [57]. In [3], periodic checkpointing prevents applications from loosing all computation data in case of a failure. However, there is a trade-off. Excessive checkpointing results in performance degradation, while deficient checkpointing still incurs an expensive recovery overhead.

Although mobile agent technology has been promoted as an emerging technology that makes it much easier to design, implement and maintain distributed system, especially in low-bandwidth, low reliability networks, the theoretical analysis of mobile agent's behavior is in its infancy. In [99, 100, 118], mathematic models for mobile agent-based network routing were established and stochastic analysis on the behavior of mobile agents were provided. In [118], the population growth of mobile agents was analyzed. In [99, 100], analysis of both the population growth and the probability of success were provided. In [31, 139], a newly proposed method, the cross-entropy method, was applied to guiding the behavior of mobile agents. However, all the existing works focused on networks that did not contain faults.

3.3 The Execution Model of Mobile Agents

The ability of fault tolerance in a mobile agent system depends strongly on the way that failures occur, on the effects to agents, and on the resources consumed by the approach which prevents agents from being blocked by a system failure. As we have mentioned, both the checkpointing approach and the replication approach have their own strong points and deficiencies. Based on an analysis of the available literature, we propose a new execution model which is robust and cost-efficient.

Suppose that in an agent-driven system (as we have described in chapter 2), once an agent reaches a node, it leaves a checkpoint in a temporary storage area and executes locally. After finishing its execution, the checkpoint is updated by a replica. The agent moves to a neighboring node. When it reaches the neighboring node, it also leaves a checkpoint in the temporary storage and executes there. During the agent's execution, communication is maintained between the previous node and the current one. Both the previous node, say h_{i-1}, and the current node work as surveillances, monitoring the states of each other and the agent. If the agent completes its execution on the current node, the current node is renamed as h_i. The previous node retires from the surveillance and deletes the replica on it. The checkpoint on h_i is updated by a replica of the agent. If a system failure takes place on the current host, the agent's execution will be influenced and the surveillant h_{i-1} will detect the situation. On the other side, if h_{i-1} fails during the agent's execution, the current host can also detect it.

If a failure takes place on the selected route of an agent, the agent is blocked. Different kind of failures will cause different effects on mobile agents and different reactions of mobile agents. Based on the analysis of the available literature, we mainly divided system failures into two classes, on-site and frontal, according to their effects on mobile agents:

- An *on-site failure* is a node's failure during an agent's execution within it. It may block the agent and alter the agent's state.
- A *frontal failure* is a node's failure before an agent's entrance to it which blocked the agent's entry, but did not alter the agent's state.

The following lists several failure scenarios that agent may encounter.

- A link breaks off. All agents running through it are crashed.
- A link is blocked or closed down. All agents wishing to pass through it have to select other paths.

- A node is out of condition because of a failure. All agents on it are lost or executing abnormally.
- A node is down, all agents moving to it are blocked. They must go back to the previous node and select another neighboring node to continue their operation.
- The agent fails. It ceases its execution or executes in a wrong way.

Obviously, the first and third scenarios are on-site failures, while the second and fourth scenarios are within the range of frontal failures.

When an on-site failure befalls the selected node during an agent's execution, it may influence the executing agent. As a result, the agent's state becomes unreliable even if the agent survives in the failure. Suppose that $NB(i-1)$ is the set of neighboring nodes of node h_{i-1} and h_i^j is the j-th node selected by the agent in $NB(i-1)$. After finishing its execution in node h_{i-1}, the agent selects one node in $NB(i-1)$, named h_i^1, and moves to it. If the agent dies or works in an abnormal way (due to a system fault) on h_i^1, the surveillant node h_{i-1} will detect it. A replica agent is generated on h_{i-1} by the copy of the previous agent, takes over the execution, selects another node in $NB(i-1)$, named h_i^2, and moves to it. This course will continue until the agent finally completes its execution in one of the neighboring node of h_{i-1}. A timer may be used to prevent the agent from executing for too long. An agent will die whenever its allowed execution time expires.

When a frontal failure occurs, the agent can not enter the selected node. Then, it can return to the node it just left. Since a frontal failure will not change the agent's state, the agent is reliable and reusable. So, if the agent can not enter h_i^1 due to a frontal failure, it should go back to h_{i-1}, select another node in $NB(i-1)$, named h_i^2, and move to it.

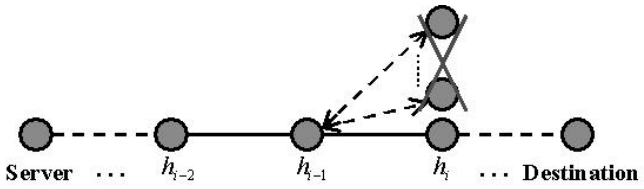


Figure 3.1: An Example of Agent's Execution

Figure 3.3 shows an example of an agent's execution. The neighboring node of h_{i-1} on which the agent finishes its execution is called h_i. In the worst case, a mobile agent will be blocked in set $NB(i-1)$ if all the neighboring nodes of node h_{i-1} are down, or if node h_{i-1} is suddenly down after the agent moves to a neighboring node which has failed.

3.4 Analysis

Most Internet users want the network to have a high fault-tolerant capability and high work efficiency. That is, the assigned task should be finished correctly and promptly, with an acceptable computational resource consumption. In this section, we analyze some important parameters that describe the statistic behavior of mobile agents and the performance of our model, including migration time, life expectation, and population distribution. Since mobile agents react in different way to different kind of failures, which results in different network performance, we will analyze for each kind of failure, respectively.

3.4.1 Migration Time

In an asynchronous distributed system, e.g., the Internet, there are no bounds on migration delays of messages and no relative process speeds. Therefore, when a mobile agent is blocked because of a failure in an asynchronous distributed system, the agent's owner cannot correctly determine whether the agent has failed or is merely running slowly [38].

On-Site Failures

In fact, it is a family of agents, including the agent dispatched by the server and its replicas, that completes a specified task. In this context, each replica is also called "the agent" since an agent and its replicas will not execute synchronously. As shown in Figure 3.2, after

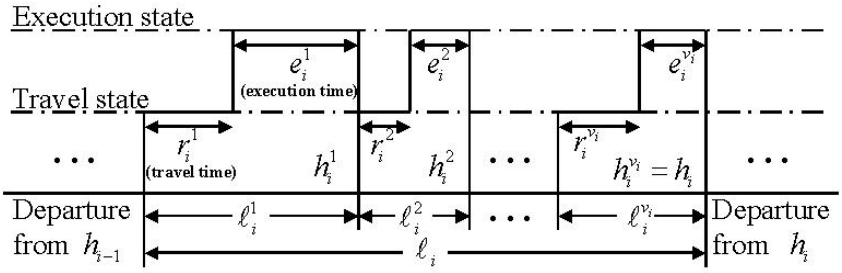


Figure 3.2: Agent's Execution Model for On-Site Failures

finishing execution in node h_{i-1}, "the agent" travels to the firstly selected neighboring node h_i^1 and executes on it. If the agent's execution on h_i^1 fails due to a failure, "the agent" travels to the secondly selected neighboring node h_i^2 and executes on it. This process will proceed repeatedly until the agent finally completes its execution. Thus, the migrating activity between two nodes on the agent's itinerary can be decomposed into two components: travelling and executing. Therefore the migration time between two neighboring set can also be decomposed into travel time and execution time. When the probability distributions of travel time and execution time are known, the probability distributions of migration times is calculable. The migration time of an agent between node h_{i-1} and the node h_i^j, denoted by ℓ_i^j, is defined as $\ell_i^j = r_i^j + e_i^j$ where r_i^j indicates the travel time from node h_{i-1} to node h_i^j and e_i^j is the execution time on the node h_i^j. Suppose that the migration time probability density function (pdf) is given by $\ell(t)$, the travel time pdf is given by $r(t)$, and the execution time pdf is given by $e(t)$. Since r_i^j and e_i^j are two independent random variables, the pdf of ℓ_i^j, $\ell(t)$, can be expressed as the convolution of $r(t)$ and $e(t)$, i.e., $\ell(t) = r(t) * e(t)$. Taking Laplace transform on both side, we have

$$L^*(s) = R^*(s) \cdot E^*(s). \tag{3.1}$$

Here,

$$L^* = E\left(e^{-s\ell_i^j}\right) = \int_0^\infty l(t)e^{-st}dt, \tag{3.2}$$

$$R^* = E\left(e^{-sr_i^j}\right) = \int_0^\infty r(t)e^{-st}dt, \tag{3.3}$$

$$E^* = E\left(e^{-se_i^j}\right) = \int_0^\infty e(t)e^{-st}dt. \tag{3.4}$$

As shown in Figure 3.2, the migration time of the agent between two neighboring sets, i.e., the migration time for moving from h_{i-1} to h_i, denoted by ℓ_i, can be expressed as $\ell_i = \sum_{j=1}^{v_i} \ell_i^j$ where v_i is the number of nodes selected by the agent in $NB(i-1)$. It is easy to see that $h_i^{v_i} = h_i$. Let $f(t)$ be the pdf of the migration time ℓ_i and $F^*(s)$ be the Laplace transform of $f(t)$, the distribution of the migration time ℓ_i satisfies

$$\begin{aligned} F^*(s) &= E\left(e^{-s\ell_i}\right) = \int_0^\infty f(t)e^{-st}dt \\ &= \sum_{l=1}^{d_i} E\left(e^{-s\ell_i}\middle|\, v_i = l\right) P\left(v_i = l\right) = \sum_{l=1}^{d_i} E\left[e^{\left(-s\cdot\sum_{j=1}^{v_i}\ell_i^j\right)}\middle|\, v_i = l\right] P\left(v_i = l\right) \\ &= \sum_{l=1}^{d_i} \left[E\left(e^{-s\ell_i^j}\right)\right]^\ell P\left(v_i = l\right) = \sum_{l=1}^{d_i} \left[L^*(s)\right]^l P\left(v_i = l\right), \end{aligned} \tag{3.5}$$

where d_i denotes the number of nodes in $NB(i-1)$.

Let X_i^j be a 0 or 1 valued random variable, with probability $\rho_o = P(X_i^j = 1)$ for $i = 1, 2, \cdots$, $j = 1, 2, \cdots, d$. The event $\{X_i^j = 1\}$ indicates that the agent can not complete its execution at the j-th selected node in $NB(i-1)$. Then the parameter ρ_o measures the incidence of on-site failure in the network. Based on the fact that $\{v_i = l\}$ is equivalent to $\{X_i^l = 0, X_i^{l-1} = \cdots = X_i^1 = 1\}$, we have $P\{v_i = l\} = (1-\rho_o)\rho_o^{l-1}$ for $1 \le l \le d_i$, which is a geometric distribution. Therefore, we have

$$F^*(s) = \sum_{l=1}^{d_i} \left[L^*(s)\right]^l (1-\rho_o)\rho_o^{l-1} = \frac{(1-\rho_o)L^*(s)}{1-\rho_o\cdot L^*(s)}\left[1-\left(\rho_o\cdot L^*(s)\right)^{d_i}\right]. \tag{3.6}$$

Thus, the probability distribution of migration time can be easily gained by a reverse Laplace transform. Due to the fact that

$$\left.\frac{\mathrm{d}F^*(s)}{\mathrm{d}s}\right|_{s=0} = \left.\frac{\mathrm{d}\left[\int_0^\infty f(t)e^{-st}\mathrm{d}t\right]}{\mathrm{d}s}\right|_{s=0} = -\int_0^\infty t\cdot f(t)\mathrm{d}t = -E(\ell_i), \tag{3.7}$$

$$\left.\frac{\mathrm{d}^2F^*(s)}{\mathrm{d}s^2}\right|_{s=0} = \int_0^\infty t^2\cdot f(t)\mathrm{d}t = E(\ell_i^2), \tag{3.8}$$

the mean and the variance of migration time are given as follows:

$$E(\ell_i) = -\left.\frac{\mathrm{d}F^*(s)}{\mathrm{d}s}\right|_{s=0}, \tag{3.9}$$

$$D(\ell_i) = E(\ell_i^2) - [E(\ell_i)]^2 = \left.\frac{\mathrm{d}^2F^*(s)}{\mathrm{d}s^2}\right|_{s=0} - \left[\left.\frac{\mathrm{d}F^*(s)}{\mathrm{d}s}\right|_{s=0}\right]^2, \tag{3.10}$$

where $D(\ell_i)$ is the variance of ℓ_i.

In case that the explicit distribution of execution time and travel time are unknown, we still can estimate the mean of migration time if the means of execution time and travel time are available. Since $\ell_i^j = r_i^j + e_i^j$ is independent of v_i for $1 \le j \le v_i$, the average migration time of the agent in the neighboring set $NB(i-1)$ of node h_{i-1} satisfies the following equation

$$E(\ell_i) = E[E(\ell_i|v_i)] = E\left[E\left(\sum_{j=1}^{v_i}\ell_i^j\middle|\, v_i\right)\right] = E(v_i)E(\ell_i^j) = E(v_i)(r+e). \tag{3.11}$$

Here $r = E(r_i^j)$ and $e = E(e_i^j)$. Denoting the average connectivity of the network by d, we have

$$E(v_i) = \sum_{j=1}^{d} j \cdot P(v_i = j) = \frac{1-\rho_o^d}{1-\rho_o} - d\rho_o^d. \tag{3.12}$$

Thus, the average migration time of an agent satisfies:

$$E(\ell_i) = (r+e)\left[\frac{1-\rho_o^d}{1-\rho_o} - d\rho_o^d\right]. \tag{3.13}$$

Frontal Failures

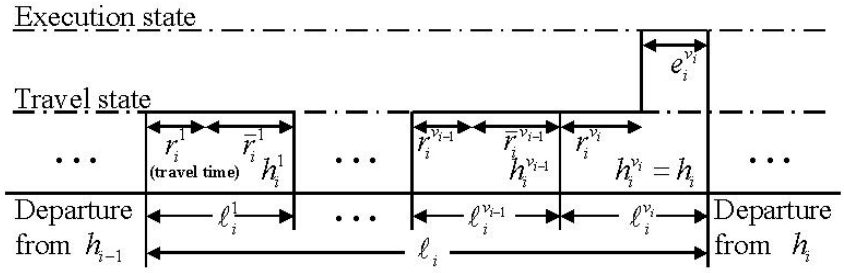


Figure 3.3: Agent's Execution Model for Frontal Failures

In the on-site failure case, the agent can enter any selected neighboring node and execute on it (either fails or successes). On the contrary, the agent can not enter any selected neighboring node but the last one in the case of frontal failures. As a result, the distribution of travel time from failed neighboring node to the previous node must be taken into consideration. As shown in Figure 3.3, the migration time of a mobile agent from node h_{i-1} to node h_i^j, denoted by ℓ_i^j, is defined as:

$$\ell_i^j = \begin{cases} r_i^j + e_i^j & \text{when } j = v_i; \\ r_i^j + \bar{r}_i^j & \text{otherwise,} \end{cases} \tag{3.14}$$

where $\bar{r}_i^j$ is the travel time from node h_i^j back to node h_{i-1} with pdf $\bar{r}(t)$. For $j = 1, \cdots, v_i - 1$, the migration time pdf, $\ell(t)$, is $\ell(t) = r(t) * \bar{r}(t)$ and the Laplace transform of $\ell(t)$ is $L^*(s) = R^*(s) \cdot \bar{R}^*(s)$; while for $j = v_i$, $\ell(t) = r(t) * e(t)$ and $L^*(s) = R^*(s) \cdot E^*(s)$. Here, $\bar{R}^*(s) = \int_0^\infty \bar{r}(t) e^{-st} dt$. Again, let $F^*(s)$ be the Laplace transform of the migration time $f(t)$, we have

$$F^*(s) = \sum_{l=1}^{d_i} E\left[e^{-s\ell_i} \middle| v_i = l\right] P(v_i = l) = \sum_{l=1}^{d_i} [R^*(s) \cdot \bar{R}^*(s)]^{l-1} \cdot [R^*(s) \cdot E^*(s)] P(v_i = l) \tag{3.15}$$

e_i^j is the execution time at node h_i^j, and v_i is the number of nodes selected by the agent in $NB(i-1)$. Then, the migration time of an agent in $NB(i-1)$, denoted by t_i, can be expressed as $t_k = \sum_{j=1}^{v_i} t_i^j = \sum_{j=1}^{v_i} r_i^j + \sum_{j=1}^{v_i - 1} r_i^{-j} + e_i^{v_i}$. Therefore, similar to that in Section 4.1.2, we have

$$E(t_i) = E(v_i)E(r_i^j) + [E(v_i) - 1]E(r_i^{-j}) + E(e_i^{v_i})$$

For descriptive clarity, two scenarios are considered in the following analysis, namely optimistic and pessimistic. We call a system optimistic if node h_{i-1} will not crash before the agent enters node h_i for $i = 1, 2, \cdots$. Otherwise, it is called pessimistic.

