Anycast Services and Its Applications

by

Shui Yu

B.Eng (University of Electronic Science and Technology of China)
M.Eng (University of Electronic Science and Technology of China)

Submitted in Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Supervised by: Professor Zhou, Wanlei
School of Information Technology
Deakin University
Melbourne, Victoria
Australia

October, 2004

DEAKIN UNIVERSITY

CANDIDATE DECLARATION

I certify that the thesis entitled **Anycast Services and Its Applications** submitted for the degree of **DOCTOR OF PHILOSOPHY** is the result of my own research, except where otherwise acknowledged, and that this thesis in whole or in part has not been submitted for an award, including a higher degree, to any other university or institution.

Full Name: **Shui YU**

Signed

……..

Date

……..

Acknowledgement

First of all, I would like to present my honest thanks to my supervisor, Professor Wanlei Zhou for his igniting the wonderful research paradise for me, providing lots of suggestions through the whole period of my PhD program, correcting my numerous manuscripts, and reading the thesis draft thoroughly many times. I am fortunate to have been able to work with him since the beginning of my PhD study.

Thanks go to Associate Professor Weijia Jia, who introduced the anycast topic to me in 2001. We have discussed some issues about anycast research, and I have learned a lot from the discussions. Dr Jia gave me numbers of valuable suggestions in the area and also showed me the spirit of a very competitive researcher.

I would like to thank the academia staff in school of IT, Professor Andrzej Goscinski, Dr Jackie Silcock, Dr Honghua Dai, Dr Jinyu Hou, Dr Morshed Chowdhury, Dr Changgui Chen, Mr Atul Sajjanhar, and so on for their supporting and encouragement.

I appreciate the PhD students and research fellows in our school, include Dr Jiyuan An, Mr Xiaoshu Hang, Mr Mingjun Lan, Mr John Casey, Miss Elicia Lanham, Miss Naomi Augar, Miss Ruth Raitman, and so on. I have shared happy time with these colleagues.

I am grateful to School of Information Technology, Deakin University, which has provided an excellent environment for learning and experiencing. Also, my gratitude goes to Kathy Giulieri, Jeff McDonald, Anne Sharma, Robert Ruge and other staff in the school for their help.

Finally, I would like to present my thanks to my parents, for their long-term and remote understanding and encouragement. I also like to thank my brother, Ping Yu, and my sister, Qing Yu, for their supporting and caring to our parents, which allows me to have more time to focus on my study.

Publications

The following is a list of published papers which are fully referenced and included in international conferences' proceedings and journals during this PhD program:

1. **Yu, S.,** and Zhou, W. (2004) A Novel Middleware Based Web Database Model, The 2004 IEEE International Conference on Web Intelligence, September 20 – 24, 2004, Beijing, P. R. China.

2. **Yu, S.,** Cassey, J. and Zhou, W. (2003) A Load Balanced Algorithm for Web Based Distributed System, *Proceedings of the 2nd International Conference on Grid and Cooperative Computing, GCC2003,* pp121-128, Springer-Verlag, Germany.

3. **Yu, S.**, Zhou, W. and Jia, W. (2003)Fault-tolerant servers for anycast communication, *Volume III Proceedings of the International Conference on Parellel and Distributed Processing Techniques and Applications: PDPTA'03*, pp. 1244-1250, CSREA Press, Las Vegas, Nevada, USA.

4. **Yu, S.**, Zhou, W. and Chowdhury, M. (2003) Quality-of-Service Routing for Web-Based Multimedia Servers, *Proceedings of the 7th Joint Conference on Information Sciences*, pp. 1349-1353, Association for Intelligent Machinery, United States.

5. **Yu, S.**, Zhou, W., Lan, M. and Wu, Y. (2003) An architecture of Internet based data processing based on multicast and anycast protocols, *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 104-110, IEEE Press, USA.

6. Zhao, Y., Zhou, W., Huang, J., **Yu, S.** and Lanham, E. (2003) Self-Adaptive Clock Synchronization for Computational Grid, *Journal of Computer Science and Technology*, Vol 18, No 4, pp. 434-441, Allerton Press, China.

7. **Yu, S.**, Zhou, W., Zhao, Y., Lan, M. and Xiang, Y. (2002) A web-DB model on Multicast and Anycast, *Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing*, pp. 412-415, IEEE Computer Society, USA.

8. **Yu, S.**, Zhou, W. and Wu, Y. (2002) Research on network anycast, *Proceedings of Fifth International Conference on Algorithms and Architectures for Parallel Processing* , pp. 154-161, IEEE Computer Society, USA **.**

9. **Yu, S.**, Zhou, W., Huang, F. and Lan, M. (2002) An efficient algorithm for application-layering anycasting, *Distributed Communities on the Web: Proceedings of the 4th International Workshop, DCW 2002*, pp. 74-83, Springer-Verlag, Germany.

10. **Yu, S.**, Chen, C. and Zhou, W. (2001) An automatic generation model for web-based database applications, *Proceedings of the IASTED International Conference: Parallel and Distributed Computing and Systems*, pp. 62-66, ACTA Press, USA.

Abstract

Anycast in next generation Internet Protocol is a hot topic in the research of computer networks. It has promising potentials and also many challenges, such as architecture, routing, Quality-of-Service, anycast in ad hoc networks, application-layer anycast, etc. In this thesis, we tackle some important topics among them.

The thesis at first presents an introduction about anycast, followed by the related work. Then, as our major contributions, a number of challenging issues are addressed in the following chapters. We tackled the anycast routing problem by proposing a requirement based probing algorithm at application layer for anycast routing. Compared with the existing periodical based probing routing algorithm, the proposed routing algorithm improves the performance in terms of delay. We addressed the reliable service problem by the design of a twin server model for the anycast servers, providing a transparent and reliable service for all anycast queries. We addressed the load balance problem of anycast servers by proposing new job deviation strategies, to provide a similar Quality-of-Service to all clients of anycast servers. We applied the mesh routing methodology in the anycast routing in ad hoc networking environment, which provides a reliable routing service and uses much less network resources. We combined the anycast protocol and the multicast protocol to provide a bidirectional service, and applied the service to Web-based database applications, achieving a better query efficiency and data synchronization. Finally, we proposed a new Internet based service, minicast, as the combination of the anycast and multicast protocols. Such a service has potential applications in information retrieval, parallel computing, cache queries, etc. We show that the minicast service consumes less network resources while providing the same services. The last chapter of the thesis presents the conclusions and discusses the future work.

Contents

List of Figures

Chapter 1
Introduction

It is a dream to create something which can operate intelligently as men for human beings. Fifty years ago, John Von Neumann, the Hungarian-born mathematician, made the dream clearer than before; he wrote “First Draft of a Report on the EDVAC” in which he outlined the architecture of a stored-program computer. From then on, the computer world experienced the dramatic hardware development and explosive software surge. Many brilliant ideas, countless exciting events and progresses were presented, are presenting and will be presented.

Since the early 1990s, the Internet has an explosive development. New networks and IP nodes have being attached to the Internet at a breathtaking rate, as a result, the 32-bit IPv4 address space would become exhausted in 2008 [SOL96]. The Internet Engineering Task Force began an effort to develop IP next generation (IPv6), a successor to the IPv4 protocol, in the 1990s to solve the problem; at the same time, the designers also took this opportunity to improve other aspects of IPv4, based on the accumulated operational experience with IPv4.

Anycast is an important proposed service in the IP next generation. The anycast service can find the “best” server among the mirrored or replicated servers for a given user. It is very powerful in the next generation Internet, in which the same or similar servers are distributed all over the world to provide fast and low network resource

consuming services. The research on anycast has started about 10 years ago, and some excellent results have been obtained already , but it is only a small part of the mountain.

In this thesis, we try to explore some aspects of the anycast service. It includes the related work, anycast routing problems, fault-tolerance issues, load balance problems among the anycast servers, anycast related research in ad hoc network environment, and anycast application on Web based applications. Furthermore, we propose a new Internet service --- minicast.

1.1 Background

With the dramatic development of computer network technology, a lot of new application requirements have emerged, and researchers are trying to develop new protocols and models to meet the ever increasing and changing requirements. Resource location is one of the demands when users expect a specific service in networks.

In the 1980s, researchers focused on wide-area resource location, and achieved some good results. In Boggs' Ph. D. thesis [BOG82], the author defined a directed broadcast mechanism, by which a host could submit a broadcast to a particular network. The author also defined an expanding ring broadcast mechanism, which allowed broadcast to increasingly large concentric rings around the broadcast host. Later on, Xerox embedded expanding ring broadcasts into their Network Binding Protocol [XER86].

There were also a lot of projects exploring this field in the 1980s. Here we will only present two notable projects among them. Accetta [ACC83] presented a server location protocol trying to be used on top of whatever network protocols were available. The Eden system [ALM85] deployed a combination of broadcasts and hint caching to locate objects. All these systems focused on local area networks, therefore, from the viewpoint of the current Internet architecture, a main limitation of these projects is that none of them considered network topology.

In the early 1990s, the research on wide-area resource location was conducted on the Internet, because of the surge in popularity of the Internet. Much of the work during that period concentrated on the mirrored or replicated servers on the Internet, trying to locate the "best" service among the servers.

A number of systems were developed to deal with network topology considerations using geographical information in the early 1990s. [BRA94] focused on working on the DNS mapping scheme, and [GWE94] offered the push-caching scheme. The approaches involved in these systems are effective when there are few replicas, but it may not work well with the increasing number of replicas.

Hotz, in his Ph. D. thesis [HOT94], examined network triangulation as a possible approach for routing and server location. The author defined an N-tuple $< s_1, s_2, s_3, \ldots, s_N>$ as the distance to a node from each of N measurement beacons, and then defined the distance between a server at coordinates $S = < s_1, s_2, s_3, \ldots, s_N >$ and a client at $C = < c_1, c_2, c_3, \ldots, c_N >$ with the AVG function:

$$AVG(S,C) = (MAX(S,C) + MIN(S,C))/2$$

Where MAX and MIN are defined as:

$$MAX(S,C) = \min(s_1 + c_1, s_2 + c_2, s_3 + c_3, ..., s_N + c_N),$$

$$MIN(S,C) = \max(|\, s_1 - c_1 \,|, |\, s_2 - c_2 \,|, |\, s_3 - c_3 \,|, ..., |\, s_N - c_N \,|)$$

Here the *max()* and *min()* are the normal arithmetic maximum and minimum. The formulas provide the upper and lower distance bounds. The paper also showed that the AVG function generally produces a better result than either MAX or MIN.

In [GUS97], the authors mentioned resource allocation in a view of traffic dispersion [KIS93] [KIS94a] [KIS94b]. In these papers, Krishnan and Silvester claimed that it is necessary to adjust routing decisions in ATM networks to the needs of each specific type of traffic. They discussed making resource allocation on the cell, burst, or connection level, depending on traffic characteristics, thereby achieving finer control of the network resources. [KIS93] presented formulae for calculating the queuing delay, the queue blocking probability, and re-sequencing delay. The formulae were extended in [KIS94a] to contain a larger number of sources, and were applied to an ATM multiplexer with traffic dispersion. The results showed that dispersion decreases the average and standard deviation of the queuing delay as the number of multiplexed sources increases. Increasing the number of parallel paths can also improve the queuing performance. Furthermore, the results in [KIS94b] showed a decreased loss of probability due to dispersion, and also revealed that the number of sources that can be supported at a multiplexer for a given quality of service increases with dispersion.

Guyton and Schwartz [GUY94] studied the Network Time Protocol (NTP), which is widely used to synchronize computer clocks throughout the Internet. They pointed out that the NTP distributed software did not provide tools to discover the nearest NTP server or help debug the NTP system as a whole, therefore they developed a prototype tool that attempted to discover relevant information about every NTP site on the Internet. The data produced by the tool can be used for a variety of purposes, including locating nearby accurate time servers and computing aggregate and long-term evaluations of the size and health of the NTP system. Most importantly, the tool provided a means by which new NTP server administrators could make informed choices from among the possible servers as to with which to synchronize, balancing the need for accurate time with the need to distribute server load.

Myers, Dinda and Zhang [MYE99] presented some fascinating findings by testing 9 clients scattered throughout the United States, and retrieving over 490,000 documents from 47 production web servers, which mirror three different web sites. They discovered several interesting findings that may aid in the design of protocols for choosing among mirror servers. Though server performance varies widely, the authors had observed that a server's performance relative to other servers is more stable and is independent of time scale. In addition, a change in an individual server's transfer time is not a strong indicator that its performance relative to other servers has changed. They also found that clients wishing to achieve near-optimal performance may only need to consider a small number of servers rather than all mirrors of a particular site.

With the development of the computer network, several problems with the current generation of networks are identified: the difficulty of integrating new technologies and

standards into a shared network infrastructure, poor performance due to redundant operations at several protocol layers, and the difficulty of accommodating new services in the existing architectural model. Therefore, the concept of an active network [TEN97], [CAL98] emerged from discussions with the broad DARPA research community in 1994 and 1995. Active networks are an approach to network architecture in which the switches of the network perform customized computations on the messages following through them. [TEN97] discussed two approaches to the realization of active networks: the programmable switch approach and the capsule approach. [CAL98] introduced a model and nomenclature for active networks, described some possible approaches in terms of that nomenclature, and presented various aspects of the architecture being developed in the DARPA-funded active networks program.

The above research forms a basis and resource of anycast. Anycast is a natural continuation of resource location in the IP next generation. We can find the range of resource location from local area networks to wide area networks, and finally to the Internet. As a result of becoming the most popular application platform, the current Internet can not meet the increasing demands on it. Then there is IPv6, the next generation of the Internet protocol, in which the IP address resource space is extended dramatically, the protocol is more secure, and more services are provided. Anycast is one of the new services of the IPv6.

Partridge, Mendez, and Milliken [PAR93] originally proposed the idea of anycasting in the network layer to deal with replicated servers. They defined IP anycasting as a service to deliver an anycast datagram to one of the members of an anycast group. The

idea of anycast meets the requirements of mirrored or replicated servers on the Internet, therefore a large number of research was quickly conducted in the area.

Anycast architecture is a critical issue. [PAR93] extended the architecture of the current Internet for the anycast service. The paper recommended assigning anycast its own address space. It also pointed out the major problems in the proposed architecture. [OE00] pointed out that a point-to-point communication in which its end point is specified by an anycast address does not work well, especially with connection-oriented protocols, and presented two mechanisms, Source Identification Option and Anycast Address Mapper, to deal with the issue. [KAT00] considered the traditional belief that IP-anycast should be routed similarly to unicast had hampered the acceptance and deployment of anycast, and proposed a scalable architecture for global IP-anycast (GIA in short).

Anycast routing is another important issue in anycast services. [YAM01], [MIU01] presented a server selection policy for anycasting by Round Trip Time. It is simple and practical, and its performance is better than that of the server load-based algorithm and the network latency-based algorithm. Furthermore, it is not necessary to take care of all the responses from the servers except from the first response server, and it is not necessary to calculate the possibility. It is essential that the first response tells us which server among the anycast group is the “best” one.

The solutions presented in [XUA00] are practical for the IPv6, and they are simple and compatible to the current Internet and use a comparable amount of information with those protocols that are currently used to route other types (unicast and multicast)

of packets. Similar to the traditional routing algorithms in IPv4, it is passive and needs more consideration of Quality of Service (QoS in short). Fortunately, [XUA01] [JIA04] discusses the anycast flow with QoS requirements by using the methodology of Distributed Admission Control. One result of this research is that it shows that the performance of DAC is very close to the performances of those algorithms, which handle global and dynamic status information, but the cost is dramatically reduced. [XUA01] mentions only three aspects that affect anycast QoS: route distance, local admission history, and available bandwidth information. We should use more aspects of anycast QoS because of the increasing and different applications, such as accuracy, security, and so on.

GeoTORA [KO00] is a customized algorithm for a wireless network, in which the network topology is very dynamic, as a result, it is not compatible with the fixed Internet. On the other hand, it implies that the anycasting problem can be an issue of Graph Theory. There are two sets in the graph, the anycast server set and the client set, the issue is how to find the "best" way between the two sets. It points out a very promising direction, in which we can apply their findings to solve anycast routing problems.

By the middle of the 1990s, some researchers found limitations in the network-layer anycast, for example, inflexibility and limited support by current routers, hence, they presented the idea of application-layer anycast [BHA96], [BHA97], [FEI98] focusing research on anycast in the application layer. The application-layer anycast is compatible with the nature of current Internet facilities and meets current application requirements too.

Nowadays, research on anycast attracts a lot of attention, and it offers a promising future. Roughly there are two main categories of anycast, network-layer anycast and application-layer anycast. As the result of previous hard work, researchers have proposed some efficient models and algorithms, and found some defects in the models and algorithms as well. At the same time, researchers have realised that the related research on anycast has just begun, and there is still much more work to do in the near future, such as the fault tolerant issue, the performance issue, application problems, and so on. That is the reason that we focus on the research on anycast.

1.2 Main Issues in Anycast

We have introduced anycast and roughly the anycast research so far. In fact, anycast is an important and indispensable service on the Internet, and possesses huge potential applications in the near future. However, to fully realise the potential of anycast, more research is needed. We list some critical and interesting topics of anycast research here.

- Research on anycast architecture

Researchers are currently focusing on two categories of anycast: network-layer anycast and application-layer anycast. At the same time, these anycast technologies have their own limitations. Compared with application-layer anycast, network-layer anycast has several disadvantages, such as inflexible, need router supporting, and so on. Application-layer anycast, of course, has its own shortcomings:

1) Application-layer anycast is not as efficient as network-layer anycast in routing, although the former is more flexible.

2) It is difficult to manage application-layer anycast resolvers in the environment of Internet, because of different metrics applied to different resolvers.

3) When the current Internet upgrades to IPv6, then application-layer routing might not be necessary at all.

Since there are two options, the question remains: which one will dominate the coming future of anycast services in the Internet? With the history and the current foundation of the Internet, it is our opinion that it is more likely that, in most cases, the network-layer anycast will be more preferred than the application-layer anycast. The application-layer anycast service possesses advantages of its own, which network-layer anycast can not replace, therefore it may be used extensively in Internet applications.

Therefore network-layer based anycast architecture is a foundation research topic. The proposed anycast architecture should get rid of the inherit disadvantages of the current Internet, and be compatible with the next generation of the Internet. Furthermore, the proposed architecture should be simple and efficient.

- Anycast routing algorithms

Some efficient anycasting routing algorithms have been proposed. But we should mention that the foundation of these algorithms is based upon the research results and

environment of the current Internet. With the dynamic change of the Internet environment, some features of the new Internet will be different from the current Internet, such as traffic characters, speed, even architectures. For this reason, researchers and industry practitioners should pay more attention to this issue because of the constantly changing Internet.

- Anycast service quality guarantee

There is an issue of quality of service for anycast service. One important problem is the fault tolerance in anycast, which can provide continuous and high quality transaction service to clients.

- Anycast in ad hoc network

In mobile ad hoc networks there is no pre-existing network infrastructure, and the topology in such networks may be highly dynamic. In this environment, anycast mechanism can be used to find the "best" peer computer to connect to the networks. This may happen in battle fields, future home appliances, and etc.

- Anycast in service discovery

Anycast mechanism can be used for service discovery. Service discovery is getting more and more important in the diverse Internet. Anycast mechanism can serve it very

well on this purpose, and it can extract the "best" service among the same or similar services.

- New protocol based on the anycast protocol

Anycast mechanism is needed by the Internet now, and will be more important in the future. With the ever changing and developing Internet requirements, new Internet protocols based on anycast or derived from anycast will become necessary.

- Services based on anycast and other protocols

It is a good idea to mix anycast and other related services, such as multicast [DEE90], PAMCast [CHA02], geocast [KO00] [NAV97], and so on. For example mixing anycast with multicast together can provide bi-directional services. The mixed service can be used widely to meet the requirement of the Internet. Multicast can synchronize distributed objects in a multicast group on the Internet; at the same time, the anycast service responses to information searching and location in the anycast group; therefore, these two services can cooperate with each other to provide new services.

Multimedia applications will be a very important issue in the future Internet. Binding anycast service with multimedia applications is a challenging research topic requiring immediate attention. For example, anycast services can be used in e-commerce to find the best answer for a query from the whole Internet; anycast services can also help the users to find the "best" Internet Telephony Gateway from the global Internet in real-

time applications; furthermore, anycast services can help us to track and communicate with mobile agents in mobile applications.

1.3 Motivation of the Thesis

Anycast is an important and promising service on the Internet, and there are a number of challenging topics to be explored. We list some of them as below:

● Effective anycast routing algorithms are always essential for applications. So far the routing algorithms for anycast are still limited, especially for advanced applications, such as multimedia applications. Without effective anycast routing algorithms, anycast service can not work properly, users can not find the “best” server for their applications.

● Few research has touched the fault tolerance issue for anycast. The working environment of anycast is unstable and dynamic, therefore, fault tolerance is necessary for anycast clients. In this case, a reliable and continuous service is impossible without a fault tolerance mechanism.

● Load balance among anycast servers is also critical in terms of system performance. It is essential to provide similar QoS to all clients, and the load balance here includes not network load balance, but also the anycast server load balance. As far as we know, no work has been done in this aspect for anycast.

● Anycast in wireless ad hoc networks. Anycast has huge potential in an ad hoc environment. Anycast mechanism meets the device or service discovery in wireless networks perfectly. Unfortunately, not too much attention about this has been attracted about anycast application in wireless environment, there are a few papers about that, but it is very limited.

● Anycast applications. Anycast is a very useful and practical service, it can be used for many kind of applications. Without anycast service, it is hard for users to handle their requirements properly with diverse Internet services.

● New service based on the anycast protocol. Based on anycast and multicast, we propose a new Internet service, minicast, which try to send a package to at least m members out of n members, and save the network resource.

1.4 Objectives and Scope of the Thesis

This thesis aims to address some of the challenging issues of anycast. The anycast service is proposed for the IP next generation, and the research on it has been going on for a decade. Researchers have obtained some good results, but there are still a number of areas waiting to be explored, such as the architecture of anycast service, load balance of anycast, fault-tolerant anycast service, and so on.

This thesis will explore several aspects of the anycast service, to our knowledge some of them have not been touched by the other researchers yet. We point out the disadvantages of network layer anycast routing, and propose an efficient application layer anycast routing algorithm; we address the fault tolerant problem in anycast service, and propose a twin server based fault tolerant model for anycast service; we deal with the anycast server load balance problem and propose a job deviation strategy; we explore the anycast service in Ad Hoc networks, of which research on it is very limited; and then propose a new Internet service, minicast, which combines the protocols of anycast and multicast.

There are a lot of challenging areas of anycast that need to be explored as we described before, and it is impossible that we deal with all of them in this thesis. We will only discuss the topics that we have listed, namely, application layer anycast algorithms, the fault-tolerance issue, the load balance problem among anycast servers, anycast applications, and a new service based on anycast protocol. We believe that the results of this thesis form a significant contribution to the research of anycast services.

1.5 Approaches of the Thesis

In this thesis we use several approaches for our research, which are listed below,

● Queuing theory. An powerful analysis tool for computer network. We use it to analyse our proposed algorithms and models. With this mathematic tool, we get some theoretical results before we conduct simulations.

● Statistics theory. We employ statistics into the analysis of some algorithms and models, for example, the Internet traffic model, balls and bins theory, and so on.

● Network simulators (ns2) as simulation tool. We use ns2 as a simulation tool in our research. ns2 is used by a lot of researchers to simulate the Internet or networks. It provides a platform for building and simulating large scale networks.

● C ++ as a simulation language. In some cases, we use C ++ for programming our algorithms, simulating the models, and conducting some experiments.

1.6 Organization of the Thesis

The structure of this thesis is outlined as follows:

● Chapter 2, related work, reports the related work so far, and builds a clear picture of the current anycast research.

● Chapter 3, routing in anycast, points out the disadvantages of network layer anycast routing algorithms, and then proposes an application layer anycast routing algorithm,

furthermore, the queuing theory based analysis shows that the proposed algorithm is better than an existing application layer routing algorithm.

● Chapter 4, fault tolerance in anycast, finds out that no research has been done on the fault tolerance in the anycast scenario, and proposes a twin service model for the anycast service.

● Chapter 5, load balance issue in anycast, presents a load balance methodology for anycast servers. Anycast bears the feature of load balance, but it based on the network, and it does not include server load, and therefore the feature is not complete. The proposed algorithm tries to solve the problem.

● Chapter 6, anycast in an ad hoc network, explores the anycast service in a wireless network environment. We propose an anycast routing algorithm combined with the multicast service.

● Chapter 7, anycast in web-based database application, shows the anycast application in web-based database systems combined with multicast. Anycast takes the responsibility of finding the "best" database server, meanwhile, multicast handles the database synchronization among the distributed database servers.

● Chapter 8, minicast service, proposes a new Internet service, minicast service, based on the anycast and multicast protocols: in the scenario of n replicated or similar servers, a message is delivered to at least m members, $1 \le m \le n$. The service can provide the same Internet service while saving network resource.

- Chapter 9, conclusion and future work. This chapter summaries the major contributions of this thesis, the novelty of the research, and points out future work in the anycast research.

Chapter 2
Related Work of Anycast

In 1993, Partridge, Mendez, and Milliken proposed the idea of anycast. Since then, the Internet has gone through a dramatic development, and the same to anycast. Some brilliant ideas, models and algorithms were presented. In this chapter, we report the two categories of anycast : network layer anycast and application layer anycast. In each category, we survey the architectures, the anycast routing algorithms, applications, and so on.

2.1 The Network-Layer Anycast

The current research on anycast can be classified into two categories: anycast in network-layer and anycast in application-layer. The original idea of anycast was on network layer, and then some researchers developed the concept of anycast into application layer. In this section, the work on network-layer anycast will be presented.

2.1.1 Anycast Definition and Architectures

The original work by Partridge, Mendez, and Milliken [PAR93] proposed the idea of anycast for the IP next generation, and discussed its network layer support. They defined IP anycast as:

A service provides a stateless best effort delivery of an anycast datagram to at least one host, and preferably only one host, which serves the anycast address.

The authors created the idea based on the IP next generation, which is an extension of the current IP networks, IPv4, and the architecture is similar to IPv4. The paper recommended assigning anycast its own address space. It also pointed out the major problems that anycast must deal with: The first challenge of anycast is the hierarchical aggregation. The second is the stateless nature of the service, an issue that makes the establishment of TCP connections on top of anycast addresses problematic.

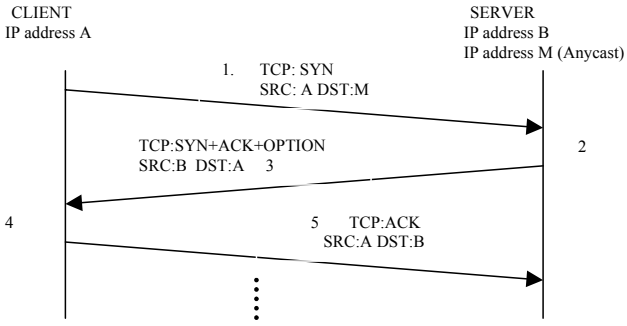

Figure 2.1: TCP Connection Establishment by "Source Identification Option"

OE and Yamaguchi [OE00] pointed out that a point-to-point communication in which its end point is specified by an anycast address does not work well, especially with connection-oriented protocols. An obvious reason is that a routing path between certain nodes may vary with changes on the network configuration. Hence, packets using an

anycast address may not be sent to the same node when changes on the routing path occur. Therefore they presented two mechanisms, Source Identification Option and Anycast Address Mapper, to deal with the issue.

The Source Identification Option mechanism uses the Internet-Draft [BOU96], "Source Identification Option", which is defined as informing any changes on source address to the peer. The proceeding of the model is detailed below and is illustrated in Figure 2.1.

1. A client sends an SYN packet to an anycast address M.
2. A server with the anycast address M receives the SYN packet.
3. The server replies to the client by sending the SYN+ACK packet with its own unicast address B as a source address and attaches "Source Identification Option" as the destination option.
4. The client checks the packet, and changes the server address from M to B.
5. The client opens the connection by sending the ACK packet to the unicast address B.

The implementation of this mechanism requires modifications in the IP layer and the transport layer of the TCP stack; therefore, any application can use the anycast address mechanism without any modification to themselves.

The Anycast Address Mapper mechanism, similar to the Source Identification Option, tries to find out the unicast address corresponding to the anycast address using the ICMP ECHO request/reply. Before establishing a connection to a server with an

anycast address, clients can find out the unicast address by using the "Anycast Address Mapper", then, the clients can use this unicast address to make a connection with the server. The process details of a TCP connection are described below and are illustrated in Figure 2.2.

1. A client sends the ICMP ECHO request packet to an anycast address M
2. A server receives the packet with the anycast address M.
3. The server sends ICMP ECHO reply with its own unicast address B as a source address back to the client.
4. The client finds that unicast address B is for the anycast address M. Then, it modifies the server address from M to B.
5. The client opens the connection to the server with unicast address B.

