AUTONOMOUS DECENTRALIZED COMMUNITY COMMUNICATION TECHNOLOGY FOR ASSURING INFORMATION DISSEMINATION

A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY OF SCIENCE
(COMPUTER SCIENCE)

Supervised by

Professor Kinji Mori

GRADUATE SCHOOL OF INFORMATION SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE
TOKYO INSTITUTE OF TECHNOLOGY

Preseneted by

Khaled Ragab Abd Eltawab Sayed Ahmed

September 2004

ABSTRACT

The enormous growth and the dynamism of the Internet initiated various new trends that reflects the need for powerful communication methods than the simpler client/server and Peer-to-Peer architecture. Despite their great potential, these systems still lack efficient data dissemination mechanisms. They deliver the information considering the users' demands regardless of their situations. There is no discernment between differences in place and time; users in any situation receive the same contents. However, situation and context-aware dissemination-oriented cooperative services motivate an increasing interest for evolving both the social and economic environments. Therefore, this thesis proposes the following community communication architecture and two community technologies to assure information dissemination by realizing the *Timeliness*, *Scalable online-expansion* and *Fault-tolerance* requirements in the large-scale and dynamic environment. The dissertation proposes an *Autonomous Decentralized Community Communication System* (***ADCCS***) and illustrates the concept, system architecture and technology of the ADCCS that permits to efficiently disseminate data according to the current situations of the system. Considered changing situations are changes of the community members' demands and situations (location, time), and the status of community nodes and logical links. The leading concept of autonomous community communication is the autonomy of the community nodes in recognizing members from non members, organizing the *Community Overlay Network* (***CON***) and achieving an efficient community communication based on local data, so that self-organized and self-adaptable ADCCS can be procured. Two techniques are proposed to satisfy the requirements mentioned before. First, *Service-Oriented multilateral community communication* technology is proposed to permit an autonomous determination of the neighbor to disseminate the community information. Thus, community information is disseminated only to its members and network congestion due to replicated messages is avoided. Second, an *autonomous decentralized community overlay network construction* technology is

proposed to construct CON under both homogenous and heterogeneous node-node latency assumptions. It constructs CON as single community so-called *Flat-CON* under homogenous node-node latency assumption. Flat-CON composed of disjoint Hamilton cycles (HC). HC forms Flat-CON with short network diameter and then achieves *timeliness*. HC allows nodes to autonomously join/leave CON with only local changes and then achieves *online-expansion*. HC maintains Flat-CON connected and non-partition able and then *fault-tolerance* is achieved. However under heterogeneous node-node latency assumption Flat-CON cannot satisfy *timeliness*. Thus, this thesis organizes CON into a multilayer of Sub-communities (Flat-CON) so-called *Multilayer-CON*. It is aware by the node-node latency to optimize the community communication delay and then satisfy *timeliness*. Moreover, it dynamically adapts to the network changes and reconstructs itself. Simulations on large-scale environment confirm that ADCCS achieves the requirements in rapid and realistic Internet settings and give evidence of the existence of the tradeoff relation between timeliness and fault-tolerance. Thus the proposed ADCCS achieves the requirements to establish a High Assurance information dissemination system to the right users, at right time and location under the changing of users' preferences and situations.

ACKNOWLEDGMENTS

My foremost thank goes to my thesis advisor Prof. Dr. ***Kinji Mori***. Without him, this dissertation would not have been possible. I thank him for his patience and encouragement that carried me on through difficult times, and for his insights and suggestions that helped me to shape my research skills. I believe that my personal life in *Japan* would not have been possible without his support.

I would like to express my thanks to my thesis committee members: Professor *E. Fujiwara*, Professor *M. Saeki*, Professor *Y. Sakai* and Professor *H. Yokota*. Their valuable feedback helped me to improve the dissertation.

I thank all the former and current members in Mori laboratory for their useful support along the course of this thesis. I thank *Moriyama* for his support to me and other students as well. Moreover, my thanks to *S. Sudo*, *N. Oohara*, *Y. Ishikawa*, *Y. Kido* and *Y. Mitsuhashi* for their kind help. I also thank the former and current members of *Community* research group: *T. Ono, N. Kaji, K. Anwar, Y. Horikoshi, H. Kuriyama, Y. Sugiyama and T. Masuishi*. I enjoyed all vivid discussions we had on various topics and had lot of fun being a member of this fantastic group. Moreover, my thanks have to go to other students Lu, Carlos and Ivan for helping me along the way of writing and proof reading of my thesis.

I am beholden to *Ministry of Education, Science, Sports, and Culture Government* of Japan for awarding me financial support for carrying out this research.

The last but not least, I thank my beloved parents, my brother, my sister and my wife's family for encouraging me and taking care of me and my family from a great distance. Special thank to my wife and kids for always being there when I need them most, and for supporting me and their patience through all these years.

Khaled Ragab

Tokyo Institute of Technology in Tokyo, Japan.
September 2004

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

In the name of ALLAH, the Most Gracious, the Most Merciful.

Oh my LORD *increase me in knowledge.* [Quran: 20-144.]

To my dear

Respective parents, beloved wife and kids.

1 INTRODUCTION

1.1 Research Field: Large-Scale information-dissemination Systems

With the extensive advances in communication technologies and network computing, the Internet has become essential for information exchange around the world. The enormous growth of the Internet is due to two reasons. First, the expected number of worldwide Internet and mobile users by the end of 2005 is more than one billion [1]. Second, the world publishes online about 300 terabytes of information every year [2]. This extreme proliferation, blended with dynamic users' and service providers' (SP) requirements, is pushing the development of new forms of e-social and e-business such as *dissemination-oriented* applications that should efficiently deliver contents to their users. The development of the communication technologies, and the availability of asymmetric, high-bandwidth links to the home, have sustained the development of a wide range of new *dissemination-oriented* applications. The information-dissemination systems are characterized by tremendous scale of the number of information flows must be delivered to users having high degree overlap not only of their demands, but also situations in terms of time and location. Cheriton coined dissemination-oriented communication systems [3]. These systems involve the delivery of data from one or more sources to a large set of receivers. Deering [4, 5] proposed the IP Multicast architecture that efficiently performs group data dissemination. However, more than a decade after the initial proposal, deployment of IP Multicast has been limited and sparse due to a variety of technical and non-technical reasons such as the overhead and complexities at the routers in the underlying network layer are increased [6]. As a result, most of the dissemination-oriented applications are still little developed or are only supported in very limited scales. In that regards, recently the research turned to implement the multicasting functionality at the application layer. In addition, the basis of the current information-dissemination systems is to provide their services to *anyone, at anywhere and anytime* based only on the users' demands without considering their situations. Moreover, they posed

the users to be passive and non-cooperative actors in the systems. *Situation and context-aware* dissemination-oriented cooperative services have motivated an increasing interest for evolving both the social and economic environments. In that aspect, there is a number of research and development undertaken in this field. For example, in our research group we have proposed a service-oriented community system that provides sale information when a particular Mall holds a sale at a specific time [7, 8].

1.2 Problem Statement

The subsequent growth and the evolving in social and economic environments promote more severe and complex requirements for the information-dissemination service systems. The dissemination model for information delivery has motivated an increasing interest. In this model, the user subscribes the desired information and then passively receives new information. The traditional forms of data distribution rely on dedicated message servers that can be setup, maintained and accessed within an enterprise infrastructure relatively easy. SonicMQ and SwiftMQ emerged as the best performing *Sun's Java Message Service* (JMS) [9] based message servers. However, such infrastructures are seldom available outside enterprises. The need for alternatives to these traditional forms of data distribution (client-server model) has been identified in modern large-scale and very dynamic networks. *Peer-to-Peer* (P2P) systems offer an alternative to traditional client-server systems for some application domains. Despite their great potential, P2P systems still lack efficient data dissemination mechanisms. Current P2P networks suffer from several problems. First, the overlay network is more or less random. Peers with long latencies to their neighbors slow down entire network branches. Second, the overlay network is static and prohibits dynamic adjusting to alternatives that would be more efficient. Third, P2P communications use the destination address to send the data. In very changing environment, the state of the peer and the stability of the connections are so unpredictable. This exhibits a non-continuous service provision. Fourth, the disseminated data has to traverse peers that are not interested by such data and are not group members but happened to be on the group communication route. This results in consuming

bandwidth of not involved peers and it slow down the communication performance, as additional nodes have to be traversed. Finally, P2P systems deliver the information considering the users' demands regardless of their situations. There is no discernment between differences in place and time; users in any situation receive the same contents. Thus, it is highly required to propose new information-dissemination system that provides users *what they want, when they want, where they want and how they want*. Moreover these system must realize the requirements/constraints of the large-scale and dynamic environment.

The goal of this thesis is to deal with these problems and provide an efficient solution to disseminate the information among end-users with considering their heterogeneous and dynamic characters. Inspired from both the social community and the *Autonomous Decentralized* (ADS) concepts [10,11], an *Autonomous Decentralized Community Communication System* (ADCCS) is proposed. It performs a communication over a proposed *Community Overlay Network*(CON). The main idea is to form an autonomous decentralized CON of end-users having not only same demands, but also same situation. Moreover, the CON is a self-organized network that is constructed and maintained with latency-awareness and dynamically adapts to network changes. It only consumes the bandwidth of the involved end-users (*community members*). This results in realizing customized and high performance information-dissemination services.

1.3 Contribution

The main contribution of this thesis is to offer tenable solutions that enable the dissemination-oriented services. This research work explicates the *Autonomous Decentralized Community Communication System* (ADCCS) concept, architecture, communication technique and community overlay network construction/maintenance techniques. The proposed ADCCS is a flexible framework that has been specifically designed for providing a solution to information-dissemination to a large set of users having same demands and situation under changing environment. It can serve as the basis for development of information systems that have in common some charac-

teristics, such as the distribution of information sources to a potentially large set of users. These users are in same situation in terms of time and location and have same demands.

The aim of the proposed ADCCS concept is to customize the dissemination service to the end-users. It is proposed to provide service to the right users, at the right location and the right time. More precisely, ADCCS provides *what we want, when we want, where we want and how we want* type of services autonomously and continually. To address the challenges of the current information systems (e.g. large-scale and dynamic environment) an autonomous decentralized architecture is proposed. Each node has autonomy in controlling its own processes and coordinating with other nodes their own resources. Moreover, each node has locality by keeping track of only local information (neighbor members' list). Locality is the key of the system scalability. In addition, by autonomy and locality the ADCCS can achieve scalable online-expansion, online-maintenance and fault-tolerance requirements of the information service. Thus it is possible to assure the service continuity and to cope with the dynamic changes.

The autonomous decentralized community communication technique is proposed to address the problems of the conventional communication technologies. For flexible communication, the proposed communication technique separates the logical community service's identifier from the physical address. Nodes autonomously recognize community members from non-members and only route community information to only small number of neighbor members. A multilateral community communication protocol is proposed for timely communication to achieve timeliness.

For flexible, continuous and efficient dissemination service, we design the community overlay network without support from the network level. The community overlay network (CON) is constructed under two assumptions: node-node has homogeneous latency and node-node has heterogeneous latency. A single community, *FLAT-CON*, is constructed under the node-node homogeneous latency assumption to maintain the network diameter short. This results in decreasing the data dissemination delay. Join/leave/fault-tolerance processes with low complexities are proposed for continuous service provision and utilization. The proposed fault-tolerance process

maintains the FLAT-CON as 2d-regular graph composed of d edge-disjoint Hamilton cycles(non partition-able network) [12]. Multiple communities, *Multilayer-CON*, is constructed under the node-node heterogeneous assumption. It organizes the CON as multi-layer architecture of sub-communities (Flat-CON) considering node-node latency. Sub-communities are added, disappeared, integrated or divided, layers are added or disappeared and nodes are moved to appropriate sub-community, in order to adapt and robust the dynamic changes of the networks (e.g. frequent join/leave) and latencies. This results in achieving high performance (e.g. Timeliness) of the data dissemination service over the CON.

1.4 Outline

The evolution and challenges of the distributed information systems are illustrated in chapter 2, *Background: Information Systems*. It discusses the current distributed information system architectures and structures and highlights their explosion of the scales along with the dynamic environments. It explicates the tight requirements to re-assess how to develop large-scale decentralized information systems under this emerging computing landscape. Chapter 2 introduces the messaging basic terms and concepts, communication protocols and then information delivery alternatives. A literature review of related works that tackles the problem of efficient information-dissemination services is presented. Most of these related works fail to cope with users requirements in a large-scale and dynamic environment.

Chapter 3, *Research Objectives*, presents the research objectives, including the problem formulation and the application domains. It clarifies the application and system requirements of the decentralized information-dissemination systems. At the end of this chapter, a brief description of our approach is illustrated. This approach is inspired from both the *Social Community* and the *Autonomous Decentralized Systems* to realize *High Assurance* to the dissemination information systems.

Autonomous Decentralized Community Communication System , Chapter 4, clarifies the concept of *Autonomous Decentralized Community Communication* and the architecture of the proposed Autonomous Decentralized Community Communication

System (ADCCS). Autonomous Decentralized Community Communication is defined as the ability to allow users (*Community members*) that have same demands and situations (time and place), under large-scale and changing environment, to cooperate and share the information-dissemination penalties and benefits as well. This goal is realized through loosely connected, loosely control, self-organizing and self-adapting *Autonomous Decentralized Community Architecture*. In that aspect, node architecture contains five *Autonomous Control Processor*(ACP) sub-systems (modules) and *Node Data Field* (NDF) that assures communication between connected modules. These autonomous subsystems are in charge of autonomous membership management, service customization and information-dissemination communication service. This chapter presents concept, requirements and design issues of the *Community Overlay Network* (CON). To assure the efficient communication among the autonomous nodes the CON is constructed in Chapter 6.

Chapter 5 presents the autonomous decentralized community communication technology. In this chapter, the $1 \rightarrow N$ community communication is proposed to permit cooperative users to disseminate information timely. Each community node receives, filters and then routes the received messages to only its neighbors. The evaluation of the $1 \rightarrow N$ community communication is provided. The robustness of community communication techniques should account for the nodes heterogeneity, autonomy, relying on self-inspection and adaptation to exploit the differences in the nodes' characteristics, behavior, and incentives.

Chapter 6 elucidates the construction technology of the CON. A single community so-called *Flat-CON* is self-organized autonomous overlay network that sustains the online-expansion and fault-tolerance of the community network. Flat-CON composed of disjoint *Hamilton cycles*. This chapter clarifies the Flat-CON's construction, fault-tolerance and maintenance techniques. It ensures that these techniques process with low complexities. Furthermore, Chapter 6 studies the effectiveness of the proposed community communication delay, network traffic and fault-tolerance associated with Flat-CON. The results confirm that ADCCS, which operates over Flat-CON, can achieve the fault-tolerance, timeliness and scalability of the large-scale information-

dissemination systems under the homogenous end-node to end-node latency assumption. However, in the realistic Internet environment the node-to-node latency is heterogeneous. The question then is: can ADCCS support large number of members with different communication cost? Then this chapter answers this question by considering the latency between community nodes as an important criterion that need to be optimized. In that regards, multi-layer of multiple community overlay networks (*Multilayer-CON*) is proposed. Multilayer-CON organizes the community overlay network into a number of homogeneous sub-communities (*Flat-CON*) thereby reducing the communication delay and join overhead. Each sub-community has two special members: *Leader* and *Disseminator*. The latency from any node to the leader and the disseminator is bounded by specific value α. Furthermore, this chapter defines the sub-community and then describes how to step by step construct and maintain sub-communities and the multi-layer structure too. To cope with the dynamic changes of the network (frequent join/leave) and latency this chapter presents two technologies: Sub-Community Division/Integration technology and Self-Adapting Multilayer-CON technology. Finally, this chapter presents the simulation results that demonstrate the effectiveness of the proposed techniques in realistic Internet settings. The results confirm that ADCCS-Multilayer that operates over Multilayer-CON achieves the fault-tolerance, timeliness and scalable online-expansion of large-scale information-dissemination systems under the heterogeneous node-to-node latency assumption. In addition the results give an evidence of the existence of the trade-off relation between fault-tolerance and timeliness along with the number of *Hamilton cycles*.

Chapter 7 presents a comparative analysis between ADCCS and the other conventional application level communication systems. Finally, Chapter 8 draws the conclusion and then describes some potential future works to address.

Finally the thesis manifests that the proposed autonomous decentralized community communication system achieves the requirements to establish a High Assurance information dissemination system.

2 BACKGROUND: INFORMATION SYSTEMS

2.1 History, Changes and Challenges

2.1.1 History

Information System (IS) is defined as a system that manipulates data and normally serves to collect, store, process and exchange or distribute data to the users within or between enterprises or to users within narrow or wider society [13]. In a rapidly changing social and business environment these users need to work cooperatively and share information in a convenient way without being bothered by geography or physical distribution of users, data and machines. Information science developers applied computers to manipulate documents and document records in information storage and retrieval systems. This began almost as soon as computers became available in the 1950s. Computers were large and expensive. Starting in the mid-1980s two advances in technology began. The first was the development of the powerful microprocessors. The second development was the invention of high-speed network [14] and *Open System Interconnection model* (OSI) [15, 16]. These technologies realize the users' needs by establishing computing systems that are composed of large number of CPUs connected by a high-speed network. These system are usually called *distributed information systems* in contrast to the *centralized information systems*. The Internet topology was developed in 1960s by DoD[1]. It is a powerful example of a successful distributed system environment. The digital data in the information systems are formed as documents. Ordinarily the word "*document*" denotes a textual record that contains digital information [17]. Moreover, the formats of the digital information are varied from text to rich multimedia. Multimedia and hyperlinked objects on the *World Wide Web* (WWW) represent some of the new kinds of information and new ways of knowing in the digital realm that bring together the traditional forms of information and transform their use. The development of computer hardware and

[1]US Department of Defense

software has also generated other new kinds of information objects, including the products of simulation, remote sensing, *computer-aided design* (CAD) and *geographic information* (GIS) systems. These objects come into being and exist as creatures of the digital environment; if developed well, digital technologies will certainly produce still other kinds of information objects, which we can now only anticipate.