• Optimistic Case

In this case, node h_{i-1} will not crash before the agent enters node h_i. Let Y_i^j be a sequence of 0 or 1 valued random variables, with probability $\rho_f = P(Y_i^j = 1)$ for $i = 1, 2, \cdots, m$ and $j = 1, 2, \cdots, d_i$. Let the event $\{Y_i^j = 1\}$ indicate that the j-th node is down, then the parameter ρ_f measures the incidence of frontal failure in the network. The probability $P\{v_i = l\}$, that the agent enters the j-th selected node in $NB(i-1)$, is $P(v_i = l) = P\{Y_i^l = 0, Y_i^{l-1} = \cdots = Y_i^1 = 1\} = (1-\rho_f)\rho_f^{l-1}$ for $1 \le l \le d_i$. and the probability that the agent can not enter any neighboring node is q^d. Therefore, $E(v_i)$, the average number of nodes selected by the agent in $NB(i-1)$, can be evaluated as follows:

$$E(v_i) = \sum_{j=1}^{d} j \cdot P\{v_i = j\} = \frac{1-q^d}{1-q} - dq^d$$

Hence, we have

$$\begin{aligned} E(t_i) &= \left(\frac{1-q^d}{1-q} - dq^d\right)[E(r_i^j) + E(r_i^{-j})] \\ &\quad -E(r_i^{-j}) + E(e_i^{v_i}) \\ &= 2r\left(\frac{1-q^d}{1-q} - dq^d\right) - r + e \end{aligned}$$

It is easy to see that the average migration time in the optimistic case is decided by the system's connectivity and the error rate. Therefore, we have

$$\begin{aligned} F^*(s) &= \sum_{l=1}^{d_i} [R^*(s)]^l \cdot [\bar{R}^*(s)]^{l-1} \cdot E^*(s)(1-\rho_f)\rho_f^{l-1} \\ &= \frac{(1-\rho_f)R^*(s) \cdot E^*(s)}{1 - [\rho_f \cdot R^*(s) \cdot \bar{R}^*(s)]} \cdot \left[1 - \left(\rho_f \cdot R^*(s) \cdot \bar{R}^*(s)\right)^{d_i}\right]. \end{aligned} \tag{3.16}$$

The mean and the variance of migration time can be calculated similarly to (3.9) and (3.10). Similar to (3.11), when only the means of the travel time and the execution time are known, the average migration time satisfies $E(\ell_i) = E(v_i)E(r_i^j) + [E(v_i) - 1]E(\bar{r}_i^j) + E(e_i^{v_i})$ due to the fact that $\ell_k = \sum_{j=1}^{v_i} \ell_i^j = \sum_{j=1}^{v_i} r_i^j + \sum_{j=1}^{v_{i-1}} \bar{r}_i^j + e_i^{v_i}$. Here, we assume that $\bar{r}_i^j$ satisfies the same distribution with r_i^j and therefore the mean is same. Since $P(v_i = j) = (1-\rho_f)\rho_f^{j-1}$, if the average connectivity of the network, d, is available, we have $E(v_i) = \sum_{j=1}^{d} j \cdot P(v_i = j) = \frac{1-\rho_f^d}{1-\rho_f} - d\rho_f^d$. Thus, we have

$$\begin{aligned} E(\ell_i) &= \left(\frac{1-\rho_f^d}{1-\rho_f} - d\rho_f^d\right)[E(r_i^j) + E(\bar{r}_i^j)] - E(\bar{r}_i^j) + E(e_i^{v_i}) \\ &= (r + \bar{r})\left(\frac{1-\rho_f^d}{1-\rho_f} - d\rho_f^d\right) - \bar{r} + e, \end{aligned} \tag{3.17}$$

where $\bar{r} = E(\bar{r}_i^j)$.

• Pessimistic Case

In the pessimistic case, node h_{i-1} may fail before the agent come back from a failed neighboring node. Let the event $\{Y_i^0 = 1\}$ indicate that node h_{i-1} is down, the event $\{v_i = l\}$ is equivalent to $\{Y_i^1 = 1, Y_i^0 = 0, Y_i^2 = 1, \cdots, Y_i^{l-1} = 1, Y_i^0 = 0, Y_i^l = 0\}$. Thus, the probability $P\{v_i = l\}$ equals $P\{v_i = j\} = [\rho_f(1-\rho_f)]^{j-1}(1-\rho_f)$. Therefore, we have

$$\begin{aligned} F^*(s) &= \sum_{l=1}^{d_i} [R^*(s)]^l \cdot [\bar{R}^*(s)]^{l-1} \cdot E^*(s)] \cdot (1-\rho_f)[\rho_f(1-\rho_f)]^{l-1} \\ &= \frac{(1-\rho_f)R^*(s) \cdot E^*(s)}{1-\rho_f(1-\rho_f) \cdot R^*(s) \cdot \bar{R}^*(s)} \cdot \left[1 - \left(\rho_f(1-\rho_f) \cdot R^*(s) \cdot \bar{R}^*(s)\right)^{d_i}\right]. \end{aligned} \tag{3.18}$$

If the means of the travel time and the execution time, r, $\bar{r}$, and e, are available, we have

$$\begin{aligned} E(v_i) &= \sum_{j=1}^{d} j \cdot P(v_i = j) = \sum_{j=1}^{d} j \cdot x^{j-1}(1-\rho_f) \\ &= \frac{1-\rho_f}{(1-x)^2}\left[(1 - x^d - (1-x)dx^d)\right] \end{aligned} \tag{3.19}$$

where $x = \rho_f(1-\rho_f)$ and

$$E(\ell_i) = \frac{(r+\bar{r})(1-\rho_f)}{1-x}\left(\frac{1-x^d}{1-x} - dx^d\right) - \bar{r} + e. \tag{3.20}$$

For $j = 1, \cdots, d-1$, the probability that the agent will enter the j-th selected node is $[q(1-q)]^{j-1}(1-q)$ and the probability that the agent will then die is $[q(1-q)]^{j-1}q^2$ since both the j-th selected node and node h_i are down. Thus, the total probability that an agent will select exactly j nodes in the set $NB(i)$ is $[q(1-q)]^{j-1}[1-q(1-q)]$, the sum of these two probabilities. When $j = d$, the probability that the agent will enter the last selected node is $[q(1-q)]^{d-1}(1-q)$. If the agent can not enter the d-th selected node, it will die whether the node h_i is up or down, i.e., the corresponding probability that the agent will die is $[q(1-q)]^{j-1}q$. Similarly, the probability that an agent will select all the d nodes in set $NB(i)$ equals to $[q(1-q)]^{d-1}$. Therefore, the average number of nodes the agent will selected can be expressed as

$$\begin{aligned} E[v_i] &= \sum_{j=1}^{d-1} j(1-x)x^{j-1} \\ &= \frac{1 - x^d - dx^{d-1}(1-x)}{1-x} \\ &= \frac{1-x^d}{1-x} - d_i x^{d-1} \end{aligned}$$

where $x = q(1-q)$. Thus, the average migration time in the pessimistic case satisfies

$$E(t_i) = 2r\frac{1-x^d}{1-x} - r + e - 2rdx^{d-1}$$

It is easy to see that the average migration time in the pessimistic case is also decided by the system's connectivity and the error rate.

3.4.2 Life Expectancy

An agent's life expectancy is another important factor for estimating work efficiency. It is also important for defining the maximum number of hops a mobile agent can make in order to keep the agents population in the network within the system's limit. As shown

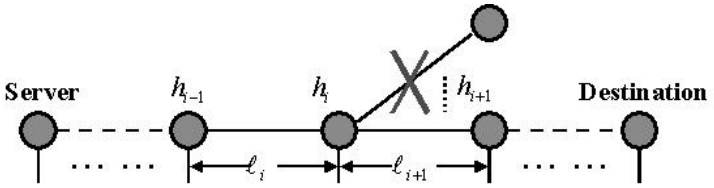


Figure 3.4: Life Expectancy of a Mobile Agent

in Figure 3.4, the life expectancy of a mobile agent, T, is defined as $T = \sum_{i=1}^{N} \ell_i$ where $N \geq 1$ is the total number of neighboring sets visited by the mobile agent and ℓ_i indicates the migration time in $NB(i-1)$. In order to obtain the distribution of the life expectancy T, taking Laplace transform on T, we have

$$\begin{aligned} G^*(s) &= E[e^{-sT}] = E[e^{-s(\ell_1+\cdots+\ell_N)}] \\ &= \sum_{l=1}^{\infty} E[e^{-sT}|N=l]P(N=l) = \sum_{l=1}^{\infty}[F^*(s)]^l \cdot P(N=l). \end{aligned} \tag{3.21}$$

Suppose the success probability that the agent reaches its destination node equals u, i.e.,

$$P(\text{agent reaches its destination}) = u, \tag{3.22}$$

then the probability $P(N=l)$ equals $(1-u)^{n-1}u$. Therefore, the Laplace transform of the life expectancy is

$$G^*(s) = \sum_{l=1}^{\infty}[F^*(s)]^l \cdot (1-u)^{l-1}u = \frac{u \cdot F^*(s)}{1-(1-u)F^*(s)}. \tag{3.23}$$

Similar to the derivation of (3.9) and (3.10), the mean and variance of life expectancy satisfies

$$E(T) = -\left.\frac{\mathrm{d}G^*(s)}{\mathrm{d}s}\right|_{s=0}, \tag{3.24}$$

$$D(T) = \left.\frac{\mathrm{d}^2G^*(s)}{\mathrm{d}s^2}\right|_{s=0} - \left[\left.\frac{\mathrm{d}G^*(s)}{\mathrm{d}s}\right|_{s=0}\right]^2. \tag{3.25}$$

Thus, we can calculate the probability distribution, its mean and variance for both on-site failure and frontal failure by substituting the value of F^* for each case in the upper formulas, respectively.

In case that we do not know the probability distribution of travel time and execution time, but only know their means, we can calculate the mean of life expectancy as follows. Since N and ℓ_i are independent of each other and ℓ_i satisfies the same probability distribution for any i, we have

$$E(T) = E[E(T|N)] = E\left[E\left(\sum_{i=1}^{N}\ell_i \middle| N\right)\right] = E(N)\cdot E(\ell_i). \tag{3.26}$$

When the average connectivity of the network, d, is available, $E(N)$ can be evaluated as follows. First, we define a binary random variable s_i as follows:

$$s_i = \begin{cases} 1 & \text{agent completes its execution in } N(i-1), \\ 0 & \text{otherwise.} \end{cases} \tag{3.27}$$

Then the probability that an agent can complete its execution in set $NB(i-1)$ is $P\{s_i = 1\} = \sum_{j=1}^{d} P\{v_i = j\}$. Since $\{N = l\}$ indicates $\{s_l = 0, s_{l-1} = \cdots = s_1 = 1\}$ and s_i are independent of each other, the probability that the agent will visit exactly l sets satisfies

$$P\{N = l\} = P\{s_l = 0\} \cdot \prod_{k=1}^{l-1} P\{s_k = l\}. \tag{3.28}$$

Thus, from the value of $P(v_i = j)$, the value of $E(N)$ can be evaluated from $E(N) = \sum_{l=1}^{\infty} l \cdot P\{N = l\}$ and the value of $E(T)$ can be evaluated from $E(T) = E(N)E(\ell_i)$.

On-Site Failures

Since $P\{v_i = j\} = (1-\rho_o)\rho_o^{j-1}$, the probability that an agent can complete its execution in set $NB(i-1)$ is equal to $P\{s_i = 1\} = \sum_{j=1}^{d} P\{v_i = j\} = 1 - \rho_o^d$ and $P\{s_i = 0\} = \rho_o^d$. Therefore, we have $P\{N = l\} = \rho_o^d \cdot \left(1 - \rho_o^d\right)^{l-1}$ and $E(N) = \rho_o^{-d}$. Thus, the average life expectancy of a mobile agent equals

$$E[T] = (r + e)\left[\frac{1 - \rho_o^d}{(1-\rho_o)\rho_o^d} - d\right]. \tag{3.29}$$

Frontal Failures

• ***Optimistic Case:*** Since $P(v_i = l) = (1 - p_f)\, p_f^{l-1}$, we have $P\{s_i = 1\} = 1 - \rho_f^d$. Therefore, $P\{N = l\} = \rho_f^d \left(1 - \rho_f^d\right)^{l-1}$ and $E(N) = \rho_f^{-d}$. The average life expectancy in the optimistic case satisfies

$$E(T) = (r + \bar{r})\left[\frac{1 - \rho_f^d}{(1-\rho_f)\rho_f^d} - d\right] - (\bar{r} - e)\rho_f^{-d}. \tag{3.30}$$

•***Pessimistic Case:***

By the result that $P\{v_i = j\} = (1-\rho_f)[\rho_f(1-\rho_f)]^{j-1} = (1-\rho_f)x^{j-1}$, we have $P\{s_i = 1\} = (1-\rho_f)\frac{1-x^d}{1-x} = y$, $P\{N = l\} = (1-y)y^{l-1}$, and $E(N) = 1/(1-y)$. Therefore, we have

$$E[T] = \frac{(r + \bar{r})(1-\rho_f)}{(1-x)(1-y)}\left[\frac{1-x^d}{1-x} - dx^d\right] - \frac{\bar{r} - e}{1-y}. \tag{3.31}$$

3.4.3 Population Distribution

In an agent-driven system, agents must be generated and dispatched frequently. Thus, they will hold a certain amount of resources. This section analyzes the population distribution of mobile agents, which reflects the occupation status of computational resources.

On-Site Failures

Mark the nodes in the system by $i = 1, 2, \cdots$ and define $r_j(t)$ as the number of requests received by the j-th node at time t. If an agent on the i-th node has finished its execution at time t, it will select a neighboring node (say, the j-th node) and move to it. At time $t + r$, it reaches the j-th node; at time $t + r + e$, it will either finish its execution in the j-th node, or fail. If the agent fails in the j-th node, the i-th node will detect it and generate a new agent immediately (at time $t + r + e$). Therefore, the agents in a node can be distinguished into three kinds: new generated by requests, old came from other nodes, or new generated for failed agents. Denote the number of agents initially in the j-th node by $p_j(0)$, the number of agents in the j-th node at time t, denoted by $p_j(t)$, can be formulized as follows:

$$p_j(t) = \begin{cases} r_j(t) & 0 \le t \le r; \\ r_j(t) + \sum\limits_{i \in NB(j)} \sum\limits_{k=1}^{p_i(t-r)} \delta_{kij}(t-r) & r \le t \le r+e; \\ r_j(t) + \sum\limits_{l=1}^{p_j(t-r-e)} \theta_{jl}(t-r-e) + \sum\limits_{i \in NB(j)} \sum\limits_{k=1}^{p_i(t-r)} \delta_{kij}(t-r) & t \ge r+e, \end{cases} \tag{3.32}$$

where $r_j(t)$ is the number of requests received by the j-th node at time t, $\theta_{jl}(t)$ is a binary random variable of the l-th agent in the j-th node at time t:

$$\theta_{jl}(t) = \begin{cases} 1 & \text{if the agent fails at time } t + r + e; \\ 0 & \text{otherwise.} \end{cases} \tag{3.33}$$

and $\delta_{kij}(t)$ is a binary random variable of the k-th agent in node i:

$$\delta_{kij}(t) = \begin{cases} 1 & \text{if the agent enters the } j\text{-th nodeat time } t + r; \\ 0 & \text{otherwise.} \end{cases} \tag{3.34}$$

The sum $\sum_{l=1}^{p_j(t-r-e)} \theta_{jl}(t-r-e)$ is the number of agents generated in the j-th node at time t for those agents which moved out from the j-th node at time $t-r-e$ and reached failed nodes at time t. The sum $\sum_{k=1}^{p_{i-1}(t-1)} \delta_{kij}(t-1)$ is the number of agents in the j-th node that came from the i-th node. Since whether a node's failure in the network is independent of other nodes and a probability ρ_o, the probability $P\{\theta_{jl}(t) = 1\} = \rho_o$. Furthermore, the probability $P\{\delta_{kij}(t) = 1\} = (1 - \rho_o)/d_i$ due to the fact that an agent can enter its selected node with probability $1 - \rho_o$ and all the neighboring nodes of the i-th node have the same probability of being selected by all the agents in the i-th node as a whole. Taking expectations on both sides of (3.32), the average number of agents in the j-th node at time t can be expressed as:

$$\hat{p}_j(t) = \begin{cases} \hat{r}_j(t) & 0 \le t \le r; \\ \hat{r}_j(t) + \sum\limits_{i \in NB(j)} \dfrac{1-\rho_o}{d_i} \hat{p}_i(t-r) & r \le t \le r+e; \\ \hat{r}_j(t) + \rho_o \cdot \hat{p}_j(t-r-e) + \sum\limits_{i \in NB(j)} \dfrac{1-\rho_o}{d_i} \hat{p}_i(t-r) & t \ge r+e. \end{cases} \tag{3.35}$$

where $\hat{p}_j(t)$ and $\hat{p}_j(t)$ $(j = 1, 2, \cdots)$ are the average number of $p_j(t)$ and $r_j(t)$ respectively. The above equation can be expressed in matrix format as follows:

$$\overrightarrow{p}(t) = \begin{cases} \overrightarrow{r}(t) & 0 \le t \le r; \\ \overrightarrow{r}(t) + A\overrightarrow{p}(t-r) & r \le t \le r+e; \\ \overrightarrow{r}(t) + \rho_o \cdot \overrightarrow{p}(t-r-e) + A\overrightarrow{p}(t-r) & t \ge r+e, \end{cases} \tag{3.36}$$

where $\overrightarrow{p}(t)$ and $\overrightarrow{r}(t)$ are column vectors defined as $(\hat{p}_1(t), \hat{p}_2(t), \cdots)^T$ and $(\hat{r}_1(t), \hat{r}_2(t), \cdots)^T$, respectively. A is a coefficient matrix with elements

$$a_{ji} = \begin{cases} (1-\rho_o)/d_i & \text{if } j \in NB(i) \text{ and } j \ne i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.37}$$