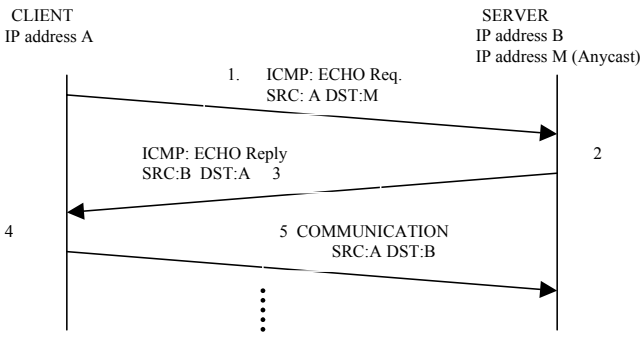

Figure 2.2: TCP Connection Establishment by "Anycast Address Mapper"

Different from the Source Identification Option mechanism, the Anycast Address Mapper is implemented as a daemon in the user memory space. The implementation modifies the application while the transport layer and IP layer remain unchanged.

The authors of [OE00] did two experiments, HTTP proxy service and DNS service, and the results confirm that the proposed mechanisms work correctly. The authors also pointed out that during the development of IPv6, using the "Anycast Address Mapper" as the anycast address resolving mechanism is better than the "Source Identification Option", because the implementation cost of the former is much lower than the later. On the other hand, after the development, it is preferable to use the "Source Identification Option" as the anycast address resolving mechanism because it needs fewer transactions.

Katabi and Wroclawski [KAT00] considered the traditional belief that IP-anycast should be routed similarly to unicast had hampered the acceptance and deployment of anycast. The anycast routing protocol should rather recognize the characteristics of IP-anycast and benefit from them to scale. Forcing anycast to obey the unicast routing paradigm wastes routing resources. Moreover, it is likely that at any given time there is a predictable set of anycast groups that users in a domain access with high probability, and that this set is much smaller than all anycast groups on the entire Internet. Based on this knowledge, they proposed a scalable architecture for global IP-anycast (GIA in short). Their design scales by dividing inter-domain anycast routing into two components. The first component builds inexpensive default anycast routes that consume no bandwidth nor storage space. The second component, controlled by the edge domains, generates enhanced anycast routes that are customized according to the beneficiary domain's interests. They conducted simulations to test performance, and the result shows that their design worked well. The authors also proved that their theory is practical by implementing the GIA in the multi-threaded routing toolkit.

[TAN03] presented that there are two type of anycasting: single-path and multi-path. The paper studied the difference between them in terms of performance, the simulations show that: the behaviour of the single-path anycasting and multi-path anycasting dependents on the source-receiver distribution very much; the multi-path anycast routing is more resilient than the single-path anycast routing when there is a rapid increase of traffic.

The originally proposed anycast is based upon the IPv6 platform, an extension of the IPv4 architecture, therefore its architecture inherits the same defects of the current Internet architecture.

[KAT00] addresses some issues about the future IP anycasting, which breaks through the existing unicast frame, and proposes a two-layer IP anycasting architecture. But several problems need to be solved in the architecture: such as, 1) The acceptance of the metric for deciding an anycast group, and 2) The compatibility of the proposed architecture with the current Internet architecture.

[OE00] points out a bug in the anycast service, and tries to fix it with two practical mechanisms for anycast on the current Internet with limited modification. It combines the protocols of the current Internet, therefore, inevitably, the architecture inherits the disadvantages of the current Internet. Moreover, protocol stack implementation requires further research in a number of areas, such as scalability, performance, security, and so on.

2.1.2 The Network-Layer Anycast Routing Algorithms

A. The RTT-based Server Selection Policy

Yamamoto, et al. [YAM01] [MIU01] presented a server selection policy for anycasting by RTT (Round Trip Time). The RTT consists of server processing delay and network delay for requesting and replying to data traversing in the network. In the active anycast, it is necessary that an active router selects an adequate server taking into account not only server load but also network latency. They proposed the probabilistic server selection based on a measured RTT at an active router. An active router measures the RTT from itself to servers by monitoring the request and replying data. Then, the active router calculates the probability of server selection based on the measured RTT. When an RTT is high, selection probability is small in inverse proportion to the RTT. With that probability, the active router decides which server to be accessed. An active router calculates P_s, a probability of selecting server s, as follows.

$$P_s = \frac{\frac{1}{RTT_s}}{\sum_{i=1}^{n} \frac{1}{RTT_i}},$$

Where n is the total number of servers serving the same service and RTTs is the RTT between the active router and server s.

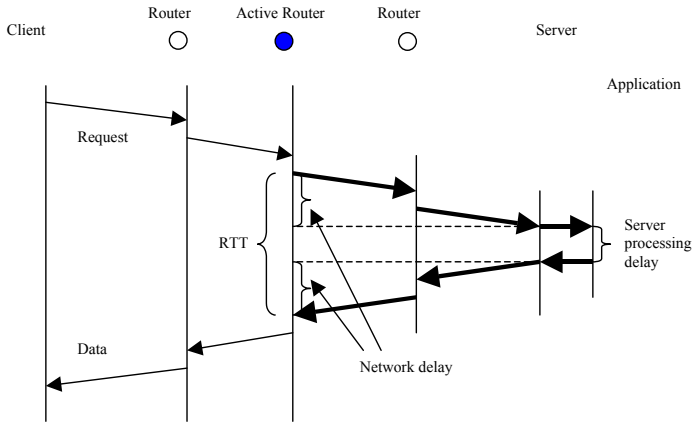

Figure 2.3 RTT Measurement

However, Figure 2.3 shows that a measured RTT at the active router is not equal to the one on the client's side, it takes the RTT (the bond line part) of the active router as the RTT of the client. This means that this RTT does not take the congestion between the client and the active router into account.

The authors conducted a simulation, which shows that the performance of RTT-based algorithm is much better than a server-load-based algorithm. Moreover, with added active routers, an RTT-based algorithm can also deal with the congestion between active routers and clients.

B. A Routing Protocol for Anycast Message

Xuan, Jia et al [XUA00] proposed and analysed a routing protocol for an anycast message. It is composed of two sub-protocols: the routing table establishment sub-protocol and the packet forwarding sub-protocol. In the routing table establishment sub-protocol, they proposed four methods, the Shortest-Shortest Path Method (SSP),

the Minimum Distance Method (MIN-D), the Source-Based Tree Method (SBT), and the Core-Based Tree Method (CBT) for enforcing an order among routers for the purpose of loop prevention. The methods differ from each other by the information used to maintain orders, the impact on QoS, and the compatibility to the existing routing protocols. In the packet forwarding subprotocol, they proposed a Weighted-Random Selection (WRS) approach for multiple path selection in order to balance network traffic. In particular, the fixed and adaptive methods are proposed to determine the weights. Both of them explicitly take into account the characteristics of the distribution of anycast recipient group while the adaptive method uses the dynamic information of the anycast traffic as well. The correctness property of the protocols was formally proven. Extensive simulation was performed to evaluate their designed protocol. Performance data showed that the loop-prevention methods and the WRS approaches have great impact on the performance in terms of average end-to-end packet delay. In particular, the protocol using the SBT or CBT loop-prevention methods and the adaptive WRS approach performed very close to a dynamic optimal routing protocol in most cases.

Xuan and Jia [XUA01] [JIA04] studied Distributed Admission Control (DAC) procedure for anycast flows with QoS requirements. The paper is the first study that addresses the issue of providing QoS support to anycast. In the paper, they designed three algorithms for destination selection in distributed admission control: Even Distributed (ED), Weighted Distributed with route Distance and local admission History information (WD/D+H), and Weighted Distribution with route Distance and available Bandwidth information (WD/D+B).

The basic idea of ED is that all the members in an anycast group will have equal probabilities to be selected as a destination of any new incoming flow, then the weights can be described as below for K members in an anycast group.

$$W_i = 1/K, i = 1,2,...,K.$$

The other two algorithms are biased weighted assignment algorithms, and there are three important factors in the algorithms: route distance information, local admission history information, and route bandwidth information.

The WD/D+H algorithm is based on route distance and local admission history. For an Admission-Control router (AC-router in short), the information on admission history is represented as a list:

$$H =< h_1, h_2, ..., h_K >$$

Where K is the number of members in an anycast group; h_i corresponds to the admission history of destination i. The value of h_i is defined as follows. At the time of initialisation, $h_i = 0$. Every time destination i is selected, the following update is made on h_i:

$$h_i = \begin{cases} 0, & if\ resourse\ \ reservation\ returns\ \ SUCCESS \\ h_i + 1, & otherwise. \end{cases}$$

That is, the value of h_i records the number of the continual failures in the most recent admission history.

At the system initialisation, the weights are assigned according to the following formula,

$$W_i = \frac{1/D_i}{\sum_{j=1}^{K} 1/D_j}$$

Where D_i is the value of the distance of the route leading to destination i.

With the processing going on, weights are updated using the following formula:

$$W_i^{'} = \begin{cases} W_i * \alpha^{h_i}, & if \quad h_i \neq 0; \\ W_i + \dfrac{\sum_{i=1}^{K} W_i * (1-\alpha^{h_i})}{M}, & otherwise, \end{cases}$$

Where α is the parameter used to adjust the impact of local admission history on weight assignment, M is the total number of destinations whose records in list H are 0.

The WD/D+B algorithm assumes that the bandwidth usage of links on routes to all destinations is available. Let r be the route from the source to destination i, Route Bandwidth B_i is defined as:

$$B_i = \min_{l \in r}(AB_l), \text{ where } AB_l \text{ is the available bandwidth of link } l.$$

The weight assignment of WD/D+B is listed as follows:

$$W_i = \frac{B_i / D_i}{\sum_{i=1}^{K} B_i / D_i},$$

Where B_i and D_i are route bandwidth and route distance of the route from source to destination i.

The authors evaluated the proposed mechanisms by mathematical analysis and computer simulation. Performance data demonstrated that in terms of admission probabilities, the heuristic DAC can perform close to those that utilize the global and dynamic status information of network, which is much more expensive and difficult to realize.

C. Anycast in Wireless Ad Hoc Networks

Ko and Vaidya [KO00] extended the Temporally Ordered Routing Algorithm (TORA in short) [PAR97] to an algorithm called GeoTORA for anycasting routing. TORA is one of a family of link reversal algorithms [GAF81] for routing in ad hoc networks. For each possible destination in the ad hoc network, TORA maintains a destination-oriented directed acyclic graph (DAG). In this graph structure, starting from any node, if links are followed in the logical direction of the links, the path leads to the intended destination.

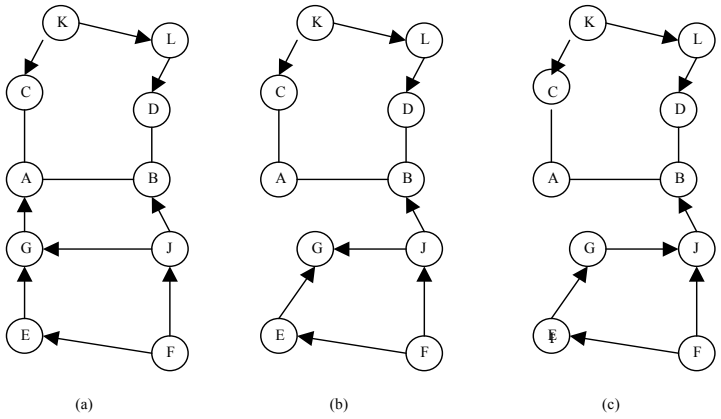

Figure 2.4 Anycasting Using Modified TORA

Figure 2.4 illustrates the anycast scheme of GeoTORA. In this example, nodes A, B, C and D belong to an anycast group. The present DAG structure is shown in Figure 2.4 (a), each node outside the anycast group can reach one of the anycast group members by the DAG. Now, suppose the link (G, A) is broken, the resulting DAG structure is shown in Figure 2.4 (b). Observe that now node G does not have any outgoing link. In response, the logical direction of the link (G, J) is reversed, resulting in the DAG shown in Figure 2.4 (c). Now all nodes that are outside the anycast group have an outgoing link (and a path to at least one node in the anycast group).

[CHO04] extended anycasting into MAC layer because the authors thought that choosing a single optimal route at the network layer may not be sufficient, and the knowledge of short-term channel conditions at the MAC layer could play an important role in improving end-to-end performance. Therefore, the paper proposed the MAC-layer anycasting for ad hoc mobile networks, in which the network layer specifies multiple downstream nodes, and the MAC layer chooses a suitable node based on instantaneous network conditions. On the other hand, [BAS03] tried to explore anycsting on higher layer, such as application modelling in dynamic mobile ad hoc networks (MANETs); they proposed the Dynamic Task-Based Anycasting to execute distributed tasks on a MANET by means of dynamic selection of specific devices that are needed to complete the tasks. The simulation shows that the proposed protocol works very well.

D. Summary of Anycast Routing Algorithms

The RTT-based server selection policy for anycast [YAM01][MIU01] is simple and practical, and its performance is better than that of the server load-based algorithm and the network latency-based algorithm. Furthermore, it is not necessary to take care of all the responses from the servers except from the first response server, and it is not necessary to calculate the possibility. It is essential that the first response tells us which server among the anycast group is the “best” one.

The solutions presented in [XUA00] are practical for the IPv6, and they are simple and compatible to the current Internet and use a comparable amount of information with those protocols that are currently used to route other types (unicast and multicast

) of packets. Similar to the traditional routing algorithms in IPv4, it is passive and needs more consideration of QoS. Fortunately, [XUA01] [JIA04] discuss the anycast flow with QoS requirements by using the methodology of Distributed Admission Control. One result of this research is that it shows that the performance of DAC is very close to the performances of those algorithms, which handle global and dynamic status information, but the cost is dramatically reduced. [XUA01] [JIA04] mention only three aspects that affect anycast QoS: route distance, local admission history, and available bandwidth information. We should use more aspects of anycast QoS because of the increasing and different applications, such as accuracy, security, and so on.

GeoTORA [KO00] is a customized algorithm for a wireless network, in which the network topology is very dynamic, as a result, it is not compatible with the fixed Internet. On the other hand, it implies that the anycasting problem is an issue of Graph Theory. There are two sets in the graph, the anycast server set and the client set, the issue is how to find the "best" way between the two sets. It points out a very promising direction, in which we can apply their findings to solve anycast routing problems. [CHE04] tries to improve the anycasting at MAC layer in ad hoc network environment, and it is a good exploration, while [BAS03] tries to conduct anycasting at higher layer.

2.1.3 The Network-Layer Anycast Metrics

Hanna, Natarajan, and Levine [HAN01] examined the server selection metrics for anycast: hop counts, autonomous system (AS in short) count, round-trip times (RTT in short), transfer times of a 10k file, and random selection. Their study indicates that network-layer metrics – minimizing hop count and AS count – perform as poorly as

predicators of file transfer times, often as poorly as random selection. RTT times are also poorly-correlated to transfer times and are not accurate predicators. They presented a novel two-step server selection method: ping set pre-selection and transfer set pre-selection. Their results demonstrated that the pre-selection techniques perform the best and require the least work of all metrics. They designed transfer set and pings set, based on a server selection with a two-step process, in which, they first pre-selected a subset of 3-5 well-performing servers; then, they selected from among the servers in that subset for downloads during a 10-day period. Transfer set isolates subsets based on ping times and then 250k file transfer; ping set isolates subsets based on ping time only.

Guyton and Schwartz [GUY95] also explored the cost and effectiveness of a variety of approaches of server location techniques, including: reactive gathering, anycast, routing table polling, route probing, and hop count probing (triangulation). They uncovered a number of tradeoffs between effectiveness, network cost, ease of deployment, and portability across different types of networks, and concluded that: 1) from the perspective of efficiency, anycast is the clear winner among the five techniques. However, anycast needs the support of IPv4 routers in practice, and 2) The authors considered triangulation would be cost effective, actually, hop counts are not sufficient enough to provide good server location choices.

Metrics are important for evaluating proposed architectures, models, and algorithms. Anycast metrics are the extension and continuation of the traditional network metrics. Research in this area is limited. More research is needed if anycast is to be widely used in advanced networks.

2.1.4 The Network-Layer Anycast Applications

Researchers in IBM [BAS97][ENG98] studied using network layer anycast for load distribution on the Internet. In their papers, they investigated how the IP anycast service can be exploited by hosts connected to the Internet without significantly impacting on the routing and protocol processing infrastructure already in place. In particular, they studied how routers running RIP and OSPF manage routes and forward packets to anycast destinations. They proposed possible enhancements to routing and forwarding to fully exploit the potential of the anycast service. They also discussed the changes required in the host protocol stack which would enable applications to transparently use the anycast service. More specially, they proposed a scheme that limits the required changes to the IP layer processing and described its implementation in the AIX TCP/IP stack. Finally they considered an application example where anycast communication is used to distribute load among a set of mirror web sites. Using traces from the IBM Olympic web site they compared several different load distribution schemes, both in terms of number of connections served as well as number of bytes transferred.

Jia et al [JIA00] proposed the idea of integrating multicast service and anycast service for Internet service: a group of replicated (or mirrored) servers that provides anycast services may also provide multicast services and need multicast to consistently update, whereas anycast routing may help multicast request to reach the ‘nearest’ member in a multicast group. The authors examined the previous work on multicast, and presented three quantities of particular interest in characterising the performance of

routing algorithms: Transmission Delay (TD in short), Bandwidth Consumption (BC in short), and Traffic Concentration (TC in short). They found that the worse values of TD, BC and TC of the Core Based Tree (CBT in short) algorithm are due to the fact that the source nodes of the multicast packets outside the CBT tree always choose the core for transmission of multicast requests. The routing of the shortest path from source to the core is not adaptive. Therefore, they designed a dynamic anycast routing algorithm for efficient transmission of anycast messages over the Internet to a group of servers, namely, for a multicast group G, when the CBT tree is built, all ontree routers (including the core) are selected to join an anycast group with an anycast address to replace the role of the core. Besides this, the authors also designed other two algorithms: (1) Integrated anycast routing with core-based tree technique based on multicast routing algorithms taking advantage of short delay, high throughput and load sharing. (2) Fault-tolerant algorithms for both anycast and multicast routing using backup paths restoring techniques. The performance figures demonstrated the benefits of anycast routing in reducing end-to-end packet delay, and attaining load balance and fault-tolerance for multicast.

Jia, Zhou, and Kaiser [JIA01] proposed a novel efficient mobile multicast protocol (MMP in short), taking advantage of the anycast routing technology. The MMP has two aims: 1) using mobility agents and anycast group to help facilitate flexible connections for mobile nodes; 2) an anycast address is configured by a group of multicast routers on the subnet that are designed to support a specific multicast group. In the MMP protocol, a multicast router can configure its interface to route both multicast and anycast packets. Each mobility agent (MA in short) maintains four lists for the dynamic memberships of mobile nodes in multicast group G: 1) the

membership list, ML(G), contains the IDs of members in group G; 2) the visitor list, VL(G), records the IDs of foreign mobile nodes that belong to G that visit this MA. 3) the away list, AL(G), records the IDs of mobile nodes in G that have departed (or disconnected) from this MA; finally, 4) the tunnelling list, TL(G), records the IDs of foreign agents that are interested in the transmission/reception of multicast packets for G. The MMP is designed in three major phases that work interactively: initialisation phase, registration and membership phase, multicast transmission phase. The simulation showed that MMP is more efficient in terms of delivery delay and throughput of multicast packets than bi-directional tunnelling and remote subscriptions. [JIA02a] applied the similar idea with a hierarchical-tree to server for a Multiple Share-Tree (MST) in multicasting on the Internet.

Geocasting [NAV97] has been proposed as a mechanism to deliver messages of interest to all hosts within a given geographical region. Ko and Vaidya [KO00] combined anycast and flooding for geocasting. The details of the algorithm, GeoTORA, have been discussed previously in this chapter. In GeoTORA, TORA (unicast) routing protocol has been modified to perform anycast and local flooding has been utilized to limit the flood to a small region. The simulation shows that the integration of TORA and flooding can significantly reduce the geocast message overhead as compared to pure flooding and LBM scheme, while achieving a high accuracy of geocast delivery.

Anycast is the result of the requirements for Internet applications, therefore it may be applied to many areas. It is natural to apply anycast in load balancing, such as in [ER097], [ENG98], and other proposed models. There is a huge potential for anycast in

network layer multimedia applications, for example, Internet based conference, video, and so on, although there are no such publications so far based on our knowledge. [JIA01] and [NAV97] show that it is a good choice to apply anycast in wireless networks and ad hoc networks. According to [JIA01], it is also a good area of research to combine anycast with mobile agent technology. Combining different services together can provide better services, such as combining anycast with multicast as in [JIA00], and combining anycast and geocast as in [NAV97]. Concast [CAL00], a counterpart of anycast, which is a network layer service that provides many-to-one channels: multiple sources send messages toward one destination, and the network delivers a single "merged" copy to that destination. If we combine anycast with concast, the Internet can provide a new bi-directional anycast-similar service. Obviously, the application of anycast will become more popular, and we are sure that there will be other services proposed by researchers.

2.2 The Application-Layer Anycast

2.2.1 The Application-Layer Anycast Architecture

Bhattacharjee et al [BHA96] [BHA97] [FEI98] examined the definition and support of the anycasting paradigm at the application layer, providing a service that maps anycast domain names into one or more IP address using anycast resolvers. The motivation of the authors derived from the limitations of network-layer-supported anycasting:

1) Some part of the IP address space must be allocated to anycast addresses.

2) The use of anycast addresses requires router support.

3) The selection of the server to which an anycast datagram is sent is made entirely within the network with no option for user selection on input.

4) Consistent with the stateless nature of IP, the destination is determined on a per-datagram basis.

5) The network layer is able to efficiently determine the shortest paths, but no other metrics.

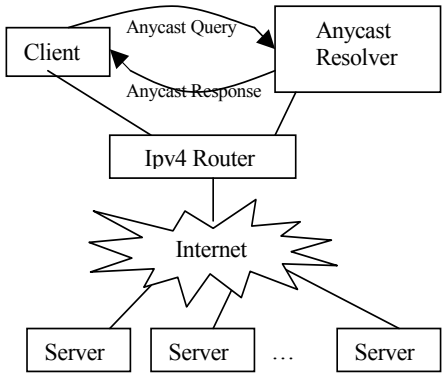

Figure 2.5. The Architecture of Application-Layer Anycasting.

They then defined the architecture of application-layer anycasting, as shown in Figure 2.5.

In Figure 2.5, a client tries to find a service from the replicated servers on the Internet. First of all, the client sends an anycast query to the anycast resolver to decide which server among the replicated servers is the "best". Then an anycast response is obtained, which consists of the "best" service server's website name or an IP address. The rest of the transaction is in the traditional unicast operations.

This model is the only one that we can find for application-layer anycast at present. As discussed this model overcomes some of the disadvantages of network-layer

anycast, and it is more feasible for use on the current Internet. There is an obvious problem for application-layer anycast architecture – all the processes are handled by one node. It is possible that a better solution would be one that distributes the anycast resolver into different places geographically or logically. As a result of such a distributed system, new issues arise: how to distribute the resolvers? How to synchronize the data among the resolvers? How to assure the accuracy for the clients? and so on. All these problems are worth investigation.

2.2.2 The Application-Layer Anycast Routing Algorithm

Chen and Mao [CHE01] investigated on how to provide anycast service in the application layer in a pure IPv6 environment.

First of all, the authors explored the Predicted Transfer Time (PTT) model,

$$PTT = K_1 \cdot RTT + K_2 \cdot (\frac{document\_size}{BW})$$

Where, RTT is the estimated Round Trip Time from client to server, document_size is the size of document the client wants to fetch, and BW is the bandwidth currently available on the links between client and server. K_1 and K_2 are the constant factors that can be calculated using some regression methods. Further more, the authors considered the Total Response Time (TRT), measured from the time the connection request is sent to the time the whole document is received, is usually a good metric for distance measurement. In practice, the client would prefer one of the replicated servers, which is located closer to it, since by going a long way through the backbone network to the

other server it could potentially face the risk of link failure and unexpected traffic congestion due to requests from other clients.

Based on these results, they also considered giving priority to servers located in the same subnet as the client localizing the traffic, which is introduced into the metric as a weight factor. The Weighted Total Response Time (WTRT) is decided as their distance metric for server selection.

$$WTRT = w[k_1 \cdot latency + k_2 \cdot (\frac{E(packets) \cdot P\_size}{BW})]$$

Where, the multiplicative factor w in front is the segregation weight (SW) to localize the traffic. The term inside the square bracket is an expression for the Total Response Time, where BW is the available bandwidth, E(packets) is the expected total number of sent packets considering packet loss and retransmission, P_size is the average TCP packet size, and K_1 and K_2 are constant linear coefficients.

The simulations showed that the WTRT model's performance is better than that of other models, such as the Predicted Transfer Time (PTT) model, and the Hop-count model. The WTRT model also has a large consideration of server load, network situations, traffic localization and client preference.

[CHE01] notices the difference between the server that is located in the same subnet and the server that is located in a different subnet with the client, and therefore give them different weight factors. The possibility that client and server that is located in the same subnet is very limited, especially in the environment of the Internet.

2.2.3 Application-Layer Anycast Applications

Wu [WU99] and his colleagues applied the application-layer anycasting model on video transmission, such as Video-On-Demand servers on the Internet. They presented an algorithm with two modules: 1) For the resolver, a table is constructed to contain system measures and characteristics referenced by each server ID in the table, and 2) the second module is implemented in each server, which needs a data structure to keep traffic data and system states for generating value for decision (VFD, in short). They specially utilise analytical techniques in economics and queuing theory, to efficiently use the system resources.

Application-layer anycast is suitable for Internet-based multimedia applications. [WU99] is a good beginning since Internet-based multimedia applications are becoming the dominant application of the Internet, we can therefore expect that anycast, both application-layer anycast and network-layer anycast, will attract more attention.

Chapter 3
Routing Issue in Anycast

The anycast communication paradigm is proposed in IPv6, and it is designed to support server replication by allowing applications to select and communicate with the "best" server, according to certain performance or policy criteria, among the replicated servers. Originally anycast researchers focused on the network layer. In this chapter we will concentrate more on application-layer anycast, because at application layer we can achieve more flexibility and scalability than that at network layer. First of all, we will describe the application-layer anycast model, and then summarize the previous work in application-layer anycast: the periodical probing algorithm for updating the database of the anycast resolver. After that, we present our algorithm, the requirement-based probing algorithm, an efficient and practical algorithm for anycast routing. In the end, we will analyse the algorithms using the queuing theory and the statistical characteristics of Internet traffic. The results show that the requirement-base probing algorithm has better performance not only in the average waiting time for all anycast queries, but also in the average time used for an anycast query.

3.1 Introduction

An anycast message is the one that should be delivered to one member in a group of designated recipients [PAR93]. Anycast is an extension of the traditional unicast message, in which there is only one server in the recipient group. With the dramatic development of the Internet, more and more applications demand anycast services, for example, when there are a number of mirrored web sites on the Internet, a user can access the "best" one transparently through an anycast service. As the result, in the latest version of the IP specification, IPv6, anycast has been defined as a standard service [DEE98]. Generally speaking, there are two categories of anycast related problems: procedures and protocols at network layer for routing and addressing anycast messages, and management methodologies at application layer for using anycast services [HIN95].

Some research has been carried out on routing anycast messages at the network layer [JIA00] [JIA01] [KAT00] [XUA00]. This research is very important for the next generation of IP, and the researchers also obtain some excellent achievements. To the various practical applications of the current Internet, however, network layer anycast routing has some inevitable disadvantages, which can be summed up as follows:

- Network layer anycast routing needs the support of routers. The original routers for IPv4 do not support anycasting, therefore we must upgrade routers, and make them recognize anycast addresses and forward packets properly. Furthermore, routers must coordinate with each other to complete the delivery of anycast packets correctly. As a result, anycast services in the network layer need a long and expensive transitional period.

- Network layer anycasting is not flexible. Routing of anycast packets are decided by the previous fixed routing algorithms residing entirely within the network, there is no possibility for users or developers to change the algorithms to meet their own special requirements.

- Network layer anycasting can not satisfy the various metrics of Internet applications. The current network layer anycasting algorithms focus on shortest path metrics, such as hop count. It is most effective to determine the shortest path, but only for this purpose. It can not handle a variety of other metrics, such as network performance, server throughput, response time, etc.

On the other hand, some research has been carried out on application layer anycasting [BHA96] [BHA97] [FEI98] [REN00]. These researches try to implement the functionalities of anycasting in application layer, which can avoid the disadvantages outlined previously, however, the application layer anycast routing methods still have some weak aspects in applications, such as the performance in the Internet environment, and so on.

There are a lot of researches have been done on the traffic characteristics of the Internet [CAO01] [PAX97] [PAX99] [SOL01]. In this chapter, we explore the algorithms on application-layer anycasting from the views of statistics and stochastic.