2.1.2 Changes

The industrial era was rapidly replaced by the new information age and technologies. In this new phase, science and knowledge are becoming the critical vectors of the so-called *value-added* economy. The world of communications is gradually changing from an economy of scarcity and government-structured controls to a free economy oriented towards abundant supply and diversity. This change quickens the place of the elimination of monopolies in the delivery and distribution of information in telecommunications. Moreover, both technology and economy are radically modifying the ways in which we use communication, both at work and in the home. Users now have powers of transmitting and receiving information undreamed of even ten years ago. The accelerating progress in new information and communication technology is essentially based on three fundamental changes: digitization of images, sounds and data; digital data compression; the growing of the power of the electronic components. Three driving forces have served to propel distributed information systems as the key development for the 2000s and beyond: advances in computer and communication technology, growing applications requirements and rapidly changing social and business environment.

2.1.3 Challenges

Over the last few years, there have been dramatic changes in the computing and communication landscape. The phenomenal growth of the Internet, the WWW and the availability of high-bandwidth links into home and on the road via satellite have revolutionized the way data is delivered and distributed between computers. The *Internet Systems Consortium*(ISC) advertised in January 2004, the worldwide number

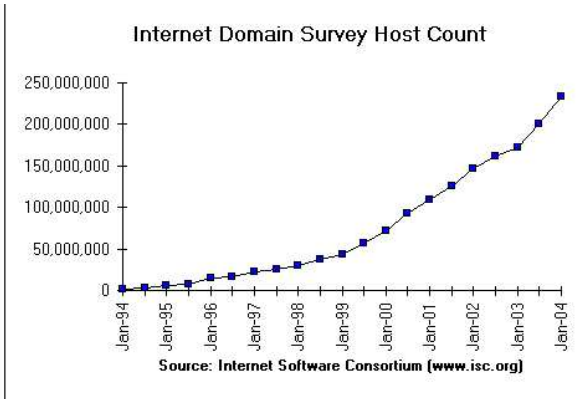

Figure 2.1. Number of Internet hosts Worldwide

of hosts touted in the DNS was 171,6 million [18], in contrast it was about 3,2 million in 1994 and 1000 in 1984, as shown in Figure 2.1. According to Forrester Research [19], it is anticipated that online advertising will reach $33 billion by 2004, one-third of which will be spent outside of the United States. Consequently, the world publishes online about 300 terabytes of information every year [2]. In addition, recent reports estimate that the number of the worldwide Internet and mobile users will exceed 1 billion by the end of 2005 [1]. This explosion in the scale of the Internet along with the dynamic data and the advances in processing power and communication is beginning to test the limit of many assumptions traditionally made when designing distributed information systems. Current distributed information systems does not benefit from the Internet's collaborative potential that resides at the edge nodes. The trend is to head towards decentralized information systems that benefit more from the computing at the edge paradigm. Moreover, in a large-scale systems users' preferences and situations change dynamically thus these systems cannot customize the service the right users at right time and location. The combination of these factors is bringing forth a number of new and interesting challenges forcing us to re-evaluate how to design and implement large-scale decentralized information systems. In this thesis, the developments of large-scale decentralized information-dissemination system and the issues that they raise are illustrated to adapt to the new tradeoffs in this emerging computing landscape.

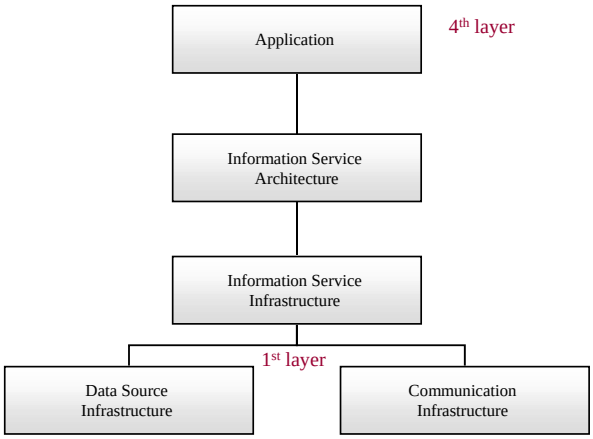

Figure 2.2. Information Systems Architecture: Layered Approach

2.2 Architectures

The layered approach has been adopted as conceptual framework for the design and maintenance of the information systems. Figure 2.2 illustrates such an approach where each layer is founded on the infrastructure provided by the lower layers. A short description of each of the layers is as follows:

- *Data Source Infrastructure* consists of various methods for managing the data quality of a single data source by using either syntactic (e.g. data mining [20]) or semantic (e.g. metadata) tools such as DAML-S [21].

- *Communication Infrastructure* consists of the seven layers of the OSI reference model [16]. The higher information service layers build on the existing telecommunication technology.

- *Information Service Infrastructure* assumes the existence of communication and data source tools and provides mechanisms for handling information from heterogeneous distributed data sources. Existing tools and standards developed at this layer include HTTP requests [22], *Java virtual machines* [23] and *Remote Procedure Call* (RPC) [24] protocols.

- An *Information Service Architecture* provides an overall architecture for handling the semantic issues involved in it.

- The top layer, the *Application layer* consists of particular classes of services such as e-commerce, intranet applications, and others.

The layered structure is useful in offering a useful method for integrating technologies. The particular interest in this thesis is second and fourth layers. Consequently, the following subsections will illustrate four models that are commonly used to structure an information system [25].

2.2.1 Master-Slave model

In computer networking, master/slave is a model for a communication protocol in which one device or process (known as the master) controls one or more other devices or processes (known as slaves). Once the master/slave relationship is established, the direction of control is always from the master to the slave(s). Slaves exhibit very little intelligence, responding to a command from a single master and exchange messages only when invited by the master. The master defines the command set and appropriate responses associated with the dialogue. The slave merely compiles the dialogue rules. An example configuration is given in Figure 2.3. This was the model on which online centralized machines running time-sharing information systems were based. The model has limited application in a distributed information technology infrastructure because it does not make best use of distributed resources and the master represents a single point of failure.

2.2.2 Client-Server model

The client/server model has become one of the central ideas of network computing. Most business applications being written today use the client/server model. It is widely used paradigm for structuring distributed information systems. Services are accessed via a well-defined interface that is made to the clients. A client requests a particular service. One or more servers are responsible for the provision of a service

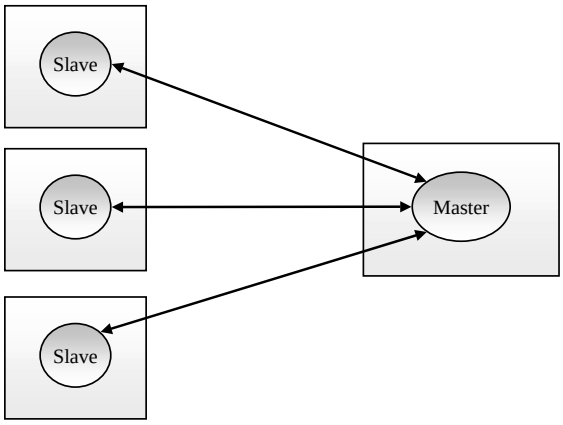

Figure 2.3. Master-slave model.

to clients. A server is normally persistent and provides services to more than one client. Obviously, client-server interaction is based on a request/reply protocol as illustrated in Figure 2.4. Both client programs and server programs are often part of a larger program or application. In the Internet, Web browser is a client program that requests services (e.g. Web pages or files) from a Web server that technically is called a Hypertext Transport Protocol [22] or HTTP server. The main distinction between master-slave and client-server models lies in the fact that client and server are on an equal footing with distinct roles and functionalities. Other client/server relationship models included master/slave, with one program being in charge of all other programs, and peer-to-peer, with either of two programs able to initiate a transaction.

2.2.3 Peer-to-Peer model

Peer-to-Peer(P2P) [26] is a communications model in which each party has the same capabilities and either party can initiate a communication session. In some cases, peer-to-peer communications is implemented by giving each communication node both server and client capabilities and functionalities as illustrated in Figure 2.5. In recent usage, peer-to-peer has come to describe applications in which users can use the Internet to exchange files with each other directly or through a mediating server. *Gnutella* [27] and *Chord* [28] are two examples of products that support the peer-to-

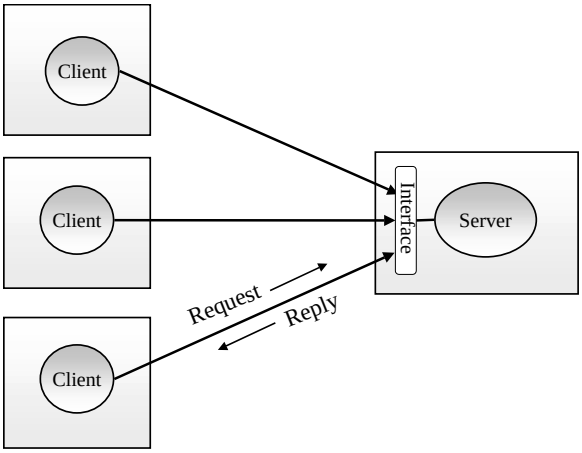

Figure 2.4. Client-Server model.

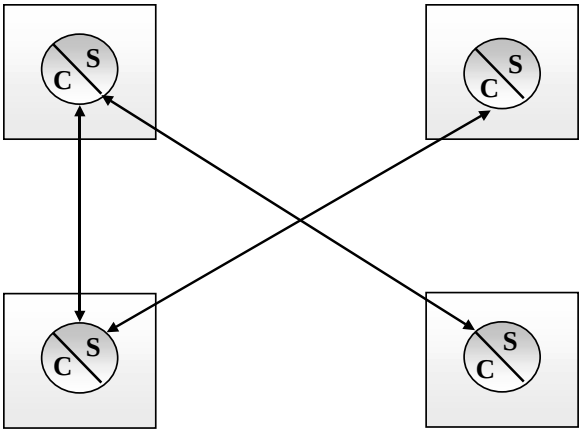

Figure 2.5. Peer-to-Peer model: C, Client function; S, Server function.

peer communication model. Corporations are looking at the advantages of using P2P as a way for employees to share files without the expense involved in maintaining a centralized server and as a way for businesses to exchange information with each other directly. The P2P model evades many problems of client-server systems but results in considerably more complex searching, node organization, security, and so on. P2P networking offers unique advantages that will make it a more effective solution to several existing client-server e-commerce applications if it can mature into secure and reliable technology [29].

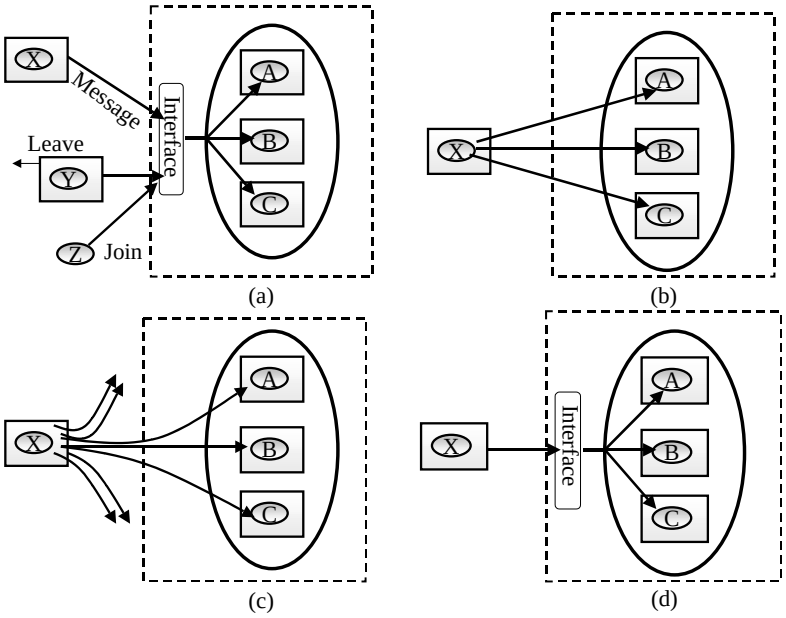

Figure 2.6. Group model.

2.2.4 Group model

A group is a collection of computers or processes that cooperate in such a way that one process may need to send a message to all other processes in the group and receive response from one or more members. A worldwide-distributed discussion system USENET [30] is an example. When a subscriber sends a message to a news item, all other subscribers receive it. The group model is illustrated in Figure 2.6-(a). When a message sent to the group interface, all members of the group receive it. There are three main approaches [31] to route the message to every member. First approach is *unicasting*, a sender sends a separate copy of the message to each member as shown in Figure 2.6-(b). Second approach is *broadcasting*, a sender broadcasts a single message to all members with a broadcast address as shown in Figure 2.6-(c). Every members receive the message and determine whether they should take action or discard it. This is not appropriate mechanism to use over the Internet and may lead to *Broadcast Storm*. A Broadcast Storm is an undesirable network event in which many broadcasts are sent at once. Broadcast storms use substantial network bandwidth and may cause network time outs. Therefore, the broadcast-based group communications

do not scale well. Third approach is *Multicasting*, a sender sends a single message to a group address that can be used for routing purposes as shown in Figure 2.6-(d). This is an efficient mechanism since the number of network transmissions is significantly less than for unicasting. It relies on an underlying network facility. The IP multicast was proposed over a decade ago [5]. However, the use of multicast in applications has been limited because of the lack of wide scale deployment and the issue of how to track the membership management. An alternative to router-dependent multicast service is to let end-nodes that form a multicast group replicate and forward messages on behalf of the group. The basis of the communication in this thesis is based on this alternative (independent of the underlying network).

2.3 Related Works

Several available centralized or distributed message/event systems aim at easing the development of communication-intensive distributed applications. This section presents the messaging basic terms and concepts, communication protocols, information delivery alternatives and then reviews some systems that have been developed to solve the problem of efficient information-dissemination services. The communication approaches of these conventional dissemination systems are client/server and Peer-to-Peer. We review their basic concepts and focus on important aspects for their communication model like efficiency, topology, scalability and robustness.

2.3.1 Messaging: Basic Terms and Concepts

The terms *message* and *event* describe a segment of data that is sent from one user and received by others to communicate with each other. The message contains some text, a picture, multimedia data, etc. The term *message* is more general than the term *event*. Event is triggered by some incidents for sending a segment of data. The difference between them might be slight, thus along the context of this thesis we will use the term *message*.

Asynchronous versus Synchronous Communication Asynchronous communication model is the transmission of message between two nodes that are not

synchronized with one another via a clocking mechanism or other technique. Basically, the sender can transmit message at any time, and the receiver must be ready to accept information when it arrives. FIFO (first-in, first-out) message queues are used to develop this model. Queues allow processing messages independently from the time when they were stored. In contrast, synchronous communication model is a precisely timed stream of data. The RPC [24] concept provides this model. The protocols that assume the system is synchronous exhibit performance degradation as the delivery delays increases [38].

Publishers/Subscribers A *publisher* is a sender of a message to multiple receivers, called the *subscribers*. They subscribe the desired information on a specific topic and then passively receive all new messages that are related with this topic. In addition, the subscribers may use message filters to decide whether messages are relevant within a specific context or not. If not, they don't have to be processed any more in this context. Often a one-to-many communication model is used to support publish/subscribe concept. In some cases, many-to-many communication model is used, if multiple publishers are taking a part of information provision. Messaging systems usually support the publish/subscribe concepts. Most of these systems rely on dedicated message servers that can be setup, maintained and accessed within an enterprise. These systems can be developed by using JMS specification [9].

Messaging Systems Messaging systems offer interfaces to send, receive, filter and route messages. They are considered as special types of middleware and are called *Message Oriented Middleware* (MOM) [39]. They reduce the complexity of developing applications that span multiple operating systems and network protocols by insulating the application developer from the details of the various operating system and network interfaces. MOM has recently received considerable attention because of its decoupled nature that nicely solves asynchronous one-to-many communications in highly dynamic distributed environments. In contrast to RPCs, MOMs do not model messages as method calls; instead they

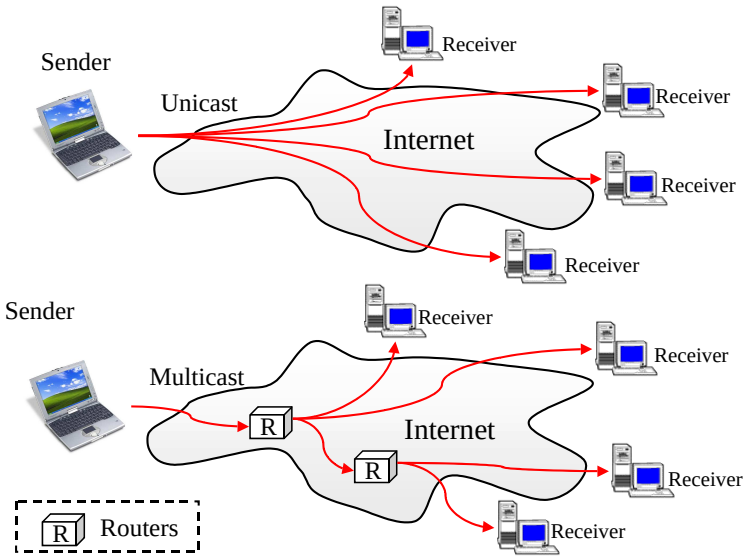

Figure 2.7. Unicast versus Multicast

model them as events in an event delivery system. Messages generated by applications are meaningful only to other clients, because the MOM itself is only a message router.

2.3.2 Communication Protocols

1. ***Unicast*** is a communication protocol between a single sender and a single receiver over a network as shown in Figure 2.7. An earlier term, *one-to-one* communication, is identical in meaning to unicast. The *Internet protocols* (IPv4) and (IPv6) support unicast. Unicast is performed over two transmission protocols *User Datagram Protocol*[2] (UDP) and *Transmission Control Protocol* (TCP). UDP does not need membership and offers unreliable message delivery. In contrast, TCP provides an explicit conversation based-model that requires synchronous rendezvous and explicit membership. Thus, TCP offers reliable source-ordered message delivery.

[2]UDP used a self-contained, independent entity of data (Packet) carrying sufficient information to be routed from the source to the destination.