Frontal Failures

Let $p_j(t)$ be the number of agents running in the j-th node at time t, the number of agents running in the i-th node can be given by the following equation:

$$p_i(t) = \sum_{k=0}^{d_i} \sum_{l=1}^{p_i(t)} \delta_{ilk}(t), \tag{3.38}$$

where δ_{ilk} indicates the situation of the l-th agent running on the i-th at time t, i.e.,

$$\delta_{ilk}(t) = \begin{cases} 1 & \text{if it is the } k\text{-th time that the } l\text{-th agentreturns back to the } i\text{-th node;} \\ 0 & \text{otherwise.} \end{cases} \tag{3.39}$$

It is easy to see that $\sum_{k=0}^{d_i-1} \delta_{ilk}(t) = 1$ and $\sum_{l=1}^{p_i(t)} \delta_{ilk}$ is equal to the total number of agents running on the i-th node that come back at the k-th time. Especially, $\delta_{il0} = 1$ implies that the l-th agent never moves out from the i-th node. These agents can be further subdivided into two kinds: newly generated in the i-th node or came from neighboring nodes. Let $g_i(t)$ be the number of agents newly generated in the i-th node at time t and $p_{ij}(t)$ be the number of agents running on the i-th node which come from the j-th node at time t. Then, the dynamical change of the agents in the network can be given by the following stochastic equation:

$$\begin{aligned} p_i(t) &= \sum_{l=1}^{p_i(t)} \delta_{il0}(t) + \sum_{k=1}^{d_i} \sum_{l=1}^{p_i(t)} \delta_{ilk}(t) = g_i(t) + \sum_{j \in NB(i)} p_{ij}(t) + \sum_{k=1}^{d_i} \sum_{l=1}^{p_i(t)} \delta_{ilk}(t) \\ &= A_i(t) + B_i(t) + C_i(t). \end{aligned} \tag{3.40}$$

Suppose that $r_i(t-1)$ is the number of requests received by the j-th node at time $t-1$, we have $g_i(t) = r_i(t-1)$, since an agent will be generated for each request. As the error rate of each neighboring node is same, all the nodes in set $NB(i)$ have the same probability to be selected. Therefore, the average number of $p_{ij}(t)$ equals $(1-\rho_f)p_j(t-r)/d_j$. Since each time an agent returns to the previous node cost $2r$ time, $\delta_{ilk}(t) = 1$ indicates that the l-th agent running on the i-th node at time t must come from other nodes or be generated in the i-th node at time $t - 2kr$. Therefore, the average number of $\sum_{l=1}^{p_i(t)} \delta_{ilk}(t)$ equals $[A_i(t) + B_i(t)]P\{\delta_{ilk}(t) = 1\}$. Taking expectations on both sides of (3.40), we have

$$E[p_j(t)] = E[A_i(t)] + E[B_i(t)] + E[C_i(t)], \tag{3.41}$$

where

$$E[A_i(t)] = E[r_i(t-1)]; \tag{3.42}$$

$$E[B_i(t)] = \sum_{j \in NB(i)} \frac{1-\rho_f}{d_j} E[p_j(t-r)]; \tag{3.43}$$

$$E[C_i(t)] = \sum_{k=1}^{d_i} E[A_i(t)] P\{\delta_{ilk}(t)=1\} + \sum_{k=1}^{d_j-1} E[B_i(t)] P\{\delta_{ilk}(t)=1\}. \tag{3.44}$$

In compact form, it can be rewritten as follows

$$\begin{aligned} \overrightarrow{p}(t) = \overrightarrow{r}(t-1) + A\overrightarrow{p}(t-r) + \sum_{k=1}^{d_j} \overrightarrow{r}(t-2kr-1) P\{\delta_{ilk}(t)=1\} \\ + \sum_{k=1}^{d_j-1} A\overrightarrow{p}(t-2kr-1) P\{\delta_{ilk}(t)=1\}, \end{aligned} \tag{3.45}$$

where $\overrightarrow{p}(t) = (E[p_1(t)], \cdots, E[p_n(t)])^T$, $\overrightarrow{r}(t) = (E[r_1(t)], \cdots, E[r_n(t)])^T$, and $A = (a_{ij})_{n \times n}$ is an n by n matrix with elements

$$a_{ij} = \begin{cases} (1-\rho_f)/d_j & \text{if } j \in NB(i); \\ 0 & \text{otherwise.} \end{cases} \tag{3.46}$$

• Optimistic Case

In this case, the event $\{\delta_{jlk}(t)=1\}$ is equivalent to $\{X_j^1 = 1, \cdots, X_j^k = 1\}$. Therefore, $P\{\delta_{jlk}(t)=1\} = \rho_f^k$. Thus, the dynamic change on the number of agents running in the network can be expressed as follows

$$\begin{aligned} \overrightarrow{p}(t) &= \overrightarrow{r}(t-1) + A\overrightarrow{p}(t-r) + \sum_{k=1}^{d_j} \rho_f^k \overrightarrow{r}(t-2kr-1) + \sum_{k=1}^{d_j-1} \rho_f^k A\overrightarrow{p}(t-2kr-1) \\ &= \sum_{k=0}^{d_j} \rho_f^k \overrightarrow{r}(t-2kr-1) + \sum_{k=0}^{d_j-1} \rho_f^k A\overrightarrow{p}(t-2kr-1). \end{aligned} \tag{3.47}$$

• Pessimistic Case

Similar to that of optimistic scenario, since the probability $P\{\delta_{jlk}(t)=1\} = P\{X_j^1 = 1, X_j^0 = 0, \cdots, X_j^k = 1, X_j^0 = 0,\} = [(1-\rho_f)\rho_f]^k = x^k$, the dynamic change on the number of agents satisfies

$$\overrightarrow{p}(t) = \sum_{k=0}^{d_j} x^k \overrightarrow{r}(t-2kr-1) + \sum_{k=0}^{d_j-1} x^k A\overrightarrow{p}(t-2kr-1). \tag{3.48}$$

3.4.4 Analysis for Non-Failure Case

Consider the situation in which there is no system failure during the agents' execution. Our model can be redescribed as follows. Once a request is received from a server, an agent is generated and dispatched into the network. Once the agent reaches a node, it executes locally. If the current node is its destination, the agent's trip is finished. Otherwise, after its execution, it selects a neighboring node and moves to it. Since no system failure occurs during its route, the analytical result of migration time, life expectancy, and population distribution will be different from the above analysis.

Migration Time and Life Expectancy

In the non-failure case, an agent will never select a failing node or a node about to fail. In other words, an agent will not die until it finds its destination. This means that the migration time of an agent equals to $r + e$ and the life expectancy of an agent will be infinitely long until it completes all its tasks.

Population Distribution

Under the assumption that nodes will not fail, the analysis of population distribution becomes greatly simplified. Define a binary random variable of the l-th agent in the i-th node as follows:

$$\eta_{kij}(t) = \begin{cases} 1 & \text{if the } k\text{-th agent in the } i\text{-th nodeselects the } j\text{-th node to go;} \\ 0 & \text{otherwise.} \end{cases} \tag{3.49}$$

The dynamic change on the number of agents in the network satisfies

$$p_j(t) = \begin{cases} r_j(t) & 0 \le t \le r; \\ r_j(t) + \sum\limits_{i \in NB(j)} \sum\limits_{k=1}^{p_i(t-r)} \eta_{kij}(t-r) & t \ge r. \end{cases} \tag{3.50}$$

Suppose that an agent will randomly select a neighboring node with probability $P(\eta_{kij}(t) = 1) = P(j|i) = P(i \to j) = \frac{1}{d_i}$ and taking expectations on both side of the upper equation, we have

$$\hat{p}_j(t) = \begin{cases} r_j(t) & 0 \le t \le r; \\ r_j(t) + \sum\limits_{i \in NB(j)} \dfrac{p_i(t-r)}{d_i} & t \ge r, \end{cases} \tag{3.51}$$

or in matrix-vector form

$$\hat{\overrightarrow{p}}(t) = \begin{cases} \overrightarrow{r}(t) & 0 \le t \le r; \\ \overrightarrow{r}(t) + A\overrightarrow{p}(t-r) & t \ge r, \end{cases} \tag{3.52}$$

where

$$\overrightarrow{p}(t) = \begin{bmatrix} p_1(t) \\ \cdots \\ p_n(t) \end{bmatrix}, \quad \overrightarrow{r}(t) = \begin{bmatrix} r_1(t) \\ \cdots \\ r_n(t) \end{bmatrix}$$

and $\hat{\overrightarrow{p}}(t) = E[\overrightarrow{p}(t)|\overrightarrow{r}(t), \overrightarrow{p}(t-r)]$. $A = (a_{ji})_{n \times n}$ is an coefficient matrix with element

$$a_{ji} = \begin{cases} 1/d_i & \text{if } j \in NB(i),\ j \ne i; \\ 0 & \text{otherwise.} \end{cases} \tag{3.53}$$

By defining $\hat{\overrightarrow{p}}(t) = E[\overrightarrow{p}(t)|\hat{\overrightarrow{r}}(t), \hat{\overrightarrow{p}}(t-r)]$, the dynamic change on the expected number of agents in the network satisfies

$$\hat{\overrightarrow{p}}(t) = \begin{cases} \hat{\overrightarrow{r}}(t) & 0 \le t \le r; \\ \hat{\overrightarrow{r}}(t) + A\hat{\overrightarrow{p}}(t-r) & t \ge r. \end{cases} \tag{3.54}$$

3.5 Experimental Results

Our mobile agent system consisting of n nodes. Once a request is received in the system, a mobile agent is generated. The average number of requests is assumed to be m. The average execution time and the average migration time on each node are assumed to be e and r, respectively. The target node for the next hop is selected randomly from neighboring nodes. The average connectivity of the system is d. Both the probability of on-site failure and the probability of frontal failure are ρ. When a node fails, all of the agents running on the node fail as well

To measure the influence of system parameters on the network performance, the migration time and the life expectancy are obtained for four cases respectively.
Basic Case: The system environment with the following parameter values is simulated: $n = 40$, $m = 10$,$r = 1$, e=2, $d = 3$, and $\rho = 0.01$.
Case 1: The value of d changes to 5.
Case 2: The value of d changes to 8.
Case 3: The value of d changes to 12.
Cases 1-3 differs only one system parameter value compared to the Basic Case.

Figure 3.5 shows the experimental results. We can see that both the migration time and the life expectancy are very sensitive to the changes of the network connectivity and the probability of failure. The figures in the first column show the results of migration time. Note that there is an inflexion in the figure. The migration time before the inflexion increase when the probability of failure increase, which means that agents need more time to enter the next node when the probability of failure is larger. The migration time after the inflexion is a decreasing function on the probability of failure, which coincides our intuition that if the probability of failure is too large, the agent may fail at the earlier migration. It also can be observed that the migration time is a monotonic increase function on the connectivity. A larger connectivity will result in a longer migration time. The figures in the second column show the results of life expectancy. It is easy to see that the migration time is a monotonic increasing function on the connectivity and a monotonic decreasing function on the probability of failure. The figures in the first row analyze the influence of the connectivity for the on-site failure case, the figures in the second row analyze the influence of the connectivity for the optimistic frontal failure case, and the figures in the third row analyze the influence of the connectivity for the pessimistic frontal failure case. Comparing those figures, we can see that figures for on-site failure case and optimistic frontal failure case are very similar. The difference is determined by the value of e and r. For the pessimistic frontal failure case, the inflexion of the curve for migration time comes earlier. Both the migration time and the life expectancy are greatly smaller than that for the on-site failure case and the optimistic frontal failure case.

3.6 Comparison

Our model effectively combines available techniques and achieves better cost-effectiveness than existing models by eliminating redundant consumption of computational resources. Table 3.1 shows the comparison among checkpointing approach, replication approach, and our model, where storage consumption indicates the rolled storage during an agent's execution, L is the total nodes from the source to the destination, K is the period to store a checkpoint, m is the number of replication servers per step, communication cost refers to

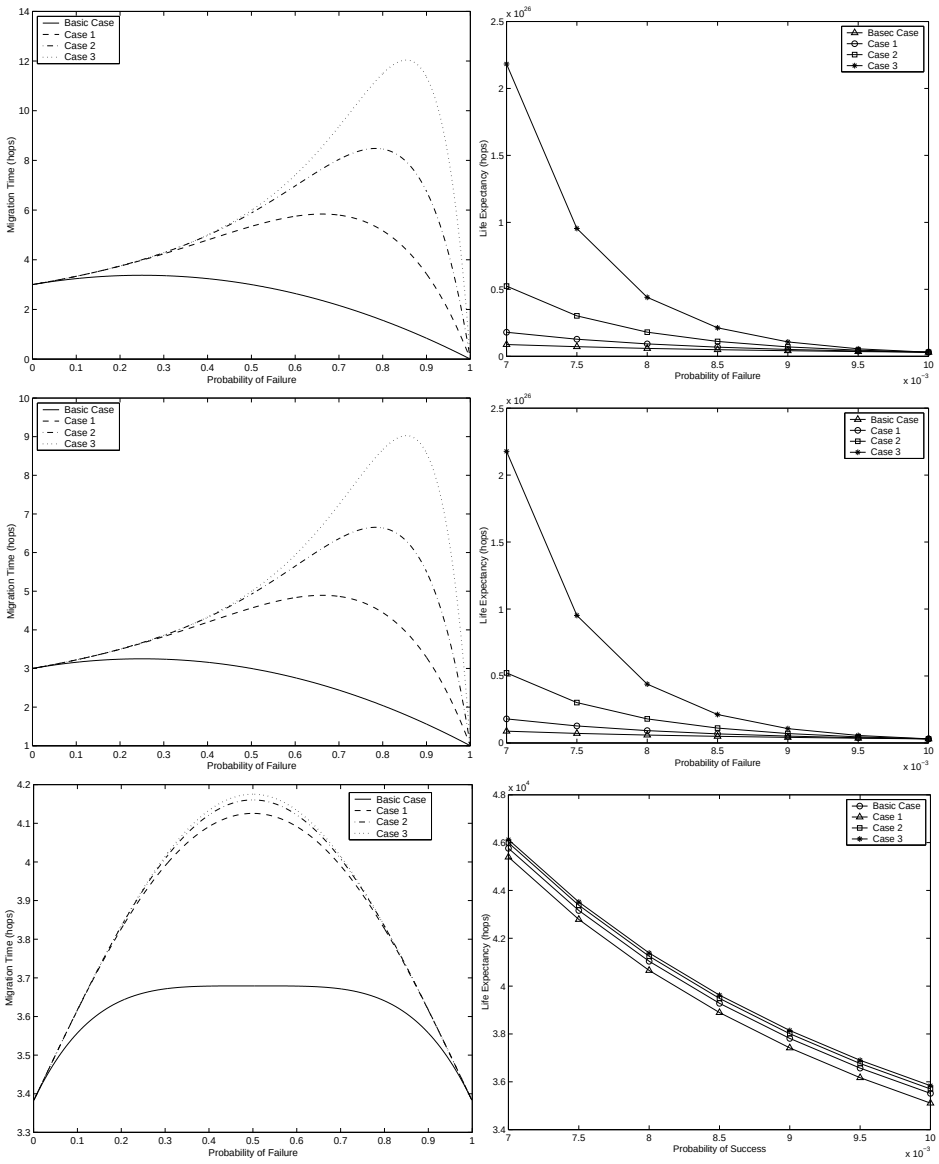


Figure 3.5: Results for Migration Time and Life Expectancy

the number of communicating nodes for one step, and the failure free degree expresses the ability of each approach that how many failures they can tolerate. The checkpoints in the

	Checkpointing	Replication	Our Algorithm
Storage Consumption	$[L/K]+1$	m	2
Communication	2	m	3
Failure Free Degree	1	$m-1$	3

Table 3.1: The comparison among checkpointing, replication, and our algorithm.

checkpointing approach are stored periodically to resume the execution of mobile agents after crash. Even if no failure occurs, multi-node communication must be kept until the agent reaches its destination. This will generate a considerable overhead. Furthermore, unlocking the communication generates additional overhead. On the contrary, our model stores a checkpoint in each node where the agent stays. Once the agent completes its execution at the node, the checkpoint is deleted. During the agent's trip, only 2-node communication is needed. Thus, the overhead in our model is greatly reduced. Moreover, when the executing agent is crashed due to a system failure, our model can resume the agent's execution from the previous step, whereas in the checkpointing approach, the agent will resume its execution from the most recent saved checkpoint. This makes the agent's migration time in our model less than that in the checkpointing approach. In the replication approach, agent replicas are sent to a set of nodes at the same time and keep communications among them. Execution on all nodes will bring more overhead. Even if only one node works at a given time, the multi-node communication overhead occurs at each execution, regardless of the occurrence of failures. In our model, only one replica is stored in the previous node and 2-node communication is maintained during the whole process. Besides, the number of agent/replicas is greatly reduced. These result in a less overhead in our model than in the replication approach.