The rest of this chapter is organized as follows. Section 2 discusses related research on application-layer anycasting, and also the statistical features of the Internet. In section 3, after describing the previous research, we will present our novel algorithm

on application-layer anycasting. We will compare the performance of the application-layer anycasting algorithms in section 4. Finally, in section 5, remarks and future work will be presented.

3.2 Related Work and Background

3.2.1 Characteristics of Internet Traffic

Network traffic properties have been intensively studied for a quite long time. Examples of analysis of typical traffic behaviours can be found in [CAC89][PAX97].

Traffic variables on an uncongested Internet wire exhibit a pervasive nonstationarity. As the rate of new TCP connections increases, arrival processes (packet and connection) tend locally toward Poisson, and time series variables (packet sizes, transferred file sizes, and connection round-trip times) tend locally toward independent [CAO01]. Here the Poisson arrivals are given by

$$P\{X = k\} = \frac{\lambda^k}{k!} e^{-\lambda}, k = 0,1,2,... \qquad \text{.....(3.1)}$$

The analysis later in this chapter is based upon Poisson arrival, which is described in this formula.

The statistical properties of Internet congestion reveal long-tailed (lognormal) distribution of latencies [SOL01]. Here latency times T_L is given by

$$P(T_L) = \frac{1}{T_L \sigma \sqrt{2\pi}} \exp\left(- \frac{(\ln T_L)^2}{2\sigma^2} \right) \quad \text{……..(3.2)}$$

Where σ represents the load of the network. Latencies are measured by performing a series of experiments in which the round-trip times of ping packets are averaged over many sent messages between two given nodes.

3.2.2 The Application-Layer Anycasting Model

Application-layer anycasting [BHA96] [BHA97] is a research topic that began with the server or resource discovery problem. Initially, with low to moderate server loads, the problem was how to find the desired resource over the network knowing only its name or property. In 1999, the Service Location Working Group of the IETF considered the design of the Service Location Protocol, which allows a user to specify a set of service attributes, which can be bound to a server's network address in a dynamic fashion [BHA97].

With the dramatic development of the Internet, many users now have to constantly find the "best" service from among many content-equivalent servers. There are several outstanding studies in this area: 1) Partridge, Mendez and Milliken [PAR93] proposed the idea of anycasting at the network layer. 2) A study by Guyton and Schwartz [GUY95] which addresses the problem of locating the nearest server. 3) Xuan, Jia, Zhao and Zhu [XUA00] presented a simple and practical routing protocol for anycast message at the network layer. And 4) Zegura and the research group [BHA96]

[BHA97] analysed the limitations of network layer anycasting, which also can be found in chapter 2, and offered an idea about application layer anycasting.

The architecture of application-layer anycast routing has been shown in Figure 2.5 in the previous chapter. There is an anycast resolver, which stores the information about the "best" servers among each anycast group, and it works on the application layer. All the work is based on the underneath IPv4 network.

The procedure of an anycast service is listed as below:

1. A client initials an anycast requirement and sends an anycast query to the anycast resolver to decide which server among the replicated servers is the "best".

2. The resolver searches its database to find the "best" server for the query based on its information in the database or its probing.

3. An anycast response is obtained, which consists of the "best" service server's website name or an IPv4 address, and feed back to the client.

4. The client tries to connect to the given server using the traditional IPv4 mechanism, and finish the transaction on an IPv4 network.

An anycast resolver is the kernel of the application layer anycast architecture. It makes use of the anycast domain names (ADNs). The function of an application-layer

anycast service is to map an anycast domain name into one or more (unicast or multicast) IP addresses.

3.3 Algorithms for Application-Layer Anycasting

3.3.1 Periodical Probing Algorithm

The critical problem of application-layer anycasting is how to map an anycast query into one or more IP addresses. [BHA97] presents 4 metrics on how anycasting performs: 1) server response time, 2) server-to-user throughput, 3) server load, and 4) processor load. Here we identify four possible approaches to maintaining replicated server performance information in the anycast servers' database:

1. Remote Server Performance Probing: By this methodology, probing agents query the replicated servers periodically to determine the performance that will be experienced if a client were to actually request service.

2. Server Push: The replicated server sends (or pushes) the relevant local performance information onto anycast resolvers.

3. Probing for Locally-Maintained Server Performance: Each replicated server maintains its own locally monitored performance metrics in a globally readable file, remote probing locations can then read the information in the file to obtain the desired message.

4. User Experience: This technique is to collect information on past experience, and then offers a coarse method of maintaining server performance.

As described in [BHA97], the foundation of anycast resolver algorithms is the remote server performance probing based on periodical probing, the periodical probing algorithm (PPA in short). [BHA97] mixed the different methods together in practical applications. There are several disadvantages of the periodical probing algorithm:

- Accuracy problem. We suppose that the period of probing is ΔT, then during ΔT time, the anycast resolver makes all its decisions based on the result of the last probing. As we know that the Internet changes quickly, therefore the longer the ΔT is, the worse the accuracy is.

- Network load problem. In order to improve accuracy, the ΔT should be as short as possible, but, on the other hand, there must not be too many probing packets, as this will generate a heavy network load.

- Completeness problem. Periodical probing can represent the performance of the servers, but it can not tell the resolvers the performance of the current network circumstance, which is also an important element of the whole performance.

- Resolver server load problem. The periodical probing algorithm probes for all anycast groups, including the anycast groups which are not used in the coming period. This part of job is not necessary, and it can degrade the resolver's performance.

3.3.2 Requirement Based Probing Algorithm

Now we propose a new algorithm, the requirement-based probing algorithm (RPA in short), which can overcome the disadvantages of the periodical probing algorithm, which is mentioned in section 3.3.1. The main propose of requirement-based probing algorithm is described below.

1. An anycast query is initiated by a client, and then the query is submitted to the anycast resolver.
2. Once the anycast query is received by the anycast resolver, the resolver will send probing packets (ping packets in our case) to each member in the anycast service group, respectively.
3. In this case, the probed servers will respond to the ping requirements, respectively.
4. The server, whose respond packet is the first packet received by the resolver is defined as the "best" server.
5. The "best" server's address is feed back to the client.
6. The transaction will be finished based on the traditional IPv4 mechanism.

List 3.1 shows the pseudo-code of the algorithm.

The Requirement-base Probing Algorithm

```
Initialize the anycast groups;

If (newQuery is true) then
//extract the anycast group ID
  GID = AnycastGroupID (newQuery)
 //extract the member of the group
   GNumber = AnycastGroupMember(GID)

For ( i = 1; i <= GNumber; i ++)
{
  ping ( Anycast Server i );
}

//Determine the "best" server
Do While ( True )
{
If (Response(GID) <> null ) then
  {
  BestServer = IPExtract (GID, ResponseInformation );
  Break;
  }
continue;
}
```

List 3.1 The Requirement-base Probing Algorithm

We define the "best" server by the following analysis: If a server's load is heavy or performance is bad, then the response must last longer than a server whose load is lighter or performance is better. Therefore the probing packets can not only probe the servers' load or performance during that short period, but also the network load during the same period. Based on this analysis, we define that the first responsive server is the "best" one among the anycast service group, because the response time represents the network performance and the server performance.

The advantages of our algorithm include higher accuracy, better system performance, and less load for both network and resolvers than the periodical probing algorithm. It is

also practical and easy to implement. In section 3.4, we will present performance comparisons of the application-layer anycast algorithms.

3.4 Performance Comparison of Anycast Algorithms

In this section, we compare the two algorithms described in section 3.3 based on statistical characteristics of Internet traffic and queuing theory. There are some assumptions in the analysis:

1) Customer arrivals are Poisson arrival.

2) The time unit for both algorithms is 1.

3) During the time unit, there are N customers for both algorithms.

4) There is one server in the system acting as the resolver, and the service velocity, μ, can be obtained from the formula (3.3).

$$\mu = \frac{1}{E(x)} = e^{-\frac{1}{2}\sigma^2} \quad \text{.....(3.3)}$$

There are two important parameters to measuring the performance of a system. One is the average time used in the system for a query, denoted as T_q. Another one is the average waiting time for all queries, denoted as T_w. For both algorithms, we will calculate these two parameters respectively.

3.4.1 Performance of the Periodical Probing Algorithm

For this algorithm, we make the following two reasonable assumptions:

1) There are two segments in one period, p and 1-p, as shown in Figure 3.1. During time points 0 and p, there is no query arrival; during time points p and 1, there are N query arrivals, and the rule is Poisson arrival.

2) During time points 0 and p, the anycast resolver provides service to clients, and during time points p to 1, there is no service to clients. During its duration, the anycast resolver updates its database.

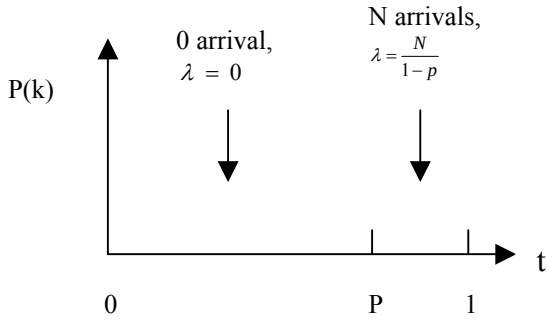

Figure 3.1 Time Segments Assumption for Periodical Probing Algorithm.

We can obtain the following results using queuing theory.

During time points p to 1:

The Poisson arrival velocity λ,

$$\lambda = \frac{N}{\Delta T} = \frac{N}{1-p} \quad \text{……..(3.4)}$$

Combining (3.3) and (3.4), we can obtain ρ, ratio of usage.

$$\rho = \frac{\lambda}{\mu} = \frac{N}{1-p} e^{\frac{1}{2}\sigma^2} \quad \text{……..(3.5)}$$

Further,

$$T_{qp2} = \frac{\frac{1}{\mu}}{1-\rho} = \frac{e^{-\frac{1}{2}\sigma^2}}{1-\frac{N}{1-p}e^{\frac{1}{2}\sigma^2}} \quad \text{……..(3.6)}$$

$$T_{wp2} = \rho.T_{q2} = \frac{\frac{N}{1-p}e^{\sigma^2}}{1-\frac{N}{1-p}e^{\frac{1}{2}\sigma^2}} \quad \text{……..(3.7)}$$

During time points 0 to p:

$\lambda = 0$, then $\rho = 0$, and further,

$T_{qp1} = 0$ ……..(3.8)

$T_{wp1} = 0$ ……...(3.9)

Then the weighted average for T_{qp} and T_{wp} are:

$$T_{qp} = pT_{qp1} + (1-p)T_{qp2} = \frac{(1-p)e^{\frac{1}{2}\sigma^2}}{1-\frac{N}{1-p}e^{\frac{1}{2}\sigma^2}} \quad \text{…….(3.10)}$$

$$T_{wp} = pT_{wp1} + (1-p)T_{wp2} = \frac{Ne^{\sigma^2}}{1-\frac{N}{1-p}e^{\frac{1}{2}\sigma^2}} \quad \text{…….……(3.11)}$$

3.4.2 Performance of the Requirement-Based Probing Algorithm

For the requirement-based probing algorithm, the Poisson arrival velocity is,

$$\lambda = \frac{N}{\Delta T} = \frac{N}{1} = N \quad \text{……..(3.12)}$$

The service velocity is the same one described by formula (3.3), then

$$\rho = \frac{\lambda}{\mu} = Ne^{\frac{1}{2}\sigma^2} \quad \text{……..(3.13)}$$

Further,

$$T_{qr} = \frac{\frac{1}{\mu}}{1-e} = \frac{e^{\frac{1}{2}\sigma^2}}{1-Ne^{\frac{1}{2}\sigma^2}} \quad \text{…….. (3.14)}$$

$$T_{wr} = \rho T_{qr} = \frac{\rho}{\mu(1-\rho)} = \frac{Ne^{\sigma^2}}{1-Ne^{\frac{1}{2}\sigma^2}} \quad \text{…… (3.15)}$$

Now, we can derive two conclusions.

Conclusion 1. $T_{qr} < T_{qp}$ **,** $t \in (0, p_e)$

Based on formula (3.10) and (3.14), we can get the curves shown in Figure 3.2.

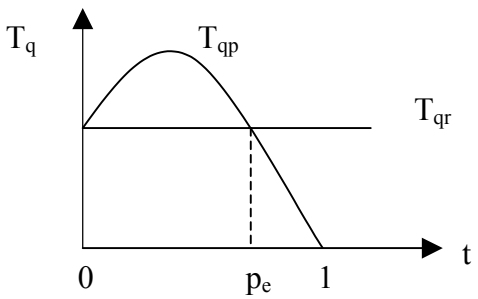

Figure 3.2 The Comparison of T_{qp} and T_{qr}

Where

$$P_e = \frac{1 - 2Ne^{\frac{1}{2}\sigma^2}}{1 - Ne^{\frac{1}{2}\sigma^2}} \quad \text{..... (3.16)}$$

Figure 3.2 shows that: if P locates in (0, p_e) then $T_{qr} < T_{qp}$, and if p locates in ($p_{e,1}$] then $T_{qr} > T_{qp}$. That means when the network load becomes heavy ($\sigma \uparrow$), or there are more customers ($N \uparrow$), or both of these events happen, then P_e becomes smaller. That is when the above situation(s) happen, in a system's view, T_{qp} is less than T_{qr}, but in practice, we hope that P_e is close to time point 1, that means we hope that the resolver's database update period is only a small part of the whole time unit, because during [P_e, 1], the resolver will focus on database updating, therefore the performance of the service is poor. Based on this analysis, generally speaking, in most of the time unit, (0, P_e), the performance of the requirement-based probing algorithm is better than that of periodical probing algorithm; only in a very small part of the time unit, (P_e, 1), will the performance of the requirement-based algorithm be worse than that of the periodical probing algorithm.

Conclusion 2: $T_{wr} \le T_{wp}$

p = 0 then $T_{wr} = T_{wp}$. This is easy to obtain.

p > 0 then $T_{wr} < T_{wp}$

Proof:

p >0 $\Rightarrow$ -p <0

$\Rightarrow$ 1 – p < 1

$$\Rightarrow 1 < \frac{1}{1-p}$$

$$\Rightarrow -1 > -\frac{1}{1-p}$$

$$\Rightarrow 1 - Ne^{\frac{1}{2}\sigma^2} > 1 - \frac{N}{1-p}e^{\frac{1}{2}\sigma^2}$$

$$\Rightarrow \frac{1}{1 - Ne^{\frac{1}{2}\sigma^2}} < \frac{1}{1 - \frac{N}{1-p}e^{\frac{1}{2}\sigma^2}}$$

$$\Rightarrow \frac{Ne^{\sigma^2}}{1 - Ne^{\frac{1}{2}\sigma^2}} < \frac{Ne^{\sigma^2}}{1 - \frac{N}{1-p}e^{\frac{1}{2}\sigma^2}}$$

$$\Rightarrow T_{wr} < T_{wp}$$

Done #

This shows that T_{wr} is always less than or equal to T_{wp}, namely the average waiting time of the requirement-based probing algorithm is always less than or equal to that of the periodical probing algorithm.

3.5 Summary

With the dramatic development of the Internet, anycasting will become an important part of forthcoming applications. Initially, anycasting was presented in the network layer. Application-layer anycasting is a practical choice for many current applications.

In this chapter, we described the application-layer anycast model and the current algorithms for the critical part of anycasting, namely how to decide the "best" service server. We proposed our requirement-based probing algorithm. The analysis shows that: 1) the average waiting time for all anycast queries of the requirement-based probing algorithm is better than that of the periodical probing algorithm. 2) Generally speaking, in normal network situations, the average time used in a system for an anycast query of the requirement-based probing algorithm is better than that of the periodical algorithm, except when the network load is very heavy, or there are too many anycast queries, or both.

Chapter 4
Fault Tolerant Issue in Anycast

Plenty of research has been done on anycast service, but there is very little research on the fault-tolerant problem to the best of our knowledge. In this chapter, we propose and analyse a fault-tolerant model, the twin server model, for anycast communication to provide reliable and continuous anycast services. We select a twin server among an anycast group for a given anycast server (the primary server). If the twin server suspects that its primary server is dead, it will take the unfinished job(s) of its primary server. We propose two algorithms: the server failure detecting algorithm and the server failure broadcasting algorithm for the proposed model. We will then analyse the performance changes when a primary server fails using queue theory and obtain some interesting conclusions.

4.1 Introduction

With the dramatic development of computer network technology, a lot of new application requirements appear, and researchers are trying to develop new protocols and models to meet its ever increasing and changing requirements. Partridge, Mendez, and Milliken [PAR93] originally proposed the idea of anycasting in the network layer. They defined IP anycasting as a service to deliver an anycast datagram to one of the members of an anycast group. The idea of anycast meets the requirements of mirrored

or replicated servers in the Internet, therefore a lot of research is rapidly being conducted in this area.

So far, a number of algorithms [KO00] [XUA00] [XUA01] [JIA04] and models [KAT00] [OE00] have been designed, but to the best of our knowledge, few of these touches the fault-tolerant issue of anycast servers. Fault-tolerant distributed systems are designed to provide transparent, reliable and continuous service despite the failure of some of its components. Anycast servers are mirrored and distributed servers in the Internet environment. As we know, the Internet is dynamic and unstable with possible server crashes and link failures, therefore an anycast service needs reliable and continuous service guarantee for anycast users. In this chapter, we will explore the fault-tolerant issue of anycast servers.

The remainder of the chapter is organized as follows. Section 4.2 introduces related works. In section 4.3, we present a fault-tolerant model for anycast communication. We propose two algorithms, server failure detecting algorithm and twin server selecting algorithm in section 4.4, and we will also present a queueing theory based analysis for the server failure case. Finally, section 4.5 summaries the chapter.

4.2 Related Work of Anycast Fault Tolerance

The topic of fault tolerance for distributed systems has been explored for many years. [CHA96] introduced the concept of unreliable failure detectors and studied how they can be used to solve the consensus problem in asynchronous systems with crash failures. The paper characterised unreliable failure detectors in terms of two properties:

completeness and accuracy. The study showed that the consensus problem could be solved even with unreliable failure detectors that made an infinite number of mistakes, and which detectors could be used to solve consensus problems despite any number of crashes, and which detectors required a majority of correct processes.

[CHE00] studied the quality of service (QoS) of failure detectors. The paper focused on two issues in terms of QoS: a) how fast the failure detector detects actual failures, and b) how well it avoids false detections. The paper first proposed a set of QoS metrics to specify failure detectors for systems with probabilistic behaviours, such as the detection time for how fast a detector detects crashes, and query accuracy for how well a detector avoids mistakes. The paper then presented a new failure detector algorithm and analysed its QoS in terms of the proposed metrics.

[ZHOU97] researched the fault-tolerance problem in the scenario of distributed operating system, and tried to provide continuous services in the case of a server or even a host failure, with or without impact on the whole distributed system. For each service, two servers (twin servers) are maintained to provide the fault-tolerance service. If one server dies, its twin will continue its job, and select a new twin. The background of the research is that the servers are located "near" each other, such as in a local area network and all the twined servers are symmetric computers.

[JIA00] investigated the problem of fault-tolerance for multicast and anycast algorithms. It is possible that the routers/links for anycasting may fail or become disconnected, therefore the paper proposed an offtree fault-tolerant subprotocol. The subprotocol is responsible for detecting the related faults and for reconfiguring the

anycast path once faults are detected, then when a fault occurs, there is an alternative path to bypass the failed component.

[FRI03] proposed three models to deal with server failures in video-on-demand system. The models guarantee continuous streaming to clients despite server failures while utilizing very low network's bandwidth and a small client's buffer. Fault tolerance issue is researched a lot in wireless environment as well, such as routing [DAT03] [ZHE02], server crash [CHE02], etc.

In this chapter, we extend the twin server model [ZHOU97] from the local area network to the Internet environment for anycast communication, furthermore, we propose two algorithms: the server failure detecting algorithm and the server failure broadcasting algorithm for the model in the Internet environment. We apply the queueing theory to analyse the changes that occur when an anycast server failure occurs, and obtain some interesting results.

4.3 The Fault-Tolerant Server Model for Anycast Communication

In order to provide transparent and high performance services, we design a fault-tolerant server model for anycast communication. For each server in an anycast group, say S_P, we try to find a backup server, say S_T, among the other anycast servers in the same anycast group of S_P. Once S_P fails, then S_T will continue the unfinished services of S_P. We name server S_P as the primary server, and server S_T as the twin server.

For each service, the two servers (S_P and S_T) are maintained to provide the fault-tolerant service. The proposed model is shown as Figure 4.1.

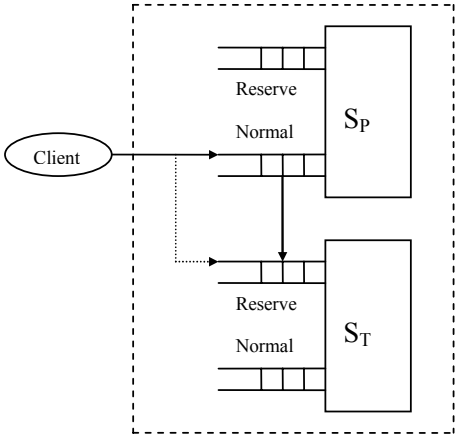

Figure 4.1 The Fault-Tolerant Server Model

For each server, there are two queues, the normal queue and the reserve queue for the incoming requests. If there is an anycast request for server S_P, then the request will be stored in the normal queue of server S_P, at the same time, a copy of the request will be sent to S_P's twin, S_T, and be stored in the reserve queue of S_P. each server takes the requests from its normal queue and executes the requests respectively. There is a pointer from the normal queue of the primary server to the reserve queue of the twin server to indicate the progress of the execution of the requests in the primary server. Once a request is executed successfully by the primary server, the request will be deleted from the reserve queue of the twin server through moving the pointer. Once the twin server finds the primary server is down, then it will push the request(s) in the reserve queue into its normal queue and continues the unfinished service(s) of the primary server.

We propose a server failure detecting algorithm for detecting the crash of an anycast server in the next section. An interesting topic is also how to find the twin server for a given server in the anycast group. Obviously, we need to find the "best" server in the anycast group as a twin server for a given server, therefore we propose an anycast based algorithm to carry out this job in section 4.4 as well.

4.4 Algorithms and Performance Analysis

4.4.1 The Failure Detecting Algorithm and Its Performance Analysis

In this section, we present a novel algorithm for server failure detecting, and then analyse the performance changes when a server failure happens. We hide the "*I am alive*" message for server failure detecting in the job progressing messages for each executed request in the primary server.

Algorithm 1. The Server Failure Detecting Algorithm

The purpose of the sever failure detecting algorithm is to synchronize the normal queue of the primary server and its counterpart reserve queue of the twin server, and to detect the crash failure of the primary server. The algorithm is shown in list 4.1.

The Algorithm at the Primary Server (Sender)

T_s = the mean service time for requests of primary server;
P = the pointer for the normal queue;

```
t = 0; // t is the service time for a request
while ( True )
        if ( t < 2Ts) then
            MessageSend (p)
             t = 0;
        endif
        if ( t >= 2Ts ) then
            MessageSend (" I am alive");
             t = 0;
       endif
end while.
```

The Algorithm at the Twin Server (Receiver)

T_s = the mean service time for requests of primary server;
P = the pointer for the reserve queue;
T_d = the network delay between the two servers;

```
t = 0; // t is the interval between two coming messages
while (True)
      if ( t < 2Ts + Td) and  (MessageReceive () <> null) then
        p = MessageReceive (); // update the pointer
        t = 0;
       end if
      if  ( 2Ts + Td <= t < 2*(2Ts + Td) and (MessageReceiver() = " I am alive" ) then
         t = 0;
      end if
       if ( t > = 2* (2Ts + Td)) and ( MessageReceive() =  null ) then
      // suspect that the primary server is dead
          QueueAppend (Normal Queue + Reserve Queue)
end while
```

List 4.1. Algorithms for Server Failure Detecting.

The main idea is that the primary server sends messages to the twin server, and the twin server decides to trust the primary server (*the primary server is alive*) or suspect the primary server (*the primary server is dead*). In order to avoid a request is executed twice by the primary server and the twin server respectively, once a request is finished successfully, the primary server will send that information to the twin server immediately. Therefore the frequency of that kind of message transmission is high. If the twin server gets one of those messages, then it can be sure that the primary server is

alive. It is possible that there is no message transmission of any kind for a long time. For this reason, we set two timers in the primary server and the twin server respectively to calculate time consuming. Once a request is processed in the primary server, the server will send a message about the pointer to the twin server, and the later will adjust its pointer of the reserve queue. If a request's service time is longer than a given time (*2Ts in this chapter, Ts is the mean service time of the primary server*), then the primary server will send a message (*I am alive*) to the twin server to hint that the primary is alive. On the other hand, if the twin server does not receive a message from the primary server for a long time (*2*(2Ts+T_d) in this chapter, T_d is the average network delay*) then it will suspect the primary server and it will take over the primary server's unfinished duties.

Analysis 1. Server Failure Performance Analysis

Network traffic properties have been intensely studied for quite some time, and examples of analysis of typical traffic behaviours can be found in [CAC89] [CAO01] [PAX97]. These papers indicate that most customer requests obey Poisson distribution. Here the Poisson distribution is given by

$$P\{X = k\} = \frac{\lambda^k}{k!} e^{-\lambda}, k = 0,1,2,... \quad \text{..... (4.1)}$$

We model our fault-tolerant model using the queueing theory, as shown in Figure 4.2. Where S_1 is the primary server, and S_2 is the twin server, then Q_1 and Q_2 are the normal queues for S_1 and S_2 respectively. Generally speaking, S_1 serves all incoming

requests with parameter λ_1, and S_2 serves all incoming requests with parameter λ_2. When S_1 crashes, the remaining request in Q_1 will be moved to Q_2 to be served. The specifications of the related parameters are described below.

λ_i , $i = 1, 2$. arrival rate of Poisson arrival.

μ_i, $i = 1, 2$. mean service rate for each arrival.

T_i, $i = 1, 2$. mean service time for each arrival.

T_{qi}, $i = 1,2$. mean time a request spends in the system.

Here, $\mu_i \cdot T_i = 1$. The service time T_i includes two parts: the average network delay of traffic and the average computing delay of the server.

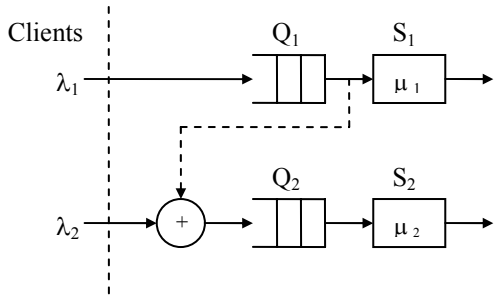

Figure 4.2 Fault-Tolerant Server Model for Anycast Messages

As we know that anycast mechanism has the property of automatic load balance among the anycast group, therefore, we will first study the mathematic expression for the model.

Without loss generality, the proofs in this chapter will focus on the scenario of two computers as default.

Assertion 1. If the work loads of n computers ($n \geq 2$) are balanced, then in a given period [*0, T*] (*T* is sufficiently big), the sums of the related service time T_s of each computer are equivalent.

Proof: For the term of a long time, we can express assertion 1 as the following equations,

$$\int_0^T \frac{\lambda_1}{\mu_1} dt \quad = \quad \int_0^T \frac{\lambda_2}{\mu_2} dt \qquad \text{or} \qquad \sum_{i=1}^{k} T_{s_1 i} = \sum_{i=1}^{m} T_{s_2 i} \quad \text{.....(4.2)}$$

Where *k* and *m* are two large integers; $T_{s_1 i} \quad (i = 1,2,...,k)$, and $T_{s_2 i} \quad (i = 1,2,...,m)$ are the individual service time of T_s for server 1 and server 2, respectively. There are three cases in favour of the issue as listed below, any other situations are the combination of the three of them.

Case 1. There are no requests in the two queues and $\lambda_1 = \lambda_2 = 0$ for the period [*0, T*].

It is obvious that the equations are correct.

Case 2. $\lambda_1 > \mu_1$ and $\lambda_2 > \mu_2$ for the period [*0, T*].

This means that both arrival rates are bigger than the service rates respectively, namely, the two servers are busy for the whole period,

$$\sum_{i=1}^{k} T_{s_1 i} = T\ , \quad \sum_{i=1}^{m} T_{s_2 i} = T$$

The assertion is correct.

Case 3. Without loss generality, suppose there is no request in Q_1 and there is/are one or more request(s) in Q_2 at a given point time $t, t \in [0, T]$.

Because of load balance, if a new request comes, the request will be dispatched to Q_1, this situation may happen from time to time. Therefore, if T is sufficiently big, the assertion is correct.

Each of the three cases are correct, therefore the assertion is correct for any combination of them. For the case of $(n > 2)$, the proof is the same.

Assertion 2. If the work load of n computers $(n \geq 2)$ are balanced, then in a given period [*0, T*] (T is sufficiently big), the ratios of the arrival rate to the service rate for each computer are the same.