2. ***Broadcast***, in general, is to cast or throw forth something in all directions at the same time. It is sometimes used in e-mail or other message distribution for a message sent to all members, of a group such as a department or enterprise. It is supported by most local area network (LAN) technologies, such as *Ethernet* and *Token Rings*. Senders direct an IP broadcast to 255.255.255.255 to indicate all nodes on the local network (LAN) should pick up that message. This broadcast is limited in that it does not reach every node on the Internet, only nodes on the LAN.

3. ***Multicast*** is a communication protocol between a single sender and multiple receivers on a network as shown in Figure 2.7. Multicast is also used for programming on the *MBone*[3] [40] systems that allow users at high-bandwidth points on the Internet to receive live video and sound programming. In addition to using a specific high-bandwidth subset of the Internet, Mbone multicast also uses a protocol that allows signals to be encapsulated as TCP/IP packet when passing through parts of the Internet that cannot handle the multicast protocol directly. Deering [4] proposed the IP Multicast architecture in 1990. IP Multicast is a networking transmission protocol that allows packets to simultaneously transmitted over the MBone to a selected set of destinations. This is faster than sending packet to each individual node by using unicast. In March 1992, the MBone carried out its first event with 20 nodes worldwide-received multicast audio streams from a meeting of the Internet Engineering Task Force (IETF) in San Diego [41]. Since the initial proposal and the deployment of IP Multicast, it has been limited and sparse due to a variety of reasons. For example, the overhead and complexities at the routers are increased [6]. In addition, IP Multicast requires additional mechanisms for congestion control (e.g. MTCP [42] and PGMCC [43]) and reliability (e.g. RMPTP [44]).

[3]It is a multicast Internet that is an arranged use of a portion of the Internet for Internet Protocol (IP) multicasting. It was set up in 1994.

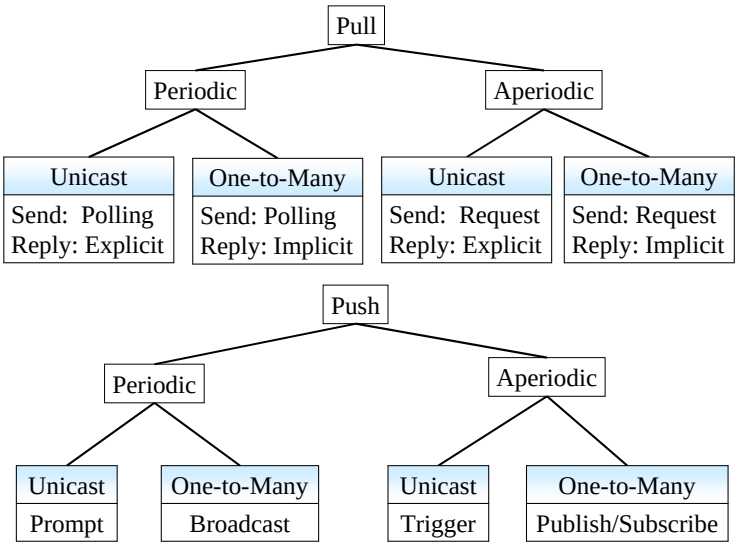

Figure 2.8. A taxonomy of information delivery methods

2.3.3 Information Delivery Alternatives

This section classifies some existing information delivery techniques. Figure 2.8 clarifies the information delivery methods according to the *pull-based* or *push-based* protocols. With pull-based delivery, a user sends a request to a server. When a request is received at the server, it sends the required information to the user. In contrast, with push-based delivery, the server initiates the transfer to the users. Both push and pull can be performed *aperiodically* or *periodically*. Periodic delivery is performed according to some schedules that had arranged in advance. Contrary, the aperiodic delivery is event-driven. The event can be either data update for push or user request for pull. For example, stock prices' server sends out stock information on either regular basis (periodic-push) or only when information is updated (aperiodic-push). The periodic push has been widely used for data dissemination in many systems, such as TeleText, VideoTex [45] and Datacycle [46]. In thesis system, the periodic push uses two communication models: *one-to-one* (Unicast) or *one-to-many*. Two types of one-to-many data delivery can be distinguished: *broadcast* and *multicast* as were described in section 3.4.2. Some leaves in Figure 2.8 show an interesting classification of the information delivery techniques that will be discussed as follows.

Request/Polling Users aperiodically send their requests to the server. In the other hand, the system may periodically poll the other sites to detect updated values (e.g. remote sensing systems).

Explicit-reply/Implicit-reply They are two types that can be used to reply users' requests. First, the server explicitly replies only to the requester by using one-to-one communication model. Second, the server replies not only to the requester, but also to the other users in the system by using one-to-many communication. Thus, users implicitly obtain information that they did not ask for. The particular interest in this thesis is the *Implicit-reply* information delivery technique.

Publish-subscribe The demand for publish/subscribe protocols is growing at an incredible rate. Publish/subscribe is push-based; data flow is initiated by the data source, and is a periodic, as there is no predefined schedule for sending data as shown in Figure 2.8. It is typically performed on one-to-many communication model but other systems are performed on one-to-one communication model (unicast), as is done for *triggers* in active database systems. For example, the *Stanford Information Filtering Tool* (SIFT) is a tool for wide area information-dissemination [47]. To customize the service to the SIFT users, the connection between SIFT server and user is push-based, unicast, and aperiodic.

2.3.4 Dissemination Systems' Structures

During the past decade, much work has been done in demonstrating the usefulness of publish and subscribe event infrastructures to large distributed systems. The publish and subscribe model is powerful because it provides a named handle on a conversation between any number of distributed parties. The structures of the information-dissemination systems are classified into client-server and peer-to-peer as follows.

Client/Server

A very large number of publish and subscribe systems have been developed, are almost universally structured as client-server and are in use today in commercial distributed systems. However, in wide-area settings these platforms clearly suffer from scalability problems. Several of these systems are presented briefly hereafter.

1. ***Information Bus.*** The Information Bus [48] was developed as a commercial distributed system infrastructure in the early nineties. The system provides publish and subscribe style distribution to applications requiring zero down time and upgradeability. Sample applications cited by the authors are stock floor systems and integrated circuit manufacturing plant systems. The architecture was built on TCP/IP, and used specialized servers to handle message queuing. Ethernet broadcast was used as an optimization for group communication in local subnets. Information Bus provides appropriate performance only within an enterprise infrastructure.

2. ***SIFT.*** The SIFT [47] system was developed at Stanford University as a way to disseminate documents to a user community. It combines data management ideas from information retrieval with a publish/subscribe model for dissemination. SIFT has three components: the document source, the SIFT server and a SIFT client (one of potentially many). The document source component aperiodically pushes the document to the SIFT server using unicast communication protocol. Then the SIFT server aperiodically pushes the new document to each client using unicast. In this case, the client profile that is held at the SIFT server is customized for each client. It consists of a series of keywords that describes the documents interest to that client. The SIFT server is responsible to index client profiles. The index technique should allow the server to accommodate a large client population with reasonable performance. Clearly, SIFT is not scalable solution for information-dissemination. Increasing the number of clients increases the overhead to store and index the clients' profiles.

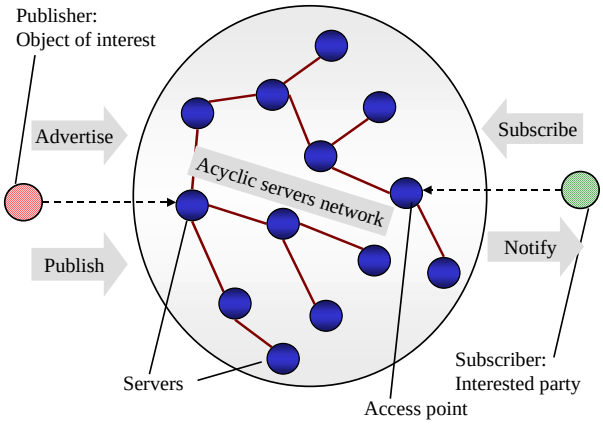

Figure 2.9. SIENA: distributed event notification architecture.

3. ***SIENA.*** It is a distributed content based publish/subscribe system that was developed at the *University of Colorado* [49]. It is composed of interconnected servers (brokers, proxies), each one serving some subset of the clients of the service as shown in Figure 2.9. The clients are of two kinds: objects of interest, which are the generator of events, and interested parties, which are the consumers of the event notifications. Clearly, a client can act as both an object of interest and an interested party. Both kinds of clients interact with a locally accessible server that functions as an access point to the network-wide service. Creating network of servers, routing algorithms and processing strategy are the main issues having been studied in SIENA. The servers are organized into an acyclic peer-to-peer architecture. However, the external clients of the service use the standard client/server protocol. The network of servers is not able to cope with failures because of the static logical topology. SIENA relies on a global broadcast operation to disseminate advertisements through the entire network, which limits its scalability.

4. ***Java Message Service (JMS).*** Enterprise messaging is now recognized as an essential tool for building enterprise applications. JMS [9] is a specification developed by *Sun Microsystems*. By combining Java technology with enterprise messaging, the JMS API provides a new, powerful tool for solving enterprise

computing problems. *IBM's MQSeries* and *Talarian's SmartSockets* are two message systems have been implemented based on JMS. Java Message Service was originally developed to provide a common Java interface (API) to legacy *Message Oriented Middleware* (MOM) [39] products. This API brings portability of Java code which facilitates the replacement of underlying message service without affecting exiting code. JMS offers two models for messaging among clients: one-to-one (using a queue) and publisher/subscriber (by means of topics). The publishers and subscribers communicate indirectly with each other to achieve a high degree of decoupling. To do so, an application server is in control of setting up the connections among publishers and subscribers. JMS was incorporated as an integral part of the *Enterprise Java Beans* (*EJB*) component model in the *EJB 2.0* specification by defining a new bean type, known as message-driven bean (MDB). This new bean acts as a message consumer providing asynchrony to *EJB-based* applications.

Peer-to-Peer (P2P)

Some number of publish and subscribe systems have been developed based on the peer-to-peer architecture. They deliver the information to the subscribers using multicast. In P2P architecture, the multicast functionality is pushed to the *end-nodes* (end-hosts) actually participating in the multicast group. The communication in this group is performed over an *overlay network*. This section will clarify motivation, overview, architecture and discussion of two well-known application level-multicast systems, ALMI [50] and Narada [51]. Both are dedicated for collaborative applications with a small number of group members.

1. ***Application Level Multicast Infrastructure (ALMI)***

   Motivation There exist some applications whose requirements are substantially different from the design point of IP multicast. Such applications

include shared white-board, video-conferencing, multi-party games and private chat rooms. These applications usually contain a small number of group members, and the groups (e.g. multi-party games) are often created and destroyed relatively dynamically. The number of such groups that are concurrently active can be fairly large. ALMI is motivated by the need to support group communication among small group of nodes without relying on the IP-Multicast mode.

Overview To meet the requirements of such emerging applications, ALMI offered a solution for multi-sender communication that scales for groups with small number of members, and did not depend on multicast support of the routers. It provided a multicast middleware that is implemented above the socket layer. The participants of ALMI multicast session were connected via a virtual multicast tree. The tree was formed as a *Minimum Spanning Tree* (MST), where a cost of each link is the round-trip application level delay between nodes.

Architecture An ALMI session consists of two kinds of sessions. First, *Controller session* is a program instance located at a special purpose server that is easily accessible by all members. This server can be installed within a corporate or an *Internet Server Provider* (ISP) network. The controller session handles membership management and maintains of the multicast tree. Moreover, it ensures the efficiency of the multicast tree by periodically constructing the MST. Second, *Members' sessions* are organized into a multicast tree. A member session in charge of receiving/sending data as it would be in IP multicast session. In addition, it also forwards data to designated adjacent members. Thus, data reaches all session members through this communication technique in a cooperative fashion. The multicast tree is a shared-tree amongst members with bidirectional links.

Discussion Despite ALMI great potential, it takes a centralized approach to the tree creation problem. Even this approach may reduce the overhead during a change of membership or a recovery from a node failure; clearly, it

constitutes a single point of failure for all control operations related to the group. The choice to have multiple backup controllers, operating in stand-by mode may address the controller fault problem, but raise other problems like non-consistency of membership data among them. In addition, the required network traffic for control messages near to the controller increases with increasing the control operations. Thus, ALMI fails to distribute the network traffic evenly across the physical links.

2. ***Narada: End System Multicast***

Motivation End system multicast, Narada, shares with ALMI the application domain. Contrary to ALMI, it takes a distributed approach for membership management. The participants *end-nodes* of Narada multicast session, join, leave or fail relatively dynamic. That leads to either network partition or degradation of data dissemination performance. Narada shares much of the motivation of ALMI. Moreover, it is motivated by the need to construct a self-organized overlay network for achieving efficient data dissemination and fault-tolerance.

Overview Narada challenges the appropriateness of using IP Multicast for all forms of group communication. It argues the need for nodes to construct overlay networks for efficient data dissemination and fault-tolerance. Thus, Narada constructs an overlay structure among participating end-users' nodes in a self-organizing and fully distributed manner. It is robust to the failure of end-users' nodes and to dynamic changes in group membership. Narada continually refines the overlay structure, as more network information is available.

Architecture Narada constructs the overlay network based on the mesh-first approach. In this approach, end-nodes first organize themselves into overlay mesh[4] topology in a distributed fashion. Multiple paths exist on the mesh between a pairs of end-nodes. Second, Narada construct spanning

[4]Richer connected graph

trees of the mesh, each tree rooted at the corresponding source using well-known routing algorithms (e.g. Reverse Path Forwarding [52]). This mesh-first approach is motivated by the need to support multi-source applications. To keep the mesh connected, every end-node maintains a list of the other end-nodes in the multicast group. Every end-nodes' list needs to be updated when a new end-node joins or an existing node leaves or fails. To handle this, Narada requires that each end-node periodically generate refresh message with monotonically increasing sequence number, which is disseminated along the mesh. These refresh messages are the key point to repair the mesh partitions. Distribution of refresh message to all other end-nodes leads to relatively high control overhead (O(N^2) aggregate control overhead, where N is the group size). In addition, Narada improves the mesh quality by adding and/or dropping of overlay links. It performs data delivery as follows. Each end-node not only maintains the routing cost to every other end-nodes, but also maintains the path that leads to such a cost.

Discussion The overall results of Narada suggests that it can achieve good performance for small and medium sized groups involving tens to hundreds of end-nodes [51]. The self-organizing and improving overlay approaches incur overhead due to the network traffics and active measurements respectively. While maintaining full group membership information helps to achieve Narada goals, it leads to the concern that the cost of maintaining such information may be prohibitively expensive for large sized groups. As multicast group size increases, it is not clear whether Narada can keep probe overhead low and construct efficient overly quickly or not. Thus, Narada is not scalable for data dissemination for large number of end-nodes.

The membership management of these previous related works varied from centralized to decentralized. In the client/server structure, clients have to subscribe in a dedicated server. Similarly, some peer-to-peer information-dissemination systems,

such as ALMI and Bayux have some sub-entities are able to grasp the total system for membership management. In addition, other peer-to-peer systems, such as Narada, each sub-entity has to know the total system. In large-scale and rapidly changing environments, it is potential that system scalability and adaptability can only be obtained by assuring the locality and autonomy of the all entities constituting the system. The trend is to head towards decentralized models that benefit more from the computing at the edge paradigm. Thus, our research work sustains the same direction as previous researches on ***Autonomous Decentralized Systems*** (ADS). In addition, this thesis extends the concepts of autonomy, locality and decentralization to large-scale information-dissemination services. The following chapter will briefly introduces autonomous decentralized systems.

2.4 Summary

This chapter illustrated the evolution and the challenges of the communication in the information systems. Then, it discussed the system architecture and structures of the current information systems. The explosion of the Internet scale along with the dynamic environments manifests a number of challenges. They enforce us to reassess how to develop large-scale decentralized information systems. The messaging basic terms and concepts, communication protocols and then information delivery alternatives are introduced. Some systems that have been developed to solve the problem of efficient information-dissemination services have reviewed. We classified these systems into two categories: based on client-server approach and peer-to-peer approach. We have shown that most of these systems are still the subject of important research efforts. Most of these systems are fail to cope with users' requirements in a large-scale and dynamic environment. We believe that this trend will continue, since information-dissemination system required many important needs. Finally, this thesis will illustrate the development of a large-scale information-dissemination system under this emerging computing landscape.

3 RESEARCH OBJECTIVES

The previous chapters asserted that the decentralized and dynamic natures of the information services exhibits severe technical challenges for any technology that attempts to provide coherent and compact information. One of these challenges is in a rapidly changing environment how the information service systems provide what users want, when they want, where they want and how they want. In other words, information needs to be efficiently disseminated to the right users at right time and right location. The problem becomes therefore how to provide a continuous and efficient information-dissemination services along with customizing the services to the right users, at right time and right location. This chapter illustrates the problem statement and clarifies the application and system requirements of the decentralized information-dissemination systems. Moreover, it presents the *Autonomous Decentralized system*'s concept and architecture that are the basis of this research.