3.7 Concluding Remarks

The reliable execution of mobile agents is an important design issue for building a mobile agent system and many fault-tolerant execution schemes have been proposed along roughly two approaches: replication and checkpointing. In this chapter, we proposed a new fault-tolerant execution model and analyzed its statistical properties. Our model effectively combines available techniques and achieves better cost-effectiveness than existing models by eliminating redundant consumption of computational resources.

In our model, failures are classified into two classes: on-site failure and frontal failure. This classification captures the intrinsic difference between these two classes of failures which causes different effects on mobile agents' behaviors. An on-site failure is a node's failure during an agent's execution within it. It may block the agent and alter the agent's state. On the other hand, a frontal failure is a node's failure before an agent's entrance to it which blocked the agent's entry, but did not alter the agent's state. For each kind of failures, an exceptional-event handling method is adopted. It can greatly decrease network resources consumption and improve the overall performance during mobile agent's execution.

For the first time, the behaviors of mobile agents in networks that may contain faults are statistically analyzed in a quantitative way, which exploits a new approach to assessing the performance of agent-based systems. Several important parameters on system performance, including migration time, life expectancy, and population distribution of mobile agents, are analyzed. We also considered the performance of our model in reliable networks. The analytical results reveal the theoretical insights into the fault-tolerant execution of mobile agents and show that our model can outperform the existing fault-tolerant models. The analytic method proposed in this chapter may also be applied for performance analysis in other complex systems. Our model is reliable and cost-efficient, which offers us a promising way for designing reliable mobile agent system.

Chapter 4

An Execution Prototype of Mobile Agent-Based Peer-to-Peer Networks

4.1 Introduction

Peer-to-peer networks are one of the trends in the field of internetworking. In p2p-networks, the hosts, or peers, act as client and server. Milojicic et al. [80] lists some characteristics of p2p-system. They are: decentralization; scalability; anonymity; self-organization; cost of ownership; ad-hoc connectivity; performance; security; transparency and usability; fault resilience; and interoperability.

Probably the most significant of these are Decentralization, Scalability and Ad-Hoc connectivity. As the clients act as servers in p2p-systems, there is no need for central management, which in traditional client-server model is done by servers. Scalability is achieved, as hosts can join or leave the network easily without having to register into a database. As hosts can also be up or down at every instant, ad-hoc connectivity is achieved.

Peer-to-peer networks can be divided into pure p2p-networks and hybrid networks. In pure p2p-networks, there is no kind of central servers as in hybrid model where some servers are offered for e.g. locating the resources.

The most known p2p-systems are probably file-sharing networks, such as Napster, kazaa, DC++, Gnutella and Bittorrent. But p2p is also used in e-commerce, distributed computing and in instant messaging (such as MSN Messenger).

It can be seen that in a large communication network such as Internet, agents have to be generated frequently and dispatched to the network. Thus, they will certainly consume a certain amount of bandwidth of each link in the network. If there are too many agents migrating through one or several links at the same time, they will introduce too much transferring overhead to the links. Eventually, these links will be busy and indirectly block the network traffic. Therefore, there is a need of developing routing algorithms that consider about the traffic load. Since the state of different links may change dynamically over time, the agents have to dynamically adapt themselves to the environment, which increases the difficulty for both algorithm design and theoretical analysis. In [19], the network state is monitored by launching an agent at regular intervals from a source to a certain destination. In [22], the agent was enabled to estimate queuing delay without waiting inside data packet queues. In [71], the authors showed that the information needed in [19, 22] for each destination is difficult to obtain in real networks. In [9],

a mechanism of handling routing table entries at the neighbors of crashed routers was proposed which significantly improved the algorithm proposed in [19, 22]. In [16], the authors formulated a method of mobile agent planning, which is analogous to the travelling salesman problem [39] to decide the sequence of nodes to be visited by minimizing the total execution time until the desired information is found.

In this chapter, we propose an agent-based routing algorithm in which the traffic cost for each link is considered. To balance the traffic load on each link, we introduce the maximum entropy theory into our algorithm to find an optimal probability distribution that makes inference on the known traffic information and balances the traffic load. Theoretical analysis shows that our derived probability distribution that an agent on an intermediate node may select a neighboring node and move to satisfies these two requirements. The remainder of this chapter is structured as follows. Section 4.2 introduce 4 most popular P2P systems. Section 4.3 presents the motivation of our research, i.e., why use mobile agents in peer-to-peer networks. Section 4.4 presents our algorithm. Section 4.5 introduces the maximum entropy theory. Section 4.6 provides theoretical analysis, Section 4.7 shows some simulation results, and Section 4.8 gives our conclusions.

4.2 P2P Systems

A variety of different P2P systems has been developed. Below we will take a brief look at the 4 most famous/popular systems.

4.2.1 Napster

Napster was perhaps the first P2P network to gain widespread recognition and use. It gained support due to its ease of use, its accuracy, and of course the large demand that had developed for MP3 music files. Napster itself is a closed application, but there is an open protocol known as OpenNap that is based on Napster [86]. We will refer to both of these collective as Napster.

The architecture of Napster is that it maintains a central database on a server with details of available MP3 music files and their locations [108]. When a user withes to download a file the database is queried. If a match is found the client is given the details of where the file, and downloads it. After a successful download the central database is updated with the client address where a new copy of the file is now located.

So Napster is extremely fast and accurate for locating the correct files due to the central database it maintains. However it is this centrally located database that has led to Napster's legal problems [37], and the rapid loss of users [127]. It is unlikely that any future P2P system would use a similar architecture that leaves it open to similar legal actions.

4.2.2 Gnutella

Gnutella is an open, text based protocol that is designed to allow for the sharing of all file types over the Internet [41]. A computer participating in Gnutella network is acting as both client and server, and hence is called a *servent*. There are many different client applications that support the Gnutella protocol such as BearShare [10] and Limewire [72].

A computer wishing to participate in a Gnutella network does so by contacting another Gnutella servent [25]. The acquisition of this other servent's address is not actually part of the Gnutella protocol. Once connected, servents seek out other servents by sending Pings, which may be responded to with a Pong. When a servent receives a Ping it forwards it to all the servents that it knows of , and sends the resulting Pongs back to the servent that initiated the ping. To search for a file a Query is sent to all known servents. If the servent has a matching file it responds with a Hit, otherwise it forwards the query on to all the servents that it knows of. Once the file has been found it is downloaded outside of the Gnutella network using the Hyper Text Transfer Protocol.

Perhaps the biggest problem with Gnutella is that each servent uses so much bandwidth for sending and receiving Ping/Pong and Query/Hit messages that are not related to it. This is critical considering that these messages grow exponentially. It is also believed that most users on the Gnutella network are not making any files available for sharing, thus making the network very inefficient [2]. Another problem is that each servent only ever sees a relatively small portion of the entire network.

To address some of these problems Clip2 developed a Reflector system [26]. The Reflector is a super peer located between a group of end servents and the rest of the Gnutella network. When servents connect to the reflector an index is created on teh reflector with all the details of shared files on the servent. The reflector thus is able to shield the connected end servents from all Ping/Pong and Query/Hit messages while also providing servents with greater network coverage.

4.2.3 Morpheus/KaZaA

Morpheus and KaZaA [63] are two new file sharing application being developed by FastTrack [36] and based on the same closed protocol. Both are currently attracting quite an amount of interest. They appear to operate in a similar way to a Gnutella network with Reflectors. Peers are self-organizing, meaning that when they detect that they have enough bandwidth, they can take on the responsibilities of acting in a way similar to the Reflectors.

4.2.4 Project Juxtaposition (JXTA)

Sun [120] has entered the Peer-to-Peer market with the recent launch of Project JXTA, an open community layered set of protocols [61]. JXTA differs from other P2P projects for a number of resons [42]. The JXTA protocols are completely hardware and language independent. Therefore two JXTA peers may communicate with each other over Bluetooth without ever needing TCP/IP, unlike any of the previously mentioned systems.

It is envisaged that many peers will not be PC based, but that they will be either enterprize systems or consumer-orientated small, systems such as PDAs. For this reason the protocols have been kept as thin as possible. Also, each peer need not implement every protocol. Project JXTA provides a layer on top of which P2P applications and services can be built, just like TCP/IP does. Finally, all the protocols defined within JXTA are extensible, allowing application builders to use their own implementations. For example some application builders may require weak security, while others may use their own strong security. A complete description of JXTA can be found in [121].

JXTA consists of six protocols and uses a small subset of XML [142]. However, peers need not be aware of this. Indeed, they can use predefined XML messages alone.

As of now JXTA is still being developed, and there are no applications based on it in widespread use yet. Project JXTA seems to be very well designed, but only time will tell how well it will succeed.

4.3 Motivation

Mobile agents reduce the need for bandwidth. Very often peers using a distributed protocol establish a communication channel between themselves, and then perform multiple interactions over this channel. Each of these interactions generates network traffic. Mobile agents allow these interactions to be packaged together, and sent as a discrete piece of network traffic. This then allows all the interactions to take place locally. Mobile agents also encapsulate all the required data within themselves. Therefore when a mobile agent arrives on a computer it has all its data with it, and does not need to communicate with any other computers. In a conventional search protocol all the raw data travels over the network to be processed, even though only a subset of this data may be needed. In this scenario, mobile agents reduce the network traffic by moving the processing to the raw data, instead of moving the raw data to the processing. Finally, mobile agents can be very small in size, but can grow dynamically as they need to accommodate more data.

Mobile agents are asynchronous. Therefore when a mobile agent is dispatched there is no need to wait for it to return. Indeed the original peer does not even need to remain connected to the network while the mobile agents are out. The mobile agents can wait until the original peer is back on the network before attempting to return to it.

Mobile agents are autonomous. This particularly suits peer-to-peer networks, because the mobile agent is learning about the network as it progresses through it. The mobile agent will visit peers that were unknown when it was originally dispatched. At each peer it can make decisions based on its history of visited peers and the current peer.

Information is being disseminated at every peer that the mobile agent visits. Every peer benefits from accepting a visiting mobile agent, because the mobile agent will have either new or more recent information about resources. Also, every mobile agent benefits from visiting a peer because it will learn of either new or updated resources . If the mobile agents do not contain any new information they may be destroyed. Accepting and hosting mobile agents requires the use of physical resources, such as memory and computer cycles. Should thee become critically limited, it is easy for the peer to refuse further requests to accept mobile agents until more physical resources become available.

Mobile agents may easily be cloned and dispatched in different directions. This allows them to function in parallel. Although this causes more mobile agents to be active on the network, it does ensure that the network resource discovery is completed sooner, and therefore the mobile agents spend less time on the network.

A mobile agent based solution is very fault tolerant. Even if some of the mobile agents are destroyed, all surviving ones will have a positive impact. Indeed, the destroyed mobile agents will have benefited every peer up to the point where they were destroyed.

Finally, a mobile agent based solution can be combined with successful features from other P2P based systems to provide an improved final solution.

4.4 Algorithm

Large scale distributed system consists of many different virtual organizations and private networks which often shows social or organizational structure. Therefore they do not naturally support a centralized point of control, which results in that the resource descriptors can't be stored at a single node in the system. Because of this peer-to-peer computing is well suited for this kind of systems. Agent-based computing is used because it offers the desired characteristics to implement a peer-to-peer system.

In this section, we will briefly describe the organizational principles of our model. Our algorithm focuses on how the mobile agents will behave while they roam around the network and can be sketched as follows:

1. The network is organized into peer groups. Note that a peer in a peer group is only only a peer, but also a node in the network. For a specified peer (also called a node), peers in the same peer group consists of its neighborhood and called neighbor nodes of the peer. When a node does not belongs to any peer group, we say the peer group consists of only one node.

2. There are three kinds of agents employed in our algorithm, namely inner agents, outer agents, and worker agents. Inner agents are periodically generated by a peer/node. They roam inside the peer group, collect and disseminate information. Outer agents are generated for when a user's request cannot be completed in the current peer group. They are dispatched into the network, roam among peer groups and search for destination peer group for executing the request. Worker agents are generated after the outer agents find the destination. They carry the requests to be completed to the destination peer group, follow a desired route that the outer agents have explored, and send the results to the users. A peer is known as the "creator peer" for the mobile agents and all future clones of these mobile agents.

3. Each peer periodically sends an inner agent, by broadcasting replicas of it to each neighbor node. For a inner agent, a journey-time is set in terms of the maximum number of peers that may be visited before it must return to its creator peer. Also a branching factor is set, that is used to determine how many times the inner agent may be cloned at any one peer. These two parameters control the depth and breadth of the search, and therefor the maximum number of peers that may be visited.

4. Initially the inner agent is given the addresses of some other peers participating in the network. Typically the number of addresses provided should be small, and should be less than the branching factor. For best results these addresses should come from different sources. This increases the chances that these peers are operating in separate sub networks that are as yet unaware of each other.

5. After arriving at each peer the inner agent decrements its journey-time, and updates the peer with information about its creator peer, and other peers encountered along the route so far. The inner agent also updates itself with information on the current peer. If two inner agents from the same creator peer arrive at the current peer within a preset time period, then the second mobile agent destroys itself. This may lead to to less information being acquired about the network. However, it keeps the information up to date, and yet prevents peers that form cycles from having to deal repeatedly with inner agents from the same creator peer.

6. If the inner agent's journey-time has expired, then it returns to its creator peer, and updates its creator peer with all the information it has collected on its journeys. Otherwise the inner agent clones itself enough times to allow a clone of itself to be sent to each peer that was known to the current peer before the inner agent arrived. By excluding previously known peers, the incidence of revisits is reduced.

7. When a peer as a node in the network receives a request, it firstly check whether the request can be fulfilled inside its peer group. If so, it generates a worker agent and forward it to the suitable site to execute the request. Otherwise, a number of outer agents are generated and launched to nodes outside its neighborhood. For a outer agent, a journey-time is also set in terms of the maximum number of nodes that may be visited before it must return to its creator peer.

8. Each outer agent while travelling, collects and carries path information, and that it leaves, at each node visited, the trip time estimate for reaching its creator node from this node over the incoming link. Once an outer agent reaches a node, the node tells it whether the request can be fulfilled in its peer group. If so, the outer agent goes back to the creator peer along the same route it has explored, updates the routing table along its route, and submits its report of its journey. Thus each node in the network maintains current routing information for reaching nodes outside its neighborhood. This mechanism enables a node to route a data packet (whose destination is beyond the neighborhood of the creator node) along a path toward the destination node.

9. After a certain number of outer agents has come back, the creator peer selects a desirable route based on some rules, generates a worker agent and forward it to the destination together with the request. The worker agent executes the request on the destination node and returns back to the creator peer with the result. Finally, the creator peer submits the result to the user.

Inner agents and outer agents make routing decisions using the specialized routing strategy by selecting the next target to visit in their exploration. The routing strategy is based on the consideration of traffic balance of the network, that is, the next hop for an agent is selected in a probabilistic manner according to the quality measure of the traffic situation. We will introduce our adopted probabilistic in detail in the following.

Let $G = \{V, E\}$ be a graph corresponding to a fixed network, where $V = \{\nu_1, \nu_2, \cdots\}$ is the set of vertices (hosts) and E is the set of edges. We assume that the topology of a network is a connected graph in order to ensure that communication are able to be made between any two host machines. For an inner agent, $NB(i)$ is the set of peers in the peer group of the creator peer ν_i and $|NB(i)|$ is the number of peers in $NB(i)$. For an outer agent, $NB(i)$ is the set of neighboring nodes outside the peer group of the creator peer ν_i and $|NB(i)|$ is the number of nodes in $NB(i)$. Originally, each peer has no information about the traffic situation of its peer group and the network. Therefore, each vertex in set $NB(i)$ has the same probability to be selected by the inner agent as a target to move to, i.e., $1/|NB(i)|$. This uniform probability distribution of agents' neighboring node-selection will be updated with time going. The new probability distribution should satisfies two constrains:

1. It makes inference on all the known traffic information.

2. It is unbiased. That is, the probability should mostly balance the traffic cost on each link.

To find a probability distribution that both makes inference on the known information and approximates to the unbiased (uniform) distribution, We mathematically model the constrains as follows. The effect of the known traffic information on the agent's migrating decision making can be expressed by a min-max problem as follows:

$$\min_{x \in R^n} f_{\max}^{(j)}(x), \tag{4.1}$$

where x is a random variable with n entries, which denotes n items to be considered to the cost of a link. The objective function $f_{\max}^{(j)}(x)$ is the traffic cost function defined as follows:

$$f_{\max}^{(j)}(x) \equiv \max_{i \in NB(j)} \{f_{ji}(x)\}. \tag{4.2}$$

Here, $f_{ji}(x)$ is the traffic cost function from node ν_j to ν_i. Without losing generalization, we assume that functions $f_{ji}(i \in NB(j))$ are differentiable. Obviously, the maximum value function $f_{\max}^{(j)}(x)$ is an undifferentiable function.