Proof: Based on equation (4.2), it is easy to obtain the following result.

$$\frac{\lambda_1}{\mu_1} = \frac{\lambda_2}{\mu_2} \quad \text{…….. (4.3)}$$

This assertion implicates the relationship between the arrival rate and the service rate is fixed when the load of the system is balanced.

Assertion 3. If the work load of n computers ($n \geq 2$) are balanced, then in a given period [*0, T*] (*T* is sufficiently big), the relationship between T_q, *mean time a request spends in the system*, and the arrival rate λ is reciprocal.

Proof: Based on the equations of queue theory, we can get the T_q in terms of λ and μ, as shown below,

$$\begin{cases} T_q = \dfrac{1}{(1-\rho)\cdot\mu} \\ \rho = \dfrac{\lambda}{\mu} \end{cases} \quad \Rightarrow \quad T_q = \frac{1}{\mu - \lambda} \quad \text{…….. (4.4)}$$

then,

$$\begin{cases} T_{q1} = \dfrac{1}{\mu_1 - \lambda_1} \\ T_{q2} = \dfrac{1}{\mu_2 - \lambda_2} \end{cases} \quad \Rightarrow \quad \frac{T_{q1}}{T_{q2}} = \frac{\mu_2 - \lambda_2}{\mu_1 - \lambda_1} \text{ …… (4.5)}$$

From equation (4.2), we can obtain,

$$1 - \frac{\lambda_1}{\mu_1} = 1 - \frac{\lambda_2}{\mu_2} \Rightarrow \frac{\mu_1 - \lambda_1}{\mu_1} = \frac{\mu_2 - \lambda_2}{\mu_2}$$

$$\Rightarrow \frac{T_{q1}}{T_{q2}} = \frac{\mu_2 - \lambda_2}{\mu_1 - \lambda_1} = \frac{\mu_2}{\mu_1} = \frac{\lambda_2}{\lambda_1}.$$

This assertion implicates the relationship between the arrival rate and the mean time a request spends in the system when the load of the system is balanced.

Assertion 4. In our proposed fault tolerant anycast server model in Figure 4.2, if $\mu_2 >> \mu_1$, when the primary server crashes, in the following crash processing period, T_q for the requests of S_2 is decreased, but very close to that before the crash; T_q for the unfinished request(s) of S_1 is dramatically increased for the clients.

Proof: We obtain the mean number of requests (w_1 and w_2) in Q_1 and Q_2 in terms of λ and μ.

$$\begin{cases} \rho = \lambda \cdot T_s = \dfrac{\lambda}{\mu} \\ w = \dfrac{\rho^2}{1-\rho} \end{cases} \Rightarrow \begin{cases} w_1 = \dfrac{\lambda_1^2}{\mu_1^2 - \lambda_1 \mu_1} \\ w_2 = \dfrac{\lambda_2^2}{\mu_2^2 - \lambda_2 \mu_2} \end{cases}$$

Once S_1 crashes, the remaining requests of Q_1, namely w_1, will be appended to Q_2 to be processed. Because the switching is done in one computer, S_2, we assume the switching time is zero. The w_1 requests arrive at Q_2 in bulk distribution, but from the viewpoint of the clients, we can simplify the case that the two incoming queues to Q_2 are Poisson distributions, and the numbers of requests are w_1 and w_2, respectively. We also assume that the arrival rate of Q_2 still be the same as that before the crash. Then the time to process the crash case, T_f, is expressed as below.

$$T_f = \frac{1}{\mu_2}(w_1 + w_2)$$

If the crash happened at time point *T*. we would try to find the performance changes for both the clients of S_1 and S_2 during $[T, T + T_f]$. We use T_{q1} and T_{q2} to denote the

mean time a request spends in the system before the crash, and T'_{q1} and T'_{q2} to denote the mean time a request spends in the system during [T, $T + T_f$]. Based on the equation (4), we can obtain the results of the four parameters as below,

$$\begin{cases} T_{q1} = \dfrac{1}{\mu_1 - \lambda_1} & t < T \\ T_{q2} = \dfrac{1}{\mu_2 - \lambda_2} & t < T \\ T'_{q1} = T'_{q2} = \dfrac{1}{\mu_2 - \lambda_2 - \lambda_1} & T \le t \le T + T_f \end{cases} \quad \text{.....(4.6)}$$

It is obvious that $T'_{q2} > T_{q2}$, that means the performance for each customer is decreased when a crash happens.

Let us assume that the twin server's service rate is much faster than that of the primary server, namely, $\mu_2 >> \mu_1$. We will discuss the change in performance under this assumption.

Based on equation (4.3) and the assumptions of this assertion, it is obvious that

$$\lambda_2 >> \lambda_1 \quad \text{……(4.7)}$$

then

$$\lambda_2 \approx \lambda_2 + \lambda_1 \Rightarrow \quad T'_{q2} \approx T_{q2}$$

It has been proven that although the performance of the twin server is dropped, it is still very close to the original performance before the crash.

Based on assertion 3, we obtain:

$$\frac{T_{q1}}{T_{q2}} = \frac{\lambda_2}{\lambda_1} \quad \text{..... (4.8)}$$

Combine equation (4.8) with equation (4.7), we obtain

$$T_{q1} >> T_{q2},$$

Further, we obtain this:

$$T_{q1}^{'} = T_{q2}^{'} \approx T_{q2} << T_{q1}$$

This means that the performance for a crashed server's unfinished requests is greatly improved from the viewpoint of the clients. This conclusion is reasonable based on the practical applications

4.4.2. The Twin Server Selecting Algorithm

So far, we have established that the twin server model can handle the responsibility of fault tolerance for anycast communication. There are two issues that needed to be addressed further: 1) how can we find a twin server in the anycast group for a given

anycast server in the Internet environment; and 2) how can a primary server know that its twin server is down.

We denote n as the number of the servers in an anycast group G_A. Generally speaking, $n \geq 2$.

Algorithm 2. Anycast Twin Server Selecting Algorithm

To select a twin server for a given server, it is obvious that we need to find the "best" server from the anycast group except the server itself. In other words, we need to find a "best" server from the other $n-1$ servers. It is natural to use the anycast mechanism to serve this requirement.

Based upon the proposed model and its analysis, we find that there are four critical properties for twin server selecting:

- The delay of the link between the two servers;
- The reliability of the link between the two servers;
- The performance difference between the two servers.
- The reliability of the selected server.

Therefore, in order to find a twin for a given anycast server, we can use the following equation to make the decision,

$$P(Twin) = w_1 \cdot P(T_d) + w_2 \cdot P(Performance) + w_3 \cdot P(LinkReliability) + w_4 \cdot P(ServerReliability)$$

P(Twin) is the possibility of a server being selected as the twin for the given server; $P(T_d)$ is the possibility of network delay between the two servers; P(LinkReliability) is the reliability of the link; and P(Performance) is the possibility of any performance difference between the potential twin server and the given server; P(ServerReliability) is the reliability of the potential twin server. w_i, $i = 1,2,3,4$, are the weights of the four components.

It is difficult to obtain all the details on the four components, especially in the ever changing Internet environment. If we assume that the reliability of each link in the anycast group are the same, and the reliability of the anycast servers are the same, then we can simplify the last equation as follows:

$$P(Twin) = w_1 \cdot P(T_d) + w_2 \cdot P(Performance)$$

Our previous research [YU02] provides a practical and efficient method for anycast routing, which integrates the network delay and the server performance as a criterion for finding the "best" server. The main idea is that when sending a probing packet, such as the ping packet, to all the members in an anycast group, the first responding server is the "best" server, because the probing time includes network delay and the server processing delay.

The Twin Server Selecting Algorithm

Initialize anycast group G_A;
Initialize T; *// T is the interval for twin server selecting*
Timer = 0;
R_q = twin server selecting request;
Qr = the queue for twin server selecting request(s);
while *(Timer = T or Qr is not Null)*
    R_q = ***head***(Q_r) *// take the head component of the queue.*
    SendProbingPacket*(G_A);*
    R_r = ***FirstRespondingPacket****(Server);*
    TwinServer = *Server;//find the "best" server.*
    Timer = 0;
    MoveNext*(Qr) //Move the head of the request queue.*
end while.

List 4.2. Algorithms for Twin Server Selecting.

We run the twin server selecting algorithm periodically because of the dynamic environment. This algorithm may also be triggered when it is necessary, such as in the event of the failure of a twin server. The algorithm is listed in list 4.2.

Algorithm 3. Twin Server Failure Broadcasting Algorithm

After the selection of the twin servers using the previous anycast twin server selecting algorithm, the relationships of all the servers can be presented as a directed graph. Some of the possible components are listed in the following diagram.

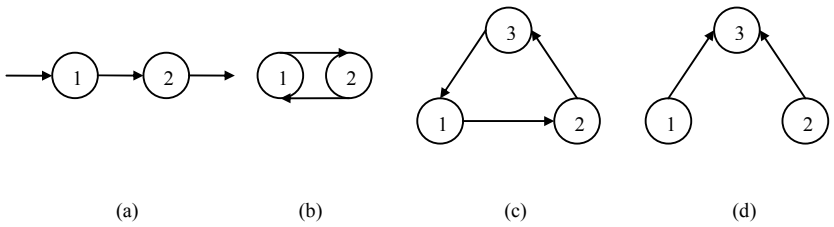

Figure 4.3 The Possible Directed Graphs for Twin Server Selecting

In Figure 4.3, a circle represents a server and the arrow denotes the data follow, namely, an arrow comes from a primary server and heads to the related twin server.

Based on this selection algorithm, when a server, say server 3 in Figure 4.3 (c), is dead, its primary server, server 2, can not notice the change, which means server 2 is not secure. Fortunately, the crashed server's twin server, server 1, will find the change. In this case, we us the twin server *(server 1)* of a failed server *(server 3)* to notify the failed server's primary server *(server 2)*. As a result, the primary server *(server 2)* will select a new twin server to keep the system fault tolerant. We also note that a server may have multiple primary servers, which means once the server is down, there are multiple servers, say m, $1 \le m \le n-1$, that need to be notified, such as in the situation in Figure 4.3 (d). At the same time, the relationship between a primary server and its twin server is dynamic based on our twin server selecting algorithm and the ever changing Internet environment.

Based on the above analysis, we designed a simple multicast based twin server failure broadcasting algorithm for server failure notification. In this algorithm, we configure a multicast group, G_m, that equals the anycast group,

$$G_m = G_A$$

Once a server finds that its primary server is dead then it will broadcast the information to the multicast group G_m , therefore the failed server's primary server(s) will receive the information and take some action. The algorithm is listed in list 4.3.

The Twin Server Failure Broadcasting Algorithm

//in this algorithm, we assume that $Server_P$ *is the crashed server,*
//$Server_{PP}$ *is* $Server_P$*'s primary server,*
//$Server_T$ *is* $Server_P$*'s twin server.*

Initialize the twin server directed graph;
Initialize anycast group G_A;

$G_m=G_A$; //G_m is a multicast group.
$Server_T$.ServerFailureDetecting($Server_p$) = *True; //*$Server_p$ *is dead.*
$Server_T$.Multicasting($Server_p$, *Gm);*
$Server_{PP}$.GetMessage($Server_P$)
$Server_{PP}$.TwinServerSelecting(G_A)

List 4.3. Twin Server Failure Multicasting Algorithm.

4.5 Summary

In this chapter, we proposed a twin server model for fault tolerance in anycast services, and analysed the performance changes using queueing theory. For each anycast server, we find a twin server for it, and the twin server will continue any unfinished task(s) if the primary server is down. We define the load balance for the two different functioning computers (primary and its twin) in queuing theory as equation (4.2): If two computers are load balanced during a given period, the sum of the service times should equal. Based on our definition and the queueing theory, we have obtained some interesting conclusions, and the conclusions are reasonable in practical application.

We developed the twin server selecting algorithm to try to find the "best" server as a twin server for a given primary server using an anycast mechanism. The selection is

based on four components: the network delay, link reliability, server reliability and performance difference between the two servers. In optimal conditions, namely, limited network delay, the links and the servers are reliable, and the twin server's performance is greatly better than the primary server. When the primary server is down, the performance of the system remains almost the same, and even better for the crashed server's unfinished task(s).

We also proposed a twin server failure notification algorithm using the multicasting method. Once a twin finds its primary server is down, it will multicast the information in the anycast group, then the failed server's primary server(s) will know about the failure and try to find a new twin server(s) for itself/themselves.

Chapter 5
Load Balance in Anycast

Anycasting has the attribute of load balancing according to the original definition, but the conclusion is based on the view point on load of network bandwidth only, not including the load of the anycast servers. In this chapter, we study the complete load balance issue for anycast services: network bandwidth load balance and anycast server load balance. Load balance has been explored for many years in terms of network bandwidth, therefore we focus on the load balance of anycast service in terms of the load of the mirrored servers under the prerequisite of the balanced network bandwidth. We propose a Balanced Load Queue (BLQ) model, which combines the queuing theory and hydro-dynamic theory, to model the distributed anycast servers. Based on the BLQ model, we obtain an assertion that if the system is in the state of global fairness, then the performance of the whole system is the best. We propose a load balanced algorithm based on the model: the algorithm tries its best to keep the system in the global fairness status using job deviation. We present three strategies: best node, best neighbour, and random selection, for job deviation. A number of experiments have been conducted for the comparison of the three strategies, and the results show that the best neighbour strategy is the best among the proposed strategies, furthermore the proposed algorithm with best neighbour strategy is better than the traditional round robin algorithm in term of processing delay, while the proposed algorithm needs very limited system information and robust.

5.1 Introduction

Anycast is a new type of network service which always tries to find the "best" server among the anycast group [PAR93] [XUA00] [JIA01]. Anycast mechanism provides an automatic load balance capability among an anycast group, but the conclusion is based on the viewpoint of load of network bandwidth only, not including the load of the anycast servers. A lot of research has been done in terms of network load balance, such as [WAN96], [MA97], and [CHE98], but the area of load balance among the anycast servers has not been touched by researchers based on our knowledge.

The anycast servers belong to Web based distributed systems essentially. One issue of Web based distributed systems is the load balance among the distributed servers. Most of the previous load balance algorithms [JOS97] [DRI02] [MIT97] based on the background of static environment, but in the situation of Web based distributed systems, the circumstance has been changed because of the unstable Internet traffic, congestions, user requests, and so on.

[HUI96] [HUI97] applied hydro-dynamic approach in the load balance issue of networks, and it works well. But the hydro system describes a continuous world, while the computer network systems belong to a discrete environment. As a result, we have to use a discrete methodology to model and analyse computer networks. Queueing theory is a practical analysis tools for computer networks. The computer networks involve buffers and queues, therefore the queueing theory is a suitable tool to model computer networks.

In this chapter, we try to take advantage of the benefits from both of the two methodologies, combine the queuing theory and the hydro-dynamic approach to model the anycast servers.

The rest of this chapter is organized as follows. Section 5.2 refers to the modelling network load balance, and Section 5.3 discusses the network load balance issue for anycast. Our anycast server load balance system modelling is presented in section 5.4. A novel algorithm is proposed in section 5.5 based on the Balanced Load Queue model. The performance evaluation is discussed in section 5.6. Finally section 5.7 summaries the paper and presents the future work.

5.2 Modelling Load Balance

Graph theory is one of the methods of analysing the load balance issue. [JOS97] studied the application of edge colouring of graph theory into load balance issue in distributed systems. An edge-colouring of a graph *G* with *K* colours is defined as the assignment of *K* distinct colours to the edges of the graph such that no two adjacent edges have the same colour. Statistics is a useful tool for the load balance research as well [DIR02] [MIT97] [AZA99].

[MIT96] presented the supermarket model: customers arrive as a Poisson stream of rate $\lambda n, \lambda < 1$, at a collection of *n* servers. Each customer chooses some constant *d* servers independently and uniformly at random from the *n* servers, and waits for service at the one with the fewest customers. The service time for a customer is

exponentially distributed with mean 1, and the service protocol is first-in first-out. Furthermore, the paper pointed out that the supermarket model is difficult to analyse because of dependencies: knowing the length of one queue affects the distribution of the other queues. Then the authors first developed a limiting, deterministic model representing the behaviour as $n \to \infty$, and then translated the results from that model to results for large, but finite, values of n.

Balls and bins model is also used for load balance research [DRI02] [MIT97]. The problem is described as follow: suppose that n balls are thrown into n bins, with each ball choosing a bin independently and uniformly at random, then the largest number of balls in any bin is approximately $\log n / \log \log n$ with high probability. [AZA99] proposed a approach of online load balance based on the balls and bins model. The paper considered the scenario in which a user or a process has to choose between a number of identical resources on-line. One method is to check all the loads and find the least one, this method is very expensive; the second approach is to send the task to a random resource. The disadvantage of this method is that the difference in load between different servers will vary by up to a logarithmic factor. If each user samples the load of two resources and sends his or her request to the least loaded one, the total overhead is small, and the load on the n resources varies by only a $O(\log \log n)$ factor.

[HUI96] [HUI97] introduced a hydro-dynamic approach to solve the dynamic load balancing problem on a network of heterogeneous computers. The authors modelled a computer as a cylinder, the diameter represents the computing capability of the computer and liquid in the cylinder denotes the work load on the computer. Their

conclusion is that when the system achieves the global fairness, namely the heights of all the cylinders are the same, the system is load balanced.

5.3 Network Load Balance Algorithms for Anycasting

A number of anycasting algorithms have been proposed in the network layer [XUA00] [XUA01] [JIA04]. Xuan et al. proposed four methods for anycasting in the network layer: the Shortest-Shortest Path method (SSP), the Minimum Distance method (MIN-D), the Source-Based Tree method (SBT), and the Core-Based Tree method (CBT). According to these algorithms, each request of the clients will be routed to the "best" server, therefore the work load of a given anycast group is balanced on the Internet.

Quality-of-service routing on network bandwidth can also be used for anycasting, such as [WAN96] [MA97] [CHE98]. For the aim of QoS, first of all, we need to find the feasible paths. If there exist a path, $P = \{i\ , i_{2,}..., i_n\}$, the maximal reservable bandwidth (MRB in short) on the path P is the minimum of the reserveable bandwidth of all links on the path. A path P is feasible if the MBR_P is no less than the requested bandwidth B, namely, $MBR_P \geq B$.

Based on the previous research, there are four algorithms on path selection for traffic with bandwidth guarantee:

1. Widest-shortest path: a path with the minimum hop count among all feasible paths. If there is more than one of that kind of paths, then the one with maximum reservable bandwidth is selected.

2. Shortest-widest path: a path with the maximum bandwidth all among all feasible paths. If there is more than one of that kind of paths, then the one with minimum hop count is selected.

3. Shortest-distance path: a feasible path with the shortest distance. The distance is defined by

$$dist(P) = \sum_{j=1}^{k} \frac{1}{R_{i_j}}$$

where R_{i_j} is the bandwidth available on link i_j.

4. Dynamic-alternative path: if *k* is the minimum-hop path when the network is idle, then a dynamic-alternative path is a widest-shortest path with no more than *k+1* hops.

Previous work has shown that a routing algorithm that gives preference to limiting the hop count algorithm, such as widest-shortest, performs better when the network load is heavy, while an algorithm that gives preference to balance the network load, such as shortest-widest algorithm, performs slightly better when the network load is light. In this chapter, we can use any one of the algorithms to guarantee the bandwidth.

It is clear that we can take the proposed network layer anycasting algorithm or modify the network QoS aimed algorithms for anycasting, all of them can guarantee the network load balance for anycasting.

5.4 Balanced Load Queue Model for Anycast Servers

[HUI96][HUI96] modelled distributed systems in hydro-dynamic approach; it is very effective for analysis and research of distributed systems. However, the liquid system is a continuous system, while the situation in our computer systems is discrete, as a result, the proposed model has to be modified to solve the problem in the computer world. Queuing theory is a powerful tool for modelling the computer systems, therefore we combine the two distinct theories together to model the distributed anycast servers.

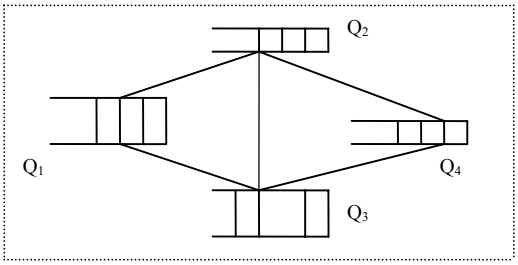

Figure 5.1 An Example of Anycast Servers by the Queuing Model

In this chapter, we model each computer in the anycast system as a queue, and the queues are connected by networks. Figure 5.1 shows an example of anycast servers with four computers and five connections.

In Figure 5.1, the queues, Q_1, Q_2, Q_3, and Q_4, represent anycast servers, and there exists a network connecting them together. For each queue the width of the queue denotes its computing capability: the wider, the more powerful. In order to simplify the explanation, we describe some concepts here, which will be used in the rest of this chapter.

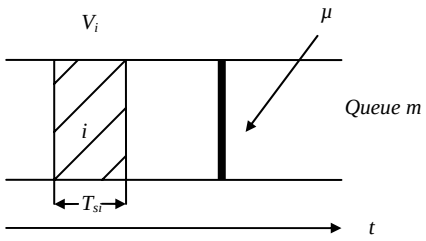

Figure 5.2 A Queue and the Related Concepts

In Figure 5.2, the parameter μ indicates the moving speed of the requests in the queue *m*, actually, μ is the service rate of the related anycast server. V_i is a request in the queue, and T_{si} is the service time for V_i in the queue.

Based on the definitions in Figure 5.2, we can find that during the processing of request *i*, at any time point *t*,

$$V_i^{'} = \int_0^t \mu \cdot dt = \mu \cdot t$$

Where $V_i^{'}$ is the duration of processing the request.

When the processing finished, then

$$V_i = \mu \cdot T_{si}$$

Definition 1. Global Fairness *(GF).* In a distributed anycast server group D_n, *n* is the number of the servers, if the sum of service time T_{si}^m (where $m \in n$, is the number for a computer, and *i* is the sequence number for requests in a queue) in each queue are equal, then we call the anycast system is in a state of global fairness.

The definition can be expressed in the follow equation.

$$\sum_{i=1}^{k_1} T_{si}^1 = \sum_{i=1}^{k_2} T_{si}^2 = ... = \sum_{i=1}^{k_{n-1}} T_{si}^{n-1} = \sum_{i=1}^{k_n} T_{si}^n \quad \text{…….. (5.1)}$$

If a distributed system D_n is in the state of global fairness, then the current requests in all the queues will be finished at the same time, and further, that each computer is equivalent for a new incoming request.

Assertion 1. If the work load of a distributed anycast system with n servers $(n \geq 2)$ is balanced, then in a given period [*0, T*] (*T* is sufficiently big), the system must be in the state of global fairness, namely, the equation (5.1) is correct.

Proof: There are three cases for this issue listed follows, any other situations are the combination of them.

Case 1. There are no requests in the queues and $\lambda_i = 0, i = 1,2,...,n$ (λ_i is the arrival rate of requests for queue *i*) for the period [*0, T*].

It is obvious that the equations are correct.

Case 2. $\lambda_i > \mu_i, i = 1,2,...,n$ for the period [*0, T*].

That means all the arrival rates are greater than the service rates respectively, namely, all the servers are busy for the whole period, then

$$\sum_{i=1}^{k_1} T_{si}^1 = T, \quad ..., \quad \sum_{i=1}^{k_n} T_{si}^n = T$$

The assertion is correct.

Case 3. Without loss generality, suppose there is no request in Q_1 and there is/are one or more request(s) in Q_i , $i=2, \ldots, n$, at a given time point $t, t \in [0,T]$.

For the reason of load balance, if there comes a new request, the request will be dispatched to Q_1 by the overloaded queue(s), this situation may happen from time to time. Therefore, if T is sufficiently big, the assertion is correct. In all of the three cases the assertion are correct, therefore the assertion is correct for any combination of them, as a result, the assertion is correct for any situation.

Assertion 2. When a distributed anycast system is in the state of global fairness, then the performance of the whole system is the best among all the situations.

Proof: assume that there are n servers, and the service rates are $\mu_1, \mu_2, ..., \mu_n$. If the anycast system is not idle, in the state of global fairness, the total service rate is $\mu_T = \sum_{i=1}^{n} \mu_i$, if the system is not in the state of global fairness, after a period of time, T, there will be at least one computer having no jobs to do, then the total service rate $\mu_T^{'} = \mu_T - \sum_{m=1}^{k} \mu_m, \mu_m$ is/are the server/servers that has/have no jobs to do. It is obvious that $\mu_T > \mu_T^{'}$, therefore the assertion 2 is correct.

Assertion 3. In an $n (n \geq 2)$ servers distributed anycast system, if the system is in the state of balance, i.e., work load of n servers are balanced, then during a given period [*0, T*] (T is sufficiently big), the ratios of arrival rate λ to the service rate μ for each server are the same.

Proof: If the system is in the state of balance, then equation (5.1) is correct. And we know that $T_s = \lambda / \mu$. We ignore the switching time of processes, then in a long term view, we can obtain the following result.

$$\frac{\lambda_1}{\mu_1} = \frac{\lambda_2}{\mu_2} = ... = \frac{\lambda_n}{\mu_n} = k \quad \text{..... (5.2)}$$

Where *k* is a constant; it represents the ratio for convenience.

This assertion implicates that the relationship between the arrival rate and the service rate is fixed when the load of the system is balanced. Furthermore, parameter *k* implies the average waiting time for the users when the whole system is fully loaded. When *k* is bigger, the average waiting time is longer in that scenario.

Assertion 4. If the work load of *n* servers ($n \geq 2$) are balanced, then during a given period [*0, T*] (*T* is sufficiently big), the relationship between T_q, *mean time a request spends in the system*, and the arrival rate λ is reciprocal.

Proof: Assume that *n=2*, based on the equations of queuing theory, we can get the T_q in terms of λ and μ, shown as below,

$$\begin{cases} T_q = \dfrac{1}{(1-\rho)\cdot\mu} \\ \rho = \dfrac{\lambda}{\mu} \end{cases} \quad \Rightarrow \quad T_q = \frac{1}{\mu - \lambda}$$

then,

$$\begin{cases} T_{q1} = \dfrac{1}{\mu_1 - \lambda_1} \\ T_{q2} = \dfrac{1}{\mu_2 - \lambda_2} \end{cases} \quad \Rightarrow \quad \frac{T_{q1}}{T_{q2}} = \frac{\mu_2 - \lambda_2}{\mu_1 - \lambda_1}$$

From equation (5.1), we can obtain,

$$1 - \frac{\lambda_1}{\mu_1} = 1 - \frac{\lambda_2}{\mu_2} \quad \Rightarrow \quad \frac{\mu_1 - \lambda_1}{\mu_1} = \frac{\mu_2 - \lambda_2}{\mu_2}$$

$$\Rightarrow \quad \frac{T_{q1}}{T_{q2}} = \frac{\mu_2 - \lambda_2}{\mu_1 - \lambda_1} = \frac{\mu_2}{\mu_1} = \frac{\lambda_2}{\lambda_1}.$$

When $n>2$, the proof is the same, then in general,

$$\frac{T_{qi}}{T_{qk}} = \frac{\lambda_k}{\lambda i}, i, k \in n, i \neq k \quad \ldots\ldots \quad (5.3)$$

This assertion indicates that the relationship between the arrival rate and the mean time a request spends in the system when the load of the system is balanced.

Despite with network delay, when we combine assertion 3 and assertion 4 together, then we find that the mean waiting time for a faster server is less than that of a slower server when system is in the state of balance, therefore we do not need to worry about a bigger request arrival rate.

5.5 Load Balanced Algorithm for Anycast Servers Based on the BLQ Model

The balanced load queue (BLQ) model can achieve load balance for distributed anycast servers, but it is expensive because we need to know the states of all the queues. Based on assertion 3, we found that if the system is balanced, then the ratio of the arrival rate and the service rate for a given server is fixed. As we know, the service rate of a server is a constant value, for a given value k, if $\lambda_i \le k \cdot \mu_i, i = 1,2,...,n$, that means the server is approaching to the state of balance; On the other hand, if $\lambda_i > k \cdot \mu_i, i = 1,2,...,n$, that means the server is overloaded, and the incoming requests should be dispatched to the other server in order to get the system back to the balanced state. The main advantage of this idea is that we just need to set a reasonable k when the system is initiated, and then each server can judge wether it is necessary to deviate the incoming request or not without the information of the whole network and any information about other servers.

We assume that the whole performance of the system is satisfied by the users, which means the k in equation (5.2) is fixed, then we get a boundary for arrival rate λ for each server, respectively. When there comes a new request to server i, the server will calculate its own k_i, if $k_i < k$, then it does nothing, otherwise, it deviates the incoming request to one of the other peer servers.

How to decide the destination to process the deviated requests is an interesting issue, we design three strategies here for the job deviation.

● Random Selection Strategy. Choose one server randomly from the other servers.