3.1 Problem Statement

Recent studies have shown that an increasing fraction of the data on the WWW is dynamic. Dynamic data can be defined by the way the data changes. The main issue in the dissemination of dynamic data is the deliver of the right information to the right users in a right location and time. Data changes rapidly, changes can even be of the order of one change every few seconds; it also changes unpredictably, making it very hard to use simple prediction techniques or time-series analysis. Some examples of the applications that disseminate dynamic data are stock prices, sports scores, national election results and traffic or weather data. A scenario of an example of unpredictable dynamic data as shown in Figure 3.1 is as follows. A major event, such as an earthquake happen in a specific location says Tokyo, at specific time. All residences in Tokyo are involved in this event. They have critical demands to know information such as: Where to gain water?, Where to gain food?, Where to find health-care?, etc. The changes of this information are unpredictable. Such kinds

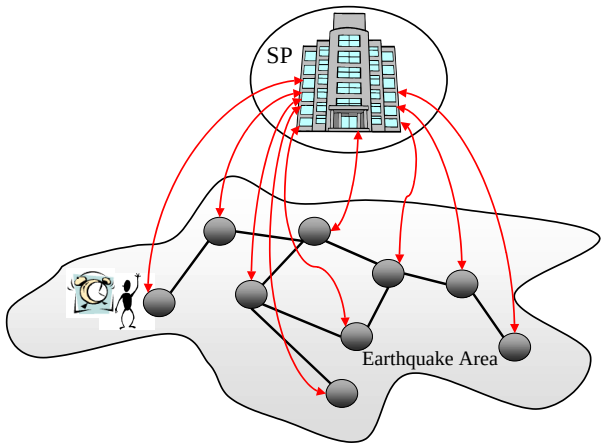

Figure 3.1. Event-based dissemination example: Earthquake

of data are generally used in decision-making (for example, earthquake event, stock trading or weather forecasting) and hence the timeliness of delivery of this data to its users becomes not only very important, but also critical. For example, the delay to deliver health-care information on time is the cause of someone who needs health-care to die. The problem of the current information-dissemination system is how to timely and continuously disseminate information that meets the user's demands and situations. Figure 3.1 illustrates a *flash crowd* that is typically triggered by events of great interest, whether planned ones such as sport events (e.g. FIFA 1998 world cup event [32]) or unplanned ones such as, earthquake and terrorists attack in September 11, 2001 that overwhelmed major news sites such as MSNBC and CNN [33]. For example, peoples in earthquake area send their requests to a city center' server to get information then a rapid and sharp surge in the volume of requests arriving at a server often results in the server being overwhelmed and/or failed and response time is shooting up. In contrast if the volunteers in the earthquake area cooperate in information dissemination then they relieve the city server from this task and alleviate a load on the server by distributing the load among them. The next section further clarifies the application requirements.

3.2 Application Requirements

The situations and demands of the users formulate their requirements. The users' situations are continuously changing; from school to home and so on. The main characteristics of the dissemination-oriented application domains described in last section are: *large-scale number of users*, *dynamic environment* and *high degree of overlap among the users demands and situations*. Users require well-customized service timely, continual and available information services [34]. Moreover, they require enriching their experiences and getting to know new information without solicitation. In that respect, an information service that does not disseminate the information to the users at right time and location is virtually equivalent to one that is unavailable.

In short, the users' requirements are: non-stop and continuous services, high response time and receive the information that is aware of their demands and situation.

3.3 System Requirements

The previous section has discussed the users' requirements. It has been observed from the users' requirements that there are three views of service provision: ***customization, situation and quality*** [35]. A system that meets these requirements is highly required. This section discusses the system requirements that must be realized.

Online-Expansion The system is expanded due to some users/nodes joined into the system. The joining process should be done with low complexity, without stopping the system and maintained high-response for the users. In other words, only a few already existed users in the system should be involved in the joining process. This is the key of the system scalability.

Fault-tolerance To cope with the dynamic changes in the network a fault-tolerance process is required that should be done with low complexity. Node failures must not lead the network partitioning or disconnecting or severely hampering the service provision. The fault-tolerance system provides continuous service even if part of a system is failed.

Timeliness It is an essential component in modern high-assurance[1] and large-scale information systems [36]. It provides a service to the users with high-response. To achieve the timeliness property, the system should be scalable to meet the requirements of a large number of users without suffering any degradation of the system performance. In addition, the system should consider the node to node latency to optimize the communication delay.

Customization The system should fit not only the users' demands, but also their situations. In the Internet users have different daily life requirements such as *what to do?, what/when/where to buy or sell?*, etc. Obviously, these requirements are influenced by their situations. For example, people during weekend or *Golden Week holidays* in Japan are keen to enjoy their time by discovering new services that are suitable to their economy, location, etc.

Finally, the system that simultaneously satisfies the online-expansion, fault-tolerance and timeliness requirements under evolving situations is called *Assurance Communication System* [36, 37]. Thus, the information dissemination system that is able to realize the online-expansion and fault-tolerance with low complexities, timeliness and customization under changing conditions is called a ***High-Assurance Information Dissemination Service System***.

3.4 Autonomous Decentralized Systems (ADS)

Motivated from the molecular biology and the remarkable progress of microelectronics in LSI and communication technology, the autonomous decentralized system [11] has been proposed.

3.4.1 ADS Concept

The constantly evolving system contains entities (subsystems) that are dynamically added, modified and others failed. This results in the total system cannot be

[1]A system is a high-assurance system if it does what it is supposed to do all of the time and when it cannot do so, it fails in a safe way.

known in advance. A system, Autonomous Decentralized System that needs to provide non-stop and continuous services in a rapidly evolving environment must satisfy the following properties:

- ***Autonomous Controllability*** Each entity is able to manage itself, carry out its functions and continue its operations regardless to the other entities.
- ***Autonomous Coordinability*** If an entity added, failed, repaired or modified, the other entities are able to coordinate among themselves to satisfy their goals.

The evaluation of these properties has been shown in [53]. In order to achieve autonomous controllability and autonomous coordinability each entity must satisfy both three conditions and three on-line properties as following respectively:

I. Conditions

- ***Equality*** All entities in the system are equal without having master-slave relationships.
- ***Locality*** Each entity manages itself and coordinates with others entities only on the base of its own local information. In the sense, no entity knows the total system. This is the key of the system scalability.
- ***Self-containment*** Each entity is self-contained in carrying out its operations. It takes its decisions based on its local information.

II. On-line Properties

- ***On-line Expansion*** The typical example of system expansion is the addition of application modules and entities. This process must be realized on-line without stopping the system. Thus, the system maintains high degree of availability to the service.
- ***Fault-tolerance*** Both hardware and software are faulty, thus the service availability cannot be guaranteed. Thus, it is highly required that the system

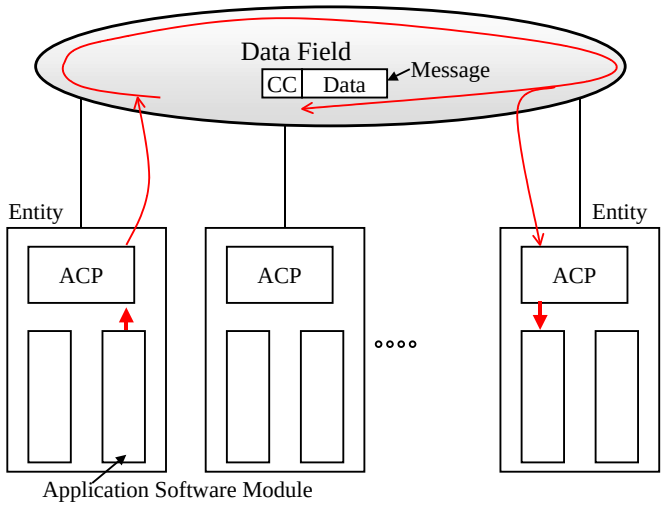

Figure 3.2. ADS Architecture

continues its operation even if a part of the system is faulty. In other words, if parts of the system are fail to work properly, the overall dynamics can continue to be the same or similar so that the system will not even come to a partial or complete halt.

- ***On-line Maintenance*** The system should carry out test procedure without stopping services to avoid failure.

3.4.2 ADS Architecture

The ADS architecture mainly relies on three main parts:

Atom Each entity has its local management system, called *Autonomous Control Processor* (ACP) and its application modules. The Atom registers the contents of data it can process to the ACP. Then, ACP filters the incoming data based in their contents specified in the message header.

Data Field It is the communication medium that allows the participated atoms in the system to communicate as shown in Figure 3.2. It serves as a common environment of coordination between the Atoms. The data circulates around the application modules in the Atom, and the data field in the atom is called

Atom Data Field (ADF). This results in excluding the mutual dependency among application modules.

Content Code In the data field, each data is specified by a unique identity based on its contents, called *Content Code* (CC). CC is the basis of the communication among atoms. Any atom sends its data on the data field by attaching content code CC to the data.

The communication on the data field relies on a data-driven mechanism. On the other hand, the conventional communication relies on address-driven mechanisms. In address-based protocols such as HTTP, each host sending a message must know in advance the address of the receiver. Contrary, the message sent on the data field is broadcasted to all atoms participated in it. Then message is then processed or discarded according to the receiver Atom. Each entity decides whether or not to accept a message. The data field architecture can satisfy the data communication requirements for various application domains. ADS shows a great success in transportation control systems and manufacturing control systems such as *Automated Train Control System* [54]. The main goal of this thesis is to present a proposition for large-scale information-dissemination system with dynamic natures.

3.5 Summary

This chapter introduced the research objectives including the problem formulation and the application domains. It clarified the application and system requirements of the decentralized information-dissemination systems. A brief description of our approach has been illustrated. In the following chapters, a new approach inspired from both the social community and the Autonomous Decentralized Systems is proposed to realize *High Assurance* to the information-dissemination systems.

4 AUTONOMOUS DECENTRALIZED COMMUNITY COMMUNICATION SYSTEM

Chapters 2 and 3 investigate that a new communication model for information-dissemination systems is required to address the issues of information-dissemination in large-scale systems with dynamic characteristics. Most communication models illustrated in the previous chapter failed to address the most important problems of the information-dissemination systems such as system scalability, service availability and continuity, and service customization. They show performance degradation with the increasing of the number of the participants in the information-dissemination service. Thus, they cannot achieve one of the essential components in modern high-assurance and large-scale information systems, the timeliness. End-users in most of these systems except P2P systems are passive actors (i.e. only utilize information). Moreover, these systems including P2P systems lacked to the multilateral cooperation among each other. In this thesis we have identified that the constructive cooperation among end-users assures the well-customized service's provision and utilization. Next paragraphs will briefly demonstrate *Why Cooperate?*, *Individual and Community*, and *Cooperative Information Systems*.

Why cooperate? Almost everything we use and depend on in our everyday life is produced and brought to us by the coordinated actions of many other people. Almost everything made by humans is produced cooperatively. We are also totally dependent on cooperation within us. We are composed of cooperative living processes. If the living processes that make up our bodies did not cooperate, we would not exist. Their cooperation is us. The key to the success of cooperation is that combinations of individuals whose activities are coordinated can do things better than individuals, and can do things that individuals cannot. Cooperation is also able to exploit the fact that combinations often have new features that their components do not. A further very general advantage of cooperation is that it can prevent the harmful effects of destructive

competition. The advantage of cooperation mean that a whole world of new adaptive opportunities is opened whenever living processes team up to form a cooperative *community*.

Individuals and community The word "*Community*" comes from the Latin term, *Communis*, meaning fellowship or common relations and feelings. In fact there are large number of community definitions from the social point of view. For example, *Hillery* [55] and *MacIver* [56] pointed out that the concept of community is based on the locality of human life, and is the counter of association, where people share a common goal. *Community* brings new importance to peoples. It makes them actors as well as spectators. The value of the individual is enhanced, if emphasizes each person has unique contributions. However, the value of individual differences does not lead to violent individualism. Instead, member-to-member relation raises the value of community. There is no such thing as a single individual. The value of a member grows together with its ties to all the other members on the system; thus, the member gains value insofar as he or she contributes to a community and reaps the community's benefits. There are levels of cooperation among members as follows. Cooperative members who work toward satisfying the same goal versus members that are self-motivated and try to maximize their own benefits. Community is an intermediary case where self-motivated members join together to work toward a common goal.

Cooperative Information Systems One of the greatest challenges for computer science is building computer systems that can work together. The integration of automated systems has always been a challenge, but as computers have become more sophisticated, the demands for coordination and cooperation have become more critical. Cooperative information system is a relatively young research area [57]. In distributed systems and cooperative computing, the word *cooperation* has a natural meaning. Cooperation presupposes users that have goals and can act upon them. Moreover, cooperation among users entails these users having some common goals and act forwards their fulfillment. Information systems can

be thought of as collections of human and computerized agents that can carry out actions such as requesting information. *Ishida, et al.* [58] pointed out that the *Community computing* is a research field for creating mechanisms and tools that support social interaction via computer networks such as the Internet and mobile computing systems.

Inspired from the cooperative models in the social communities, this thesis will illustrate a cooperative information system that is cooperatively able to disseminate the information efficiently.

The next section illustrates the leading concept of the *Autonomous Decentralized Community Communication System* (ADCCS).

4.1 ADCCS: Concept

In large-scale and extreme dynamism of the operating environment, it is difficult to disseminate the information efficiently. Therefore it is necessary to maintain the locality and autonomy of the information-dissemination. To achieve so, based on the spirit of cooperation in the social community and the autonomous decentralized system concept [11,59], this thesis proposes the concept of *Autonomous Decentralized Community Communication System* (ADCCS). In addition, the basis of the ADCCS concept is to provide the information to specific users at specific place and specific time. In contrast, current information systems provide the information to anyone, anywhere and anytime. In this thesis, we define *Autonomous Community* as a communication environment for a coherent group of autonomous members having common demands and interests not only in specific location, but also at specific time. The community members are autonomous, cooperative and active actors and they mutually cooperate to enhance their objectives. Each community member plays a dual role as information sender and receiver at the same time. Furthermore, each message from a participant is meaningful to all members and at the same time every member is typically interested in data from all senders in the community.

The ADCCS concept is realized if the following aspects are satisfied:

- ***Autonomous Controllability*** Each member is able to manage himself, carry out its functions and continue its operations (e.g. routing messages) regardless to the other members.

- ***Autonomous Coordinability*** If a new member joined, leaved, his node failed, his software repaired or modified, the other members are able to coordinate among themselves to satisfy their own goals and to disseminate messages timely.

- ***Mutual Cooperation*** The constructive cooperation among members is the vehicle of the successful community. Cooperation is not merely coordination. Through the mutual cooperation, the community members attain the useful information and achieve their own objectives (i.e. information-dissemination) with low efforts (i.e. network traffic). In other word, the community benefits have to cover its liability in order to be attractive for both its members and the other users have intention to join it.

These aspects assert that the community structure dynamically changes, and each member has autonomy for interactive communication and information processing. Furthermore, the community members cooperate for utilization and provision of the community services and information-dissemination and sharing under the evolving situations.

The ADCCS concept can be realized with autonomous controllability, autonomous coordinability and mutual cooperation. Moreover, each member is required to satisfy the following conditions:

- ***Equality*** Each member must be equal and able to handle his objectives without being directed by or giving the directions to the others. In other words, there is no master-salve relation among the community members. All the community members have same rights and same responsibilities (e.g. routing messages) in the community. In fact the fairness among the community members has to be satisfied because unfairness provokes them to leave the community.

- ***Locality*** Each member handles his objectives and coordinates with others based only on his local information. For example, each member based on his local

information takes the decision whether to route the received messages to his neighbors or not. Furthermore, he also takes the decision to continue in or to leave such community based on his own local information (preferences).

- ***Self-Containment*** Each community member is self-contained in managing his objectives whereas he coordinates with the others.

- ***Synergy*** Cooperation among people, and organizations would be the main characteristic of this era. Prompted from the dictum that *one for all and all for one*, each community member must cooperate with the other community members. Furthermore, the community's success is subjected to the mutual and productive cooperation among its members. Owing to the synergy among community members, a continual and timely information delivery can be implemented.

ADCCS is a promising concept for information-dissemination services operating at the edge of the network. It realizes the large-scale information-dissemination system that successfully able to carry out and enhance the objectives of the community members (e.g. timely information-dissemination and sharing) in a very dynamic environment. It guarantees the constructive cooperation among the community members with a very high degree of autonomy among them. We have developed system architecture, ADCCS that fosters the concept of the autonomous decentralized community information-dissemination system.

4.2 ADCCS: Architecture

4.2.1 System Architecture

Applications are moving away from tightly-coupled systems towards systems of loosely-coupled, dynamically bound components. This trend requires to integrate autonomous, heterogenous components (nodes) into complex systems by means of discovering and exchanging information. The architecture of the ADCCS can be developed as follows. The nodes of the community members will be connected on a

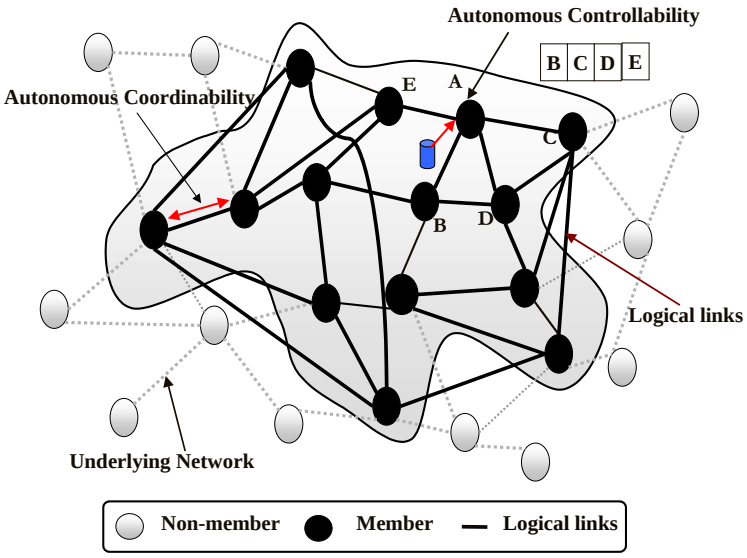

Figure 4.1. Autonomous Decentralized Community Communication System Architecture

bilateral basis. The bilateral logical contact between two community nodes will occur considering that the users of these nodes have the *same interests and demands*, *at specific time and location*. It can be defined that in bilateral contacts, community nodes connect each other and share information. The community network is a self-organized and self-adaptable logical topology. It is a set V of nodes of the end-users that considers the symmetric connectivity and the existence of loops. Each node keeps track of its immediate neighbors in a table. The immediate neighbors set of the community node x is defined as the set of nodes

$$INS_x = \{y; x, y \in V, h(x, y) = 1\} \tag{4.1}$$

Where *h(x, y)* is the number of logical hops between nodes x and y. For example, Figure 4.1 shows that the immediate neighbors' set of node A is INS_A= {B, C, D, E}. Each node knows its neighbor nodes and shares this knowledge with other nodes to form a loosely connected large number of nodes. In Figure 4.1 the solid bold lines represent the logical links among nodes. Each node judges autonomously whether to join/leave the community network by creating/destroying its logical links with its neighbor members based on its user's preferences. The community's boundaries

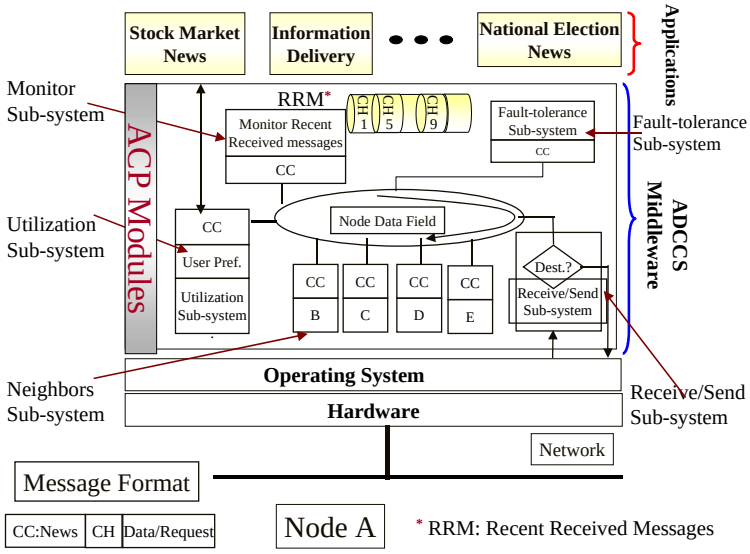

Figure 4.2. ADCCS: Node Architecture

change with the dynamic change of its members' requirements. ADCCS architecture has no central server whatsoever, as can be seen in Figure 4.1.