At the same time, the unbiased requirement is expressed by the maximum entropy function (as shown in the next section). Solving the combinatorial optimization problem results in a probability distribution that can be expressed as follows:

$$p_{ji} = \frac{\exp\{\theta f_{ji}(x)\}}{\sum_{l \in NB(j)} \exp\{\theta f_{jl}(x)\}},\ j = 1, 2, \cdots; i \in NB(j), \tag{4.3}$$

where p_{ji} is the probability that an agent on node ν_j migrates to node ν_i, $\theta \geq 0$ is a weight coefficient defined according to the effect of the known traffic state of the network.

In Section 4.5, we will detail the reduction and show the rationality of this probability distribution.

4.5 Maximum Entropy Theory

In [110], Shannon first introduced the concept of entropy into informatics as a measurement of uncertainty. Suppose that there are a set of possible events whose probabilities of occurrence are $\lambda_1, \lambda_2, \cdots, \lambda_n$. These probabilities are known but that is all we know concerning which event will occur. Can we find a measure of how much "choice" is involved in the selection of the event or of how uncertain we are of the outcome? Shannon pointed that if there is such a measure, say $H(\lambda_1, \lambda_2, \cdots, \lambda_n)$, it should have the following properties:

1. H should be continuous on λ_i.

2. If all λ_i are equal, i.e., $\lambda_i = 1/n$, then H should be a monotonically increasing function of n.

3. If a choice is broken down into two successive choices, the original H should be the weighted sum of the individual values of H.

In [110], it is proved that the entropy function $H = -k\sum_{i=1}^{n}\lambda_i \ln \lambda_i$ is the only function that can satisfies all the requirements, where k is a positive constant decided by measurement units. Usually, k is set to be 1. The Shannon entropy has the following properties:

1. $H_n(\lambda_1, \lambda_2, \cdots, \lambda_n) \geq 0$;
2. If $\lambda_k = 1$ and $\lambda_i = 0$ $(i = 1, 2, \cdots, n; i \neq k)$, then $H_n(\lambda_1, \lambda_2, \cdots, \lambda_n) = 0$;
3. $H_{n+1}(\lambda_1, \lambda_2, \cdots, \lambda_n, \lambda_{n+1} = 0) = H_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$;
4. $H_n(\lambda_1, \lambda_2, \cdots, \lambda_n) \leq H_n(1/n, 1/n, \cdots, 1/n) = \ln n$;
5. $H_n(\lambda_1, \lambda_2, \cdots, \lambda_n)$ is a symmetrical concave function on all variables.

where $H = -\sum_{i=1}^{n}\lambda_i \ln \lambda_i$.

E. T. Jaynes found that in many probabilistic executions, the resulting probability distribution cannot foreknown; thus, the entropy cannot be calculated. But he also claimed that the probability distribution could be induced by the accumulated test data such as the mean and the variance. In [59], E.T.Jaynes proposed the maximum entropy theory: "in making inference on the basis of partial information we must use that probability distribution which has maximum entropy subject to whatever is known. This is the only unbiased assignment we can make; to use any other would amount to arbitrary assumption of information which by hypothesis we do not have". Notice that "entropy" is a measurement of the degree of uncertainty and the great the entropy's value, the less known information, the maximum entropy theory can be mathematically expressed as follows:

$$\begin{cases} \max & H = -\sum_{i=1}^{N}\lambda_i \ln \lambda_i \\ s.t. & \sum_{i=1}^{N}\lambda_i = 1; \\ & \sum_{i=1}^{N}\lambda_i g_j(x_i) = E\left[g_j\right], j = 1, 2, \cdots, m; \\ & \lambda_i \geq 0, i = 1, 2, \cdots, N, \end{cases} \tag{4.4}$$

where $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_n)$, $g_j(j = 1, 2, \cdots, m)$ is some predefined constrained function, and $E[\cdot]$ is the mean of these constrained function.

Templeman et al. [123] first applied maximum entropy theory to solve optimization problems in which the objective function is unanimously approximated by a smooth one. By solving the resulting problem, an approximate solution of the original problem can be obtained. The purpose of deploying maximum entropy theory in agents' searching process is to find a probability distribution that both satisfies the known routing information and mostly approximate to the unbiased (uniform) distribution.

4.6 Properties of Approximating Function

Let's look at the following Lagrange function:

$$\ell_j(x, p_j) = \sum_{i \in NB(j)} p_{ji} f_{ji}(x) \ \ \forall x \in R^n, p_j \in \Delta_j, \tag{4.5}$$

where $p_j = (p_{j1}, p_{j2}, \cdots, p_{j,|NB(j)|})^T$ is the vector of Lagrange multiplier, Δ_j is a simplex set defined as follows:

$$\Delta_j \equiv \left\{ p_j \in R^{|NB(j)|} \left| \sum_{i \in NB(j)} p_{ji} = 1, p_{ji} \geq 0 \right. \right\}. \tag{4.6}$$

It is easy to see that no matter which value the multiplier vector p_j is chosen, the value of the Lagrange function $\ell_j(x, p_j)$ is less than or equal to the maximum value function $f_{\max}^{(j)}(x)$, i.e.,

$$\ell_j(x, p_j) \leq f_{\max}^{(j)}(x). \tag{4.7}$$

From the definition of Lagrange function $\ell_j(x, p_j)$, we have the following lemma:

Lemma 7 *The maximum value function $f_{\max}^{(j)}(x)$, defined in (4.2), can be expressed as follows:*

$$f_{\max}^{(j)}(x) = \sup_{p_j \in \Delta_j} \ell_j(x, p_j) = \max_{p_j \in \Delta_j} \ell_j(x, p_j). \tag{4.8}$$

Proof For $\forall x \in R^n$ and $\forall p_j \in \Delta_j$, it is easy to see that

$$\sum_{i \in NB(j)} p_{ji} f_{ji}(x) \leq f_{\max}^{(j)}(x). \tag{4.9}$$

Therefore,

$$\sup_{p_j \in \Delta_j} \ell_j(x, p_j) \leq f_{\max}^{(j)}(x). \tag{4.10}$$

Let $I_{\max}^{(j)}(x)$ be the indicator set of element functions $f_{ji}(x)(i \in NB(j))$ that equal to the maximum value function $f_{\max}^{(j)}$ at point x, i.e.,

$$I_{\max}^{(j)}(x) := \{k | f_{jk}(x) = f_{\max}^{(j)}(x)\}. \tag{4.11}$$

If $k \in I_{\max}^{(j)}(x)$, then for arbitrary $x \in R^n$ and $p_j \in \Delta_j$, we have

$$\sup_{p_j \in \Delta_j} \ell_j(x, p_j) \geq \sum_{i \in NB(j)} \bar{p}_{ji} f_{ji}(x) = f_{\max}^{(j)}(x) b \tag{4.12}$$

where

$$\bar{p}_{ji} = \begin{cases} 1, & i = k; \\ 0, & i \neq k. \end{cases} \tag{4.13}$$

From (4.10) and (4.12), the first equality in (4.8) is hold. Consider that Δ_j is a tight set and $\ell_j(x, p_j)$ is a continuous function on p_j, the second equality in (4.8) is also hold. □

From Lemma 7, it can be seen that since the Lagrange function $\ell_j(x, p_j)$ is a linear function on variable p_j, (4.8) has multi-solutions. Therefore, function $f_{\max}^{(j)}(x)$ defined in (4.2) is an undifferentiable function.

From Lemma 7, it can also be seen see that since the multiplier vector p_j is limited inside the simplex Δ_j, the Lagrange function $\ell_j(x, p_j)$ can be interpreted as a convex

combination of all element functions $f_{ji}(x)$ $(i \in NB(j))$ and multipliers p_{ji} are the combination coefficients. Therefore, Problem (4.1) can be solved by solving an equivalent problem of finding a set of value of $p_{ji}, (i \in NB(j))$ such that the Lagrange function $\ell_j(x, p_j)$ approximates to the maximum value function, i.e., to find the optimal combination $\hat{p}_j$ from all combinations that satisfies (4.6) such that (4.7) becomes the following equality:

$$\ell_j(x, \hat{p}_j) = \sum_{i \in NB(j)} \hat{p}_{ji} f_{ji}(x) = f_{\max}^{(j)}(x). \tag{4.14}$$

On the other hand, if the Lagrange multipliers p_{ji} $(i \in NB(j))$, also called the combination coefficients, is endued with a probability sense, i.e., describing them as the corresponding probabilities such that the element function $f_{ji}(x)$ becomes the maximum value function $f_{\max}^{(j)}(x)$, then from the concept of probability, Problem (4.1) can be transferred into a maximized problem of finding the optimal probability distribution that satisfies:

$$\begin{cases} \max\limits_{p_j \in R^{|NB(j)|}} & \ell_j(x, p_j) \\ s.t. & \sum\limits_{i \in NB(j)} p_{ji} = 1; \\ & p_{ji} \geq 0, \ \ i \in NB(j). \end{cases} \tag{4.15}$$

Now, we begin to find a smooth function to approximate to the maximum value function. According to the analysis above, there are two object functions to be maximized:

1. To maximize the Lagrange function through selecting the optimal multiplier vector;
2. To maximize the entropy function by finding an unbiased probability distribution.

Consider the maximum entropy theory that introduced in the previous section, the optimal probability distribution should also satisfies:

$$\begin{cases} \max\limits_{p_j \in R^{|NB(j)|}} & H(p_j) = -\sum\limits_{i \in NB(j)} p_{ji} \ln p_{ji} \\ s.t. & \sum\limits_{i \in NB(j)} p_{ji} = 1; \\ & p_{ji} \geq 0, \ \ i \in NB(j). \end{cases} \tag{4.16}$$

Therefore, the problem to be solved is a multi-objective problem as follows:

$$\begin{cases} \max\limits_{p_j \in R^{|NB(j)|}} & \{\ell_j(x, p_j), H(p_j)\} \\ s.t. & \sum\limits_{i \in NB(j)} p_{ji} = 1; \\ & p_{ji} \geq 0, \ \ i \in NB(j). \end{cases} \tag{4.17}$$

By the weighting coefficient method, the multi-object problem (4.17) can be transformed into a single-object problem as follows:

$$\begin{cases} \max\limits_{p_j} & L_\theta^{(j)}(x, p_j) = \sum\limits_{i \in NB(j)} p_{ji} f_{ji}(x) \\ & \qquad -\dfrac{1}{\theta} \sum\limits_{i \in NB(j)} p_{ji} \ln p_{ji}; \\ s.t. & \sum\limits_{i \in NB(j)} p_{ji} = 1; \\ & p_{ji} \geq 0, \ \ i \in NB(j), \end{cases} \tag{4.18}$$

where $\theta \geq 0$ is a weighting coefficient. Obviously, when θ is small, the second item of the object function $L_\theta^{(j)}(x, p_j)$ is dominative. Then the gained probability distribution mainly reflects the requirement of unbiased distribution. With the increase of θ's value, the effect of the first item increases; thus, the object of maximizing the Lagrange function becomes dominative.

To solve Problem (4.18), we first consider the following problem:

$$\sup_{p_j \in \Delta_j} \left\{ L_\theta^{(j)}(x, p_j) := \ell_j(x, p_j) - \theta^{-1} \sum_{i \in NB(j)} p_{ji} \ln p_{ji} \right\}. \tag{4.19}$$

Based on the knowledge of convex analysis and the property of entropy function, we can prove that the function defined by (4.19) has the following property:

Theorem 19 *The function $F_\theta^{(j)}(x)$ defined by (4.19) is differentiable and uniformly approximate to function $f_{\max}^{(j)}(x)$ on the whole space R^n.*

Proof From the definition of indicator function, Problem (4.19) can be reduced as

$$\begin{aligned} \sup_{p_j \in R^{|NB(j)|}} \Bigg\{ & \sum_{i \in NB(j)} p_{ji} f_{ji}(x) \\ & -\theta^{-1} \sum_{i \in NB(j)} p_{ji} \ln p_{ji} - \delta(p_j | \Delta_j) \Bigg\}, \end{aligned} \tag{4.20}$$

where $\delta(p_j|\Delta_j)$ is an indicator function on the closed convex set Δ_j. From the strictly convex property of entropy function $\sum_{i \in NB(j)} p_{ji} \ln p_{ji}$, it can be seen that for arbitrary fixed $x \in R^n$, the object function of the maximum problem (4.20) is a closed normal strictly concave function on variable p_j and the effective region is the tight set Δ_j. Therefore, from the Weiertrass theory, the maximum problem exists an unique solution $p_j^*(x, \theta)$ and reaches its finite optimal value on the unique solution. That is, Problem (4.19) defines a real value function on R^n as follows:

$$F_\theta^{(j)}(x) := \sum_{p_j \in \Delta_j} L_\theta^{(j)}(x, p_j) = L_\theta^{(j)} \left[x, p_j^*(x, \theta) \right]. \tag{4.21}$$

Consider that function $-\sum_{i \in NB(j)} p_{ji} \ln p_{ji}$ is unnegative on the bounded closed convex set Δ_j and has an upper bound $(\ln m)/\theta$, that is

$$1 \leq -\sum_{i \in NB(j)} p_{ji} \ln p_{ji} \leq \frac{\ln m}{\theta}, \forall p_j \in \Delta_j, \tag{4.22}$$

we have

$$\begin{aligned} \sup_{p_j \in \Delta_j} \ell_j(x, p_j) & \leq \sup_{p_j \in \Delta_j} L_\theta^{(j)}(x, p_j) \\ & \leq \sup_{p_j \in \Delta_j} \ell_j(x, p_j) + \frac{\ln m}{\theta}. \end{aligned} \tag{4.23}$$

Thus, from (4.8) and (4.21), we have

$$f_{\max}^{(j)}(x) \leq F_{\theta}^{(j)}(x) \leq f_{\max}^{(j)}(x) + \frac{\ln m}{\theta}. \tag{4.24}$$

This indicates that the function $F_{\theta}^{(j)}(x)$, defined by the maximum problem (4.19), is uniformly approximate to $f_{\max}^{(j)}(x)$ on the whole space R^n.

At the same time, if function $K(p_j)$ is defined as follows:

$$K(p_j) := \begin{cases} \sum\limits_{i \in NB(j)} p_{ji} \ln p_{ji} & p_{ji} \geq 0; \\ +\infty & p_{ji} < 0, \end{cases} \tag{4.25}$$

then it is a closed normal strictly convex function on $R^{|NB(j)|}$ and $\mathrm{ri}(\mathrm{dom}K) \cap \mathrm{ri}(\Delta_j) \neq \phi$. Therefore, from (4.20) and the definition of convex conjugate function, we have

$$\begin{aligned} F_{\theta}^{(j)}(x) &= \theta^{-1} \cdot (K + \delta)^*(\theta F_j(x)) \\ &= \theta^{-1} \cdot (K^* \diamond \delta^*)(\theta F_j(x))), \end{aligned} \tag{4.26}$$

where $F_j(x) := (f_{ji}(x))_{i \in NB(j)}^T$ is a vector function and K^* is the convex conjugate function of K. Since $K \in Leg(R_+^m)$, i.e., $K(p_j)$ is a Legendre convex function, $K^* \in Leg(\mathrm{int}(\mathrm{dom}K^*))$ and $(K^* \diamond \delta^*)(\cdot)$ is essentially smooth. Thus, from (4.26) and the property that $F_{\theta}^{(j)}(x)$ is a real value function on R^n, we have $\mathrm{dom}(K^* \diamond \delta^*) = R^{|NB(j)|}$. Due to the continuous differentiable property of $F_j(x)$, $F_{\theta}^{(j)}(x)$ is a smooth function.

Combine the above two aspects, the theorem is proven. □

In the following, we will prove some properties of function $F_{\theta}^{(j)}(x)$.

Theorem 20 *For $\forall x \in R^n$, function $F_{\theta}^{(j)}(x)$ has the following properties:*

1. $f_{\max}^{(j)}(x) \leq F_{\theta}^{(j)}(x) \leq f_{\max}^{(j)}(x) + (\ln m)/\theta$.
2. $\lim_{\theta \to \infty} F_{\theta}^{(j)}(x) = f_{max}^{(j)}(x)$.
3. *If all the functions f_{ji} $(i = 1, 2, \cdots, m)$ in the original problem (4.1) are convex, $F_{\theta}^{(j)}(x)$ is a convex function too.*
4. $\nabla_x F_{\theta}^{(j)}(x) = \sum_{i \in NB(j)} \hat{p}_{ji}(x) \nabla_x f_{ji}(x)$.
5. $-(\ln m)/\theta^2 \leq \partial F_{\theta}^{(j)}(x)/\partial\theta \leq 0$.
6. $F_{\theta}^{(j)}(x) \leq F_{\vartheta}^{(j)}(x)$, $\forall \theta \leq \vartheta$.