● Best Node Strategy. Choose the best one from all the distributed servers.

● Best Neighbour Strategy. Choose the better one from the current server's nearest two neighbours.

The details of the algorithm is shown in list 5.1.

The LBQ Based Deviation Algorithm

Initialize the system;
Initialize parameter k, μ;
If $k_i \leq k$ then
QueueAppend (R, Q_i)
Else
RequestDeviate(R, Q_m) *// m = 1,2,…,n, m<>i*
Endif

//For the Deviation, there are three strategies:
//Random Selection Strategy:
//Choose a server randomly
Q_m = ***RandomQueue()*** *// m = 1,2,…,n, m<>i*

//Best Node Strategy:
// choose the lightest workload server
G_A = {all the distributed servers } // build an anycast group G_A
Q_m = ***Anycasting*** *(G_A)*

//Best Neighbour Strategy:
// choose the better performance server from its two nearest neighbours
Q_m = ***Minimum*** *{*
for *i = 1 to 2*
{ ***PerformanceProbing(i)*** *}*
}

List 5.1 the BLQ Based Deviation Algorithm

We must point out that for the best neighbour strategy and the random selection strategy, there exists a potential problem of deviation loop. For example, server A deviates a job to a random selected server, say server B, but server B needs to deviate the incoming job(s) as well, unfortunately, server B selects server A as a deviate

destination, then there exists a deviation loop until one of the servers stops the deviation or the loop is broken. The probability of deviation loop is high when there are a small number of servers.

The implementation of the algorithm is not difficult. For the first deviation strategy, it is not necessary to hold the system state information, instead, we just keep the information of how many servers in the anycast group and their addresses respectively;

The second strategy needs the support of anycasting. If the network provides anycast service, then the only job the algorithm needs to do is putting all the anycast servers into an anycast group and maintaining the group information, in this chapter we use our previous result achieved in [YU02].

The third strategy needs to send *two* probing packets (such as ping packets) to the *two* nearest neighbours, the first responding server is the deviation destination.

5.6 Performance Analysis

We have conducted some experiments on the Internet in order to demonstrate our proposed algorithm and to compare the performance of the three strategies for job deviation. Moreover, we use a central controlled algorithm with round robin strategy [WAN00] [MAR01] [LIU02] as a benchmark to evaluate our algorithm. The benchmark algorithm has a central controller to accept all the incoming requests of the distributed system and dispatches the requests to the distributed servers using the round robin strategy.

The scenario for our algorithm is that requests are generated everywhere in the Internet and target to one of the distributed anycast servers randomly. We know an estimated processing time for each job on a given server. Because of the delay of the deviation, there exists a delay of processing compared with the estimated processing time, we name it as Processing Delay.

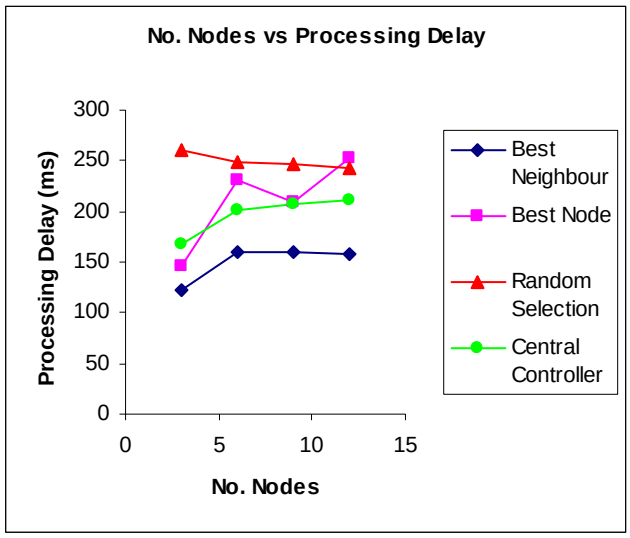

Figure 5.3 Number of Nodes versus Processing Dealy

We use more than ten computers as anycast servers distributed in two campuses of 100km apart, to act as the distributed anycast servers. Each computer connects to the local area network using a 100Mb ethernet, and the two campus LANs are connected by the Internet. In the experiments, when a deviation is needed: the best neighbour strategy probes the nearest two anycast servers, and selects the lighter workload one to serve for the deviated job(s); the best node strategy probes all the members of the anycast group and deviates the job(s) to the lightest workload server to be executed; the random selection strategy selects a server randomly from the anycast group as a

deviation destination. In the rest of this section, we present and compare several factors, which have impact on the performance of the whole system.

Figure 5.3 shows that when the number of nodes in a system increases, the processing delay of the best neighbour strategy is almost constant and is less than the two other strategies. While the number of nodes increases, the processing delay of the best node strategy has a trend of increasing, because of more probing overhead; the change of the processing delay of the random selection strategy is smooth, and the processing delay is higher than that of the best neighbour strategy, because the chosen server is not the "best" or a "better" one in most of the cases. Generally, only the best neighbour strategy of the proposed algorithm is better than the central controller algorithm. The reason is obvious as the best node strategy is expensive while the random selection strategy has no quality control.

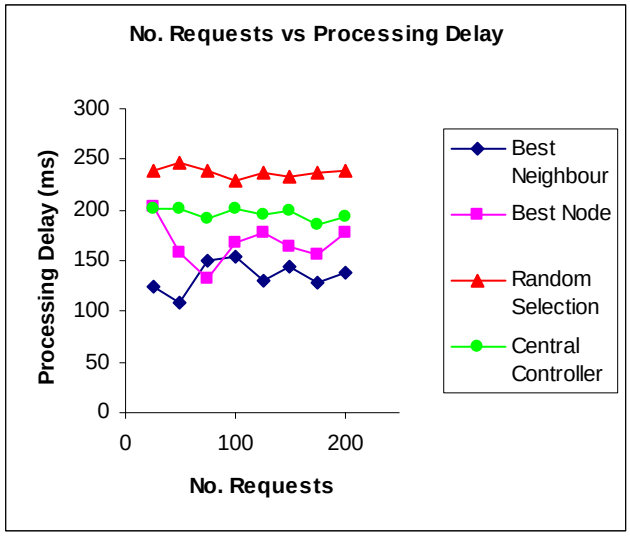

Figure 5.4 Number of Requests Versus Processing Delay

If the arrival rates are stable, then the number of requests can reflect the general performance in terms of time. Based on Figure 5.4, we conclude that generally the average processing delays of the three strategies and the central controller algorithm

are close to a constant value respectively despite the unexpected Internet traffic. In terms of the performance, the best neighbour strategy is better than the best node strategy, and is much better than the random selection strategy. In fact, both strategies with quality control are better than the central controller algorithm in terms of processing delay.

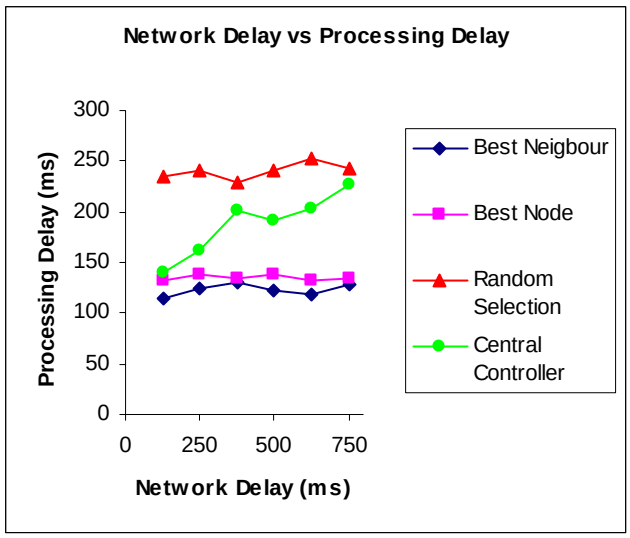

Figure 5.5 Network Delay versus Processing Delay

The impact of network delay on the processing delay is shown in Figure 5.5. We find that the performance of the best node strategy and that of the best neighbour strategy is very close and both of them are much better than the performance of the random selection strategy or that of the central controller algorithm. The best node strategy finds the lightest loaded server to deviate but pays for the network delay, while the best neighbour strategy pays less in network delay, but the performance of the deviation destination could be worse than that of the best node strategy.

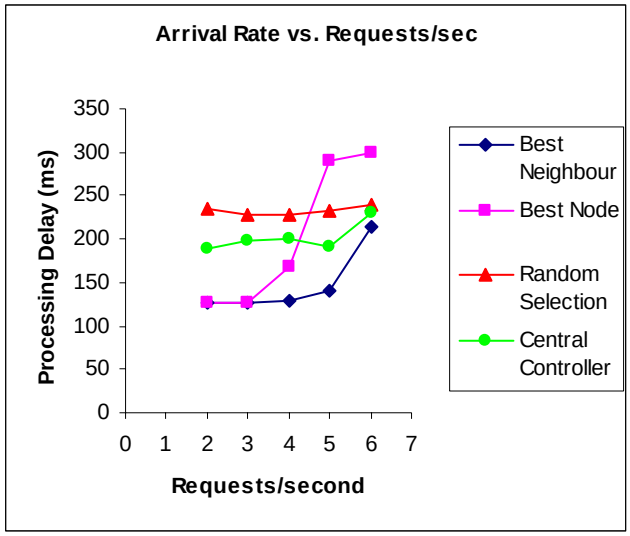

Figure 5.6. Arrival Rate versus Processing Delay

Arrival rate is a component which reflects the concentration of the Internet traffic. The relationship of processing delay and the arrival rate is shown in Figure 5.6. Based on the result, we can conclude that the performance of the best neighbour strategy is better than that of the best node strategy, and the processing delay of the best neighbour strategy is getting closer to that of the random selection strategy when the arrival rate increases. We notice that the processing delay of the best neighbour strategy increases fast when the arrival rate approaches to 6, the main reason is the deviation loop may occur as analysed previously. Performance of the best neighbour strategy is better than that of the central controller algorithm, while the best node strategy becomes worse than the central controller algorithm when the arrival rate increases.

5.7 Summary

In this chapter, we proposed the balanced load queue model, which takes advantage of the queuing theory and the hydro-dynamic approach to model the load balance of the Internet based anycast systems. As we know that a computer network is a discrete rather than a continuous environment, therefore the discrete methodologies are more suitable than the continuous ones in computer network modelling. Base on the proposed model, we obtained that when the distributed anycast system stays in the global fairness status, the whole system performance is the best.

We proposed a load balancing algorithm based on our balanced load queue model, which tries its best to keep the system in the global fairness status using job deviation. We presented three strategies: best node, best neighbour, and random selection for job deviation. At the same time, in order to reduce the overhead of keeping the global fairness, we predefined a threshold in the algorithm for each distributed server (queue) respectively, which depends on a reasonable delay to users. If one queue's jobs exceeds the predefined threshold, then the deviation will be employed.

Our experiments show that the best neighbour strategy is the best among the three strategies at several aspects: number of nodes, number of requests, network delay and arrival rate. In terms of processing delay, the best neighbour strategy is better than that of the central controller algorithm in the examined four properties, while another quality control strategy – the best node strategy is better than that of the central controller algorithm in most of the situation.

Based on the experiments, we conclude that the performance of the proposed algorithm with the best neighbour strategy is better than the central controller algorithm with round robin strategy, and the proposed algorithm can work with very limited system information. Moreover, the proposed algorithm can work independently from network traffic, link breaches, and so on.

Some further researches need to be done, for example, the dynamic adjustment for the threshold for a distributed system is an important issue for the whole system performance. Moreover the deviation loop is a critical and interesting topic for further research.

Chapter 6
Anycast in Ad Hoc Networks

Wireless networks have become increasingly popular since their emergence in the 1970s. There are currently two categories of mobile wireless networks: infrastructured mobile networks and ad hoc networks (infrastructureless mobile networks). An infrastructured mobile network is a network with fixed and wired gateways, named as base stations as well. The base station is a bridge for all the mobile nodes in its communication radius.

Ad hoc mobile network is an infrastructureless network. Ad hoc networks have no fixed routers or gateways, and all the nodes are capable of movement and can be connected dynamically in an arbitrary manner, furthermore, each node can act as a router to provide the routing service for the others. There are huge application prospects of ad hoc networks, such as emergency search-and-rescue operations, meetings in which people wish to quickly share information, and data acquisition operations in inhospitable terrain.

Key issues in the ad hoc network include efficient routing algorithms, effective applications, security, and so on. In this chapter, we focus on the anycast routing issue in the ad hoc networks.

6.1 Introduction

Ad hoc networks have wide applications. For example, it can serve for the temporary communication for the intelligent electronic facilities in homes, factories, and the field; it also has huge potential applications in the military, and so on. Compared to current popular wireless networks, ad hoc networks have unique characteristics, such as, changing network topology; every node should act as a router, etc. The challenges of ad hoc networks include:

● Routing protocols. The routing foundation is totally different in ad hoc networks; therefore new routing protocols should be designed explicitly for unicast, multicast and anycast. It is essential that the designed routing algorithm is simple efficient and has minimum control message exchanging.

● Security and reliability. The "routers" are variable, even unavailable, and the topology is volatile, therefore, the security and reliability are quite challenging issues for ad hoc network researchers.

● Power consumption. One inherent feature of mobile devices is the limited power capability. The power shortage can cause the disappearing of the nodes and it further impacts the topology of the ad hoc networks. It is very important that power consumption should be properly distributed among the mobile hosts.

● Quality of service. The resource, such as network bandwidth, is scare in ad hoc

networks; it is a big challenge to provide different QoS guarantees based on the limited resource.

Mesh-based routing methods have been proposed for unicast and multicast in ad hoc network environments [GAR99] [LEE99] [LEE00], It is a reliable method in the wireless networks, because the mesh offers the sources more routes to the receivers, and the mesh has a strong recovery capability of local link failures. Mesh based routing method is very suitable for routing service in ad hoc network, in which robustness is a critical issue, however, to the best of our knowledge, there is no research has been presented about anycast routing in ad hoc networks.

In this chapter, we focus on the anycast routing issue in ad hoc networks. We propose a mesh based anycast routing protocol (MARP), which provides reliable and efficient anycast routing service in ad hoc networks. The mesh architecture makes the routing service reliable; moreover, the proposed protocol can prevent the traffic storm in the network, and can result in less bandwidth consumption by reducing the control packets delivery.

The rest of this chapter is organized as follows. Section 6.2 presents the related work of routing algorithms in wireless networks. A new mesh based anycast routing protocol is proposed in Section 6.3. Section 6.4 describes more details in implementation. Finally, Section 6.5 summarises for the chapter.

6.2 Related Work

[ROY99] reviewed the routing algorithms for ad hoc mobile wireless networks, and classified them into two categories: table-driven and source-initiated (demand-driven), which are shown in Figure 6.1.

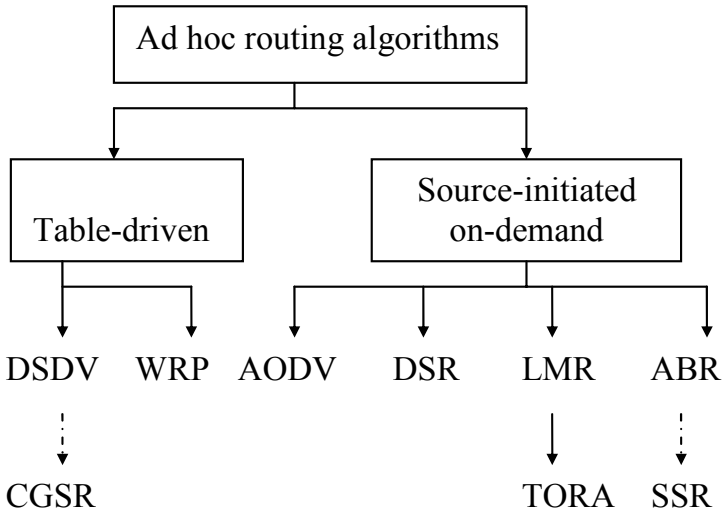

Figure 6.1 Categorization of Ad Hoc Routing Algorithms

Table-driven ad hoc routing algorithms include Destination-Sequenced Distance-Vector (DSDV) routing [PER94] and Wireless Routing Protocol (WRP) [MUR96]. The Clusterhead Gateway Switch Routing (CGSR) protocol [CHI97] is derived from DSDV. All the table-driven algorithms try to maintain consistent, up-to-date routing information for every node in the network. Each node keeps one or more tables to store routing information, once there is a change in the topology, the change will be propagated throughout the network in order to keep a consistent network view. In an ad hoc environment, if the topology changes frequently, then there will be a lot of routing control packets, which may cause traffic storms, as a result, these traffic storms can degrade the network performance.

Source-initiated on-demand algorithms have four protocols: the Ad hoc On-demand Distance Vector (AODV) [PER99], Dynamic Source Routing (DSR) [JOH96], Temporally Ordered Routing Algorithm (TORA) [PAR97] which is similar to the Lightweight Mobile Routing (LMR) [COR95], Associativity-Based Routing (ABR) [TOH96], and the Signal Stability Routing (SSR) [DUB97] which was derived from the ABR. Source-initiated on-demand protocols create routes only when it is desired by a source node. When a source node requires a route to a destination, a route discovery process is initiated within the network. The process will stop once a route is found or all possible route permutations have been examined. Once a route has been established, a maintenance process will maintain the route information until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. Comparing with table-driven algorithms, demand-driven algorithms can reduce the cost of routing information maintenance, and are more suitable for ad hoc networks whose topologies are volatile. Some papers [BRO98] [JOH99] [COR99] have shown that the source-initiated on-demand algorithms outperform the table-driven ad hoc routing algorithms.

[PAR99a] [PAR99b] introduced the concept of virtual node into the anycast paradigm. The authors viewed the network topology as a graph, and introduced a virtual node to represent the anycast service. They extended link-state, distance-vector and link-reversal unicast routing protocols to provide the anycast service through the proposed method.

[GUL01] made simple extensions to the three existing routing algorithms for mobile ad hoc networks: Dynamic Source Routing (DSR), Ad-hoc On-demand Distance Vector (AODV) routing, and Temporally Ordered Routing Algorithm (TORA). All the updates locate at the sinks, for this reason, the modifications are very limited. For example, in order to make DSR anycast capable, the authors only required that the anycast group members reply to all route requests for the anycast address and accept any data intended for the anycast address, therefore, DSR easily lends itself to anycast.

[KO00] extended the Temporally Ordered Routing Algorithm (TORA) [PAR97] to an algorithm called GeoTORA for the anycast routing in mobile networks. The algorithm based on link reversal to keep the connection between sources and receivers. However, GeoTORA was not designed explicitly for ad hoc networks, it is weak in the aspects of mobility, robustness, etc.

In the mobile ad hoc environment, robustness of routing is a critical issue. Flooding is a reliable routing method in ad hoc networks. [WIL02] classified broadcasting technology into four categories: simple flooding, probability-based methods, area-based methods and neighbour-knowledge-based methods. In a high mobility ad hoc circumstance, simple flooding is the only way to achieve the full coverage, but it introduces very heavy traffic which may cause network congestion and collision. [TSE02] proposed the probability-based and the area-based methods to solve the broadcasting storm problem caused by flooding. [WU03] tried to use the neighbour-knowledge-based method to achieve the same goal.

[CHO04]extended anycasting into MAC layer because the authors thought that choosing a single optimal route at the network layer may not be sufficient, and the knowledge of short-term channel conditions at the MAC layer could play an important role in improving end-to-end performance. Therefore, the paper proposed the MAC-layer anycasting for ad hoc mobile networks, in which the network layer specifies multiple downstream nodes, and the MAC layer chooses a suitable node based on instantaneous network conditions.

[GAR99] [LEE99] [LEE00] researched on mesh-based multicast protocols, which provide alternative paths and a link failure does not trigger a recomputation of a mesh. All the multicast sources, receivers, forwarding nodes and the links establish a mesh, and the one hop away neighbours of the mesh nodes are the group neighbours. The multicast source submits a local request packet to maintain the mesh, and only the mesh nodes and the group neighbours deliver the packet. Once a request packet arrives at a receiver, the receiver responds with a reply packet back to the source along the reversal path, therefore a route is created between the source and the receiver.

6.3 A New Anycast Routing Protocol

Because of the ever changing ad hoc network topology, robustness is a critical issue that the routing service has to deal with. For this reason, we propose a mesh-based anycast routing protocol (MARP) for ad hoc networks.

6.3.1 An Overview of MARP

Mesh-based anycast routing protocol is a robust and efficient protocol. Generally speaking, the mesh-based protocols are not as efficient as that of the tree-based protocols in terms of performance, but they are robust against topology changes [GAR99] [LEE00]. In the proposed MARP, some related hosts establish a mesh, and the anycast routing service depends on the mesh.

At the initial state of an anycast service, an anycast source uses broadcasting to flood mesh-establishing messages (flooding route discovery), once an anycast server receives the mesh-establishing message, it will respond an acknowledgement message, a route between the source and the receiver will then be established. After that, an on-mesh route discovery procedure is employed for the mesh refreshment and link failure recovery. The on-mesh route discovery packet is only delivered by the mesh nodes and the group neighbour nodes (which will be defined in section 6.3.2). This can prevent the mesh maintenance packets broadcast unnecessarily in the ad hoc network. The previous research [AGG99] has proven that most link failure recoveries can be localized to a small region along a previous route, therefore, the method is feasible for on-mesh link failure recovery.

Because of the continuous changing topology of ad hoc networks, MARP performs flooding route discovery occasionally, which can refresh the whole mesh and make sure its correctness. Flooding route discovery can also deal with the network partition issue. The flooding route discovery is expensive in terms of network bandwidth.

6.3.2 Anycast Mesh Creation

When an anycast source tries to join an anycast group, it initially broadcasts a JOIN_REQ packet, the JOIN_REQ packet has an *upstream node* field. When an intermediate node caches the JOIN_REQ packet, it updates the *upstream node* field with its own address, and then forwards (broadcasts) the updated packet to the next nodes. When an anycast server receives the JOIN_REQ packet, it responds a MESH_ACK packet back to the node from which it received the JOIN_REQ packet. Once the upstream node receives the MESH_ACK packet, it adds an entry for the anycast group to its routing table, and then it forwards the MESH_ACK packet to its own upstream node. This procedure continues until the MESH_ACK packet gets to the anycast source. And then an anycast route is established between the source and the receiver. The intermediate nodes that relay the MESH_ACK packet become *forwarding nodes*. An anycast mesh of an anycast group consists of anycast sources, anycast receivers, forwarding nodes, and links connecting them. All the nodes in an anycast mesh are called *mesh nodes*.

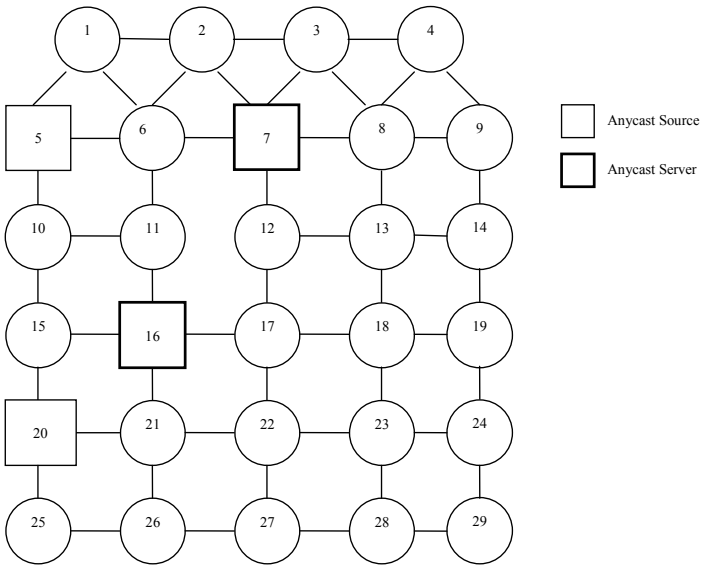

Figure 6.2 The Initial Ad Hoc Network

Figure 6.2 shows an ad hoc network as an example. In the ad hoc network, there are 29 nodes in total, includes two anycast sources (node 5 and node 20) and two anycast servers (node 7 and node 16). The link between any two nodes means that there is a network connection for the two nodes.

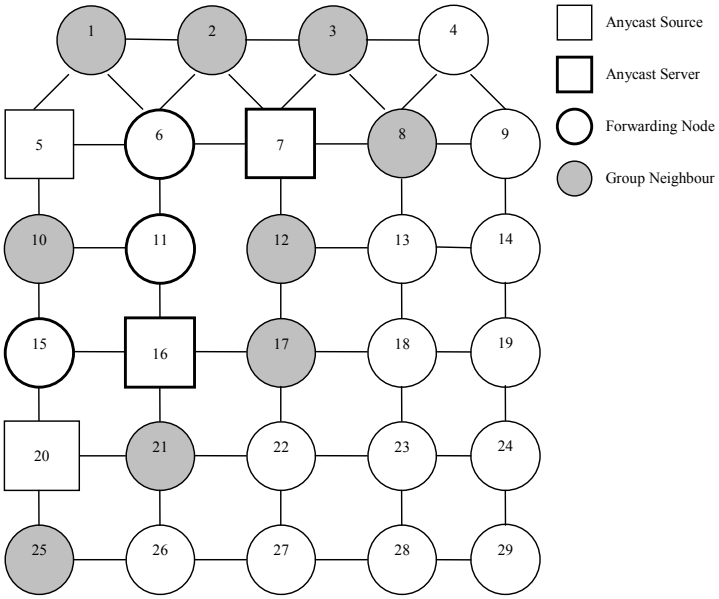

Figure 6.3 The Ad Hoc Network after Mesh Creation

Figure 6.3 demonstrates how an anycast mesh is established. In the initial network (Figure6.2), node 7 and node 16 belong to an anycast group, we express it as $G(A) = \{7,16\}$. We assume that node 5 is a new anycast source as an example, and it broadcasts the JOIN_REQ packet, which includes the ID of node 5 and a broadcasting sequence number. When the JOIN_REQ packet arrives at node 6, node 6 updates the *upstream node* field of the packet with its own address and forwards the packet to its neighbours. Once node 7, an anycast server receives the packet, it sends a MESH_ACK packet back to node 6 which is the upstream node of node 7. Then node 6 realises that it is on the anycast mesh, it updates its routing table and relays the

MESH_ACK to its upstream node, node 5. After this procedure, a route between the anycast source and one of the anycast servers is established. Similarly, there is a route between node 5 and node 16, which is another server of the anycast group.

In order to make it clear, in this chapter, we use $P\{node_1, node_2, ..., node_n\}$ to donate a path between $node_1$ and $node_n$; and *link(node$_1$, node$_2$)* to represent a link between $node_1$ and $node_2$.

MARP prefers a path that contains more existing forwarding nodes for route efficiency and maintenance reasons. For example, in Figure 6.3, there are three paths between node 5 and node 16 which have the same distance, namely $P\{5,6,11,16\}$, $P\{5,10,11,16\}$, and $P\{5,10,15,16\}$. Because of the preference rule, MARP chooses the path $P\{5,6,11,16\}$ as a route for the anycast source and the anycast server. To another anycast source, node 20, there are two paths to anycast server node 16, $P\{20,15,16\}$ and $P\{20,21,16\}$, the distances of the two paths are equivalent. In this circumstance, MARP prefers the path which makes the mesh having less neighbour nodes (which will be defined at the rest of this section), therefore $P\{20,15,16\}$ is chosen as the route to anycast server node 16. Based on this mechanism, the number of total forwarding nodes and neighbour nodes is minimised, furthermore, the cost of maintenance is reduced.

Figure 6.3 shows the result after the anycast mesh is established. Once the anycast mesh is created, an anycast source holds all the routes to the anycast group members, respectively, therefore a "best" server can be chosen based on given metrics. For example, if we choose the "best" server based on the shortest path, then the anycast server node 7 is chosen for the source node 5 on path $P\{5,6,7\}$; and the anycast server

node 16 is chosen for the source node 20 on path $P\{20,15,16\}$. When there are packet deliveries in the anycast group, the packets are only transported by the forwarding nodes among the sources and the receivers.

Anycast *Group Neighbour* nodes are defined as the nodes that are directly connected to at least one anycast mesh node. In Figure 6.3, nodes 1, 2, 3, 8, 10, 12, 17 and 25 are the group neighbour nodes. Group neighbour nodes are defined for the on-mesh broadcasting. In MARP, only the mesh nodes and group neighbour nodes forward the on-mesh broadcasting packets, while the other nodes do not forward the on-mesh broadcasting packets. Therefore MARP can effectively prevent the potential traffic storm and subsequently reduce the network load.

6.3.3 Anycast Mesh Maintenance

In order to provide the up-to-date information of network topology in ad hoc networks, the route information has to be updated in time, and kept consistent with the instant practical network status. Anycast mesh maintenance includes two parts: on-mesh route discovery and flooding route discovery.