4.2.2 Node Architecture

The node is an execution platform that is in charge of autonomous membership management, service customization and information-dissemination communication service. It contains five *Autonomous Control Processor*(ACP) sub-systems (modules) and *Node Data Field* (NDF) that assures communication between connected modules as shown on Figure 4.2 and as follows. For the communication among modules a specific message format is designed and will be illustrated in the following section.

- ***Utilization Module*** It monitors the changes of the user's preferences. A Java utility such as, Java Developer *Almanec* 1.4, can be used to form and monitor the user's preferences and then generates XML data. As soon as it detects any change of the preferences of its user, it generates an appropriate Content Code (CC) and then registers it into the ACP. Then, this node will receive any message holding the CC that is propagated in the community.

- ***Neighbors Module*** It is responsible for the membership management such as node joining process, leaving process and fault detection and recovery processes. These processes require only local changes in the community network. Thus, this module guarantees a low complexity on-line expansion and fault-tolerance using *fault-tolerance module* to the community system as will be shown in next chapters. This allows the ADCCS to scale extremely large groups and to deal with the rapid changes in the group membership efficiently. Furthermore, *Neighbors Module* not only considers the node-node latency when nodes join or leave the community, but also monitors the changes of the node-node latency to adapt these changes, as will be shown in Chapter 6.

- ***Monitor Module*** To avoid the congestion that may happen if some of the community nodes send simultaneously identical messages, each node keeps a short memory (CH) of the recently routed messages in the *Recent Received Messages* list (RRM). This module monitors the recent received messages and judges autonomously to forward only one copy of the received messages to the other neighboring nodes. Moreover, it autonomously takes the decision to whether keep or delete the short memory of the received message, based on the frequency of receiving such message. This enables community to avoid network congestion and to distribute the load more evenly overall nodes.

- ***Receive/Send Module*** It is responsible to send, receive and then route the received message to neighbor nodes except the neighbor node that delivered it. In that respect, receive/send module coordinates with the neighbors module and monitor module to identify the destinations of the message to be routed.

- ***Fault-tolerance Module*** It is responsible to detect the failure of neighbors by exchanging keep alive messages with neighbor nodes. It then recovers this failure and maintains the node connectivity with other nodes.

The communication among these modules is done through the NDF as shown on Figure 4.2. However, the processing steps for filtering and forwarding the received messages by each node, according its CC, is shown on Figure 4.3 and as follows.

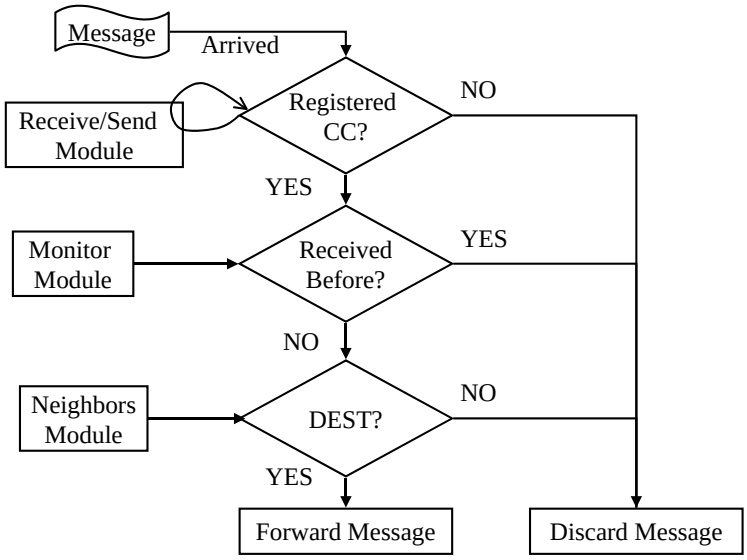

Figure 4.3. Filtering and Forwarding of received Messages

When node A received a message, it calls the *receive/send* module to check the CC and then takes a decision to accept such message or not. If CC was registered then A calls the *monitor module* that checks if the message was received before or not. If the message does not received before then A calls the *neighbors module*. The *neighbors module* determines the neighbor nodes that were registered the CC. Then A forwards the message to these neighbor nodes.

Node's Autonomy This node architecture supports two system properties, the autonomy and locality of the node as follows. Each node recognizes autonomously a member from a non-member and cooperatively forwards the community information to only its neighbor members. Community node does not forward the community information/request out of the community. Moreover, each node "*think globally and act locally*" by taking a decision autonomously based on its local information to store the relevant received information. The decision is taken not only according to the node situation (e.g. limited resources) and the importance of the offered information, but also according to the requirements of the other members. Each community node keeps a short memory of the recently routed messages to avoid the congestion in the community network.

Each node autonomously cooperates with others for locating, and/or providing the information in the community. If any node leaves, fails and joins the community, the other community members still can coordinate their individual objectives among themselves. Consequently, each member is able to operate in a coordinated fashion.

Thus, both the ADCCS architecture and its node architecture fulfill the following important four features. We believe that the system, which realizes the follows features, will assure an efficient real time and cooperative content delivery service, regardless of the extreme dynamism of its operating environment.

- ***Loosely-Connected*** Each node keeps a short memory about its neighbors as shown in Equation 4.1 and Figure 4.2. Each node shares the neighborhood information with its neighbors. Thus, a loosely connected mass of nodes can be constructed.

- ***Loosely-Control*** Figure 4.1 shows that no node is responsible for either membership management or central service provision.

- ***Self-organizing*** Each node judges autonomously to join or leave the community based on its user demands and situation.

- ***Self-adapting*** The system must be adaptable to meet the constantly and dynamically changes of the members' interests and demands. In addition, it should cope with the dynamic network changes such as member-member latency as will be shown in chapter 6.

The ADCCS architecture is a fully decentralized model, where each participated node has equal responsibilities, and does not rely on any central authority to organize the network. Thus, it does not load up any single node excessively and enables the development of the high assurance information-dissemination systems with adaptability, flexibility and high availability characteristics.

4.3 ADCCS: Community Overlay Network

4.3.1 Overlay Networks

The Internet itself was developed as an overlay on the telephone network. Several Internet overlays have been designed in the past for various purposes, easing IP multicast deployment using *MBone* [40] and providing IPv6 connectivity using the *6-Bone*. In the broadest sense, the *Overlay Network* can be defined as a set of tunnels formed among network edges to support a common packet processing function other than the ones supported in the conventional network.

One of the important goals of overlay networks is to implement multicast services. In deciding whether to implement the multicast overlay networks at the IP layer (underlying layer) or at end-nodes, there are two conflicting considerations that we need to overcome. According to the end-to-end arguments [60], the functionality should be pushed to higher layers if possible; unless implementing it at the lower layer can achieve large performance benefits that outweigh the cost of additional complexity at the lower layer. The IP Multicast requires routers to maintain per group state that not only violates the *stateless architectural principle* of the Internet original design, but also introduces high complexity and serious scaling constraints at the IP layer. Even with *Moore's law's* prediction in 1965 [61] that the processing speeds double every 18 Months, it still falls short of the speed that bandwidth capacity is growing at. So not only it is not cost-effective to add new software to router platforms in order to meet new application demands, the limited processing power at core routers also leaves little room for additional processing functions.

4.3.2 Why Overlay Network?

The most cited problems preventing *Internet Service Providers* (ISPs) from deploying a multicast-enabled network (Mbone) include: the complexity of most multicast routing protocols and their implementations; the lack of a scalable inter-domain routing protocol; and the lack of support in access control and transport services. In addition, the Mbone topology is not optimized and grows randomly. It also tends to

Table 4.1
Multicast Overlay Network: Underlying Layer vs Application Layer

	Underlying Layer	Application layer
Global Address	Yes (IP-Multicast Address)	No
Internet Stateless Architectural Principle	No	Yes
Scalability	Low	High
Congestion and Flow Control	Difficult	Easy
End-to-End Latencies	Low	High
Bandwidth Usage	Low	High

trigger miss-configurations and consequently to disrupt services. Despite these difficulties, it is undeniable that multicast is an efficient transmission mechanism to reduce network load for very large groups and to save transmission time and bandwidth of data sources even in small multicast groups as shown in Table 4.1. Moreover, there is a deficit of experiences with additional mechanisms such as congestion and flow control on top of the IP multicast overlays, so ISPs are wary of enabling multicasting service at the network layer. Thus, it is highly required to build overlay networks that enable the efficient multicast at the application layer without any support from routers.

The basic idea of the multicast overlay at the application layer as shown in Figure 4.4-(b) is that data packets are replicated at the end-nodes. Contrary, in the network layer multicast shown in Figure 4.4-(a), data packets are replicated at the *routers*. Logically, end-nodes form an overlay network to construct and maintain an efficient overlay for information-dissemination. In addition, the prime advantage of the overlay network architecture in application layer is that it does not require universal network support. This enables faster deployment of desired network functions and adds flexi-

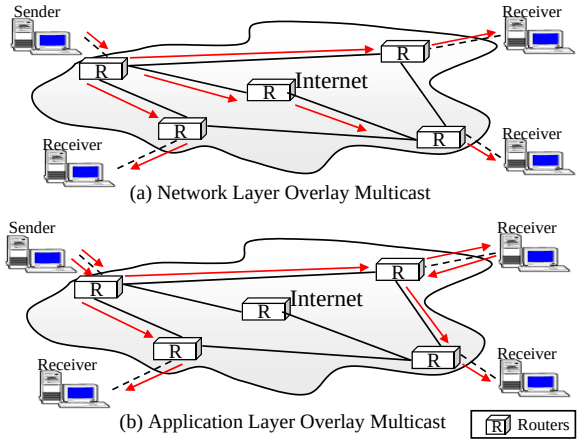

Figure 4.4. Network-layer and application layer overlay multicast.

bility to the service infrastructure. In addition, it allows the co-existence of multiple overlay networks each supporting a different set of service functions.

4.3.3 Community Overlay Network (CON)

An *autonomous Community Overlay Network* (CON) is one type of overlay networks that are formed over the application layer. The nodes comprising a CON reside in a variety of routing domains and cooperate with each other to forward data. CON provides information-dissemination services to end-users (*community members*) having same demands at specific time and location. It performs the multilateral and content-code $1 \rightarrow N$ community communication that will be illustrated in next section on top of the general Internet unicast infrastructure. Thus, CON will have a continuing role in preserving service flexibility and customization over the Internet. Despite the advantages of building the multicast overlay network at application, layer there are some disadvantages. Such that it is impossible to completely prevent multiple overlay edges from traversing the same physical link and thus same data will pass multiple of times over the same link. Moreover, communication between end-nodes involves traversing other end-nodes, potentially increasing latency. A comparison between multicast overlay networks over underlying layer and application layer is tabulated in Table 4.1. The questions then are: How to efficiently construct the

community overlay network for efficient information-dissemination?. Second, How to construct the community overlay network at application layer with efficient that converges to the high performance of the overlay networks at underlying layer?. This thesis answers these questions in the following sections. The main goal of CON is to enable community nodes to communicate with each other (i) in face of problems at the underlying Internet paths connecting them; (ii) in high performance even without underlying network supports. The following section discusses the existing discovering techniques to discover the overlay network.

Discovering CON

When an end-user wants to join a CON, he/she has to discover at least one community node *X*. The end-user's node can either use information from an out-of-band bootstrap mechanism similar to Narada [51] and CAN [62], employ network broadcasting and discovery techniques such as IP multicast or employ network random walk. In bootstrap technique, the new node look up the CON domain name in DNS to retrieve a bootstrap node's IP address. This node maintains a partial list of CON nodes it believes that they are currently in the community. Thus, bootstrap techniques prone to scalability problems and the bootstrap nodes represent single point of failures. IP multicast technique is scalable but leads to high network traffics. We believe that the random walk is efficient technique but may take long time to find a community node. In this thesis we do not address this issue.

CON: Design Issues

A community service that should be deployed and operated in the CON faces the problem of constructing a least-cost network that will meet the needs of the community members. Then, several questions in order to construct the CON should be answered. *How many connections should be assigned to each node? How do nodes join efficiently to the CON for efficient multicast under two assumptions: Homogeneous node-to-node latency and Heterogeneous node-to-node latency? How do nodes leave the CON with low complexity? How does CON achieve service continuity re-*

gardless of the failure of its nodes? How can CON be self-organized and self-adaptable to the network changes?. Each of these questions leads to a subproblem in the CON design problem. The coupling and interaction of these problems results in a complex network design. This dissertation studies these problems and provides efficient solutions. The thesis proposes an autonomous decentralized community overlay network construction/maintenance technologies in Chapter 6.

4.4 Summary

Autonomous Decentralized Community Communication is the concept proposed to allow users have same demands and situations, under large-scale and changing environment, to cooperate and share the information-dissemination penalties and benefits as well. This goal is realized through the *Autonomous Decentralized Community System Architecture*. This architecture is characterized by loosely connected, loosely controlled, self-organized and self-adaptable properties. In addition, this chapter introduced the community overlay network concept and structures for providing customized, flexible and efficient information-dissemination services. It argued that this overlay network is well-suited for the emerging trends in the Internet architecture.

5 AUTONOMOUS DECENTRALIZED COMMUNITY COMMUNICATION TECHNOLOGY

The conventional communication, typically through Web browsers, has been built on the one-to-one communication protocol. In one-to-one, data travels between two users, e.g., e-mail, e-talk. This protocol misspends the network bandwidth and makes the real time services unresponsive. Caching most popular web pages on the proxy server reduces the network bandwidth consumption and the access latency for the users. However, the web cache techniques have some disadvantages as follows. First, a single proxy server is a single point of failure. Second, the limited number of users per proxy manifests bottleneck affects. Third, data does not updated automatically. Finally, cache misses increase in the latency (i.e. extra proxy processing). In the conventional one-to-many group communication the message travels primarily from a server to multiple users, e.g., software distribution. For very large groups (thousands of members) or very dynamic multicast groups (frequent joins and leaves), having a single group controller might not scale well. Currently, there is no design for the application-level multicast protocol that scales to thousands of members. For example, Overcast [63] builds a mesh per group containing all the group members, and then constructs a spanning tree for each source to multicast information. The mesh creation algorithm assumes that all group members know one another and therefore, does not scale to large groups. Bayeux [64] builds a multicast tree per group. Each request to join a group is routed to a node acting as the root. This root keeps a list of all the group members. All group management traffic must go through that root. It generates more traffic for handling a very dynamic group membership. Bayeux ameliorates these problems by splitting the root into several replicas and partitioning members across them. But this only improves scalability by a small factor. Chapter 7 will show comparative analysis with other application level multicast systems.

This thesis proposes the *Autonomous Decentralized Community Communication Technique*. The main ideas behind this communication technique are: *content-code*

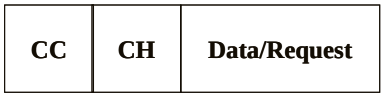

Figure 5.1. Community communication message format

communication (community service-based) and *multilateral communication*. It is proposed to realize a flexible, a timely and a productive cooperation among members. It is illustrated in the following subsections.

5.1 Service-oriented and Multilateral Community Communication

Conventional communication techniques use the destination address (e.g. unicast address, multicast address) to send data. In constantly and rapidly changing operating environment likes ADCCS, the status of nodes (i.e. end-users are frequently joins and leave) and the stability of connections are unpredictable. Therefore, these conventional communication techniques cannot guarantee the high assurance of the information-dissemination service. Instead of the destination address, this thesis encapsulates a service identifier and detailed contents of the requested/disseminated service to assure a flexible communication among members.

The first main idea behind the autonomous decentralized community communication technique is the separation of the logical community service identifier from the physical node address. In this communication technique, the sender does not specify the destination address but only sends the content/request with its interest *Content Code* (CC) to its neighbor nodes. CC is assigned on a type of the community service basis and enables a service to act as a logical node appropriate for the community service. Figure 5.1 shows the community communication message format. CC is uniquely defined with respect to the common interest of the community members (e.g. politic, news, etc.). The information content is further specified by its *Characteristic Code* (CH). For example, CC is "*Iraq news*" and CH_i is the "*prisoners abuse scandal*" at the Abu Ghraib prison in *Baghdad, Iraq* on first of May 2004. The CH is the hash of the message content. It is uniquely specified with respect to the message

content (e.g. data or request). It can be computed by the collision resistance hash function (e.g. SHA-1 [65]) that ensures a uniform distribution of CH.

The second main idea behind the autonomous decentralized community communication technique is multilateral communication for timely and productive cooperation. Multilateral communication occurs among the community members that are already networked on a bilateral basis. All members communicate productively for satisfaction of all the community members, as follows.