Proof 1. Here, we provide a different proof from Theorem 19. From the expression of $F_{\theta}^{(j)}(x)$ in (4.39), we have

$$\begin{aligned} F_{\theta}^{(j)}(x) &= f_{\max}^{(j)}(x) \\ &\quad + \frac{1}{\theta} \ln \left\{ \sum_{i \in NB(j)} \exp\left[\theta \left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right] \right\}. \end{aligned} \tag{4.27}$$

From the definition of maximum value function $f_{\max}^{(j)}(x)$ in (4.2), we have

$$1 \leq \sum_{i \in NB(j)} \exp\left[\theta\left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right] \leq m. \tag{4.28}$$

Substitute this inequality into (4.27), we have

$$f_{\max}^{(j)}(x) + \frac{1}{\theta}\ln 1 \leq F_{\theta}^{(j)}(x) \leq f_{\max}^{(j)}(x) + \frac{1}{\theta}\ln m. \tag{4.29}$$

2. Take limitation on both side of (4.29), this property is proven.
3. For any $x, y \in R^n$ and $\alpha \in (0,1)$, since all the functions f_{ji} $(i \in NB(j))$ are convex, we have

$$\begin{aligned}
&F_{\theta}^{(j)}(\alpha x + (1-\alpha)y) \\
&= \frac{1}{\theta}\ln\left\{\sum_{i \in NB(j)} \exp\left[\theta f_{ji}\left(\alpha x + (1-\alpha)y\right)\right]\right\} \\
&\leq \frac{1}{\theta}\ln\left\{\sum_{i \in NB(j)} \exp\left[\theta\left(\alpha f_{ji}(x) + (1-\alpha) f_{ji}(y)\right)\right]\right\} \\
&= \frac{1}{\theta}\ln\left\{\sum_{i \in NB(j)} \left(\exp\left[\theta f_{ji}(x)\right]\right)^{\alpha}\left(\exp\left[\theta f_{ji}(y)\right]\right)^{1-\alpha}\right\}.
\end{aligned}$$

Applying to the Hölder inequality, we have

$$\begin{aligned}
&\sum_{i \in NB(j)} \left(\exp\left[\theta f_{ji}(x)\right]\right)^{\alpha}\left(\exp\left[\theta f_{ji}(y)\right]\right)^{1-\alpha} \\
&\qquad \leq \left\{\sum_{i \in NB(j)} \exp\left[\theta f_{ji}(x)\right]\right\}^{\alpha} \\
&\qquad\qquad \cdot \left\{\sum_{i \in NB(j)} \exp\left[\theta f_{ji}(y)\right]\right\}^{1-\alpha}.
\end{aligned}$$

Combined the above two relationships, we have

$$\begin{aligned}
&F_{\theta}^{(j)}\left(\alpha x + (1-\alpha)y\right) \\
&\quad \leq \frac{\alpha}{\theta}\ln\left\{\sum_{i \in NB(j)} \exp\left[\theta f_{ji}(x)\right]\right\} \\
&\qquad + \frac{1-\alpha}{\theta}\ln\left\{\sum_{i \in NB(i)} \exp\left[\theta f_{ji}(y)\right]\right\} \\
&\quad = \alpha F_{\theta}^{(j)}(x) + (1-\alpha)F_{\theta}^{(j)}(y).
\end{aligned} \tag{4.30}$$

Hence, function $F_{\theta}^{(j)}(x)$ is a convex function.
4. Take derivation about x on both side of (4.27), we have

$$\begin{aligned}
&\nabla_x F_{\theta}^{(j)}(x) = \nabla_x f_{\max}^{(j)}(x) \\
&+ \frac{1}{\theta}\nabla_x \ln\left\{\sum_{i \in NB(j)} \exp\left[\theta\left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right]\right\}.
\end{aligned} \tag{4.31}$$

Since

$$\begin{aligned}
&\nabla_x \ln\left\{\sum_{i\in NB(j)} \exp\left[\theta\left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right]\right\} \\
&= \frac{\sum_{i\in NB(j)} \nabla_x \exp\left[\theta\left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right]}{\sum_{i\in NB(j)} \exp\left[\theta\left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right]} \\
&= \sum_{i\in NB(j)} \frac{\exp\left[\theta\left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right]}{\sum_{i\in NB(j)} \exp\left[\theta\left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right]} \\
&\quad \cdot\nabla_x\left[\theta\left(f_{ji}(x) - f_{\max}^{(j)}(x)\right)\right] \\
&= \theta \sum_{i\in NB(j)} \hat{p}_{ji}(x)\nabla_x f_{ji}(x) - \nabla_x f_{\max}^{(j)}(x),
\end{aligned} \tag{4.32}$$

where $\hat{p}_{ji}x$ is defined as (4.38). Substitute this results into (4.31), this property is proven.
5. From the expression of function $F_\theta^{(j)}(x)$ in (4.39) and the definition of $\hat{p}_{ji}(x)$ in (4.38), we have

$$\begin{aligned}
&\frac{\partial F_\theta^{(j)}(x)}{\partial\theta} \\
&= -\frac{F_\theta^{(j)}(x)}{\theta} + \frac{\sum_{i\in NB(j)} \exp\left(\theta f_{ji}(x)\right) f_{ji}(x)}{\theta \sum_{i\in NB(j)} \exp\left[\theta f_{ji}(x)\right]} \\
&= \theta^{-1}\left[\sum_{i\in NB(j)} \hat{p}_{ji}(x) f_{ji}(x) - F_\theta^{(j)}(x)\right].
\end{aligned} \tag{4.33}$$

According to (4.18) and (4.39), we have

$$\begin{aligned}
F_\theta^{(j)}(x) = &\sum_{i\in NB(j)} p_{ji}(x) f_{ji}(x) \\
&-\theta^{-1} \sum_{i\in NB(j)} p_{ji}(x) \ln\left[p_{ji}(x)\right].
\end{aligned} \tag{4.34}$$

Therefore,

$$\partial F_\theta^{(j)}(x)/\partial\theta = \theta^{-2} \sum_{i\in NB(j)} p_{ji}(x) \ln\left[p_{ji}(x)\right]. \tag{4.35}$$

Thus, from the following inequality:

$$-\frac{\ln m}{\theta} \le \theta^{-2} \sum_{i=1}^{m} p_{ji}(x) \ln\left[p_{ji}(x)\right] \le 0, \ \ \forall x \in R^n, \tag{4.36}$$

this property is proven.
6. From property 5, we can see that function $F_\theta^{(j)}(x)$ is a decreasing function on θ, thus, this property is straight forward for property 5. □

Item 1 in Theorem 20 provides error bounds of function $F_\theta^{(j)}(x)$, and item 2 shows that function $F_\theta^{(j)}(x)$ uniformly approximates to function $f_{\max}^{(j)}(x)$. Item 3 shows the convex

property of function $F_\theta^{(j)}(x)$, and item 4 is for the continuity and the differentiability of function $F_\theta^{(j)}(x)$. Item 5 provides both upper bound and lower bound of the derivation of function $F_\theta^{(j)}(x)$ on p, and item 6 shows that function $F_\theta^{(j)}(x)$ is a monotonously decrease function on θ.

Since function $F_\theta^{(j)}(x)$ uniformly converges to the objective function $f_{\max}^{(j)}(x)$, solving the original problem (4.1) is equivalent to solving the following problem:

$$\min_{x \in R^n} F_\theta^{(j)}(x). \tag{4.37}$$

As function $F_\theta^{(j)}(x)$ is differentiable, the optimal solution, $\hat{p}_j(x)$, of Ea. (4.18) can be easily derived from applying the $K-T$ condition as follows:

$$\hat{p}_{ji}(x) = \frac{\exp\{\theta f_{ji}(x)\}}{\sum_{l \in NB(j)} \exp\{\theta f_{jl}(x)\}}, \;\; i \in NB(j). \tag{4.38}$$

Substitute the analytical solution $\hat{p}_j(x)$ of the multiplier p_j in the objective function of (4.18), we have

$$\begin{aligned} F_\theta^{(j)}(x) &= L_\theta^{(j)}\left(x, \hat{p}_j(x)\right) \\ &= \frac{1}{\theta} \ln \left\{ \sum_{i \in NB(j)} \exp\left[\theta f_{ji}(x)\right] \right\}. \end{aligned} \tag{4.39}$$

4.7 Simulation Results

In this section, we present some simulation results from OMNet++ simulator [128]. We use the Japanese Internet Backbone (NTTNET) as the simulation framework (see Figure 4.1). It is a 57 node, 162 bidirectional links network. The link bandwidth is 6 Mbits/sec

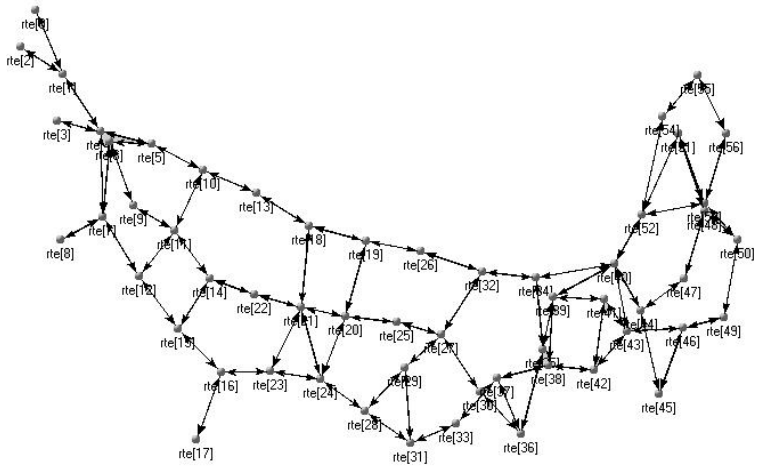


Figure 4.1: The Japanese Backbone (NTTNET)

and the propagation delay is from 2 to 5 milliseconds. The size of data packet is 512 bytes and the size of an agent is 48 bytes. Traffic is defined in terms of open sessions. A session is defined between two nodes and it remains active until a certain amount of data are transferred at a given rate. Each session is characterized completely by session size,

the generation rate of sessions (GRS), and the sending rate of the packets (SRP). In the simulation, session size is set to be 2 Mbits.

We implemented our algorithm (OA) together with OSPF [29] and AntNet [20] to evaluate and compare the efficience of these three algorithms. OSPF is a routing protocol used in Internet. It is based on the shortest path routing by using the Dijkstra algorithm. It uses only the best path towards the destination to forward the data packets and this best path is computed with the costs of edges, normally the distances of links. As OSPF ignores the queuing delays, it cannot work well during heavy traffic loads. AntNet is a dynamic routing algorithm proposed by Di Caro and Dorigo [20]. It is inspired from the foraging activity of real ants. In contrast with OSPF, AntNet measures the queuing delay and tries to find a globally optimal solution that satisfies the dynamic constraints. However, AntNet has to store all possible solutions for its statistical method. Because the heuristic measuring has a relation with a percentage of the data flow, the correctness of heuristic measuring would be worse under the condition of a low load on the network. In addition, when the queuing delay is low, a stochastic process is no more sensible. Our algorithm utilizes inner agents to disseminate the state of the network to the routers in real-time. It does not need any global information such as the structure of the topology and cost of links among routers, which not only balances the local traffic flow but also enhances fault tolerance. Compared to AntNet, this enhancement in performance is achieved with a little more routing overhead which is defined as the ratio between the bandwidth occupied by the routing packets and the total network bandwidth, but this overhead remains constant for a given topology.

Two parameters are introduced for our comparison, throughput and packet delay. The throughput is defined as the correctly delivered bits per second, which shows the ability of the algorithms for transmitting data streams. The packet delay is defined as the time interval from the creation of a data packet to its arrival at the destination, which is only calculated for the correctly received data packets. It indicates the quality of paths chosen by a data packet. Figure 4.2 and Figure 4.3 show the comparison results of the average throughput and the average packet delay between OSPF, AntNet, and our algorithm.

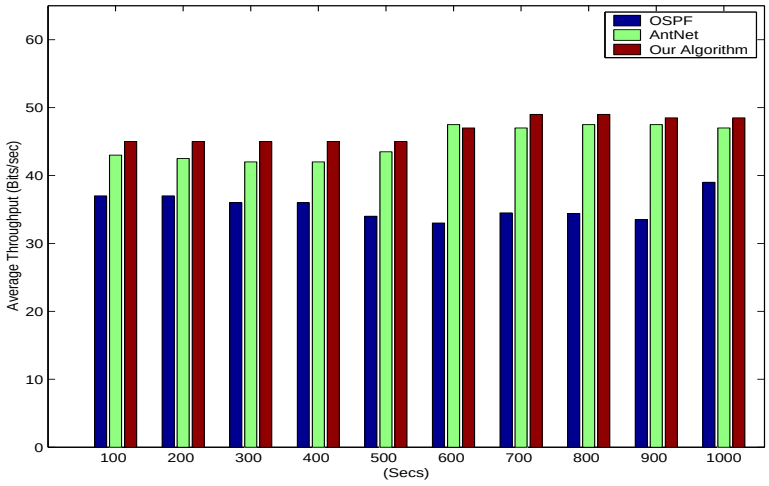


Figure 4.2: The average throughput when all nodes sent data to node 4 from 500 seconds to 1000 seconds

Originally, there is a normal load of GRS=2.7 seconds and SRP=0.3 second. From 500

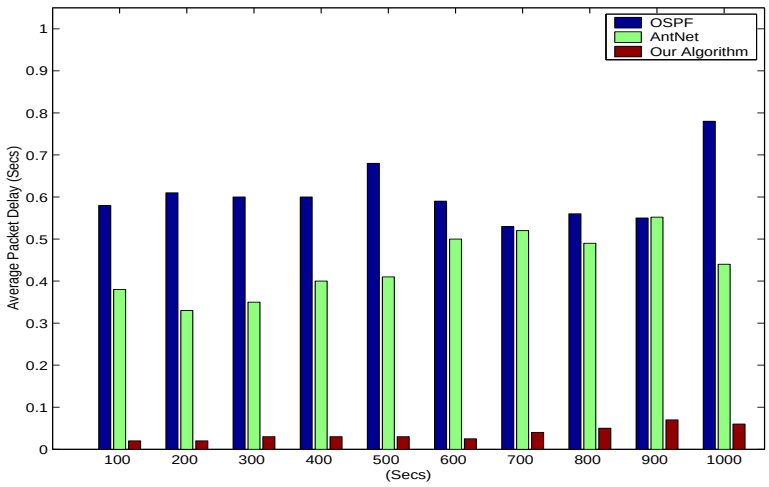


Figure 4.3: The average packet delay when all nodes sent data to node 4 from 500 seconds to 1000 seconds

seconds to 1000 seconds, all nodes sent data to node 4 with SRP=0.05 seconds. It can be seen that both our algorithm and AntNet are able to cope with the transient overload. OSPF shows the poorest performance. It can also be seen that the average packet delay for our algorithm is less than 0.1 second as compared to 0.5 second for AntNet. Again, OSPF shows the poorest performance.

We also compared the deliver rate among these three algorithms which states the proportion of packets correctly delivered. Table 4.1 shows the results of the deliver rate which is defined as the quotient of the amount of received packets to the amount of sent packets. From the simulation results, we can see that our algorithm achieves a similar

Parameters (Sec)		Deliver Rate (%)		
GRS(Sec)	SRP(Sec)	OSPF	AntNet	Our Algorithm
4.5	0.5	83.21	96.85	99.99
2.5	0.5	82.46	97.31	99.99
1.5	0.5	80.13	97.24	99.99
2.5	0.05	83.94	95.94	99.68

Table 4.1: The Comparison of Deliver Rate

performance to AntNet, which is much better than OSPF.

Figure 4.4 shows the results of the comparison between our algorithm and AntNet in which node 21 crashed at 300 seconds, node 40 crashed at 500 seconds, and both of them were repaired at 800 seconds. Here, GRS=4.7 seconds and SRP=0.05. The purpose of this experiment was to analyze the fault tolerant behavior of our algorithm. The GRS and SRP is selected to ensure that no packets are dropped because of the congestion. Based on our experimental results, our algorithm is able to deliver 97% of deliverable packets as compared to 89% by AntNet. From the figure we can see that our algorithm has a superior throughput and lesser packet delay. But once node 40 crashes, the packet delay of our algorithm increases because of higher load at node 43. From Figure 4.1 it is obvious that the only path to the upper part of the network is via node 43 once node 40

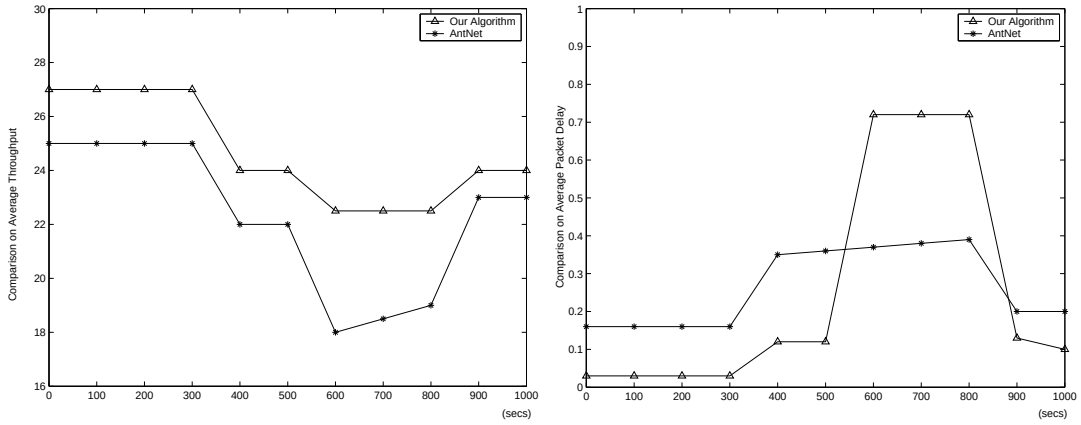


Figure 4.4: Node 21 is down at 300 and node 40 is down at 500 and both repaired at 800

crashed. Since our algorithm is able to deliver more packets the queue length at node 43 increased and this led to relatively poorer packet delay as compared to AntNet. On the contrary, although node 21 is critical, but in case of its crash still multiple paths exist to the middle and upper part of the topology.