A. On-mesh Route Discovery

Each anycast source periodically broadcasts a MESH_REQ packet, and only the mesh nodes and the group neighbour nodes forward the packet. Similar to the JOIN_REQ packet for mesh creation, a mesh node or a group neighbour node updates the *upstream node* field of the received MESH_REQ packets with its own address and

forwards the modified packets to the next nodes. When the MESH_REQ packet arrives at an anycast receiver, a MESH_ACK packet is sent back to the anycast source along the path from which the MESH_REQ packet came. After the on-mesh route discovery procedure, the updated mesh is established, and the forwarding nodes and the group neighbour nodes are refreshed as well.

Based on the rule of MARP, the nodes more than two hops away from the mesh can not receive the MESH_REQ. This mechanism can efficiently save the valuable network bandwidth in mobile ad hoc networks, and prevent the potential traffic storms. For example, in Figure 6.3, there are nearly ¼ nodes (node 14, 19, 23, 24, 27, 28, and 29) are not involved in the MESH_REQ packet broadcasting.

More importantly, on-mesh route discovery procedure can repair most link failures caused by node movements in ad hoc networks. For example, we assume that the mobile node 11 in Figure 6.3 is power off, then there are three link failures occuring, namely link (6, 11), link (10, 11) and link (11, 16), when the anycast source node 5 submits the MESH_REQ packet, node 10 will deliver it to node 15, and further to node 16, then a path $P\{5,10,15,16\}$ is established. Figure 6.4 shows the ad hoc network after the link failure recovery.

Previous research [AGG99] has shown that most of the on-mesh link failures can be repaired by on-mesh route discovery, but it can not solve all the possible link failures and network partitions. For example, if there are two link failures of link (5, 10) and link (12, 17) in Figure 6.4, the original mesh is then divided into two parts, and it can not be repaired by the on-mesh route discovery.

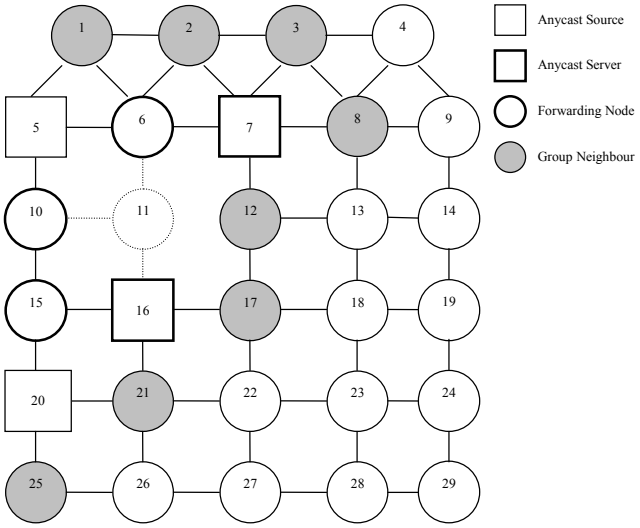

Figure 6.4. The Ad Hoc Network after Link Failure Recovery

The on-mesh route discovery procedure tries to keep all the anycast sources and the anycast servers connected with each other by the mesh. This is important to provide a reliable anycast service in ad hoc networks. For example, to the source node 5 in Figure 6.4, if the "best" receiver node 7 is not reachable, then there is an alternative receiver node 16, which can provide the same service.

B. Flooding Route Discovery

Flooding route discovery is an important procedure to maintain an anycast mesh, although it is expensive in terms of network bandwidth. When a node initiates the flooding route discovery procedure, the JOIN_REQ packet is broadcasted to all its neighbours, and every node in the ad hoc network will forward the packet. The JOIN_REQ packet will cover all nodes of the ad hoc network, therefore, it brings an up-to-date view of the network topology and the anycast mesh. MARP does not

perform flooding route discovery frequently, because of the expensive network bandwidth consumption. It happens in several cases:

● A new anycast source joins the group. When a mobile node wants to join an anycast group as an anycast source, and it is at least 2 hops away from the mesh. Therefore, it has no knowledge about the anycast mesh; it has to use flooding route discovery procedure to join the anycast mesh.

● An anycast source is separated at least 2 hops away from the original mesh caused by link failures. This is similar to the previous case.

● Network partitions. This case can only be recovered by flooding route discovery.

6.4 Detailed Description of MARP

The data structures and the details of the procedures are described in this chapter. It includes the routing table structure, the processing of duplicated data packets or control packets, joining and leaving a group, and so on.

6.4.1 Data Structure and Packet Header

The structure of MARP packet is shown in Figure 6.5. There are eight fields in the packet header. The type field indicates the type of the packet. It is one of the following values:

- DATA: data packet
- JOIN_REQ: flooding route discovery packet
- MESH_REQ: on-mesh route discovery packet
- MEM_REQ: route discovery packet sent by a new anycast receiver
- MESH_ACK: reply packet to a route discovery packet

Type	Sequence Number	Group Address	Source Address	Upstream Node	TTL	FC	NC

Figure 6.5 The Structure of MARP Packet Header

The sequence number field is used to indicate the sequence of the packet, and it is also used to identify whether the received packet is a new packet or a duplicated one for the receiving nodes. The group address field indicates the different anycast groups. The source address field records the source address where the packet was initiated. The upstream node field shows the previous node address from which the packet came from. The TTL field is the same as that in IP protocol. The FC *(Forwarding Count)* denotes the number of forwarding nodes along a path. Finally, the NC *(Non-forward Count)* shows the number of non-forwarding nodes, and this field is also used to decide a node is a group neighbour node or not.

Each node in the ad hoc network holds a routing table, and the table is updated from time to time. The structure of the routing table is shown in Figure 6.6.

Group Address	Forwarding Flag	Forwarding Timeout	GroupNeighbour Flag	GroupNeighbour Timeout
…	…	…	…	…

Figure 6.6 The Structure of Routing Table in MARP

When a node becomes a forwarding node or a group neighbour node, MARP sets the forwarding flag and goupneighbour flag, respectively. Forwarding timeout and groupneighour timeout are two timers which keep the information fresh, namely, when the fields decrease to zero, the forwarding node or the group neighbour node loses its function.

In order to deal with the duplicated broadcasting packets, MARP designs a ReqCache and a DataCache for each node. The structures of the two caches are shown in Figure 6.7 (a) and (b), respectively.

Group Address	Source Address	Sequence Number
….	….	….

(a) The Structure of DataCache

Group Address	Source Address	Sequence Number	Upstream Address
….	….	….	….

(b) The Structure of ReqCache

Figure 6.7 The Two Caches in MARP

6.4.2 Initiating and Relaying JOIN_REQ and MESH_REQ

Each anycast source needs to make sure that it is on a given anycast mesh. For this reason, every source keeps a timer, MESH_REFRESH_TIMEOUT. The timer decreases from an initial value; when the value equals to zero, the source will broadcast a MESH_REQ packet on the mesh, and only the mesh nodes and the group neighbour nodes forward the packet. The anycast receivers will send a MESH_ACK back once

they receive the MESH_REQ packet. The on-mesh route discovery can fix the local link failures as we discussed previously. At the same time, the anycast source set the MESH_REFRESH_TIMEOUT to the initial value. If there is no MESH_ACK packet received by the anycast source after a given time interval, MESH_INTERVAL, then the anycast source assumes that it is separated from the mesh, and it will initiate a flooding route discovery procedure, namely, it broadcasts the JOIN_REQ packet to all the nodes in the ad hoc network. After this procedure, the anycast source can be reconnected to the mesh.

When a node receives the route discovery packet, it consults the received packet with ReqCache to check whether the incoming packet is a new one or a duplicated one. The group address, the source address and the sequence number are used for the packet comparison. If it is a new packet, the node records the related information in its ReqCache and takes the related actions, otherwise it discards the packet.

Every node in the ad hoc network relays the JOIN_REQ packets, while only the mesh nodes and the group neighbour nodes relay the MESH_REQ packet. Before a route discovery packet is relayed, the *upstream node* field of the packet should be updated with the current node address for the later reversal path establishment. A relaying node increases FC by one if it is a forwarding node; otherwise, NC is incremented by one.

6.4.3 Initiating and Replying MESH_ACK

The MESH_ACK packet is used to build a route between a source and a receiver. Once a receiver receives a JOIN_REQ or MESH_REQ packet, it initiates a

MESH_ACK packet and sends the packet back to the anycast source according to the *Upstream Node* field information in the ReqCaches on the reversal path. When an intermediate node receives the MESH_ACK packet, it realises that it is a *forwarding node*, then it opens an entry for the route and sets the *Forwarding Flag* and refreshes the *Forwarding Timeout* in its routing table. The packet is then forwarded to the upstream node until it reaches the source.

Note that all the packet delivery in ad hoc networks based on broadcasting to 1 hop away nodes. When a MESH_ACK packet is sent back to its upstream node by an anycast receiver, actually the anycast receiver broadcasts the packet at the 1 hop range, then its upstream node becomes a forwarding node, and the other nodes which receive the MESH_ACK packet become the group neighbour nodes. The group neighbour nodes set the *GroupNeighbour Flag* and refresh the *GroupNeighbour Timeout* in its routing table.

It is possible that there are several applicable routes between a source and a receiver, MARP chooses the shortest path and prefers the path with more existing *forwarding nodes*. When a receiver obtains a first non-duplicate route discovery packet, it holds the information of the packet head in its CacheReq for a reasonable period, ROUTE_WAITING. During that period, the receiver may receive several route discovery packets from the same source, then the receiver calculates the weighted path length by the following formula

$$\alpha \times FC + \beta \times NC, \ \ (\alpha + \beta = 1)$$

The parameters α and β can balance the preference between the existing *forwarding nodes* and the hops in total of the route. Once the "best" route is decided, the receiver replaces the entry for that source with the new route in its CacheReq, and submits the MESH_ACK packet back to the source on the chosen route.

6.4.4 Receiving and Forwarding DATA Packets

Every mesh node has a DataCache, which is configured to deal with the duplicated data packets in ad hoc networks. Once a DATA packet is received, the node consults DataCache to check if the packet is a duplicated one. If the incoming DATA packet is a duplicated one, it will be discarded, otherwise, the information of the packet head will be reflected into the DataCache. The DATA packet will be relayed if the receiving node is a forwarding node.

6.4.5 Joining and Leaving a Group

It is simple when a node wants to join an anycast group as a source. If it is a mesh node or a group neighbour node, the node just initiates a MESH_REQ packet, and broadcasts it on the mesh, and then the routes to the receivers will be established; otherwise, the joining node broadcasts the JOIN_REQ packet, which has been described previously.

When a node intends to serve as a receiver, it waits for the MESH_REQ or JOIN_REQ packet for a period of MESH_INTERVAL. It will receive one route

discovery packet generally if it is a mesh node, the group neighbour node, or a node two hops away from the mesh, then the new receiver join the mesh through feeding back the MESH_ACK packet. If the node does not receive any route discovery packet during the MESH_INTERVAL period, it broadcasts MEM_REQ packet which is similar to JOIN_REQ, and tries to connect to the mesh. The *TTL* of MEM_REQ is set as a small value in order to prevent it from roaming in the ad hoc network. The node, which received the MEM_REQ packet, takes analogous operates on receiving a JOIN_REQ packet. The node needs to update an entry in its ReqCache.

Leaving an anycast group in MARP is very simple. The action needs no additional control message, the leaving nodes just do not respond the MESH_ACK packets to the subsequent route discovery packets. For the leaving anycast source, there is no MESH_REQ packet any longer, therefore, the routes to the anycast receivers will disappear with the refreshment of the mesh.

6.5 Summary and Future Work

We proposed a mesh based anycast routing protocol (MARP) for the ad hoc networks. The proposed algorithm improves the robustness for routing in the dynamic ad hoc mobile networks, furthermore, it also reduces the bandwidth consumption for control packets. These two advantages of the proposed MARP have been demonstrated in section 6.3 by the examples. We discussed the architecture, and presented the data structure in MARP, and detailed the routing methods.

Because of the time limitation, the mathematic analysis and performance evaluation of MARP through simulations is the topic for future work.

Chapter 7
Anycast in Web Database Applications

Databases are always important for computer applications. Especially when the Internet became popular in the 1990s, the scope of database applications extended widely all over the world, becoming so called Web-based databases. Unfortunately, most of the current web-based database systems suffer from poor performance, complicated heterogeneity, and difficulty in synchronization. In this chapter, we propose a novel architecture for web-based database systems based on multicast and anycast protocols to deal with these issues. We design a middleware, called castway, which locates between database server and Web server. Every castway in a distributed system operates as a multicast node and an anycast node independently. The proposed mechanism can balance the workload among the database servers, and offers the "best" server to serve for a database request. The model is independent from the Internet environment; it can synchronise the databases efficiently and automatically.

7.1 Introduction

The web-based database is an important and key component for applications on the Internet, such as, E-commerce, E-banking, etc. Because of the huge number of users, most of the popular web-based database systems suffer from poor performance, network congestion, heterogeneity, and so on. The distributed database model is a

solution which can improve the performance, but the web-based database applications introduce new problems for the model, such as job distributing, load balance, data synchronization, heterogeneous database platforms integration, and so on. For example, distributed replication provides high availability, fault-tolerance and enhanced performance. But we must pay for these benefits: replication adds great complexity to the system development and maintenance [GUE97] [NIC00]. Most of all, replication jeopardises data consistency. In turn, mechanisms have to be employed to enforce the data consistency. Maintaining the data consistency is very expensive [BIR95].

In order to meet the demands of Internet applications, researchers and industry players are working hard to develop new protocols, Internet services, and applications for the ever increasing and changing requirements. Multicast and anycast services are the results of this kind of developments.

Multicast [DEE90] is an Internet service, which tries to transmit packets to a group of hosts and guarantees the delivery with minimum bandwidth consumption. It is an important service for distributed systems and applications in terms of efficiency and robustness. The multicast communication services have been widely recognized as a very useful tool for Internet applications, such as, database synchronization, replicated database updating, audio and video conference, and so on.

Anycast [PAR93] is a service providing a stateless best effort delivery of an anycast datagram to at least one host out of the *n* mirrored servers, and preferably only one host, which serves the anycast address. Anycast service tries to find the “best” server

among the replicated or mirrored servers of the anycast group. Anycast is powerful in information retrieval essentially.

In this chapter, we propose a novel Web-based database model by taking the advantages of the multicast and the anycast services. The model is independent from the heterogeneous database platforms; it can synchronise the databases efficiently and automatically. It also improves query performance based on the advantages of multicast and anycast protocols.

The rest of the chapter is organized as follows. Section 7.2 discusses the related work about anycast and multicast applications in database. In section 7.3, we propose a novel model for Web-based databases based on the multicast and the anycast protocols. The algorithms applied for the proposed model are presented in section 7.4. Performance evaluation is presented in Section 7.5. Finally, in Section 7.6, summary and future works are described.

7.2 Related Work

Multicast [DEE90] is defined as a service which tries to send packets to every the member of a multicast group. Its capability has been recognized as an important facility for networks and the Internet because of its growing usage in distributed systems. [HOL99] presented four protocols for distributed replicated databases that take advantage of atomic broadcast systems to simplify message passing and conflict resolution in hopes of making replication efficient.

1. Broadcast all protocol. This protocol requires that a transaction initiated at a server site broadcasts all its operations (read or write) to all other sites using atomic broadcast.

2. Broadcast writes protocol. This protocol executes read operations locally and broadcast only write operations to the other sites.

3. Delayed broadcast writes protocol. This protocol attempts to completely localize transaction execution. The protocol defers update operations until commit is ready, then a single message with all updates is sent to all other sites.

4. Single broadcast transactions protocol. This protocol maintains a version number with each page in the database to make sure the correct sequence of read in databases.

These protocols can be applied to replicated database recovery as well [HOL00]. [KEM98] proposed a family of replication protocols based on multicast in order to address some of the concerns expressed by database designers regarding existing replication solutions. All these work show that the multicast service is a good solution for the data synchronization and data recovery for distributed systems.

The anycast protocol is a powerful new service on the Internet, the related work about anycast has been presented in chapter 2 of this thesis. [JIA00] proposed an idea of integrating multicast service and anycast service for the Internet service: a group of replicated (or mirrored) servers that provide anycast services may also provide

multicast services and need multicast to archive consistent update operations, whereas anycast routing may help multicast requests to reach the "nearest" member in a multicast group. [JIA01] proposed a novel efficient mobile multicast protocol (MMP), taking advantage of the anycast routing technology.

7.3 A Middleware for Web-Based Database Model

The Web-based database applications have been applied widely, therefore, any coming proposals about Web-based database architecture or model should be practical with limited modifications. Therefore, we propose our methods for Web-based database applications with the following primary rules:

- Using exist Internet protocols, services, etc.
- Limited modifications to the existing systems.
- Independent from DBMS, OS and Web server.

Based on the previous work, we find that the combination of the multicast and the anycast services can provide a bi-directional service. The multicast service provides a "one-to-many" delivery service, and it can guarantee the delivery. This characteristic can be used in data synchronization for Internet based distributed databases; meanwhile, the anycast service offers a service which can find the "best" one out of many. This feature can fill the request to find the "best" database server among the distributed group in terms of work load, performance, bandwidth, and so on. Therefore, we propose a middleware for the Web-based database (*Web-DB in short*) model based on the multicast and anycast protocols. In our proposed middleware, there are two

groups: a multicast group $G_m = \{m_1, m_2, m_3, ..., m_k\}$, and an anycast group $G_a = \{a_1, a_2, a_3, ..., a_i\}$ represents a mirrored database in the Internet, and $a_j (j = 1,2,3,..., i)$ does the same. Furthermore, we let $m_j = a_j (j = 1,2,3,...k, k = i)$ for a group of mirrored Web-based databases.

By integrating the advantages of the multicast service and the anycast service, the proposed middleware inherently has the following characteristics:

- Working with heterogeneous Internet environment.
- Balancing the work load among the mirrored database servers.
- Improving the performance in terms of request responding.

7.3.1 The Architecture of Web-DB Model

The architecture of the Web-DB model with the proposed middleware is shown in Figure 7.1. We keep the traditional three-layer Web based database model, and place the middleware, castway, between the Web server and the backend database server at every site of the mirrored database systems. Working as a middleware, the castway is independent from the DBMS, the operating system and the Web server. It integrates the functionalities of the multicast and the anycast protocols to serve for the distributed database systems.

The anycast resolver module in Figure 7.1 is optional. The module exists to provide anycast routing services for the application-layer anycast. As described previously, the

anycast routing service can also be served by the network layer [XUA00] [XUA01] [JIA04]. In this chapter, we take the application-layer anycast as a default anycast routing service, because the support of network layer anycast routing is limited currently.

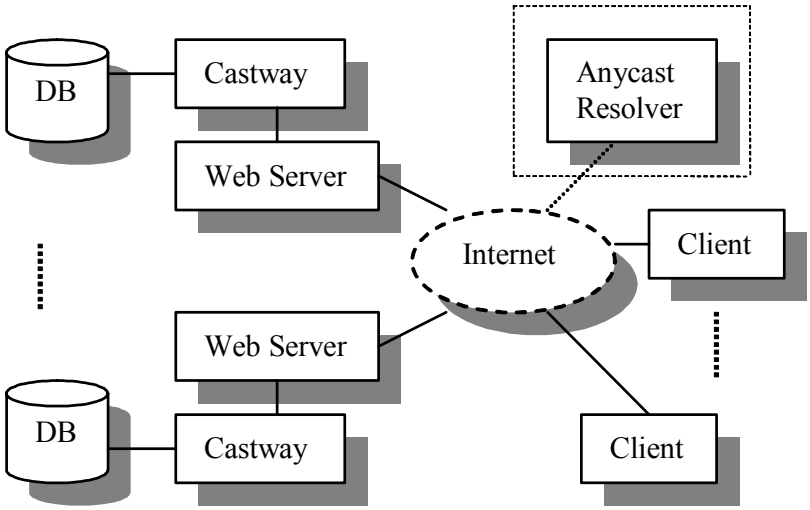

Figure 7.1 Architecture of Web-Based Database Model

Once a client initiates a database request to a distributed Web-based database system, the destination address of the request will be set as the anycast address which is related to the distributed system, the relative anycast resolver will responds the unicast address of the "best" database server among the anycast group, and the destination address of the request will be replaced with the unicast address; and then the updated database request will be forwarded to the "best" database server using the traditional IPv4 routing protocols.

We give two definitions here in order to make the following descriptions clear. An *original transaction* means that a transaction is received by a database server of the

distributed database group for the first time. The site which processes the original transaction is the *original site.* While a node receives an original transaction, it will copy the transaction and broadcasts the copies to the multicast group, these transactions are called *copy transactions*. The site serves for the copy transaction is called the *copy site.* Copy transactions are used for data synchronization among the distributed databases.

When a database request arrives at the selected site, the castway catches request, it strips the transaction from the packet and submits it to the backend database engine to execute. At the same time, the castway assembles the commands of the transaction (except the commit command) in a new packet addressing with the multicast address and broadcasts it in the multicast group G_m to synchronize the distributed databases. The details of the synchronization will be presented in section 7.4. The castway will forward the result from the backend database to the client where the request comes from.

The castway examines the received multicast packets. If the source address of the multicast packet is the same to the current node's own address, then the node discards the packet; if the two addresses are different, that means the received packets are used for database synchronization only, then the node forwards the transaction to the backend, and discards the result which is feed back by the backend database engine.

A database request can be initiated anywhere from the Internet, anycast mechanism inherently balances the workload among the mirrored databases, and also choose the "best" server to serve the clients according to the given criteria. The multicast protocol

guarantees the delivery of the copy transactions, which is employed to synchronize the databases. The two services are implemented in the middleware, castway, which is a critical component of the proposed model. We will present more details of the castway in following subsection.

7.3.2 The Structure and Mechanism of the Castway

A castway locates between a Web server and a database server, it connects the two servers and acts as a communication bridge of the two servers, but the castway is independent from the database server and the Web server.

The original CGI module of the traditional three layer architecture is extended to a castway as a middleware. The castway includes three modules: interface module, multicast module and anycast module, shown as Figure 7.2.

The interface module connects the castway to the Web server. It receivers the requests, and checks every incoming request to find that weather it is an *original transaction* or a *copy transaction*.

If the incoming transaction is a original transaction, then the interface module examines the transaction and makes sure that it is a *read only transaction* (includes select operations) or a *update transaction* (includes update, delete or insert operations). If it is a read only transaction, the requests will be forwarded to the database engine by the multicast module without multicasting the transaction; however, if it is an update transaction, the request will also be delivered to the backend database engine, and the

transaction will be broadcasted to the multicast group G_m. The database engine submits the results of transaction back to the interface module, and then it is sent to the client through the Internet.

If the incoming transaction is a copy transaction, the original node discards the transaction, because it has been executed; the copy nodes forward the transaction to the backend database engine, and the interface module discards the result from the engine. Table 7.1 summaries the actions in a castway. The interface module forwards the incoming transactions to the anycast module only when it considers that the current server is overloaded.

Castway Modules	Original Transaction	Copy Transaction
Interface Module (IM)	Forward to MM Submit Result to client Forward to AM *	Forward to MM Discard result Forward to AM *
Multicast Module (MM)	Forward to Database Engine Broadcast Multicast Packets	Forward to database engine
Anycast Module (AM)	Deviate	Deviate

Table 7.1 Summary of the Actions in Castway

The multicast module takes the responsibility to broadcast the original transaction to the multicast group, and forwards the transactions to the backend database engine. This module works independently as a multicast node in the Internet. All the multicast modules in the distributed database system use the same protocol, and the protocol can

be chosen from spanning tree, reverse path forwarding (RPF), core-based tree (CBT) [ROC00], and so on.

The anycast module could be regarded as an independent anycast node in the Internet. All nodes of the anycast group G_a work together to keep the anycast routing service. The anycast group can take any existing anycast routing algorithms, such as the Shortest-Shortest Path Method (SSP), the Minimum Distance Method (MIN-D) [XUA00], or the Requirement-based Probing Algorithm (RPA) [YU02], and so on. Once the interface module finds the work load of the backend database engine is heavy, it will forward the incoming transactions to the anycast module, rather than the multicast module, and the anycast module will deviate the transactions to the "best" server in order to improve the performance.

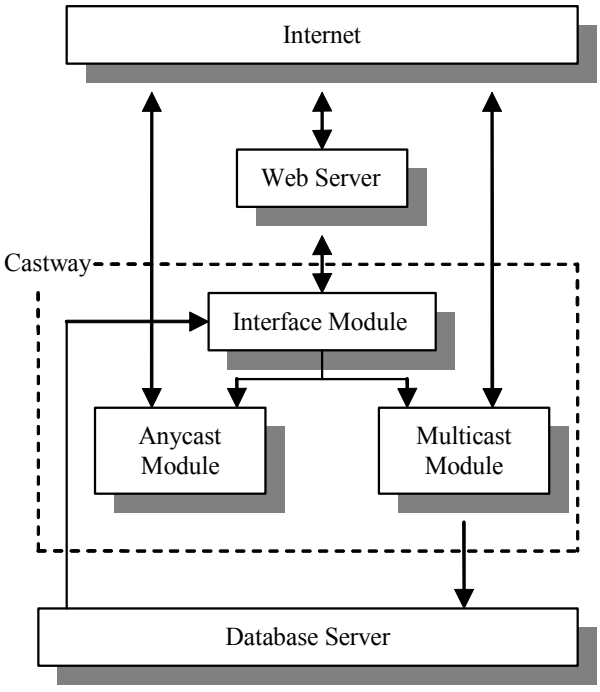

Figure 7.2. The Structure of Castway

7.4 Algorithms for the Proposed Model

In this chapter, we assume the network and servers are reliable, therefore, we do not include reliable and fault-tolerance issue in the design. We also suppose the anycast algorithms and multicast algorithms are fully developed; they can be any ones from the existing algorithms. Therefore, in this part, we only discuss the atomic multicast update algorithm in details for the proposed Web-DB model.

A transaction is a sequence of read and write operations on the data items that is executed atomically. For the atomic multicast described in this chapter, we consider it satisfying the following properties [HAD93]:

1. If a site broadcasts a message *m*, the primitive ensures that the message will be delivered to all operational sites.
2. If a site delivers message *m*, then all operational sites deliver *m*.
3. If sites *p* and *q* deliver broadcast messages *m* and *m'*, then *m* and *m'* are delivered in the same order at all sites.

The procedure of the atomic multicast update algorithm is described below:

1. An update transaction is initialised at one site (original site) with a unique transaction ID.
2. The castway of the original site passes the transaction to the backend database engine; at the same time, the castway copies all the statements in the transaction except the "commit" statement, and multicasts the copies to the multicast group.

3. The interface module of the original site discards the copy transaction.

4. The copy sites execute the copy transactions at their own backend database engines.

5. When the original transaction is committed, the multicast module multicasts the commit demand to all the members of the group.

6. The original site discards the incoming "commit" command, and the other sites commit the copy transactions. After that, all the members in the multicast group are synchronized.

The pseudo code of the atomic multicast update algorithm is shown in List 7.1.

```
The Atomic Multicast Update Algorithm
/* The algorithm keeps working all the time*/
While True
 {
  Ti = An incoming transaction.
/* To test wether the current node is the original site or not */
   If  SourceAddress (Ti) = Address (Current Site) then
/* The action for the original node */
       DiscardTransaction (Ti),
    Else if
       {
/* The actions for the copy nodes */
        ForwardTransaction(Ti, Current_DB_Engine), // Ti is forwarded to local database
         Ti = Ti -  "commit";
/* Multicast the new transaction */
         Multicasting (Gm, Ti);  // Gm is the multicast group
            While True
              {
                if  TransactionCommit (Ti, Current_DB_Engine) = True
                  { Break};
              }
          Ti = "commit";
/* Issue the 'commit' statement to every site of the system  */
          Multicasting (Gm, Ti);
       }
}
```

List 7.1 The Pseudo Code of the Atomic Multicast Update Algorithm

7.5 Performance Evaluation

In order to evaluate the performance of the proposed model, we use the *common Web transaction* defined below as a benchmark to compare the performance of the Web-DB model. The *common Web transaction* is the actions that the database administrators normally take for the data synchronization among the distributed databases or the methods to choose one of the mirrored servers for information retrievals. The *common Web transaction* is defined as following: for an anycast service, the *common Web transaction* chooses one of the distributed database server randomly; for the multicast service, the *common web transaction* builds connections to each server of the multicast group and independently delivers the packet to the servers respectively.

In order to measure the synchronization performance of distributed database systems, a *Replication Transaction Time (RTT)* is defined as follow: To a group of replicated servers $\{S_1, S_2, ..., S_i, ..., S_n\}$, the time last for each replication is denoted as T_i, then the *replication transaction time* for the distributed database synchronization is $Max\{T_i\}, i = 1,2,...,n.$

In order to compare the performance of the *common Web transaction* and the anycast based web transaction, we conducted a simulation using *ns 2* [NS02]. The scenario is that there are two LANs (bandwidth: 10M, delay: 20ms) connected by a physical link (bandwidth: 1.5M, delay: 40ms). All the servers are located in one LAN, and all the clients are located in another LAN. There is some irrelative web traffic as background. We tried to download a file from two replicated servers using the *common Web*

transaction method and the anycast method, respectively. The size of the file varies from 5Mb to 50Mb, and the result is shown as in table 7.2.