5.2 $1 \rightarrow N$ Community Communication

To disseminate information within the community, this thesis presents a communication technology so called $1 \rightarrow N$ *community communication*. A scenario of the $1 \rightarrow N$ community communication is described as follows. The community node asynchronously sends a message to its N neighbor nodes. Then, each node from these N nodes forwards the same message to another N neighbor nodes in the next layer, except the node that delivered the incoming message, and so on gradually until all the community nodes have received the message. This technique handles as the model like *Viral propagation*. Each community node executes the following instance of pseudo-code.

```
Listen (CC) {
// Installed in the Receive/Send module
// CC is the community communication content code
// The self-variable means the calling node id.
        do {
              if (received_message.cc == CC) then
                 self.Forward_message(received_message);
              } while(1);
        }
Forward_message(m) {
// m is the message structure contains the CC, CH and data
// The self-variable means the calling node id.
```

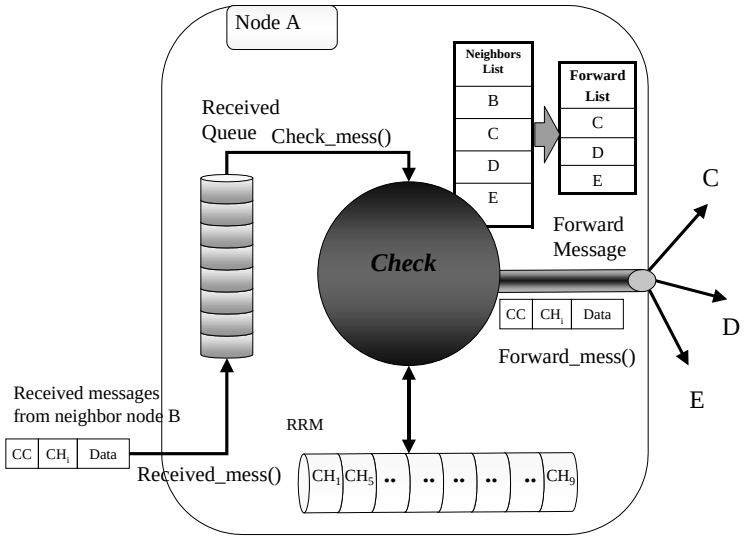

Figure 5.2. $1 \rightarrow N$ Communication: Node work flow diagram

If (self.Not_received(m)) then
  for k ∈ self.Neighbors in asynchronously do
    self.SendTo(m, k);
}

Figure 5.2 shows the $1 \rightarrow N$ communication flow diagram at a node. Each node has two queues one to keep received messages and the second to keep forwarding messages. In addition, it has two lists. The first list contains the neighbors of the node. The second is called *Recently Routed Message* (RRM) list. It contains the characteristic codes of recently forwarded messages out of the node. Each node has four routines: *Listen(), Received_mess(), Forward_mess()* and *Check_mess()*. The first three routines are installed in the *Receive/Send module* while the last one is installed in the *Monitor module*. As soon as a node joins the community, it executes the *Listen()* routine and listens to all messages propagated in the community network that hold the community content code (CC). The source of a message calls the *Forward_mess()* routine that sends the message to all its neighbors. When a node receives a message, it calls the *Forward_mess()* routine. The Check_mess() routine checks the RRM if the received message has received before or not, sets the value of the *Not_received*

variable and then adds its characterized code to RRM list if not. The Check_mess() creates a volatile list that contains the neighbors nodes except the node that delivered this message. The node running the Received_mess() routine listens to all messages has CC content code and then adds the received message to the received queue. For example, as shown in Figure 5.2 node A has received a message from node B with content code CC and characterized code CH_i. Node A checks the message and then takes the decision to forward this message to all neighbors nodes except node B. Thus, we can avoid the network congestion that may happen if some of the community nodes simultaneously send identical messages.

The $1 \rightarrow N$ communication technique does not rely on any central controller. Each community node has its own local information and communicates only with specified number (N) neighbor nodes. There is no global information such as IP multicast group address [4] or multicast service nodes [66, 67].

5.3 Community Communication Protocols

The community communication technique has two protocols: Hybrid pull/push based and request/reply-all based.

Hybrid pull/push based protocol When one of the community members has new information, he/she publishes it to all the community members using $1 \rightarrow N$. A typical application is news information sharing among users having same interests and demanding to know specific news at specific time and or location. This thesis addresses only non-multimedia contents (e.g. news) having moderate size. The hybrid pull/push based protocol offers an effective solution to the flash crowd problem as shown in Figure 5.3. The solution scenario is as follows. When community member S downloads interesting content for the community from the server (e.g. news server or government server), he/she publishes it to all community members, thereby relieving the server of this task, alleviating a load on the server and distributing the load among the community nodes. When the number of nodes increased sharply, the load at each node is increased

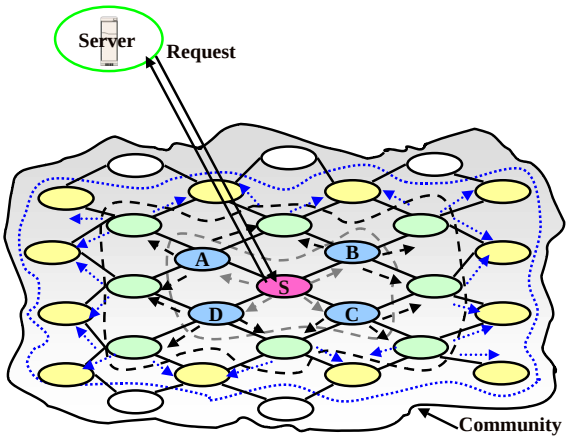

Figure 5.3. Hybrid Pull/push based protocol

slightly. The hybrid pull/push protocol represents a scalable solution for large-scale information-dissemination systems.

Request/reply-all based protocol When a community member wants to locate information, he/she sends a request message. The the other community members then cooperate to locate the requested information. When any community node receives the requested message, it processes the request. If a node finds no results, it forwards the request to its neighbor nodes using $1 \rightarrow N$. Otherwise, the node sends its results, such as pointers to the information or the actual content depending on its size. Then this node sends a reply message not only to the requesting node, but also to all community members. Figure 5.4 shows the message flow when the community node S sends a request (solid arrows) to its neighbors and node R replies (dotted arrows) to all community members by the required information *I*. The reply-all protocol affords the other community members to send the same request. Consequently, all the community members expand their knowledge, experiences and implicitly get to know new services without issuing specific requests, in which individually they cannot get to know. This is in keeping with the community's goal of multilateral benefits. The reply-all protocol also decreases the per-node traffic by avoiding multiple requests for the same content.

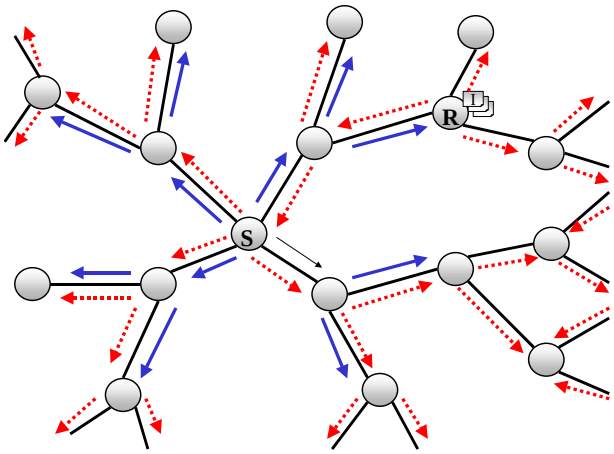

Figure 5.4. Messages flow in request/replay-all based protocol

The originality of our proposed communication technique does not come only from the service-oriented (content-based) communication, but also from the reply-all that satisfies the multilateral benefits. Contrary to peer-to-peer (P2P) communication techniques, in $1 \rightarrow N$ community communication all members cooperate on a single request to the benefit of all members. In P2P, peers cooperate only for the unilateral benefit of the requesting member. The comparisons between the ADCCS and the conventional information-dissemination systems based on client/server and peer-peer structures are tabulated in Table 5.1. From this table we conclude that the community communication is: service-based, cooperative and multilateral benefits communication. The service-based communication is not merely content-based communication because service-based communication considers not only the users' demands, but also the users' situations.

5.4 Performance Verification

The objective of the analysis in this section is to show the effectiveness of the proposed communication technique. Assume the number of the community nodes is M and each node has k neighbors. The information is broadcasted in a tree as follows. The source node sends asynchronously a message to each one of k neighbors (children) and then each neighbor forwards asynchronously the same message to another *k-1*

Table 5.1
Communication comparison

		Client/Server	Peer-to-Peer	ADCCS
Communication	Model	Address-based	Address-based & Content-based	Service-based
	Request	One-to-One	One-to-Many	$1 \rightarrow N$
	Reply	One-to-One	One-to-One	$1 \rightarrow N$
Characteristics	Benefit	Unilateral	Unilateral	Multilateral
	Users	Passive	Active	Active
	Load	Servers Congestion	Peers Congestion	No-congestion (Fairness)

neighbors nodes in the next layer and so on, until all community nodes receive this message. Thus, the number of the community nodes taking part in the broadcasting tree can be written as follows.

$$M \leq 1 + \sum_{i=0}^{L-1} k(k-1)^i \tag{5.1}$$

Where L is the number of layers or depth of the tree. Then the number of layers can be calculated as follows.

$$L \simeq \lceil \frac{log(\frac{(M-1)*(k-2)}{k} + 1)}{\log(k-1)} \rceil \tag{5.2}$$

Under the assumption that the communication cost between each node is one unit of time then, the transmission time τ to send a message from one member to all other members is bounded by $O(N * \log_N(M))$, where $N = k - 1$. Consequently, we can drive the optimal $1 \rightarrow N$ community communication as follows.

$$\frac{d\tau}{dN} = \frac{d}{dN}(N * \log_N(M)) = \log(M) * (\frac{\log(N) - 1/ln(10)}{(\log(N))^2}) \tag{5.3}$$

From Equation 5.3, we conclude that $\frac{d\tau}{dN} = 0$, $\frac{d^2\tau}{dN} > 0 \Rightarrow N \simeq 3$ and τ is concave up. For any number of nodes M, the $1 \rightarrow 3$ community communication technology is the optimal. Similarly, Figure 5.5 shows that ever-increasing the number of nodes

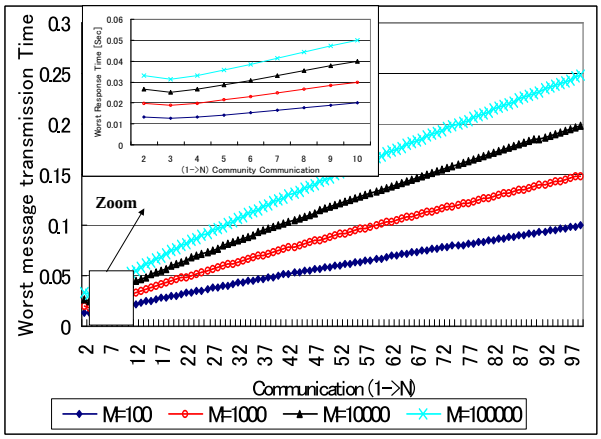

Figure 5.5. Optimal $1 \rightarrow N$ Community communication

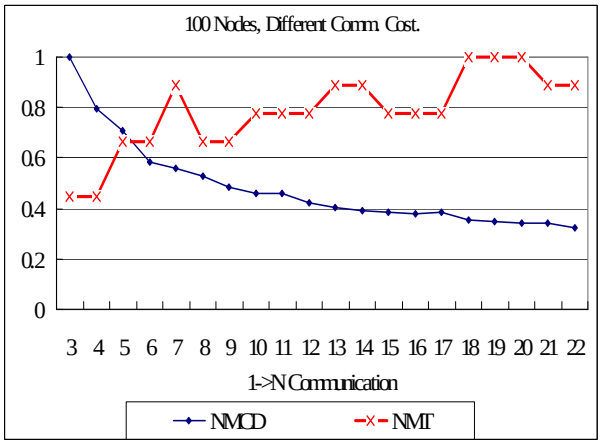

Figure 5.6. Trade-off: Communication Delay & Network traffic/link

the optimal communication is $1 \rightarrow 3$ under the assumption that the communication cost between each node is one unit of time.

We study the $1 \rightarrow N$ Community communication under the assumption that the communication cost between each node is different. Figure 5.6 shows the results of the experiment that investigates the effect of the variation of $1 \rightarrow N$ on its performance. In this experiment two metrics are used: *normalized mean communication delay* (NMCD) and the *normalized mean network traffic per link* (NMT) (Mean number of identical messages per link). Figure 5.6 shows that when $1 \rightarrow N$ increases, the network diameter decreases and network traffic per link increases. Consequently,

it verifies that the trade-off between the communication delay and the network traffic exists. Then, several questions are raised and should be answered. Such as, *how many connections should be assigned to each node?* for efficient communication and low network traffic. This thesis will answer these questions in next chapter.

5.5 Summary

This chapter demonstrated the service-oriented and multilateral community communication. It separates of the logical community service identifier from the physical node address for flexible communication. In contrast to the conventional communication system, all community members cooperate on a single request to the benefit of all members. Moreover, in this chapter the $1 \rightarrow N$ community communication is proposed to permit cooperative users to disseminate information timely. The evaluation showed that the $1 \rightarrow 3$ community communication is optimal when the communication delay between each node is one unit of time. However, under different communication cost, the experiment results verified the existence of the trade-off between the communication delay and the network traffic. The prominent strength of community communication techniques should account for the nodes heterogeneity, autonomy, relying on self-inspection and adaptation to exploit the differences in the nodes' characteristics, behavior, and incentives.

6 AUTONOMOUS DECENTRALIZED COMMUNITY OVERLAY NETWORK CONSTRUCTION TECHNOLOGIES

6.1 Single Community Construction/Maintenance Techniques

The topology design problem is how to place network nodes - routers in the conventional context and end-nodes (community members) in the CON. Moreover, it is required to decide how to connect the community nodes with respect to some network diameter and node-to-node latency constraints. This section illustrates the autonomous decentralized construction/maintenance technique of the CON as *single community* under the assumption that node-to-node latency is *homogenous*. We call this single community overlay network *Flat-CON*.

6.1.1 Goals and Requirements

Flat-CON is symmetric in the sense that each node in the network will have identical capabilities and duties on the network. In addition, the node to node communication cost is homogenous (i.e. latency from any end-node to the others is same). This excludes the existence of central servers that might be involved in organizing the network. Our first goal is to construct Flat-CON with short diameter that can support multicast efficiently. In addition, we should avoid hotspots in Flat-CON by distributing the network traffic evenly among community nodes during the broadcast. Hotspots push members to give up from the community system, as a result the availability of the system is gradually decreased and the system becomes distasteful for its users. That is required fairness among community nodes. The fairness objective dictates that all nodes must have the same degree as possible. The second goal is to make the construction and maintenance of Flat-CON be highly scalable. In other words, the complexities for joining, leaving and fault-tolerating nodes should be as low as possible. Finally, Flat-CON topology has to provide redundancy. Node failures must not lead to community network disconnecting or severely hampering broadcast

properties. Next subsection presents our proposed construction technique achieving these goals for the community topology.

6.1.2 Organizing Flat-CON: Regular Graph

Any node can join and leave the community network at any time and through a node already exist in the community network. It has been observed that the node degree distribution of the Internet with uncontrolled evolution mimics a *power-law* [68]. It occurs when no scheme is imposed on the way nodes join and leave, thus the network is likely to grow to become exponential network and may lead to some hotspots in the community network. For example, some peer-to-peer systems do not specify how many connections a peer may initiate, accept, or simultaneously maintain. Consequently some peers may have high load than others. In that respect, this thesis proposed an autonomous decentralized community construction technique for making the potential hotspots very unlikely. The thesis constructs Flat-CON as *regular-graph*[1] for three arguments as follows. First, regular graphs are chosen because it is required that all nodes having the same degree. Second, we construct the community network as an intermediate of a completely ordered regular network and a fully random network for achieving two interesting features: high clustering i.e., there is a high density of connections between nearby nodes, which is a characteristic of the regular topologies, and short network diameter. Finally, the community network composed of *Hamilton cycles*[2] having the advantage that joining or leaving processes will require only local changes in the graph. A graph that is constructed from disjoint *Hamilton cycles* is a connected graph [70]. Thus, as long as CON maintains Hamilton cycles then CON is a non partitionable network.

Flat-CON construction polices the nodes joining and leaving the community network and organizes them in a *2d-regular graph* ,$G = (V, E)$, such that V is the set of nodes with labels [M] =1, 2,..., M and E is the set of edges. Due to frequent join and leave, E and V are changing with time. The graph G is composed of independent d

[1]A graph in which all nodes have the same degree.

[2]A cycle that connect all nodes in a graph and visit each node only once [69].

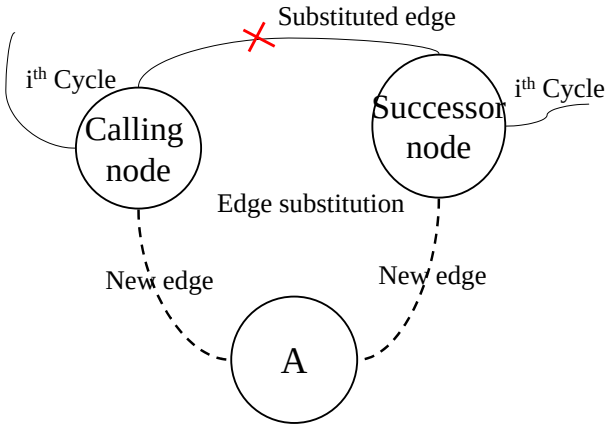

Figure 6.1. Add node A in the i-th Hamilton cycle.

edge disjoint *Hamilton cycles* [69]. Assume H_n denotes the set of all *Hamilton cycles* C_i on set [n], and $C_1, C_2, ..., C_d \in H_n$ for $n \geq 3$, E = $(C_1, C_2, ..., C_d)$. Let $H_{n,2d}$ be the set of all 2d-regular multi-graph constructed by some $C_1, C_2, ..., C_d \in H_n$. To construct Flat-CON with achieving the goals mentioned before, we shall construct 2d-regular multi-graphs in $H_{n,2d}$ for $d \geq 2$ and $n \geq 3$. Of course if $n \leq 2 * d + 1$ we might use a complete graph instead of H-graphs. Then each node has *2d* neighbors (node connectivity). Those neighbors are labeled as $r_p^{(1)}$, $r_s^{(1)}$, $r_p^{(2)}$, $r_s^{(2)}$, ..., $r_p^{(d)}$, $r_s^{(d)}$. For each i, $r_p^{(i)}$ denotes the neighbor node's predecessor and $r_s^{(i)}$ denotes the neighbor node's successor on the i-th Hamilton cycle. The advantage of using the Hamilton cycles is that nodes joining or leaving will require only local changes in the community network as will be described as follows.