4.8 Conclusion

In this chapter, we proposed a mobile agent-based execution model for use in peer-to-peer networks in which the traffic congestion is considered. We defined a traffic cost function for each link based on known traffic information of the network and found a probability distribution that mobile agents may select a neighboring node and move to. We prove that the probability distribution both makes inference on the known traffic information and approximates to a unbiased distribution. Theoretical analysis provided some properties of the approximating function (including the convergence property) and showed the rationality of our probability distribution. We also provide simulation results to evaluate our algorithm in comparison with AntNet and OSPF. Through extensive simulations we have demonstrated that our algorithm achieves a better or similar performance as compared to AntNet. In the near future we will have implemented our algorithm inside the network stack of the Linux kernel in order to then test the algorithm on real network topologies.

Chapter 5

Conclusions

5.1 Summarization

In this dissertation, we studied the behaviors of mobile agents in some application areas, including network routing, fault tolerance, and peer-to-peer networks. The main contents of this dissertation are generalized as follows:

- Routing is a key factor for network performance. In Chapter 2, we addressed the problem of network routing and management by deploying mobile agents. Several agent-based routing algorithms have been considered and analyzed, including the classical ant-like algorithm, settleable-agent algorithm, and nonsettleable-agent algorithm. For each case, we analyzed both the probability of success and the population distribution of mobile agents to evaluate the performance of these proposed algorithms. We further proposed an agent-based routing model in faulty networks and analyzed these two parameters. Experiments to validate the theoretical results were also provided. Both the theoretical and experimental results showed that the behaviors of mobile agents could be characterized by the population of mobile agents and the probability of success, and both two parameters can be controlled by tuning the number of agents generated per request and the number of jumps each mobile agent can make.

- The reliable execution of mobile agents is an important design issue for building a mobile agent system. In Chapter 3, we proposed a new fault-tolerant execution model by effectively combines available techniques. In our model, failures were classified into two classes: on-site failure and frontal failure. For each kind of failures, an exceptional-event handling method was adopted. For the first time, the behaviors of mobile agents in networks that may contain faults were statistically analyzed in a quantitative way, which exploits a new approach to assessing the performance of agent-based systems. Several important parameters on system performance, including migration time, life expectancy, and population distribution of mobile agents, were analyzed. The analytic method proposed in this chapter may also be applied for performance analysis in other complex systems. Our model is reliable and cost-efficient, which offers us a promising way for designing reliable mobile agent system.

- Peer-to-peer computing is one of the trends in the field of internetworking and also an important application area for mobile agents technology. In Chapter 4, we

proposed a mobile agent-based execution model for use in peer-to-peer networks in which the traffic congestion was considered. To balance the traffic load on each link, we introduced the maximum entropy theory into our algorithm. We defined a traffic cost function for each link based on known traffic information of the network and found a probability distribution that mobile agents may select a neighboring node and move to. We proved that the probability distribution both made inference on the known traffic information and approximated to a unbiased distribution. We also theoretically analyzed some properties of the approximating function (including the convergence property) and showed the rationality of our probability distribution.

5.2 Future Work

The following issues can be considered for future research.

- Peer-to-peer computing. Based on our previous results, we will further develop new methods and algorithms for monitoring and analyzing mobile agents in peer-to-peer networks, and build a prototype of mobile agent-based peer-to-peer system.

- Ubiquitous computing. Many ubiquitous computing applications need a constant flow of information about their environment to be able to adapt to their changing context. Mobile agents technology is expected as an applied means for collecting, aggregating, and disseminating context information.

- Grid computing. Grid computing was proposed in mid 1990s thanks to the increasing demand of large-scale scientific computation in the fields of life sciences, biology, physics, and astronomy. Since a grid connects numerous geographically distributed computers, and tasks are submitted to grid nodes in a distributed fashion, an important issue is how to evenly distribute submitted tasks to nodes. Mobile agents can be used in grid computing applications for task scheduling and resource allocation.

- Wireless sensor networks. A sensor network is usually built with a large number of small devices, each of which has limited battery energy, memory, computation, and communication capacities. Furthermore, these small devices are statically configured and cannot easily adapt to a changing context or be upgraded, which prevents the network from running other applications. Mobile agent-based sensor network can be used to undertake multiple tasks simultaneously and maintain maximum network flexibility in terms of its ability to be reprogrammed and its ability to execute multiple applications at a time.

Bibliography

[1] F. Abbattista, A. Paradiso, G. Semeraro, and F. Zambetta, "An Agent that Learns to Support Users of a Web Site," *Applied Soft Computing,* Vol. 4, No. 1, pp. 1-12, 2004.

[2] E. Adar and B. Huberman, "Free Riding on Gnutella," *First Monday*, Vol. 5, No. 10, 2000.

[3] A. Agbaria, A. Freund, and R. Friedman, "Evaluating Distributed Checkpointing Protocols," *Proceedings of the 23rd International Conference on Distributed Computing Systems,* pp. 266-273, 2003.

[4] IBM Japan Research Group, "Aglets Workbench," *http://www.trl.ibm.co.jp/aglets/index.html*.

[5] Ajanta - Mobile Agent Research Project, *http://www.cs.umn.edu/ajanta*.

[6] J. W. Baek, J. H. Yeo, G. T. Kim, and H. Y. Yeom, "Cost Effective Mobile Agent Planning for Distributed Information Retrieval," *Proceedings of the 21st International Conference on Distributed Computing Systems(ICDCS'01)*, pp. 65-72, 2001.

[7] J. W. Baek, G. T. Kim, and H. Y. Yeom, "Cost-Effective Planning of Timed Mobile Agents," *Proceedings of the International Conf. on Information Technology: Coding and Computing(ITCC'02)*, pp. 536-541, 2002.

[8] J. W. Baek, J. H. Yeo, and H. Y. Yeom, "Agent Chaining: An Approach to Dynamic Mobile Agent Planning," *Proceedings of the 22nd International Conference on Distributed Computing Systems(ICDCS'02)*, pp. 579-586, 2002.

[9] B. Baran and R. Sosa, "A new approach for antnet routing," *Proceedings of the 9th International Conference on Computer, Communications and Networks*, 2000.

[10] BearShare, *http://www.bearshare.com/*.

[11] R. Beckers, J. L. Deneubourg, and S. Goss, "Trails and U-turns in the selection of the shortest path by the ant Lasius niger," *Jorunal of Theoretical Biology,* Vol. 159, pp. 397-415, 1992.

[12] B. Benjami, "Improved Antnet Routing," *ACM SIGCOMM Computer Communication Review*, Vol. 31, No. 2, pp. 42-48, 2001.

[13] F. Bergadano, A. Puliafito, S. Riccobene, and G. Ruffo, "Java-based and secure learning agents for information retrieval in distributed systems," *Information Sciences*, Vol. 113, NO. 1-2, pp.55-84, 1999.

[14] A. Bieszczad, B. Pagurek, and T. White, "Mobile Agents for Network Management," *IEEE Communication Surveys*, Vol. 1, No. 1, pp. 2-9, 1998.

[15] F. Brazier, B. Overeinder, M. Steen, and N. Wijngaards, "Agent Factory: Generative migration of Mobile Agents in Heterogeneous Environments," *Proceedings of the ACM Symposium on Applied Computing (SAC02)*, pp. 101-106, 2002.

[16] B. Brewington, R. Gray, K. Moizumi, D. Kotz, G. Cybenko, and D. Rus, "Mobile Agents in Distributed Information Retrieval," *Intelligent Information Agents: Agents-Based Information Discovery and Management on the Internet,* M. Klusch, ed., Chapter 15, pp. 355-395, 1999.

[17] M. Brown and M. Najork, "Distributed Active Objects," *Dr. Dobb's Journal,* Vol. 22, No. 3, pp. 34-41, 1997.

[18] L. Cardelli, "A Language with Distributed Scope," *Computing Systems,* Vol. 8, No. 1, pp. 27-59, 1995.

[19] G. D. Caro G. and M. Dorigo, "AntNet: A Mobile Agents Approach to Adaptive Routing," *Technical Report IRIDIa/97-12,* Universite Libre de Bruxelles, Belgium, 1997.

[20] G. D. Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks," *Journal of Artificial Intelligence Research*, Vol. 9, pp. 317-365, 1998.

[21] G. D. Caro and M. Dorigo, "Mobile Agents for Adaptive Routing," *Proceedings of the 31st Hawaii International Conference on System Sciences*, pp. 74-83, 1998.

[22] G. D. Caro and M. Dorigo, "Two Ant Colony Algorithms for Best-Effort Routing in Datagram Networks," *Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98)*, pp. 541-546, 1998.

[23] D. Chang and D. Lange, "Mobile Agents: A New Paradigm for Distributed Object Computing on the WWW," *Proceedings of the OOPSLA96 Workshop: Toward the Integration of WWW and Distributed Object Technology,* pp. 25-32, 1996.

[24] J. Claessens, B. Preneel, and J. Vandewalle, "(How) Can Mobile Agents Do Secure Electronic Transactions on Untrusted Hosts? A Survey of the Security Issues and the Current Solutions," *ACM Transactions on Internet Technology*, Vol. 3, No. 1, pp. 28-48, 2003.

[25] Clip2, "The Gnutella protocol specification v.0.4," *http://www.clip2.com/ GnutellaProtocol04.pdf.*

[26] Clip2, "Reflector White Paper," *http://dss.clip2.com/reflector_wp.html.*

[27] K. Curran, D. Woods, N. McDermot, and C. Bradley, "The Effects of Badly Behaved Routers on Internet Congestion," *International Journal of Network Management*, Vol. 13, No. 1, pp. 83-94, 2003.

[28] M. Dalmeijer, E. Rietjens, and M. Moschgath, "A Reliable Mobile Agents Architecture," *Proceedings of the 1st IEEE International Symposium on Object-Oriented Real-Time Distributed Computing,* pp. 64-72, 1998.

[29] E. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerical Mathematics,* pp. 269-271, 1959.

[30] M. Dikaiakos and D. Gunopoulos, "FIGI: The Architecture of an Internet-based Financial Information Gathering Infrastructure," *Proceedings of the 1st International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems,* pp. 91-94, 1999.

[31] M. Dorigo, V. Maniezzo, A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, Vol. 26, No. 1, pp. 29-41, 1996.

[32] M. Dorigo and L. M. Gambardella, "Ant Colonies for the Traveling Salesman Problem," *BioSystems*, Vol. 43, pp. 73-81, 1997.

[33] M. Dorigo, M. Zlochin, N. Meuleau, and M. Birattari, "Updating ACO pheromones using Stochastic Gradient Ascent and Cross-Entropy Methods," *Proceedings of the Eve Workshop 2002,* pp. 21-30, 2002.

[34] T.C. Du, E.Y. Li, and A.P. Chang, "Mobile Agents in Distributed Network Management," *Communications of the ACM,* Vol. 46, No. 7, pp. 127-132, 2003.

[35] M. Elnozahy, L. Alvisi, Y. Wang, and D.B. Johnson, "A Survey of Rollback-Recovery Protocols in Message-Passing Systems," *ACM Computing Surveys*, Vol. 34, No. 3, pp. 375-408, 2002.

[36] FastTrack, *http://www.fasttrack.nu/.*

[37] Findlaw, *http://www.findlaw.com/napster/index.html.*

[38] M.J. Fischer, N.A. Lynch, and M.S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *Journal of the ACM*, Vol. 32, No. 2, pp. 374-382, 1985.

[39] M. Garey and D. Johnson, "Computers and Intractability: A Guide to the Theorey of NP-Completeness," *Freeman,* 1979.

[40] E. Gendelman, L.F. Bic, and M.B. Killencourt, "An Application-Transparent, Platform-Independent Approach to Rollback-Recovery for Mobile Agent Systems," *Proceedings of the 20th International Conference on Distributed Computing Systems*, pp. 564-571, 2000.

[41] Gnutella, *http://www.gnutella.co.uk/.*

[42] L. Gong, "Project JXTA: A Technology Overview," *Sun Microsystems, Inc.,* 2001.

[43] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels, "Self-Organized Shortcuts in the Argentine Ant," *Naturwissenschaften,* Vol. 76, pp. 579-581, 1989.

[44] R. Gray, "AgentTCL: A Flexible and Secure Mobile-agent System," *PhD Thesis*, Dartmouth College, 1997.

[45] R. Gray, D. Kotz, G. Cybenko and D. Rus, "Agent Tcl," *Mobile Agents: Explanations and Examples, Manning Publishing,* W. Cockayne and M. Zyda editors., pp. 58C95, 1997.

[46] R. Gray, D. Kotz, G. Cybenko, and D. Rus, "D'Agents: Security in a Multiple-Language, Mobile-Agent System," *Mobile Agents and Security,* G. Vigna, editor, pp. 154-187, 1998.

[47] R. Gray, D. Kotz, G. Cybenko, and D. Rus, "Mobile Agents: Motivations and State-of-the-Art Systems," *Technical Report TR2000-365,* Dartmouth College, 2000.

[48] C. Harrision, D. Chess, and A. Kershenbaum, "Mobile Agents: Are They a Good Idea?", *IBM Research Report*, T.J.Watson Research Center, NY 10598, 1995.

[49] M. He, N.R. Jennings, and H. Leung, "On Agent-Mediated Electronic Commerce," *IEEE Transactions on Knowledge and Data Engineering,* Vol. 15, No. 4, pp. 985-1003, 2003.

[50] P. S. Heck and S. Ghosh, "A Study of Synthetic Creativity through Behavior Modeling and Simulation of an Ant Colony," *IEEE Intelligent Systems*, Vol. 15, No. 6, pp. 58-66, 2000.

[51] B. Hölldobler and E. O. Wilson, "The ants," *Springer-Verlag*, Berlin, 1990.

[52] IBM, "Aglets," *http://www.trl.ibm.co.jp/aglets/*, 2001.

[53] InAMosh-handy, "InAMosh-handy: A Shopping Scenario for Mobile Phones Based on Intelligent Agents," *http://dai.cs.tu-berlin.de/english/forschung/projekte/InAMoSHandy/main.html*, 2002.

[54] "FIPA Interaction Protocol Library Specifications," *http://www.fipa.org/specs/fipa00025/PC00025C.html*, 2002.

[55] D. N. Jayasimha, L. Schwiebert, D. Manivannan, and J. A. May, "A Foundation for Designing Deadlock-Free Routing Algorithms in Wormhole Networks," *Journal of the ACM*, Vol. 50, No. 2, pp. 250-275, 2003.

[56] D. Johansen, R. Renesse, and F. Schneider, "Operating System Support for Mobile Agents," *Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems,* pp. 42-45, 1995.

[57] D. Johansen, K. Marzullo, F. Schneider, K. Jacobsen, and D. Zagorodnov, "NAP: Practical Fault-Tolerance for Itinerant Computations," *Proceedings of the 10th International Conference on Distributed Computing Systems (ICDCS'99)*, pp. 180-189, 1999.

[58] D. Johansen, F.B. Schneider, and R. Renesse, "What TACOMA Taught Us," *Mobility, Mobile Agents and Process Migration,* pp. 564-566, 1999.

[59] E. T. Joynes, "Information theory and statistical mechanics," *The Physical Review.* No. 108, pp. 171-190, 1957.

[60] "Jumping Beans White Paper," Ad Astra Engineering, Inc., 1998, *http://www.JumpingBeans.com.*

[61] Project JXTA, *http://www.jxta.org/.*

[62] G. Karjoth D. Lange, and M. Oshima, "A Security Model for Aglets," *IEEE Internet Computing,* Vol. 1, No. 4, pp. 68-77, 1997.

[63] KaZaA, *http://www.kazaa.com/.*

[64] S. H. Kim and T. G. Robertazzi, "Mobile Agent Modeling," *SUNY at Stony Brook Technical Report 786*, University of Stony Brook, 2000.

[65] D. Kotz and R.W. Gray, "Mobile Agents and the Future of the Internet," *ACM Operating Systems Review*, Vol. 33, No. 3, pp. 7-13, 1999.

[66] G. Karjoth, D. Lange, and M. Oshima, "A Security Model for Aglets," *IEEE Internet Computing*, Vol. 1, No. 4, pp. 68-77, 1997.

[67] D. Lange and M. Oshima, "Programming and Developing Java Mobile Agents with Aglets," *Addison Wesley,* 1998.

[68] D. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents," *Communications of the ACM,* Vol. 42, pp. 88-89, 1999.