Size(M)	Common(S)	Anycast(S)
5	1.081196	1.109124
10	2.082446	2.080206
15	3.079946	2.080206
20	4.081196	4.090374
25	5.082446	5.080206
30	6.083696	6.081456
35	7.081196	7.078956
40	8.082446	8.080206
45	9.083696	9.081456
50	10.081196	10.078956

Table 7.2 the Transaction Times of Common and Anycast Methods

In order to find the performance difference between the common Web transaction method and the anycast transaction method, we transfer the data of table 7.2 to Figure 7.3. The vertical axis denotes the value of the common Web transaction (common in short) method deducts the value of the anycast method.

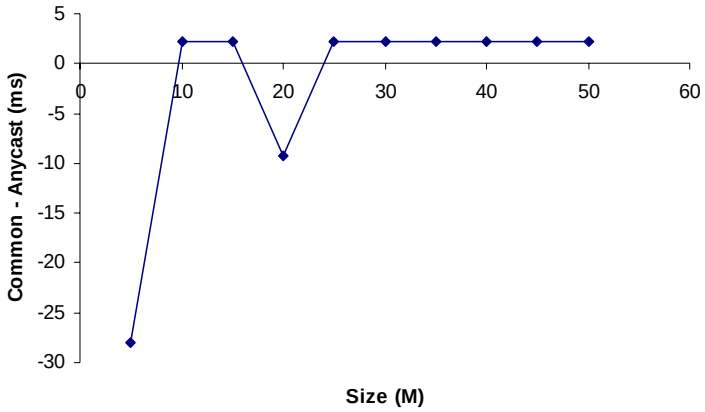

Figure 7.3 Performance Difference between Common and Anycast Method

From Figure 7.3, we found that the performance of the anycast method is better than that of the common Web transaction method in most of the individual download transactions. The difference of the first and fourth points comes from the unstable background Web traffic. Normally the difference is very limited, because the web traffic is not heavy and the resource, such as bandwidth, delay for different paths are the same in our simulation. The advantage of the anycast method for an individual transaction is limited, even worse, such as the first and the fourth data in Figure 7.3, but from the system point of view, the performance of the anycast method is better than that of the common web transaction method.

To compare the performance of the common Web transaction method and the multicast method in data replication, we conducted another simulation: the network environment is the same as the last simulation, and we tried to synchronise three replicated servers, one server locates in LAN 1, and the other two locate in LAN 2. We assume the original site is located in LAN 1, and we try to synchronise the other two servers in LAN 2. The size of data to be transported is 2.5Mb. The multicast algorithm applied in the simulation is the Dense Mode (DM) [NS02]. We vary the bandwidth between the two LANs as 1.5Mb, 2Mb, 5Mb, 10Mb, and 50Mb for each instance. The result is listed in table 7.3.

Bandwidth(M)	Common(S)	Multicast(S)
1.5	0.56711	0.56599
2.0	0.56650	0.56571
5.0	0.56554	0.56521
10	0.56521	0.56504
50	0.56494	0.56490

Table 7.3 RTT of Common and Multicast Replications

As we did previously, we transfer the data of table 7.3 to Figure 7.4. The vertical axis is the values of the RTT of the common Web transaction deduct the RTT of the multicast method, respectively.

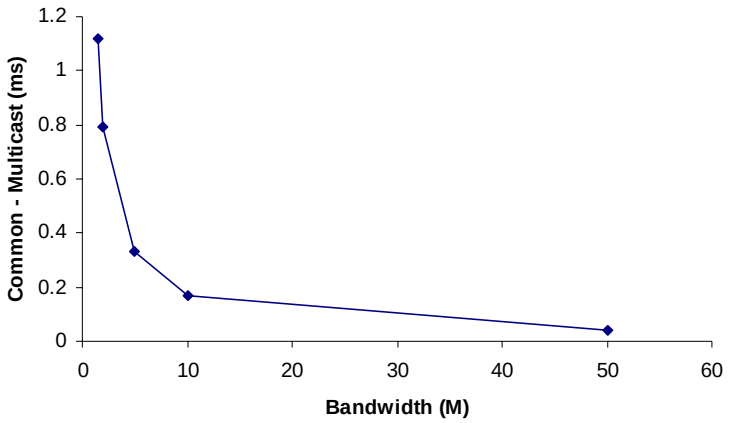

Figure 7.4 Performance Difference between Common and Multicast Replications

From Figure 7.4, we conclude that from the viewpoint of RTT, the performance of the multicast method is better than that of the common Web transaction method, especially when the network bandwidth is limited.

In another simulation, we vary the number of replica servers and examine the RTT of the multicast methods and the common Web transaction method. For the multicast method, we use two multicast algorithms: Centralized Multicast (CM) [NS02] and Dense Mode (DM). We use COM to stand for the common Web transaction method. The result is shown in table 7.4.

Nodes	DM(ms)	CM(ms)	COM(ms)
2	580.7	595.1	564.6
3	573.7	601.1	567.7
4	577.7	594.1	578.2
5	585.7	599.1	602.9
6	575.7	597.1	608.7
7	579.7	604.1	611.9
8	584.7	598.1	612.6
9	575.7	603.1	612.7
10	585.7	599.1	614.2

Table 7.4 Comparison of Multicast and Common Replication with Difference Replicas

We transfer the data of table 7.4 to Figure 7.5, which is shown below.

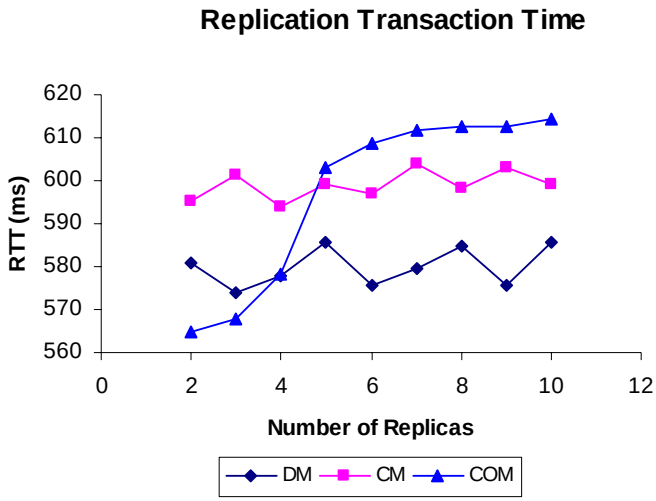

Figure 7.5 The Multicast and the Common Replications with Difference Replicas

From Figure 7.5, we find that when there are only two or three replicas, the performance of the common replication algorithm is better than that of the multicast methods; however, with the increasing number of replicas, the multicast methods are better than COM method.

7.6 Summary

Multicast and anycast are two powerful services in the Internet, in this chapter, we proposed a novel middleware for Web-based databases based on the multicast and anycast protocols. The proposed middleware possesses a number of advantages, including:

- An independent middleware, it can work in the heterogeneous environment.
- Limited modification at the Web server or the database server.
- Workload balance capability among the distributed database servers.

An atomic multicast update algorithm has been discussed in details for the proposed model. The algorithm can guarantee the data synchronization among the distributed servers based on the multicast protocol.

The simulation shows that the proposed middleware performs well. Compared with the benchmark methods, the middleware improves the performance of information retrieval among mirrored database servers, and it offers a steady and efficient solution for data synchronization.

Chapter 8
The Minicast Service

Anycast and multicast protocols are two important Internet services. The combination of the two protocols can provide new and practical services. In this chapter we propose a new Internet service, minicast service, based on the two protocols: in the scenario of n replicated or similar servers, delivering a message to at least m members, $1 \le m \le n$. Such a service has potential applications in information retrieval, parallel computing, cache queries, etc. The service can provide the same Internet service with an optimal price through the achievement of, e.g., less bandwidth consuming, less network delay and so on. In this chapter, we also design a multi-core tree based architecture for the minicast service, and further more, we present the criteria for calculating the subcores among a subset of minicast members. The simulation shows that the proposed architecture can even the minicast traffic, and the proposed service can reduce the consumption of network resources.

8.1 Introduction

The dramatic development of the Internet has led to many interesting communication paradigms which provide all sorts of Internet-based services. The multicast service delivers a packet from a source to all the n members in the multicast group [DEE90]. This service is currently applied widely in the Internet, such as data synchronization,

Internet meeting, etc. The anycast service tries to send a packet from the sender to the "best" receiver among the *n* replicated servers [PAR93]. The anycast protocol is very powerful in information retrieval; it can reduce the cost for an Internet service.

The combination of anycast and multicast protocols can provide new practical Internet service. PAMcast [CHA02] is an example of the combination. PAMcast generalizes both anycast and multicast packets and provides a message delivery service to *m* out of the total *n* group members, where $1 \le m \le n$. In this paradigm *m* is a given constant, although it can be changed from time to time. The PAMcast service emphases on message delivery more than on information retrieval, furthermore, it is not flexible with a fixed parameter *m,* and this is also costly for network resource consuming.

Information retrieval is increasingly important with the every increasing content of the Internet. It is a challenging topic to provide information retrieval with less network resource consuming.

In this chapter, we propose a new packet delivery service – minicast – which pays more attention to information retrieval, and generalizes both anycast and multicast services. The minicast service provides a service for delivery to at least any *m* out of total *n* group members, $1 \le m \le n$. Similar idea has been proposed in operating system research [ROU01]. Minicast service has potential and is applicable for a wide range of applications. For example:

• Parallel information retrieval. Consider a scenario of searching a message from a group of similar web sites, we can get the result after at least *m* probing statistically. If

we try PAMcast with parameter *m-1*, maybe we can not get the desired result for the first time, and have to start another PAMcast probing, that is expensive in terms of network bandwidth and service delay. On the other hand, the minicast submits the probing message to at least *m* receivers (*m* depends on the statistical calculation, and it can be adjusted according to the network status), the possibility of obtaining the requested information is improved, while the usage of bandwidth is more economical.

• Parallel cache queries. Assume a group of caches store a data items. A client might minicast a query to at least *m* caches in the group in hope that at least one has the desired message. We configure the parameter *m* so that the possibility of a successful hit is optimal with the least cost.

• Parallel grid computing. In the circumstance of grid computing, it is not possible to use all related computers to serve for one job when the number of members of the grid is big. There is a balance between the grid computing performance and the number of computers used (the cost). The minicast bears the computing performance in mind, and decides the suitable *m* computers in terms of price.

• Parallel downloading. Suppose there are *n* mirrored file servers. A client might minicast queries to at least *m* of the servers, requesting each of them transmit a portion of a particular file with the least network bandwidth usage or with the least network delay. The critical issue here is how to find these *m* servers to provide that kind of quality-of-service with least cost.

The rest of this chapter is organized as follows: Section 8.2 introduces the related work of anycast and multicast, and the combination of the two services as well. The minicast architecture is presented in Section 8.3. Section 8.4 discusses the core selection algorithm and the message delivery algorithm for minicast. The performance evaluation and analysis is described in Section 8.5. Finally, in Section 8.6, the conclusions and the future work are presented.

8.2 Related Work

[ROC00] presents a survey of multicast, and it covers most of the issues in multicast research. It summarises that there are five classes of multicast routing algorithms: flooding, spanning tree, reverse path forwarding (RPF), core-based tree (CBT), and solution to the travelling salesman problem (TSP); Reliable transmission service of multicast includes ARQ solution (ACK and NAK-based), FEC-based solutions, and hybrid solutions; the paper also discusses adding congestion control to multicast transmissions and some other related issues, such as, QoS-based multicasting, heterogeneity support, and so on. There are also some interesting researches on core based multicast algorithms [GUP03] [JIA02b] [THA97] [YOO00].

The related work of anycast [PAR93] [BHA97] [YU02] [JIA04] has been discussed a lot previously in this thesis, therefore, we do not repeat them here again.

[CHA02] proposed a programmable any-multicast message delivery service (PAMCast), which generates both the anycast packets and the multicast packets. The

PAMCast tries to deliver packets to m out of total n group members, where $1 \le m \le n$. This kind of service has potential applications in fault-tolerant repositories, parallel file downloading, and so on. The authors designed a shared tree for the PAMCast service management. The simulations of the paper show that the proposed service works well.

The combination of multicast and anycast service is natural, and the cooperation can provide new services for the Internet applications [JIA00] [JIA01]. The combining of anycast and multicast offers a bi-directional service for the Internet based distributed data processing systems: multicast takes the responsibility of data synchronization among the multicast group, and anycast takes the role for finding the "best" server in the anycast group; furthermore, anycast is a good methodology for server load balance and network load balance as well.

8.3 The Architecture of the Minicast Service

In this section, we propose a multicore architecture for the minicast service and the management of the multicore minicast tree. In order to make the description of the minicast service simple and clear, we do not involve the aspects of reliability of the networks in this chapter. We assume that the network for the minicast service is reliable and there is no local failure.

8.3.1 The MultiCore Minicast Architecture

The tree architecture is widely employed in network applications, such as source tree [MOY98], shared tree, and core based tree [BAL97] in multicast applications. The tree architecture is efficient in terms of performance and it is easy for management. In this chapter, we design our minicast architecture based on the tree data structure.

There are three metrics for measuring the performance of the minicast service, which is listed below:

1. Transmission-Delay (*TD*): the maximum number of links traversed by any packet in traveling from a source to a receiver.
2. Bandwidth-Consumption (*BC*): the total number of links used to deliver a packet from a source to all the receivers
3. Traffic-Concentration (*TC*): the number of packets transmitted across each link per unite time in the minicast tree when each source sends to all receivers.

In order to achieve a satisfactory level of the three metrics, especially the traffic concentration, we propose a multi-core architecture for the minicast service, which is shown as below,

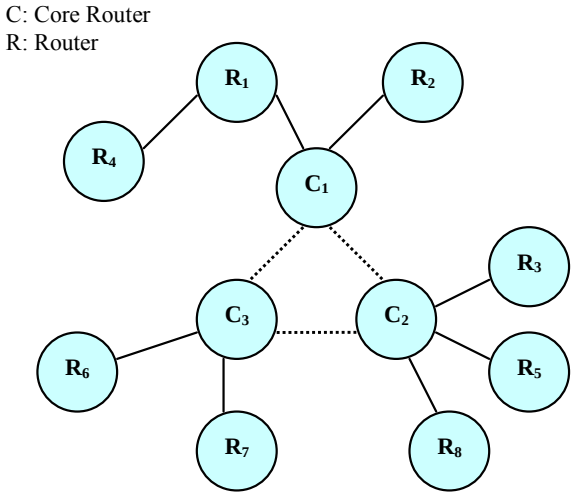

Figure 8.1. The Multicore Minicast Architecture

All the receivers of a minicast service are defined as members of a minicast group; all the minicast group members are connected by minicast routers, which are the routers in the network with the capability of providing minicast routing service. The minicast routers may be distributed anywhere in a network. We partition a network into a number of domains (say N domains) by organizations, regions or any other criteria. In each domain, we select one router from the minicast routers in that domain as a local core for the minicast service. Furthermore, we establish a local minicast tree which is rooted on the local core and the minicast members become the leaves of the local minicast tree. The local core holds all the information about the local minicast tree, such as number of members, number of hops to each member in the local tree. All the cores of a minicast group will exchange the local tree information when it is necessary. For the whole network, we have N cores for one minicast group. For performance reason, we organise the N cores of a minicast group into an anycast group. We address the minicast service with the anycast address, therefore, when a minicast client initiates a query to the minicast group, the query will be delivered to the "nearest" anycast member (one of the minicast core) by the anycast mechanism. Without loss of generality, we assume that a local minicast tree has k members; if $k \geq m$, where m denotes the minimum group members which have to receive the pack packets, we also call it as the parameter m of a minicast service. In order to make the packet to be delivered to at least m members and close to m members, the local core sets a suitable *TTL* (*Time To Live*) for the minicast packet, and multicasts the packet on the local minicast tree; if $k < m$, then the local core sets a suitable *TTL* for the minicast packet and multicasts the packet to the local tree (each member of the local tree will receive the packet), at the same time, it forwards the minicast query with parameter $m - k$ to

the nearest remote core. The remote minicast core continues the procedure until at least *m* minicast members will receive the packet.

Figure 8.2 presents a local minicast tree. All the members in the local domain are included in the local minicast tree, and there is a core for the members (the members of the local minicast tree are deep coloured in Figure 8.2). Note that there maybe a router in a local minicast tree but there is no minicast group member connecting to that router directly, such as router R_3 in Figure 8.2.

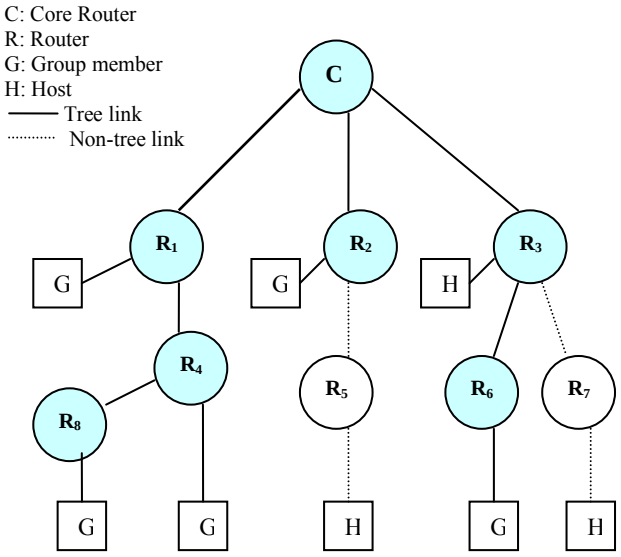

Figure 8.2. A Example of the Local Minicast Tree

The solid lines in Figure 8.2 connect all the minicast group members in the local domain in a core-based tree: node C is the core; every minicast group member attaches at least to one of the on-tree router. In this case, the local minicast tree has 5 members. The local minicast core knows the hops to each of the members on the local minicast tree respectively so that it can set the *TTL* parameter to send a message to at least *m* members of the minicast group. We define **minicast radius** in term of hops for delivering to at least *m* minicast members. For example, in Figure 8.2, we assume that

we will send a message to at least 3 minicast members, the local minicast core router sets the minicast radius as 3, then the packets will be discarded after 3 hops, and within that distance, the message is sent to 4 members on the minicast tree in the example. The minicast service meets the requirement of at least 3 members and close to 3 members as possible as it can.

8.3.2 The Management of the Multicore Minicast Tree

For the minicast tree management, we propose a Minicast Group Membership Protocol (MGMP), which is similar to the Internet Group Membership Protocol (IGMP) [FEN97] and the Core Based Tree (CBT) protocol [BAL97].

Tree creation. In order to join a minicast tree, a host must express its interest in joining a minicast group by broadcasting a request on the LAN, on receiving the request, the local router generates a JOIN_TREE packet and starts the joining procedure. The JOIN_TREE packet includes the information of the minicast group ID. If the local router is on the minicast tree, then the joining will be confirmed after a JOIN_ACK packet is received by the initial host. If the local router is not on the given minicast tree, then it tries to find the nearest router on the minicast tree, and delivers the JOIN_TREE packet to that on-tree router, and the router then forwards that packet to the local minicast core along the minicast tree, at the same time, a confirmation message, JOIN_ACK packet, is issued backward to the host.

Tree maintenance. To maintain the status of the minicast tree, the members send periodically a “keepalive” message to their upstream neighbors, and a “response”

message is fed back to the initial members respectively. If there is no response within a specific timeout, then it assumes that the upstream neighbor has become unreachable. If an upstream router is unavailable, the router sends a LEAVE_TREE packet upstream, and flushes all of its downstream branches by sending FLUSH_TREE packets, allowing them to rejoin individually if necessary. When a router exits from a minicast group, and there are no attached routers or receivers, the router submits a LEAVE_TREE packet to its upstream router in the minicast tree. The local cores of a minicast group exchange periodically the information of the number of each local members among them.

Data transmission. In terms of data transmission, the packets are delivered to the nearest local core. The local core sets a suitable *TTL* according to the minicast parameter *m*. If the local minicast tree does not have enough members to meet the parameter *m*, then the local core forwards the packets with a modified minicast parameter *m* to the nearest remote core, which has been presented at section 8.3.1.

8.4 Local Core Selection of the Minicast Tree

The quality of the minicast schema depends on two components: 1) How to find the best router to act as the local core in a domain; and 2) How to transport packets to the local core and from the local core to the rest of the members which are within the minicast radius. In this chapter, we bear the transmission delay in our mind for the protocol design and performance evaluation.

It is obvious that the local cores have a critical impact on the proposed model in terms of performance. Because of the dynamic characteristic of the Internet, such the congestion, link failure, furthermore, the ever changing properties of minicast groups on the Internet, such as number of members, locations, etc, that the best node acts as a local core of a domain in one period may not be the best candidate for the local core for the next period. In this section, we try to find an adaptive method to decide the local core for a given minicast domain.

The topology of networks (either the fixed networks or the wireless networks) can be modeled by an undirected graph $G = (V, E, W_x, W_e)$, where V is a set of the nodes, E is a set of the links among the nodes, W_x is the node weight function, and W_e is the edge weight function [9]. A instance of minicast routing issue involves a set of *receivers* R ($R \subseteq V$). We use the least delay measure and use the 1-median (median in short) problem method [13] to find the optimal local cores for the minicast service. In the median problem, a function H is defined as follows:

$$H(v) = \sum_{u \in V} W_x(u) d(v,u)$$

Where $d(v,u)$ is the cost of the shortest path from node u to v in graph G. The median of the graph is node v^* with minimum H value, namely,

$$H(v^*) = \min H(v), v \in V$$

In order to obtain the core of a minicast domain, we need to set the following preliminary

$$W_x(u) = \begin{cases} 1 & u \in R \\ 0 & u \notin R \end{cases}$$

Assume that a vertex *v* with degree *k* in a tree, $T = (V, E)$, when we remove the node v from the tree, then there exists a forest, $T - v$, with *k* subtrees, $T_{v,1}, T_{v,2}, ..., T_{v,k}$. Define

$$w(T_{v,i}) = \sum_{u \in T_{v,i}} W_x(u), i = 1,2,...,k$$

and

$$F(v) = \max_{1 \le i \le k} w(T_{v,i})$$

A vertex $v^* \in V$ is called the *centroid* of T if and only if $F(v^*) = \min_{v \in V} F(v)$. The research of [KAR97] has obtained the following conclusions:

Theorem 1. A vertex of a tree is a centroid if and only if it is a median of the tree.

Theorem 2. A vertex v of a tree $T = (V, E)$ is a centroid of T if and only if

$$F(v) \le \frac{1}{2} \sum_{u \in V} W_x(u)$$

Similar to [GUP03], we can obtain the core selection algorithm based on the previous mathematical analysis.

1. Select one node from the member in a given minicast domain randomly, and assume that the node is the local core, and build the local minicast tree base on the core.
2. The core calculates the sum of the weights of each subtree, and detects when it is no longer a centroid of the local minicast tree based on Theorem 2.
3. The current local core starts a migration towards the current centroid of the local minicast tree.

8.5 Performance Analysis

To analyze the performance, we conducted some simulations for the proposed Minicast service. The experiment environment is set up to have three local cores for a minicast group, and 17 non-core members (20 minicast members in total). If all the local minicast trees have the same or similar number of members, then we call the architecture a symmetric minicast tree; otherwise, we call it an asymmetric minicast tree.

A critical issue in the minicast service is the traffic concentration problem. We conducted traffic concentration simulations for a symmetric minicast tree and an asymmetric minicast tree, respectively. For each experiment, a minicast requirement is initiated randomly from the local minicast trees, and we count the times a local core

involved in the minicast packet delivery. We try to find the principle distribution of the traffic on the minicast tree with an increasing parameter *m*.

In the asymmetric minicast tree case, each of the three local minicast trees in our simulation (local tree 0, local tree 1, and local tree 2) has 10, 6, and 4 minicast members, respectively. The result is shown in Figure 8.3. Based on the analysis of the curves, we can conclude that in general, the packet distribution is even among the local minicast trees.

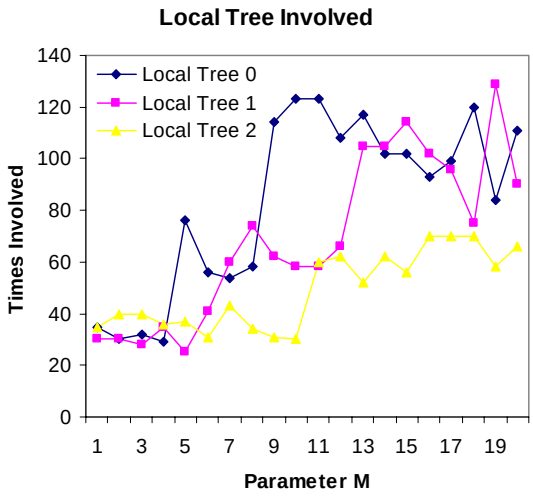

Figure 8.3. Packet Distribution in a Symmetric Minicast Tree

In order to check the traffic concentration on an asymmetric minicast tree, we arrange the three local trees with 7, 6, 7 minicast members, respectively. The result of the experiment is shown in Figure 8.4. It is obvious that the packet distribution is even among the three local minicast trees.

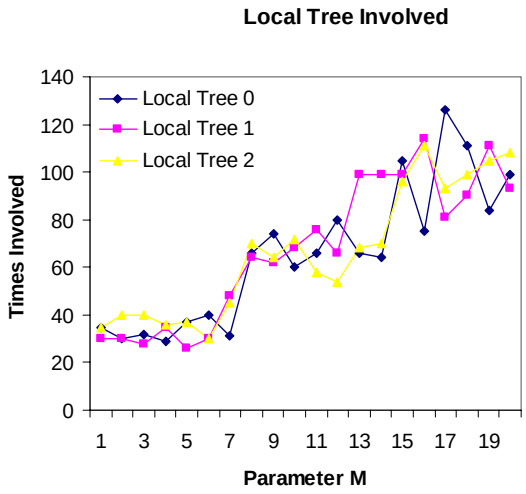

Figure 8.4. Message Distribution in an Asymmetric Minicast Tree

The previous two experiments have shown that the multicore minicast architecture has solved the traffic concentration problem, and makes an even packet distribution among the minicast members.

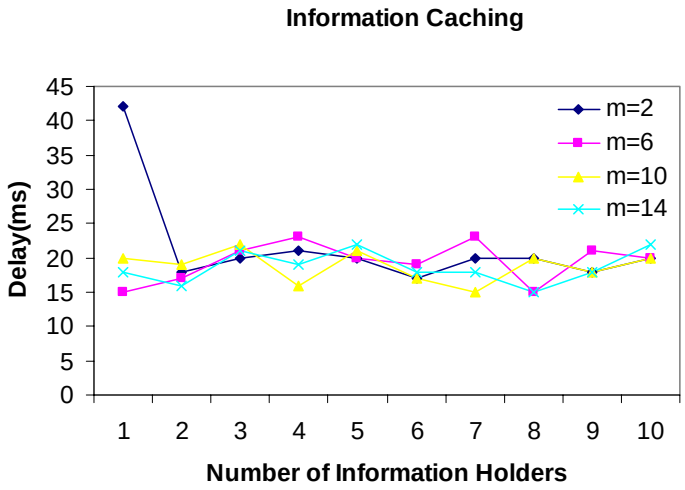

Figure 8.5. Minicast Application in Information Caching

The minicast model has potential and wide applications in information retrieval. Here we apply the minicast service in information caching applications as an example. The

simulation scenario is that there are 23 hosts in a minicast group, some of them, called information holders, hold the information in their caches, we do not know which hosts hold the information for a given query. Meanwhile, we want to save the network bandwidth consumption as much as we can, and we measure the network delay as a performance metric. We did the simulation on the asymmetric minicast tree scenario with different parameter *m* (2, 6, 10, and 4, respectively), and the result is shown in Figure 8.5.

The simulation shows that in most of the situations, the performance (the network delay) is not sensitive with the two parameters, parameter *m* of the minicast service and the number of the information holders. Therefore, in the practical applications we can configure a small *m* in order to save the network bandwidth consumption.

8.6 Summary

In this chapter, we proposed a new Internet service, minicast, in the scenario of *n* replicated servers, to deliver a message to at least *m* members, where $1 \le m \le n$. This kind of service has wide applications, such as parallel information retrieval, parallel information caching, parallel grid computing, parallel downloading, and so on. The essential advantage of this service is that the proposed model provides the same quality of service with less cost for Internet applications, such as reduce bandwidth consumption, reduce network delay for applications, etc.