6.1.3 Online Expansion and Construction: Join Process

As soon as the end-user's node H discovered a community node C, it sends a join request to node C. Node C is responsible to find *2d* neighbors for node H to connect. If some joining nodes connect to the neighbors of the same node C, then the community network diameter increases linearly (e.g. completely ordered regular graph) [26].

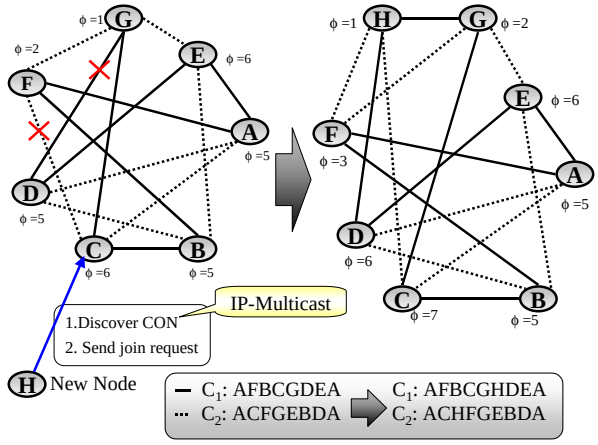

Figure 6.2. Joining Process.

In order to avoid such situation, each node autonomously determines how many new nodes have connected it within period t; we call this node join-rate $\phi(t)$. Node C broadcasts a join-request to all community nodes within $O(log_{2d}\ M)$ layers. Each node autonomously decides based on its join rate ϕ(t) whether to reply by the message "Ok to join" or not. Node C receives some "*OK to join*" messages and autonomously selects d nodes in different Hamilton cycles. The selected d nodes then call the following add() routine to add joining node H in each i-th Hamilton cycle.

Add (H, i) {

Successor_node $\longleftarrow$(Calling_node $\Longrightarrow r_s^{(i)}$);

Edge(Calling_node, H, i);

Edge (H, Successor_node, i); }

Edge (B, C, i) {

(B $\Longrightarrow r_s^{(i)}$) $\longleftarrow$ C ;

(C $\Longrightarrow r_p^{(i)}$) $\longleftarrow$ B ; }

The expression $H \Longrightarrow$ Var means that we seek the value of Var from node H, the expression ($H \Longrightarrow$ Var) $\longleftarrow$ y means that we set the variable Var of node H to value y. The add routine inserts the joining node between the calling node and the

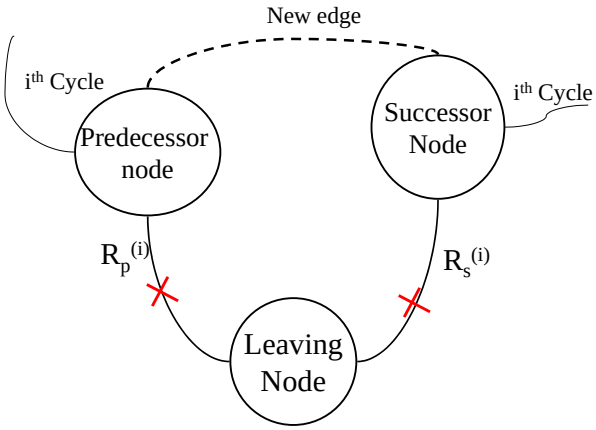

Figure 6.3. Leave node from the i-th cycle.

successor of the calling node in the i-th Hamilton cycle. The routine substitutes the edge between calling node and its successor by two edges, one between calling node and joining node and the other one between joining node and successor of the calling node as shown in Figure 6.1. The Edge(B, C ,i) routine makes C the successor of B and B the predecessor of C. It thus creates the communication session between nodes B and C. Obviously, the join process requires only local changes in the community network. Figure 6.2 shows an example for the joining process of new node H. Assume H discovers node C in community network as shown in section 4.3.3. Then node H sends join request to node C that forwards this request to all members in the community. Each node replies to node C based on its ϕ(t). Then, node C selects two nodes for example, G and F, where links (G, D) and (F, C) are from different cycles. Then, both G and F call *add()* routine. For clarity, we show only two Hamilton cycles C_1 and C_2. Node G has a successor node D in cycle 1 while, node F has a successor node G in cycle 2.

6.1.4 Maintenance and Fault-tolerance

This section illustrates the leaving and fault-tolerance processes as follows.

Leaving Process When a member leaves the community, it notifies its neighbors and calls the following leave routine to leave from each Hamilton cycle.

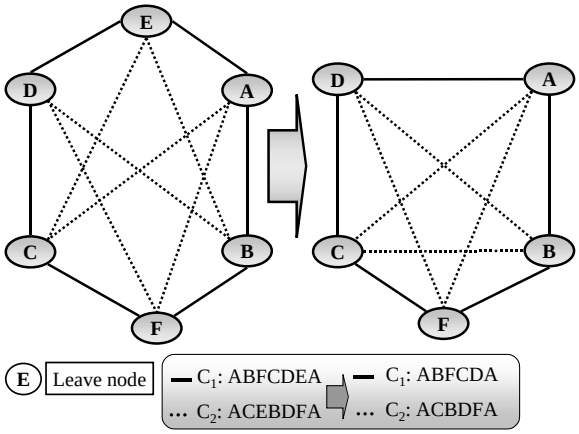

Figure 6.4. Leaving Process.

Leave () {

// LN is leaving node

For i :=1, ..., d in parallel do

Edge(LN $\Longrightarrow r_p^{(i)}$, LN $\Longrightarrow r_s^{(i)}$, i)

}

The leave routine creates edges at d cycles between the leaving node's successor and predecessor node as shown in Figure 6.3. For example, Figure 6.4 shows the leaving process of node *E*. It substitutes the edges (*E*, *A*) and (*E*,*D*) by new edge (*D*, *A*) in cycle 1. Similarly, it substitutes edges (E, B) and (E, C) by new edge (C, B) in cycle 2. Obviously, the leave process requires only local changes in the community network with *O(d)* messages.

Fault-tolerance Process It is also required to consider the difficult case of node failure. For fault-tolerance, we assume that each node knows the predecessor of its predecessor node and the successor of its successor node in each cycle. The node failure is detected locally as follows. Each node has fault-tolerance module that is periodically exchange keep alive message with its neighbors. For example, the neighboring nodes of node X that belong to INS_X, periodically exchange keep-alive message with the node *X*. If node *X* is unresponsive for a

period T, neighbor nodes presume it has failed. All neighbors of the failed node update their INS sets and execute the following instance of code.

*FT(*X, i *)*{

// X is the failed node in cycle i.

MyPredecessor $\longleftarrow$ (Calling_node $\Longrightarrow r_p^{(i)}$);

MySuccessor $\longleftarrow$ (Calling_node $\Longrightarrow r_s^{(i)}$);

If (X == MyPredecessor) then

(Calling_node $\Longrightarrow r_p^{(i)}$) $\longleftarrow$ (MyPredecessor $\Longrightarrow r_p^{(i)}$);

if (X== MySuccessor) then

(Calling_node $\Longrightarrow r_s^{(i)}$) $\longleftarrow$ (MySuccessor $\Longrightarrow r_s^{(i)}$);

}

The FT routine connects two nodes around the failed node in same cycle and sets the calling node's predecessor and successor to the failing node's predecessor and successor, respectively. This maintains the *Hamilton* cycles as well as the same number of links for all nodes. This technique scales well: a few nodes exchange messages to detect faults, and fault recovery is local, involving only a few nodes $|INS_x|$. In addition, this technique maintains the community network G composed of edge disjoint *Hamilton* cycles. If a Hamilton path connects every two nodes of G, then G is *Hamilton-connected* [70]. Thus the Flat-CON is unpartitionable and does not have any single point of failure node; to prove this, we construct *Hamilton cycles* with (M$\geq$5). For example, if we construct the community network as a 4-regular, 4-connected graph (cf. Figure 6.4) then this graph should have two edges disjoint Hamiltonian cycles [12, 71]. The community network is thus a connected graph. Because our proposed fault-tolerance technique maintains the *Hamilton cycles*, the resulting Flat-CON is connected and un partitioned.

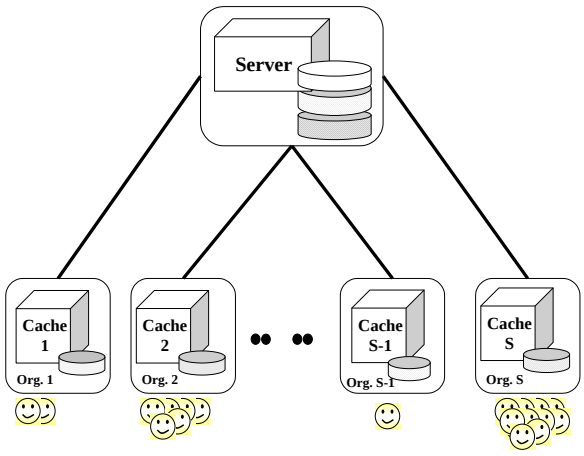

Figure 6.5. One-to-one communication simulation model based on caching proxies

6.1.5 Performance Verification

Simulation Setup

We have developed a simulation to prove the validity of the proposed ADCCS system. The target number of users of the ADCCS is 100,000. Thus, the simulation ran over Flat-CON contains 100,000 members. It is based on the assumption that the communication cost between each node is same and equal one unit of time. We simulated the one-to-one communication based on the client/server model. The experiments have been conducted over 100,000 of requests. Each client accesses the server and sends a request simultaneously to the server (e.g. news server or e-government server). Obviously, the surge of simultaneous requests arriving at the server results in the server overwhelmed and response time shooting up. Caching web pages on the proxy servers reduces the access latency for the clients. Thus, the webs caching techniques have slightly effect in the response time.

Figure 6.5 shows the one-to-one communication simulation model based on caching proxies as follows. We assume that each caching proxy is located at an organization and the requests of the clients are assigned randomly to S caching proxies. It has been proved that a caching proxy has an upper bound of 30-50% in its hit rate [72]. In addition, we simulated the proposed community communication technique

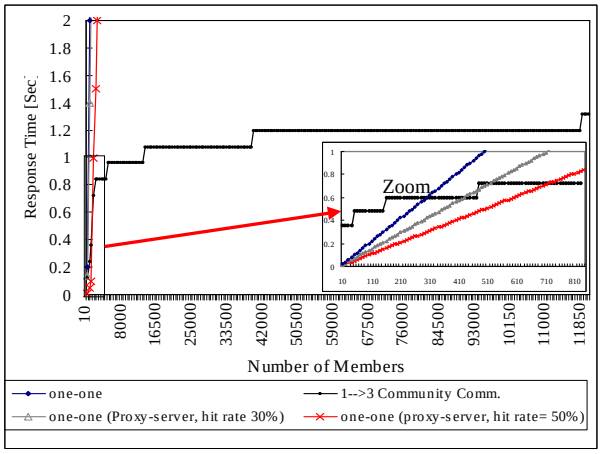

Figure 6.6. Scalable information-dissemination system

on a network spending *4-array* connectivity for each community node (Flat-CON, *2d* regular graph with d=2). The experiments have been conducted over 100,000 community members, using $1 \rightarrow 3$ communication technique and is constituted of communication cost between each node τ_{cc}= 10 ms. τ_m= 10 ms is the average time that each node needs to monitor the recent received messages to avoid the congestion. Thus, the transmission time τ to send a message asynchronously from any node to all the other community nodes is bounded by $L \times N \times (\tau_{cc} + \tau_m)$, where L is the number of layers that has been determined by Equation 5.1 and N= *2d-1*.

Flat-CON: Communication Delay

We concentrate in these experiments on the comparison between the conventional one-to-one communication techniques without and with caching proxy (hit rate of 30%, 50%) and $1 \rightarrow N$ community communication technique. As soon as one client (community member) downloads an interesting content for the community from the server, he/she publishes it to all the community members using $1 \rightarrow N$. Figure 6.6 shows the variations of the number of members in the community with the worst transmission time of a message to all members. Figure 6.6 depicts the effectiveness of the proposed communication technique, that is performed over Flat-CON, compared with the conventional communication techniques. The $1 \rightarrow N$ communication tech-

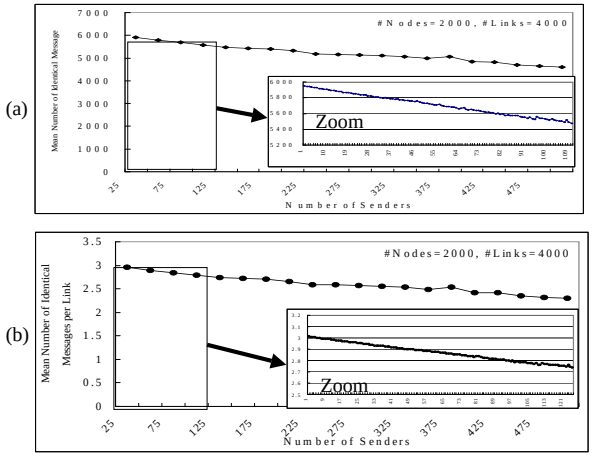

Figure 6.7. Flat-CON traffic with multiple senders

nique is able to send a message to all community members (target number 100,000) with imprecision close to 93% compared to (one-one) unicast, 91% compared to unicast with hit rate 30% and 87% compared to unicast with hit rate 50%. These results reflect that unicast with caching proxies technologies improve the communication performance slightly compared with the proposed communication technique. Thus, it shows that the community communication technique is scalable of the response time with the number of the members. As shown in the zoom part in Figure 6.6, for small number of members (less than 1000) the proposed community communication technique is not effective but it reveals meaningful results when the total number of members in the community increases sharply. The results of this simulation suggest that ADCCS with Flat-CON can achieve good performance for large number of members (target number 100,000) under the assumption that the communication cost between each node is one unit of time. The question then is: can ADCCS support large number of members with different communication cost? Next sections will answer this question.

Flat-CON: Network Traffic

We ran an additional experiment to evaluate the Flat-CON traffic in case of many nodes in the community send an identical message at once. Each node monitors the

received messages and forwards only one from the received identical messages. This experiment ran on a community network with 2000 nodes and 4000 logical links, which were generated as regular graph. The delay of each link was set to 10 ms. We ran the simulation 1000 of times to determine the *mean number of identical messages* (**MNIM**) in the community network. Each time we ran the experiment, the senders were selected randomly. Figures 6.7-(a) and 6.7-(b) plot the variation of both the *MNIM* in the community network and the *MNIM* carried by a logical link with the number of senders. The number of senders is increased from one sender to about 25% senders from the participated nodes in the community network. In addition, zoom parts in both figures 6.7-(a) and 6.7-(b) show the variation of *MNIM* with the increases of the number of senders from 1 to 125. The increase of the number of senders reduces both *MNIM* in the community network and MNIM per link. Figure 6.7-a shows about 22% improvement of *MNIM* when 25% nodes in the community send identical message at once. The standard deviation of *MNIM* per link is about 0.078. This result indicates that community system does an efficient job in distributing loads over all nodes; each node is responsible for forwarding messages only to a small number of nodes. This is important to achieve scalability with the community system size. We conclude that Flat-CON's network traffic is reduced when many members send an identical message at once.

Flat-CON: Construction/Maintenance Overhead

We conducted a simulation to evaluate the community network construction and maintenance overheads. The joining communication cost is the required communication time to forward the join request message within the community network. Each second 100 nodes join the community network with uniform distribution. We ran this simulation for 20 minutes as a result the community network size becomes 108,000 members and about 384ms is the required communication delay for the construction and maintenance of the community network. In the simulation each second 10 nodes are chosen randomly to leave the community network. Each leave node calls the *Leave()* routine with maintenance cost that varies with the communication delay to

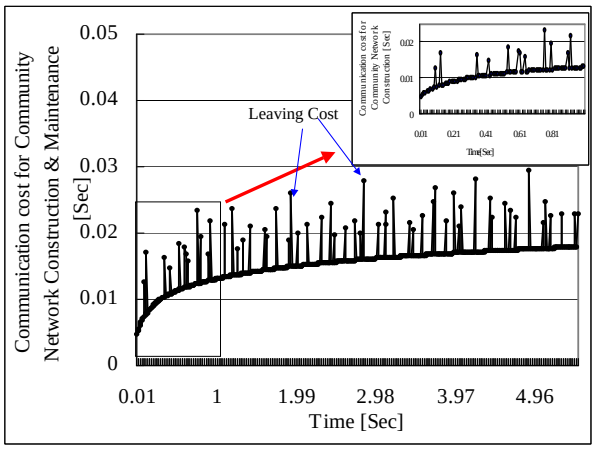

Figure 6.8. Flat-CON construction and maintenance overhead.

its successor's and predecessor's neighbors. To get better understanding of the simulation results, Figure 6.8 presents part of the simulation results of 5.5 seconds and 500 members. In this figure, the peaks represent the leaving cost. The results show that the required communication delay for the construction (joining) and maintenance of the community network is increased logarithmically with standard deviation approximately equal 0.0033. Thus, these results indicate that the construction and maintenance techniques for the Flat-CON are scalable for large number of members.