[69] Z. J. Lee, C. Y. Lee, and S. F. Su, "An Immunity-Based Ant Colony Optimization Algorithm for Solving Weapon-Target Assignment Problem," *Applied Soft Computing*, Vol. 2, No. 1, pp. 39-47, 2002.

[70] T. Li and K. Lam, "An Optimal Location Update and Searching Algorithm for Tracking Mobile Agent," *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02),* pp. 639 - 646, 2002.

[71] S. Liang, A. Zincir-Heywood, and M. Heywood, "The effect of routing under local information using a social insect metaphor," *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 1438-1443, 2002.

[72] Limewire, *http://www.limewire.com/.*

[73] Y. Ling, J. Mi, and X. Lin, "A Variational Calculus Approach to Optimal Checkpoint Placement," *IEEE Transactions on Computers,* Vol. 59, No. 7, pp. 699-708, 2001.

[74] W. Lou and J. Wu, "On Reducing Broadcast Redundancy in Ad Hoc Wireless Networks," *IEEE Transactions on Mobile Computing*, Vol. 1, No. 2, pp. 111-123, 2002.

[75] M.R. Lyu and T.Y. Wong, "A Progressive Fault Tolerant Mechanism in Mobile Agent Systems," *Proceedings of the 7th World Multi-Conference on Systemics Cybernetics and Informatics (SCI 2003),* pp. 299-306, 2003.

[76] P. Maes, R.H. Guttman, and A.G. Moukas "Agents that Buy and Sell," *Communications of the ACM*, Vol. 42, No. 3, pp. 81-91, 1999.

[77] F.J. Meyer and D.K. Pradhan, "Consensus with dual failure modes," *IEEE Trans. on Parallel and Distributed Systems,* Vol. 2, No. 2, pp. 214-222, 1991.

[78] D. Milojicic, "Trend Wars: Mobile Agent Applications," *IEEE Concurrency,* Vol. 7, No. 3, pp. 80-90, 1999.

[79] D. Milojicic, "Guest Editor's Introduction: Agent Systems and Applications," *IEEE Concurrency*, Vol.8, No. 2, pp. 22-23, 2000.

[80] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-peer Computing," *Technical Report,* HP Laboratories Palo Alto, 2002.

[81] N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes, "Hive: distributed Agents for Networking Things," *IEEE concurrency*, Vol. 8, No. 2, pp. 24-33, 2000.

[82] J. H. Moore, L. W. Hahn, M. D. Ritchie, T. A. Thornton, and B. C. White, "Routine discovery of complex genetic models using genetic algorithms," *Applied Soft Computing,* Vol. 4, No. 1, pp. 79-86, 2004.

[83] V.F. Nicola, "Checkpointing and the modeling of program execution time," *Software Fault Tolerance,* pp. 167-188, 1995.

[84] ObjectSpace, Inc. "ObjectSpace Voyager Core Package Technical Overview," *Technical Report*, ObjectSpace, Inc., 1997, *http://www.objectspace.com/.*

[85] Odyssey, *http://www.genmagic.com/*, 2001.

[86] OpenNapL Open Source Napster Server, *http://opennap.sourceforge.net/.*

[87] H. Pals, S. Petri, and C. Grewe, "FANTOMAS–Fault Tolerance for Mobile Agents in Clusters," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS) 2000 Workshop,* pp. 1236-1247, 2000.

[88] S. Papastavrou, G. Samaras, and E. Pitoura, "Mobile Agents for WWW Distributed Database Access," *IEEE Transactions on Knowledge and Data Engineering Journal (TKDE),* Vol. 12, No. 5, pp. 802-820, 2000.

[89] T. Park, I. Byun, H. Kim, and H.Y. Yeom, "The Performance of Checkpointing and Replication Schemes for Fault Tolerant Mobile Agent Systems," *Proceedings of the Symposium on Reliable Distributed Systems (SRDS02),* pp. 256-261, 2002.

[90] H. Paulino, "A Mobile Agent Systems' Overview," *Technical Report DCC-2002-1,* New University of Lisbon, 2002.

[91] S. Pears, J. Xu, and C. Boldyreff, "Mobile Agent Fault Tolerance for Information Retrieval Applications: An Exception Handling Approach," *Proceedings of the 6th International Symposium on Autonomous Decentralized Systems,* pp. 115-122, 2003.

[92] S. Pears, J. Xu, and C. Boldyreff, "A Dynamic Shadow Approach for Mobile Agents to Survive Crash Failures," *Proceedings of the 6th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing,* pp. 113-120, 2003.

[93] H. Peine and T. Stolpmann, "The Architecture of the Ara Platform for Mobile Agents," *Proceedings of the 1st Internatioanl Workshop on Mobile Agents (MA'97),* pp. 50-61, 1997.

[94] Perpetuum, "Perpetuum Mobile Procura Project," *http://www.sce.carleton.ca/netmanage/perpetuum.shtml*, 2002.

[95] A. Pitsillides, G. Samaras, M. Dikaiakos, and E. Christodoulou, "DITIS: Collaborative Virtual Medical Team for Home Healthcare of Cancer Patients," *Proceedings of Conference on the Information Society and Telematics Applications,* 1999 (Invited).

[96] S. Pleisch and A. Schiper, "Modeling Fault-Tolerant Mobile Agent Execution as a Sequence of Agreement Problems," *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems,* pp. 11-20, 2000.

[97] S. Pleisch and A. Schiper, "FATOMAS - A Fault-Tolerant Mobile Agent System Based on the Agent-Dependent Approach," *Proceedings of the International Conference on Dependable Systems and Networks,* pp. 215-224, 2001.

[98] S. Pleisch and A. Schiper, "Fault-Tolerant Mobile Agent Exection," *IEEE Transactions on Computers,* Vol. 52, No. 2, pp. 209-222, 2003.

[99] W. Qu, H. Shen, and J. Sum, "New Analysis on Mobile Agents Based Network Routing," *Proceedings of the 3rd International Conference on Hybrid Intelligence Systems (HIS'03),* pp. 769-778, 2003 (Best Student Paper).

[100] W. Qu and H. Shen, "Some Analysis on Mobile-Agent Based Network Routing," *Proceedings of The International Symposium on Parallel Architectures, Algorithms, and Networks (I_SPAN2004),* pp. 2-17, 2004.

[101] K. Rothermel and M. Strasser, "A Fault-Tolerant Protocol for Providing the Exactly-Once Property of Mobile Agents," *Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems (SRDS'98),* pp. 100-108, 1998.

[102] T. Roughgarden and E. Tardos, "How Bad is Selfish Routing?", *Journal of the ACM*, Vol. 49, No. 2, pp. 236-259, 2002.

[103] G. Samaras and A. Pitsillides, "Client/Intercept: a Computational Model for Wireless Environments," *Proceedings of the 4th International Conference on Telecommunications (ICT'97),* pp. 1205-1210, 1997.

[104] G. Samaras, M. Dikaiakos, C. Spyrou, and A. Liberdos, "Mobile Agent Platforms for Web-Databases: A Qualitative and Quantitative Assessment," *Proceedings of the First International Symposium on Agent Systems and Applications Third International Symposium on Mobile Agents (ASA/MA'99)*, pp. 50-64, 1999.

[105] G. Samaras, P. Evripidou, and E. Pitoura, "Mobile-Agents based Infrastructure for eWork and eBussiness Applications," *Proceedings of the eBusiness and Ework Conference,* pp. 1092-1098, 2000.

[106] F. Schneider, "Towards Fault-Tolerant and Secure Agentry," *Proceedings of the 11th International Workshop on Distributed Algorithms,* pp. 1-14, 1997 (Invited Paper).

[107] D. Schoder and T. Eymann, "The Real Challenges of Mobile Agents," *Communications of ACM*, Vol. 43, No. 6, pp. 111-112, 2000.

[108] Scholl, "Napster Protocol Specification by Analysis," *http://opennap.sourceforge.net/napster.txt.*

[109] R. Schoonderwoerd, O. Holland, and J. Bruten, "Ant-like Agents for Load Balancing in Telecommunications Networks," *Proceedings of the First International Conference on Autonomous Agents,* pp. 209-216, 1997.

[110] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27, No. 3, pp. 379-428, 1948.

[111] T. Sheldon, "Linktionary," *http://www.linktionary.com/.*

[112] F.Assis Silva and R. Popescu-Zeletin, "An Approach for Providing Mobile Agent Fault Tolerance," *Proceedings of the 2nd Internatioanl Workshop on Mobile Agents (MA'98),* pp. 14-25, 1998.

[113] C. Spyrou, G. Samaras, E. Pitoura, and P. Evripidou, "Mobile Agents for Wireless Computing: The Convergence of Wireless Computational Models with Mobile-Agent Technologies," *Journal of ACM/Baltzer Mobile Networking and Applicaitons (MONET),* Vol. 9, No. 5, pp. 517-528, 2004.

[114] I. Stojmenovic, S. Seddigh, and J. Zunic, "Domination Sets and Neighbor Elimination Based Broadcasting Algorithms in Wireless Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 1, pp. 14-25, 2002.

[115] Stormcast 5.0, *http://www.cs.uit.no/DOS.StormCast/*, 2002.

[116] M. Straßer and K. Rothermel, "Reliability Concepts for Mobile Agents," *Int'l Journal of Cooperative Information Systems,* Vol. 7, No. 4, pp. 355-382, 1998.

[117] M. Straßer and K. Rothermel, "System Mechanism for Partial Rollback of Mobile Agent Execution," *Proceedings of the 20th International Conference on Distributed Computing Systems,* pp. 20-28, 2000.

[118] J. Sum, H. Shen, C. Leung, and G. Young, "Analysis on a Mobile Agent-Based Ant Algorithm for Network Routing and Management," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 14, No. 3, pp. 193-202, 2003.

[119] J. Sum, H. Shen, G. Young, J. Wu, and C. Leung, "Analysis on Extended Ant Routing Algorithms for Network Routing and Management," *The Journal of Supercomputing,* Vol. 24, pp. 327-340, 2003.

[120] Sun Microsystems, Inc., *http://www.sun.com/.*

[121] Sun Microsystems, Inc., "JXTA v1.0 Protocols Specification, Revision 1.1.1," *http://spec.jxta.org/vl.0/docbook/JXTAProtocols.html/*, 2001.

[122] L. Tang and B. Pagurek, "A Comparative Evaluation of Mobile Agent Performance for Network Management," *Proceedings of the 9th Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems(ECBS'02),* pp. 258 - 267, 2002.

[123] A. B. Templeman and X. Li, "A maximum entropy approach to constrained non-linear programming," *Engineering Optimization*, Vol. 12, No. 3, pp. 191-205, 1987.

[124] W. Theilmann and K. Rothermel, "Optimizing the Dissemination of Mobile Agents for Distributed Information Filtering," *IEEE Concurrency,* pp. 53-61, 2000.

[125] Things That Think, *http://www.media.mit.edu/ttt/*, 2002.

[126] Toshiba, "Multi-Agent Framework for 100% Pure Agent System ," *http://www2.toshiba.co.jp/rdc/beegent/index.htm*, 2005.

[127] K. Truelove, "OpenNap Use Crashes," *http://www.openp2p.com/pub/a/p2p/2001/05/11/opennap.html.*

[128] A. Varga. "OMNeT++: Discrete Event Simulation System: User Manual," *http://www.omnetpp.org/.*

[129] Y. Villate, A. Illarramendi, and E. Pitoura, "Data Lockers: Mobile-Agent Based Middleware for the Security and Availability of Roaming Users Data," *Proceedings of th 5th International Conference on Cooperative Information Systems (CoopIS'2000),* pp. 275-286, 2000.

[130] H. Vogler, T. Kunkelmann, and M.L. Moschgath, "An Approach for Mobile Agent Security and Fault Tolerance using Distributed Transaction," *Proceedings of the International Conference on Parallel and Distributed Systems,* pp. 268-274, 1997.

[131] Recursion Software Voyager, *http://www.recursionsw.com.*

[132] T. Walsh, N. Paciorek, and D. Wong, "Security and Reliability in Concordia," *Mobility: processes, computers, and agents,* pp. 524 - 534, 1999.

[133] Y. Wang, "Dispatching Multiple Mobile Agents in Parallel for Visiting E-Shops," *Proceedings of the 3rd International Conference on Mobile Data Management(MDM'02),* pp. 61-68, 2002.

[134] "Webopedia Online Computer Dictionary for Computer and Internet Terms and Definitions," *http://www.webopedia.com.*

[135] J.E. White, "Telescript Technology: Mobile Agents," *Software Agents,* J. Bradshaw, ed., pp. 437–472, 1996.

[136] J.E. White, "Mobile Agents," *General Magic White Paper,*, *http://www.genmagic.com/agents*, 1996.

[137] T. White, B. Pagurek, and F. Oppacher, "ASGA: Improving the Ant System by Integration with Genetic Algorithms," *Proceedings of the 3rd Conference on Genetic Programming (GP/SGA'98),* pp. 610-617, 1998.

[138] T. White, B. Pagurek, and F. Oppacher, "Connection Management Using Adaptive Agents," *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98),* pp. 802-809, 1998.

[139] O. Wittner and B.E. Helvik, "Cross-Entropy Grided Ant-Like Agents Finding Cyclic Paths in Scarcely Meshed Networks," *Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS'2002),* pp. 123 - 134, 2002.

[140] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet, "Concordia:An Infrastructure for Collaborating Mobile Agents," *Proceedings of the 1st International Workshopon Mobile Agents*, pp. 86-97, 1997.

[141] D. Wong, N. Paciorek, and D. Moore, "Java-Based Mobile Agents," *Communications of the ACM,* Vol. 42, pp.92-102, 1999.

[142] "Extensible Markup Language (XML) 1.0 (Third Edition)," *http://www.w3.org/TR/REC-xml/*, 2004.

[143] D. Xu, J. Yin, Y. Deng, and J. Ding, "A Formal Architectural Model for Logical Agent Mobility," *IEEE Transactions on Software Engineering*, Vol. 29, No. 1, pp. 31-45, 2003.

Publications

[1] Wenyu Qu, Hong Shen, and John Sum: "New Analysis on Mobile Agents Based Network Routing", Applied Soft Computing Journal (ASOC), Elsevier, Vol. 6, No. 1, pp. 108-118, 2005. (Preliminary version was appeared in HIS03)

[2] Wenyu Qu, Hong Shen, and John Sum: "Stochastic Analysis on Mobile Agent-Based E-Shopping", International Journal of Electronic Business (IJEB), Inderscience, Vol. 3, Nos. 3/4, pp. 339-355, 2005.

[3] Wenyu Qu and Hong Shen: "Analysis on Fault-Tolerant Execution of Mobile Agents", ACM Transactions on Internet Technology (TOIT), submitted.

[4] Wenyu Qu and Hong Shen: "The Probability of Success of Mobile Agents When Routing in Faulty Networks", Proceedings of the Eighth Asia Pacific Web Conference (APWEB06), pp. 76-84, China, January 2006.

[5] Wenyu Qu, Hong Shen, and Xavier Defago: "A Survey of Mobile Agent-Based Fault-Tolerant Technology", Proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05), pp. 446-450, China, December 2005.

[6] Wenyu Qu, Hong Shen, and Yingwei Jin: "Distribution of Mobile Agents in Vulnerable Networks", Proceedings of the 4th International Conference on Grid and Cooperative Computing (GCC 2005), pp. 894-905, China, November 2005.

[7] Wenyu Qu and Hong Shen: "Theoretical Analysis on A Traffic-Based Routing Algorithm of Mobile Agents", Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'05), pp. 520-526, France, September 2005.

[8] Wenyu Qu and Hong Shen: "Performance Modelling of a Fault-Tolerant Agent-Driven System", Proceedings of the International Conference on Communications (ICC'05), Korea, May 2005 (CD ROM).

[9] Wenyu Qu and Hong Shen: "Further Analysis of Mobile Agents' Fault-Tolerant Behavior", Proceedings of the 5th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'04), pp. 582-585. Singapore, December 2004.

[10] Wenyu Qu and Hong Shen: "Mobile Agent-Based Execution Modelling", Proceedings of the 4th International Conference on Hybrid Intelligent Systems (HIS'04), pp. 148-153, Japan, December 2004.

[11] Wenyu Qu and Hong Shen: "Behavior Modelling of Mobile Agents' Fault-Tolerant Execution", Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04), pp. 377-380, China, September 2004.

[12] Wenyu Qu, Hong Shen, and John Sum: "Further Analysis on the Application of Mobile Agents in Network Routing", Proceedings of the International Conference on E-Business and Telecommunication Networks (ICETE'04), pp. 204-212, Portugal, August 2004.

[13] Wenyu Qu and Hong Shen: "Some Analysis on Mobile-Agent Based Network Routing", Proceedings of the 7th International Symposium on Parallel Architectures., Algorithms, and Networks (I-SPAN 2004), pp. 2-17, Hong Kong, China, May 2004.

[14] Wenyu Qu and Hong Shen: "New Analysis on Mobile Agents Based Network Routing", Proceedings of the 3rd International Conference on Hybrid Intelligent Systems (HIS'03), pp. 769-778, Australia, December 2003 (Best Student Paper Award).