The minicast service is based on two exist Internet protocols, anycast service and multicast services. We proposed the multicore minicast tree architecture for the minicast service. We organize the cores of a minicast tree into an anycast group. Therefore any minicast query will be delivered to the "nearest" core using the anycast mechanism. Then the minicast query will be delivered by the core through the multicast method, and this can make sure the query will be sent to at least *m* members and also close to *m* members.

Our simulations show that the multicore minicast architecture can handle the issue of traffic concentration very well. Furthermore, the simulation of information caching application shows that the minicast mechanism can reduce the network resource consumption while provide the same quality of service.

There are some further issues that need to be explored, such as the relationship between the performance and the number of local cores for a minicast group; Fault-tolerance issue in the minicast services; the performance and bandwidth consumption comparison between the minicast service and the related services, and so on.

Chapter 9
Conclusions and Future Work

This chapter outlines the contributions of this thesis, the importance of the research, and addresses what can be improved in the future.

9.1 The Main Contributions

The anycast research in IP next generation started in 1993. Researchers in the area have obtained some prestigious results, however, there are still many challenging issues in anycast to be explored. We carried out research on some of the challenges, and the main contributions of this thesis can be outlined as follows:

● Proposed an efficient application layer anycast routing algorithm. The network layer anycast can not be completely implemented in short period as we discussed in chapter 3. Fortunately, the application layer anycast routing is an alternative for current anycast applications and it is also a practical choice in the near future. We proposed the Requirement-based Probing Algorithm for application layer anycast routing. The new algorithm needs less control messages comparing with the existing Periodical Probing Algorithms. Through mathematical analysis, we proved that the new algorithm has a better performance compared to the existing algorithms.

The proposed algorithm can be implemented easily on the current Internet, and it can carry out its duty to serve for anycast applications immediately. Compared with the network layer anycast, the proposed algorithm is more flexible for users, as the result, the related users can custom their applications easily and quickly. This feature will prompt more relevant applications dramatically, such as, Internet meeting, Video on Demand on the Internet, Web-based database applications, and so on. Moreover, this proposal started the exploration of the work of on-demand anycast routing service.

● Proposed a twin-server model for fault-tolerance of anycast systems. The Internet platform is not reliable; link failures and server failures occur from time to time. Therefore, anycast, as an Internet based service, must deal with the fault-tolerance issue to offer users reliable and continuous services. We introduced the twin-server model into the anycast systems. According to the model, each anycast server is assigned a twin server, which is another server in the anycast group. Once a server failure occurs, the failure node's twin server will transparently take over the duties of the original server.

Without a fault-tolerance mechanism, the quality of anycast service will degrade. The proposed model can handle the problem with the characteristics of the mirrored severs in an anycast group. Our proposed algorithms make the job transfer smoothly, and all the actions are transparent to the users.

● Researched on the load balance problem of anycast systems. The load balance property of the current anycast is based on network status, it can not exactly represent the real situations, because it excludes the information of server workload. In order to

improve the global load balance capability of anycast service, we designed the job deviation algorithms for the anycast service. Both the mathematical analysis and the simulation show that the proposed algorithms work well.

The proposed job deviation algorithms try to guarantee that all the requests of an anycast service enjoy a reasonable certain level quality of service, namely, the service performance of an anycast group is similar to all the users on the Internet. This work prohibits the possibility of service discrimination: some clients enjoy a much better service performance than others, which may cause by locations, number of concurrent requests in a domain, network bandwidth, hardware performance, and so on.

- Proposed a reliable anycast routing algorithm in ad hoc networks. Currently there is no practical or reliable anycast routing algorithm for ad hoc networks to the best of our knowledge. We proposed the mesh-based anycast routing algorithm in ad hoc networks. Compared with the broadcast algorithms, it uses less control packets for the routing information maintenance; it has the capability of local link failure recovery.

Reliability is a critical issue of mobile computing, especially in the ad hoc networks. We introduced the mesh method for anycast routing in ad hoc networks. The mesh based anycast routing protocol improves the reliability of anycast service in the mobile ad hoc environment; furthermore, our on-mesh route discovery algorithm prevents the unnecessary traffic of control packets.

- Presented a novel middleware for the Web-based distributed databases. The combination of the anycast service and the multicast service can provide a wonderful

two direction service; therefore, we applied the two services into the Web-based distributed database applications. We integrated the anycast service and the multicast service into a middleware, the castway, which is independent from the heterogeneous Internet environment. The anycast service offers the "best" server from a group of mirrored servers, while the multicast service takes the responsibility of data synchronization among the mirrored database servers. The simulation shows that the proposed mechanism performs well.

This work demonstrates an example of integrating the existing Internet services to serve for the existing applications. The integration requires limited modifications to the original three-layer Web based database architecture. As a result of taking the advantages of the anycast and multicast protocols, the middleware improves the database performance in information retrieval and data synchronization. The philosophy of this work can be introduced to promote the diversity of the Internet applications.

- Proposed a new Internet service – Minicast. As a new Internet service, it tries to provide the same quality of service with less resource consumption. The minicast service submits a minicast packet to at least m members out of total n members ($1 \le m \le n$). A multi-core architecture is designed for the minicast service to deal with the traffic concentration issue. Besides, a local core selection algorithm is also presented.

The minicast service has prospective applications, such as in information retrieval, cache, parallel downloading, etc. The minicast service is especially valuable for the applications in the scare resource environment, such as the ad hoc networks.

9.2 The Importance of This Thesis

There are many publications on anycast since the idea appeared in 1993, but most of them focus on the routing algorithms, architectures and applications. To the best of our knowledge, not a book or a thesis purely on anycast research has been published yet, and this thesis is an exploration on that target. The importance of this thesis is summarized as follows:

● the thesis addresses several critical issues of anycast research. We categorize key research on anycast into two parts: research of anycast itself, such as routing algorithms, load balance property, architectures, and so on; and the anycast applications, such as anycast in Web-based distributed databases, anycast in multimedia applications, anycast in information retrieval, and so on. The thesis addresses four important issues of anycast research: efficient routing algorithms, reliability issues, load balance aspects, and anycast in Web-based distributed database applications.

● the thesis initiates three interesting topics in anycast research. The thesis starts the research on reliable and fault-tolerant topic on anycast itself for the first time; the thesis discovers the limitations of load balance property of anycast, especially for the network

layer anycast, and fixes the problem using job deviation methods; the thesis firstly introduces the mesh method into anycast routing in ad hoc networks to obtain reliable routing service.

- the thesis proposed a new Internet service – minicast. The minicast service has promising applications in the Internet, such as, large global information retrieval, parallel downloading, and so on. The minicast service targets on reducing the usage of network resource, and this is a very valuable characteristics for the scare resource environments, such as wireless Internet, ad hot networks, etc.

9.3 Future Work

The anycast related research is widely open, but we only touched several aspects in this thesis, and there are a number of topics to be explored. We list the possible improvements of the research that we have done, and the interesting aspects in the future to explore.

1. We have explored several aspects of anycast, but we did not go very deep in some of the issues because of time limitation. We will continue a deep exploration in the topics. For example, we will implement a prototype model for the requirement-based probing anycast routing algorithm, and apply it to the real Internet applications; the proposed middleware, the castway, for the Web database model is worth to be implemented and to be evaluated in the real distributed Web based database

applications; the mesh based anycast routing algorithm in ad hoc networks, which possesses a promising research and application perspective, has a good start already, the mathematical analysis and performance evaluation will be presented in the near papers.

2. Anycast service upgrade from IP version 4 to IP next generation. The anycast service is used in some places on the IPv6 networks, and it definitely will be popular in the near future, therefore there will exist a long transition period. How to make the transition smoothly is a practical issue for researchers and industry practitioners.

3. Anycast applications. The anycast service is a great idea, but it needs applications, otherwise the proposal is useless. We will try our best to apply anycast service into diverse applications, such as information retrieval, service discovery in the Internet or wireless networks, multimedia applications, and so on.

4. Propose new Internet services based on the anycast protocol. It may be interesting to combine the anycast service with the other Internet service to propose new efficient and more reliable Internet services.

Bibliography

[ACC83] Michael Accetta, RFC887: Resource Location Protocol, December 1983.

[AGG99] G. Aggelou and R. Tafazolli, "RDMAR: A Bandwidth-Efficient Routing Protocol for Mobile Ad Hoc Networks," Proc. of the WoWMoM'99, Seattle, Washington, Aug. 1999.

[ALM85] Guy T. Almes, Andrew P. Black, Edwards D. Lazowska, and Jerre D. Noe, "The Eden System: A Technical Review," IEEE Transactions on Software Engineering, SE-11(1): 43-59, January 1985.

[AZA99] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli, Upfal, "Balanced Allocations," SIAM J. COMPUT. Vol. 29, No.1, pp180-200, 1999.

[BAL97] A. Ballardie, "Core Based Trees(CBT CBT Version 2) Multicast Routing Architecture," RFC2201&RFC2189, September 1997.

[BAS97] Erol Basturk, Robert Engel, Robert Haas, Vinod Peris, and Debanjan Saha, "Using Network Layer Anycast for Load Distribution in the Internet," IBM Technology Report, 1997.

[BAS03] P. Basu, W. Ke and T. Little, "Dynamic Task-Based Anycasting in Mobile Ad Hoc Networks," Mobile Networks and Applications 9, pp. 593-612, 2003.

[BHA96] Samrat Bhattacharjee, Mostafa H. Ammar, Ellen W. Zegura, Viren Shah, and Zongming Fei, "Application-Layer Anycasting," Technique Report of Georgia Institute of Technology, GIT-CC-96/25, 1996.

[BHA97] Samrat Bhattacharjee, Mostafa H. Ammar, Ellen W. Zegura, Viren Shah, and Zongming Fei, "Application-Layer Anycasting," IEEE INFOCOM'97, Kebe, Japan, April 1997.

[BIR95] Kenneth P. Birman and Bradford B. Glade, "Reliability Through Consistency," IEEE Software, pp 29-41, May 1995.

[BOG82] David R. Boggs, "Internet Broadcasting", Ph. D. thesis, January 1982.

[BOU96] J. Bound and P. Roque, "IPv6 Anycasting Service: Minimum requirements for end nodes,"draft-bound-anycast-oo.txt, August 1996.

[BRA94] Hans-Werner Brau and Kimberly C. Claffy. "An Experimental Means of Providing Geographically Oriented Responses Relative to the Source of Domain Name Server Queries," Technical Report, San Diego Supercomputing Centre, 1994.

[BRO98] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," Proc. of the 4th Annual ACM/IEEE Inter. Conf. on Mobile Computing and Networking, ACM, Dallas, TX, Oct. 1998.

[CAC89] R. Caceres, "Measurements of wide-area Internet Traffic," Tech. Report. UCB/CSD 89/550, Computer science Department, University of California, Berkeley, 1989.

[CAL98] K. L. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz, "Directions in Active Networks," IEEE Communications Magazine, October 1998.

[CAL00] K. L. Calvert, J. Griffioen, B. Mulllins, A. Sehgal, and S. Wen, " Concast: Design and Implementation of an Active Network Service," IEEE Transactions on Networking, Vol. XX, No. Y, Month 2000.

[CAO01] Jin Cao,William S.Cleveland, Dong Lin, and Don X. Sun, "On the Nonstationarity of Internet Traffic," Proc. ACM Sigmetrics `01, 102-112, 2001.

[CHA96] T. D. Chandra and S. Toueg, "Unreliable Failure Detectors for Reliable Distributed Systems." Journal of the ACM, 43(2): 225-267, March 1996.

[CHA02] Youngsu Chae, Ellen W. Zegura, and Haris Delalic, "PAMcast: Programmable Any-Multicast for Scalable Message Delivery" in Proceedings of IEEE OpenArch 2002.

[CHE98] Shigang Chen, Klara Nahrstedt, An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions, IEEE Network, Special Issues on Transmission and Distribution on Digital Video, Nov./Dec. 1998.

[CHE00] Wei Chen, Sam Toueg, and Marcos K. Aguilera, "On the Quality of Service of Failure Detectors," Proceeding of International Conference on Dependable Systems and Network, New York, USA, June 25-28, 2000.

[CHE01] Minghua Chen, and Wei Mao, "Anycast By DNS Over Pure IPv6 Network," Project Report, University of California, Berkeley. 2001.

[CHE02] S. Chessa and P. Santi, "Crash Faults Identification in Wireless Sensor Networks," Computer Communications, Vol. 25(14), pp. 1273-1282, 2002.

[CHI97] C. E. Chiang, "Routing in Clusterhead Multihop, Mobile Wireless Networks with Fading Channel," Proceedings of IEEE SICON'97, April 1997, pp.197-211.

[CHO04] R. R. Choudhury and N. H. Vaidya, "MAC-Layer Anycasting in Wireless Ad Hoc Networks,"ACM SIGCOMM Computer Communications Review, Volume 34, pp.75-80, Number1: January 2004.

[COR95] M. S. Corson and A. Ephremids, "A Distributed Routing Algorithm for Mobile Wireless Networks," ACM/Baltzer Wireless Networks J., vol. 1, no. 1, Feb. 1995, pp.61-81.

[COR99] S. Corson and J. Macker, "Mobile Ad Hoc Networking (MANET) : Routing Protocol Performance Issues and Evaluation Considerations," RFC 2501, IETF, Jan. 1999.

[DAT03] A. Datta, "Fault-tolerant and Energy-efficient Permutation Routing Protocol for Wireless Networks," Proceedings of the Internet Parallel and Distributed Processing Symposuim, 2003.

[DEE98] S. Deering and R. Hinden, "Internet Protocol Version 6(Ipv6) Specification," RFC 2460, Dec. 1998.

[DRI02] Eleni Drinea, Alan Frieze, and Michael Mitzenmacher, "Balls and Bins Models with Feedback," Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp 308-315, 2002.

[DUB97] R. Dube et al., "Signal Stability Based Adaptive Routing for Ad Hoc Mobile Networks," IEEE Personal Communications, Feb. 1997, pp36-45.

[ENG98] R. Engel, V. Peris and D.Saha, "Using IP Anycast for Load Distribution and Server Location," Proc. of IEEE Globecom Global Internet Mini Conference, Nov. 1998.

[FEI98] Zongming Fei, Samrat Bhattacharjee, Ellen W. Zegura and Mostafa Ammar, "A Novel Server Selection Technique for Improving the Response Time of a Replicated Server," INFOCOM'98.

[FEN97] W. Fenner, "Internet Group Management Protocol, Version 2," RFC1112, November 1997.

[FRI03] R. Friedman, L. Baram and S. Abarbane, "Fault-Tolerant Multi-Server Video-on-Demand Service," Proceedings of the International Parallel and Distributed Processing Symposium, 2003.

[GAF81] E. Gafni and D. Bertsekas, "Distributed algorithms for generating loop-free routes in networks with frequently changing topology," IEEE Transactions on Communications, vol. C29, no. 1, 1981.

[GAR99] J. J. Garcia-Luna-Aceves, and E. L. Madruga, "The Core-Assisted Mesh Protocol," IEEE Journal on Selected Area in Communications, Special Issue on Ad Hoc Networks, Vol. 17, No. 8, August 1999.

[GUE97] Rachid Guerraoui and Andre Schiper, "Principles of Transaction-Oriented Database Recovery," IEEE Computer, pp 68-74, April 1997.

[GUL01] Vivek Gulati, Aman Garg and Nitin Vaidya, "Anycast in Mobile Ad Hoc Networks," Technical Report, Texas A&M University, April 2001.

[GUP03] S.K.S. Gupta, P.K. Srimani, "Adaptive Core Selection and Migration Method for Multicast Routing in Mobile Ad Hoc Networks," IEEE Transactions on Parallel and Distributed Systems, pp27-38, Vol. 14, NO. 1, January 2003.

[GUS97] Eva Gustafsson, Gunnar Karlsson, "A Literature Survey on Traffic Dispersion," IEEE Network, March/April 1997.

[GUY94] James D. Guyton and Michael F. Schwartz, "Experiences with a Survey Tool for Discovering Network Time Protocol Servers", Proceedings of SIGCOMM'94.

[GUY95] James D. Guyton and Michael F. Schwartz, "Locating Nearby Copies of Replicated Internet Servers", Proceedings of SIGCOMM'95.

[GWE94] James Gwertzman and Margo Seltzer, "The Case for Geographical Push-Caching," Technical report, Harvard University, 1994.

[HAN01] K. M. Hanna, N. Natarajan, and B.N. Levine, "Evaluation of a Novel Two-Step Server Selection Metric," Proceeding IEEE ICNP 2001. November 2001.

[HIN95] R. Hinden and S. Deering, "IP Version 6 Addressing Architecture," RFC 1884, Dec. 1995.

[HOT94] S. M. Hotz, "Routing Information Organization to Support Scalable Interdomain Routing with Heterogeneous Path Requirements," Ph. D thesis, 1994.

[HUI96] Chi-Chung Hui, and Samuel T. Chanson, "A hydro-dynamic approach to heterogeneous dynamic load balance in a network of computers," Proceedings of the 1996 International Conference on Parallel Processing, pp. III-140-147, 1996.

[HUI97] Chi-Chung Hui, and Samuel T. Chanson, "Efficient load balancing in interconnected LANs using group communication," Proceedings of the 17th International Conference on Distributed Computing Systems, pp.141-148, 1997.

[JIA96] Weijia Jia, Jorg Kaiser, and Edgar Nett, "RMP: Fault-Tolerant Group Communication," *IEEE Micro*, Vol. 16, No. 2, April 1996, pp. 59-67.

[JIA00] W. Jia, G. Xu, and W. Zhao, "Integrated Fault-tolerant Multicast and Anycast Routing Algorithms," IEE Proc.-Comput. Digit. Tech. Vol. 147, No. 4, July 2000.

[JIA01] W. Jia, W. Zhou, and J. Kaiser, "Efficient Algorithm for Mobile Multicast Using Anycast Group," IEE Proc.-Commun., Vol. 148, No. 1, February 2001.

[JIA02a] Weijia Jia, P-O. Au, G. Xu, and W. Zhao, "Scalable Multicast Routing Protocol using Anycast and Hierarchical-Trees," The 27th Annual IEEE Conference on Local Computer Networks (LCN), November 6-8, 2002, Tampa, Florida, USA, pp. 572--581.

[JIA02b] Weijia Jia, G. Xu and W. Zhao, "Efficient Internet Multicast Routing Using Anycast Path Selection", *Journal of Network and Systems Management*, 12(4), Dec. 2002, pp.417-438.

[JIA04] Weijia Jia, D Xuan, W. Tu, L. Lin and W. Zhao, "Distributed Admission Control for Anycast Flows", *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, VOL. 15, NO. 6, JUNE 2004.

[JOH96] D. B. Johnson and D. A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks," Mobile Computing, T. Imielinski and H. Korth, Eds., Kluwer, 1996, pp. 153-181.

[JOH99] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, "Scenario-Based Performance Analysis of Routing Protocols for Mobile Ad-Hoc Networks," MobiCom'99, Aug. 1999.

[JOS97] Bharat S. Joshi, Seyed Hosseini, and K. Vairavan, "On a load balancing algorithm based on edge coloring," Proceedings of the 29th Southeastern Symposium on System Theory, pp174-178, 1997.

[KAR79] O. Kariv and S.L. Hakimi, "An Algorithmic Approach to Network Location Problem. ii: the P-Medians," *Proc. SIAM J. Applied Mathematics, vol. 37, no.3*, pp539-560, Dec. 1979.

[KAT00] Dina Katabi, and John Wroclawski, "A Framework for Scalable Global IP-Anycast (GIA)," SIGCOMM'00, Stockholm, Sweden, 2000.

[KEM98] Bettina Kemme, Gustavo Alonso, "A Suite of Database Replication Protocols based on Group Communication Primitives," The 18th International Conference on Distributed Computing Systems, P156-164. May 1998, Amsterdam, The Netherlands.

[KO00] Young-Bae Ko and Nitin H. Vaidya, "Anycasting and Geocasting in Mobile Ad Hoc Networks," Technical report, Texas A&M University, June, 2000.

[KRI93] R. Krishnan and J. A. Silvester, "Choice of Allocation Granularity in Multipath Source Routing Schemes," Proceeding IEEE INFOCOM'93, March 1993.

[KRI94a] R. Krishnan and J. A. Silvester, "Resource Allocation in Broadband Networks – Cell, Burst or Connection Level," Proceeding ICC'94, May 1994.

[KRI94b] R. Krishnan and J. A. Silvester, "The Effect of Multipath Routing on the Loss Performance of Multiplexed ON-OFF Sources," Proceeding ITC-14, June 1994.

[LEE99] S. Lee, W. Su, and M. Gerla, "Ad Hoc Wireless Multicast with Mobility Prediction," IEEE ICCCN'99, Boston, MA, Oct. 1999.

[LEE00] S. Lee and C. Kim, "Neighbor Supporting Ad Hoc Multicast Routing Protocol," IEEE VTC'2000, August 2000.

[LIU02] Jing Liu, Hung Chun Kit, Mounir Hamdi, and Chi Ying Tsui, "Stable Round-Robin Scheduling Algorithms for High-Performance Input Queued Switches," Proceedings of the 10th Symposium on High Performance Interconnects Hot Interconnects, 2002.

[MA97] Qingming Ma and Peter Steenkiste, *On Path Selection for Traffic with Bandwidth Guarantees*, Fifth IEEE International Conference on Network Protocols, Atlanta, IEEE, October 1997.

[MAR01] Tamas Marostis, Sandor Molnar, and Janos Sztrik, "CAC Algorithm Based on Advantage Round Robin Method for Qos Networks," Proceedings of the Sixth IEEE Symposium on Computers and Communications, 2001.

[MIT96] Michael Mitzenmacher, "Load balancing and dependent jump markvo processes," Proceedings of the 37th Annual Symposium on Foundations of Computer Science, pp213-222, 1996.

[MIT97] Michael Mitzenmacher, "The Power of Two Choices in Randomized Load Balancing," Ph.D thesis, 1997.

[MIU01] Hirokazu Miura and Miki Yamamoto, "Server Selection Policy in Active Anycast," IEICE Trans. Commun., Vol. E84.B, No. 10 October 2001.

[MOY98] J.Moy, OSPF: Anatomy of an Internet Routing Protocol, Addison-Wesley, Reading MA, 1998.

[MUR96] S. Murthy and J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks, Oct. 1996, pp.183-197.

[MYE99] Andy Myers, Peter Dinda, and Hui Zhang, "Performance Characteristics of Mirror Servers On the Internet," Proceedings of INFOCOM '99, 1999.

[NAV97] J. C. Navas and T. Imielinski, "Geocast – geographic addressing and routing," in proceeding of ACM/IEEE Intel. Conference on Mobile Computing and Networking (MOBICOM), 1997.

[NIC00] Matthias Nicola and Matthias Jarke, "Performance Modelling of Distributed and Replicated Databases," IEEE Transaction on Knowledge and Data Engineering, 12(4):645-672, July/August 2000.

[NS02] The Network Simulator 2 Manual, http://www.isi.edu/nsnam/ns/, April, 2002.

[OE00] Masafumi OE, and Suguru Yamaguchi, "Implementation and Evaluation of IPv6 Anycast," the proceeding of the 10th Annual Internet Society Conference. Yokoham, Japan, July 2000.

[PAR93] C. Partridge, T. Mendez, and W. Milliken, " Host Anycasting Service," RFC 1546, November 1993.

[PAR97] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in Proceeding of IEEE INFOCOM'97, 1997.

[PAR99a] V. D. Park and J. P. Macker, "Anycast Routing for Mobile Services," Proc. Conference on Information Sciences and Systems' 99, January 1999.

[PAR99b] V. D. Park and J. P. Macker, "Anycast Routing for Mobile Networking," Proc. IEEE MILCOM 99, Nov. 1999.

[PAX97] Vern Paxson, "Measurements and Analysis of End-to-End Internet Dybamics," Ph.D. thesis, University of California Berkeley, 1997.

[PAX99] Vern Paxson, "End-to-End Internet Packet Dynamics," IEEE/ACM Transactions on Networking, Vol.7, No.3, pp. 277-292, June 1999.

[PER94] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," Computer Communication Review, October, 1994, pp. 234-244.

[PER99] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," Proc. 2nd IEEE Workshop on Mobile Comp. Sys. and Apps., Feb. 1999, pp.90-100.

[REN00] Robbert V. Renesse, "Scalable and Secure Resource Location," the Proceedings of the Hawaii International Conference on System Sciences, Maui, Hawaii, January 2000.

[ROC00] Roca Vincent,Casta Luis,Vida Rolland, Dracinschi Anca and Fdida Serge, "A Survey of Multicast Technologies," Technical Report, http://www-rp.lip6.fr September 2000.

[ROU01] J. Rough and A. Goscinski, "A Group Communications Facility for Reliable Computing on Cluster," Proceedings of the ISCA 14th International Conference of Parallel and Distributed Computing Systems, Texas, USA, pp 19-24, August , 2001.

[ROY99] Elizabeth M. Royer, Chai-Keong Toh, "A review of current Routing Protocols for Ad Hoc Mobile Wireless Networks," IEEE Personal Communications, April 1999.

[SOL96] F. Solensky, "IPv4 Address Lifetime Expectations," IPng: Internet Protocol Next Generation, Addison-Wesley, Reading, MA, 1996.

[SOL01] Ricard V. Sole, and Sergi Valverde, "Information Transfer and Phase Transitions in a Model of Internet Traffic," Ricard V. Sole and Sergi Valverde, Physica A 289 595-605, 2001.

[TAN03] K. H. Tan, M. Yaacob, T. C. Ling, and K. K. Phang, "IPv6 single-path and multi-path anycast routing protocols performance analysis," Proceedings of the 1st international symposium on Information and communication technologies, September, 2003.

[TEN97] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall and G. Minden, "A Survey of Active Network Research," IEEE Communications Magazine, Vol.35, No.1, January 1997.

[THA97] David Thaler and C. V. Ravishankar, "Distributed Center Location Algorithms", *IEEE Journal on Selected Areas in Communications,* Vol. 15(3), pp. 291-303, April 1997.

[TOH96] C-K. Toh, "A Novel Distributed Routing Protocol to Support Ad-Hoc Mobile Computing," Proc. IEEE 15th Annual Int'l. Phoenix Conf. Comp. and Commun., Mar. 1996, pp.480-486.

[TSE02] Y.C. Tseng, S.Y. Ni, Y.S. Chen, and J.P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," Wireless Network, vol. 8, no. 2/3, 2002.

[WAN96] Zheng Wang and Jon Crowcroft, "Quality-of-Service Routing for Supporting Multimedia Applications," IEEE JSAC 14(7): 1288-1234, September 1996.

[WAN00] Jie Wang and Yonatan Levy, "Managing Performance Using Weighted Round-Robin," Proceedings of the Fifth IEEE Symposium on Computers & Communications, 2000.

[WIL02] B. Williams and T. Camp, "Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks," Proc. of the 3th ACM International Symposium on Mobile Ad Hoc Network and Computing, 2002.

[WU99] Wu, Z.D., C. Noble and D. Huang, "Optimal Video Distribution Using Anycasting Service", Proc. of the Internet Global Summit (INET'99) CD-ROM by Internet Society, San Jose, USA, June 1999.

[WU03] Jie Wu and Fei Dai, "Broadcasting in Ad Hoc Networks Based on Self-Pruning," IEEE INFOCOM 2003, 2003.

[XER86] Xerox Corporation. Network Binding Protocol. Technical report, June 1986.

[XUA00] Dong Xuan, Weijia Jia, Wei Zhao, and Hongwen Zhu, "A Routing Protocol for Anycast Message," IEEE Transaction on Parallel and Distributed System, VOL. 11, NO. 6, June 2000.

[XUA01] Dong Xuan, and Weijia Jia, "Distributed Admission Control for Anycast Flows with QoS Requirements," IEEE International Conference on Distributed Computing Systems, 2001.

[YAM01] Miki Yamamoto, Hirokazu Miura, Kenji Nishimura, and Hiromasa Ikeda, "A Network-Supported Server Load Balancing Method: Active Anycast," IEICE Trans. Commun., Vol. E84.B, No. 6, June 2001.

[YOO00] Jaehee Yoon, Azer Bestavros, and Ibrahim Matta, "SomeCast: A Paradigm for Real-Time Adaptive Reliable Multicast", *in the IEEE Real-Time Technology and Applications Symposium (RTAS) 2000*, Washington D.C., May 31-June 2000.

[YU02] Shui Yu, Wanlei Zhou, Fuchun Huang, and Mingjun Lan, "An Efficient Algorithm for Application-Layer Anycasting", The Fourth International Conference on Distributed Communities on Web (DCW2002), Sydney, April 2002.

[ZHA02] Li Zhang, "Fault-Tolerant Meshes with Small Degree," IEEE Transactions on Computers, Vol. 51, NO. 5, May 2002.

[ZHOU97] Wanlei Zhou, Andrzej Goscinski, "Fault-Tolerant Servers for RHODOS System," Journal of Systems and Software, Elsevier Science Publishing Co., Inc., New York, USA, Vol. 37, No. 3, pp 201-214, June, 1997.