Flat-CON: Fault-tolerance

We ran an additional experiment to evaluate the fault-tolerance process of the Flat-CON. Each node has 4 neighbors and resides on two *Hamilton* cycles. In this simulation we assume that the community node-node TCP connections provide data reliability on a hop-by-hop basis, which implies that message losses due to network congestion and transmission errors are eliminated. Instead, the main reason for message losses in ADCCS are due to transient network link failures, or node failures. In each run of the simulation, one community node randomly selected as a source of a message. In addition, a number of nodes are randomly selected to act as failed nodes simultaneously. Then the *Functional Reliability (FR)*, *fraction of mean num-*

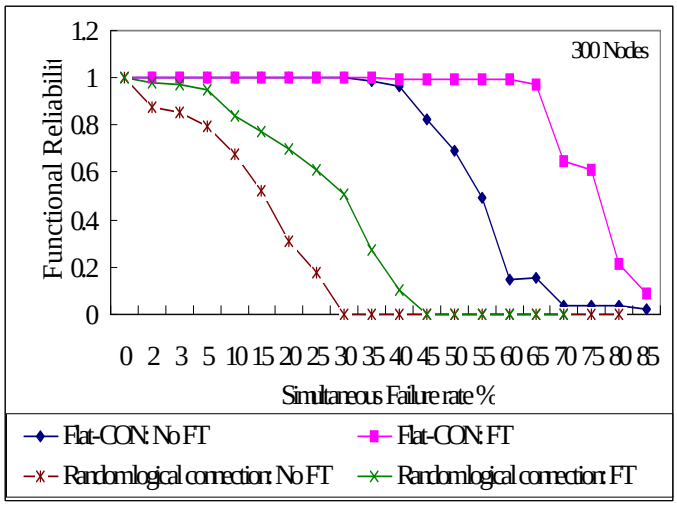

Figure 6.9. Flat-CON Fault-tolerance verification.

ber of nodes received a message is evaluated. Figure 6.9 shows the variation of the *FR* with the simultaneous failure rate. It shows that Flat-CON with fault-tolerance improves the *FR* by about 18% compared to Flat-CON without fault-tolerance and 45% compared to random logical connection overlay network with fault-tolerance. In this random overlay network each node has four neighbors. In addition, the neighbors of the failed node recover the failure by connecting each others randomly. Moreover, the fault-tolerance of the Flat-CON demonstrates that the catastrophic failure of Flat-CON starts when about 65% of random nodes simultaneously failed. We argue the improvement of the Flat-CON to the fault-tolerance process that maintains the Hamilton cycle (i.e. connected graph). However due to the randomization of the fault-tolerance processes of the random logical overlay network, the network suffers from partition and could not keep the network connected.

6.2 Multiple Communities Construction/Maintenance Techniques

In the Internet environment, the communication cost from node to node is heterogeneous. The goal of this section is to take advantage of this heterogeneity. It develops an autonomous community overlay network that is aware of the communi-

cation cost heterogeneity in order to optimize the end-to-end communication delay. This section constructs CON as a multiple of communities so-called *Multilayer-CON*.

6.2.1 Goals and Requirements

The performance of the community communication could be improved if the application level connectivity between community nodes is congruent with the underlying IP-level topology [73], but this is not the approach we have decided to follow. The ADCCS-Multilayer deliberates the end-to-end node latency as an important criterion that must be optimized. Thus, we have turned to construct the autonomous community overlay network considering node-node latency awareness. The problem of constructing an optimal overlay network is known to be NP-hard [67, 74].

Designing an efficient overlay network faces several challenges. Some important designing issues are as follows. First, the end-to-end delay from source to a receiver may be increases because the message may have to go through a number of intermediate community members. The receivers with long latencies to their neighbors slow down entire network branches. Moreover, the end-to-end delay may also be long due to an occurrence of bottleneck at the communication tree. Therefore, it is important to have the node degree bounded. Second, the behavior of the community nodes is unpredictable; they are free to join and fail/leave the community at any time. Thus, to prevent the community service interruption, a robust technique has to provide a quick and efficient recovery when failure occurs. Third, community nodes have to store some local information and exchange state information with small number of neighbors in order to maintain the connectivity and to improve the efficiency of the community overlay network. This control overhead should be kept minimum. This is an important issue to achieve the scalability of ADCCS for a large number of members. To address these issues above, this thesis proposes Multilayer-CON. Details of construction, maintenance and adaptation techniques are described hereafter. To take advantage of heterogeneity of the community nodes-to-nodes communication delays, the following sections describe the details of Multilayer-CON's construction, maintenance and adaptation techniques. Multilayer-CON organizes the community

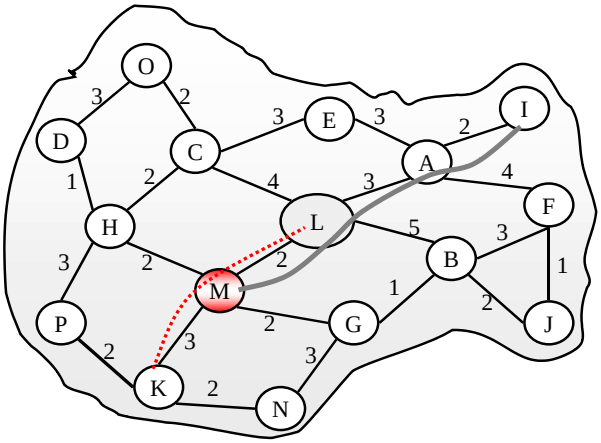

Figure 6.10. An example of sub-community structure ($\alpha_j^i = 10ms$).

network into a multi-layers of sub-communities. Next subsection illustrates the sub-community definition, structure and the step by step construction technology.

6.2.2 Sub-Community: Definition and Structure

$S_i^{(j)}$ denotes the i-th sub-community at layer j. Each sub-community $S_i^{(j)}$ has two special members: *Leader* and *Disseminator*. The leader $L_i^{(j)}$ is responsible for membership management of the $S_i^{(j)}$. The disseminator $M_i^{(j)}$ is responsible for transmitting contents from/to the sub-community $S_i^{(j)}$. The disseminator $M_i^{(j)}$ is one of the neighbors of $L_i^{(j)}$ that has the smallest communication cost to $L_i^{(j)}$. It keeps a list of the Leader's neighbors. In the sub-community $S_i^{(j)}$ all nodes have communication delay to the leader and the disseminator that is bounded by a selected value $\alpha_i^{(j)}$. Thus, the sub-community $S_i^{(j)}$ can be defined as a set of nodes x_0 that satisfies the ***Latency Awareness Condition*** (LAC) as follows:

$$S_i^{(j)} = \left\{ \begin{array}{ll} x_0; \quad \sum_{Path1,k=0}^{m-1} \delta(x_k, x_{k+1}) \leq \alpha_i^{(j)} & \text{if } (x_{m-1} = M_i^{(j)} \wedge x_m = L_i^{(j)}) \\ or & \\ x_0; \quad \sum_{Path2,k=0}^{m-1} \delta(x_k, x_{k+1}) \leq \alpha_i^{(j)} & \text{if } (x_{m-1} = L_i^{(j)} \wedge x_m = M_i^{(j)}) \end{array} \right\} \quad (6.1)$$

Where $\delta(X, Y)$ denotes the current end-to-end delay from X to Y that is measured by the delay of the round-trip message. The dotted line that is shown in Figure 6.10

represents the $Path_1 = \{x_0, x_1,...,x_{m-1}= M_i^{(j)}, x_m = L_i^{(j)}\}$ from node x_0=K to L through the disseminator node M. The gray line in Figure 6.10 represents the $Path_2 = \{x_0, x_1,..., x_{m-1} = L_i^{(j)}, x_m = M_i^{(j)}\}$ from node x_0= I to M through node A and leader node L. Figure 6.10 shows that the communication delay from any node to the leader and the disseminator in the sub-community is less than or equal $\alpha_i^{(j)}$ =10ms (i.e. All nodes satisfy the LAC). The leader $L_i^{(j)}$ of the sub-community $S_i^{(j)}$ autonomously determines $\alpha_i^{(j)}$ and adapts it to cope with the changing of the nodes communication delay. We conclude that the sub-community is a set of nodes considering LAC, the existence of loops and the node's connectivity is bounded by π.

Sub-Community: Step by Step Construction

To construct a sub-community step by step, we have developed a join_sub(X, $S_i^{(j)}$) routine. It inserts the new node X to the sub-community $S_i^{(j)}$. The join_sub() scenario is as follows. The leader $L_i^{(j)}$ of the $S_i^{(j)}$ forwards join-request to its neighbor nodes and then waits their replies. Each node receives the join-request processes the instance of the function Node_joinCheck that is given as follows. Each node then autonomously decides whether the new node X can connect to itself or not based on each node join-rate ϕ(t) that has been defined in the previous section. As soon as the leader of the $S_i^{(j)}$ received some replies, it selects the nodes that have the smallest latency to X and then X connects to π nodes from them.

*Node*_joinCheck (X, rf)

{

// X: new joining node and rf: received from node.

Id=my_node_id; // Id ϵS_i^j

i= Node.getSubCommunityId;

j= Node.getSubCommunityLayer;

If (connectivity(Id) $> \pi$) then

{

Forward(join-request(X), {neighbors { Id } \ { rf } });

Return 0;

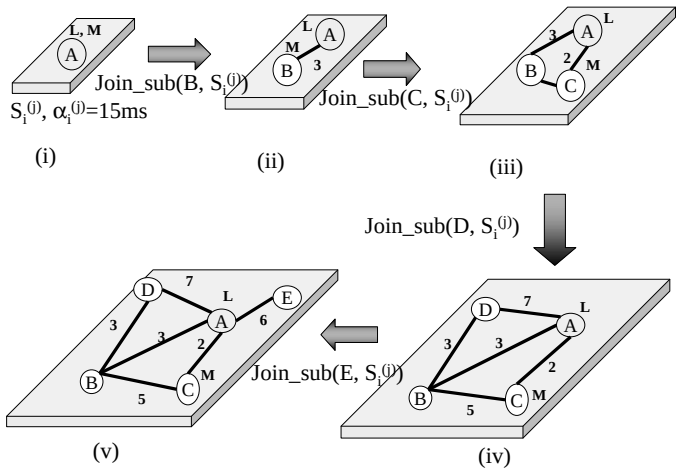

Figure 6.11. Step by step sub-community construction.

}

Else {

// Path ={$x_0 = Id$, x_1, ...,x_m= $L_i^{(j)}$}

$\tau = \sum_{x_i \in Path} \delta(x_i, x_{i+1}) + \delta(Id, X)$

If ($\tau < \alpha_i^{(j)}$) then

{

Send ($L_i^{(j)}$, "*Ok to join*", τ);

Return 1;

}

Else

Return 0;

}

}

Figure 6.11(i) shows that node A is the leader of the sub-community $S_i^{(j)}$ with $\alpha_i^{(j)}$ =15ms. When new node B wants to join $S_i^{(j)}$, it calls the join_sub(B, $S_i^{(j)}$). Figure 6.11(ii) shows that B has joined to $S_i^{(j)}$ because δ(A, B)< $\alpha_i^{(j)}$. The new node C joined the sub-community $S_i^{(j)}$ as shown in Figure 6.11(iii) because $\delta(C, B) + \delta(B, A)$ < $\alpha_i^{(j)}$ and δ(C, A) < $\alpha_i^{(j)}$. In addition, C becomes the disseminator of $S_i^{(j)}$ instead

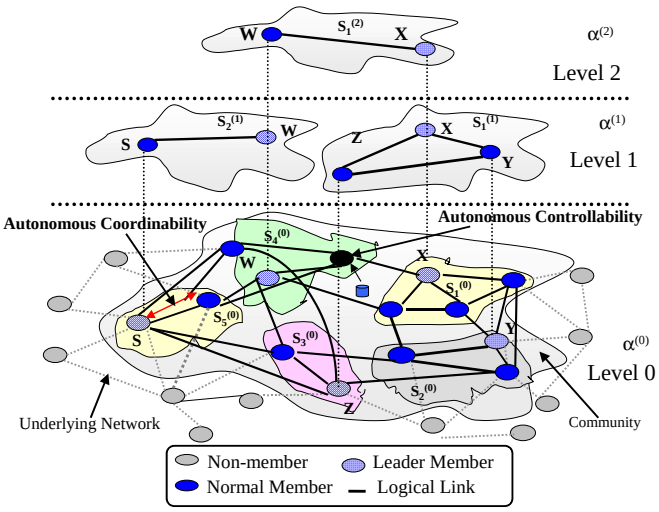

Figure 6.12. Multilayer-CON: architecture.

of B because δ(C, A)< δ(B, A) as shown in Figure 6.11(iv). Similarly, nodes D and E satisfy the ***LAC*** of $S_i^{(j)}$ and then they join it as shown in Figure 6.11(v).

6.2.3 Organizing Multilayer-CON

This section presents the proposed ADCCS-Multilayer structure that is developed to organize N community nodes into multi-layers of sub-communities. It is recursively defined as follows (where β_j is the number of sub-communities at layer j and K is the number of layers):

1. Layer 0 contains all nodes that are currently partaking in the community. It is partitioned into β_0 sub-communities.

2. Layer *j+1* contains all leaders of the sub-communities at layer *j*. It is partitioned into β_{j+1} sub-communities. Obviously, $\beta_j > \beta_{j+1}$; j= 0, 1, ..., *k-1*.

3. The leaders at layer j automatically become members of the sub-community of leaders at layer *j+1*, if they satisfy the ***LAC*** at layer j+1. For $j \geq 0$, the number of nodes at layer *j+1* is β_j. Layer K consists of a few sub-communities (e.g. one or two).

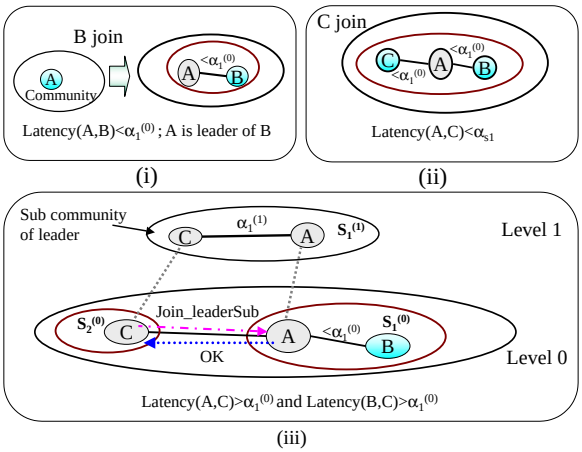

Figure 6.13. Step-step construction multi-layer community structure.

4. If a node belongs to layer j then it must be in one sub-community in each of the layers 0, 1, ..., *j-1*. Furthermore, any node at layer $j > 0$ must be a leader of the sub-community and belongs to all lower layers.

5. For any i and j, $\alpha_i^{(j)} < \alpha_i^{(j+1)}$.

This scheme is used to map the community nodes into layers as shown in Figure 6.12. This figure shows that the multi-layer community structure consists of three layers. Layer 0 contains 14 nodes and organized into five sub-communities. The leaders at layer 0 form layer 1 and they are organized into two sub-communities. Finally, layer 2 contains only one sub-community that contains two nodes.

Multilayer-CON: Step by Step Online Expansion and Construction

This section illustrates the construction of the community overlay network as multi-layer structure. Figure 6.13(i) shows that node A initiates the community of an interest, creates a sub-community $S_1^{(0)}$ and becomes a leader of $S_1^{(0)}$. Then, node B has the same interest and wants to join the community. Node B sends join request to node A. As soon as node A receives join request it checks the round-trip latency to the joining node B. If the joining node satisfies the LAC (δ (A, B)$< \alpha_1^{(0)}$), then A connects B by a logical link. Similarly, the joining node C sends a join request to a

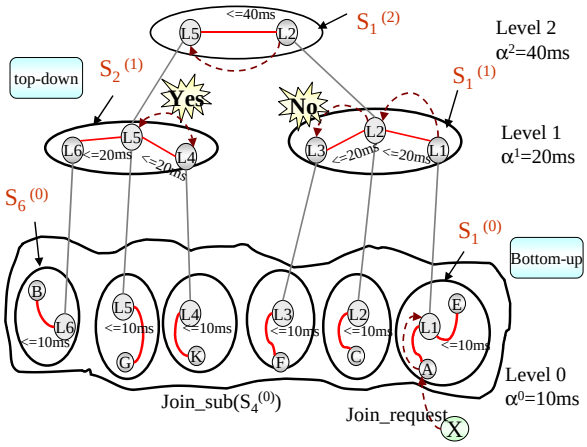

Figure 6.14. Example: Join process in control tree.

node in the community (e.g. B). Node B forwards the join request to the leader of the sub-community it belongs to (i.e. node A). The leader checks whether the joining node satisfies the LAC or not. Figure 6.13(ii) shows the join process of node C when the LAC is satisfied. Otherwise, Figure 6.13(iii) shows that the joining node C, joins the community and becomes a leader of its own created sub-community $S_2^{(0)}$. Node C then sends a join_leadersub ($S_1^{(1)}$) request to the neighbor leader (e.g. node A). Then, A checks if C satisfies the LAC at the upper level. If δ(A,C)$< \alpha_1^{(1)}$ then C joins the sub-community of leaders $S_1^{(1)}$ at the upper level otherwise C creates a new sub-community of leaders $S_2^{(1)}$ at the upper level. Recursively, new nodes join the community and consequently the community multi-layer structure is constructed.

6.2.4 Joining Process: Bottom-up & Top-down

When new node X wishes to join the community, Multilayer-CON assumes that X is able to get at least one community node A. The joining node X sends a join request to one community node A as shown in Figure 6.14. The join request is redirected along the multi-layer community structure *bottom-up* and *top-down* to find the appropriate sub-community as follows. Node X joins the community temporarily by contacting a first contact node, FN (e.g. FN = A) that belongs to $S_i^{(j)}$ at layer j=0. Thus, X can receive community information during the joining process. X then calls the join

function to find an appropriate sub-community. The scenario of this function is as follows.

1. Node X sends join request to the first contact node, FN that belongs to $S_i^{(j)}$ at layer j=0.

2. Node FN checks:

   (a) If $(\delta(X, FN) + \delta(X, L_i^{(j)})) < \alpha_i^{(j)}$ then Create_link(X, FN).

   (b) Otherwise, forwards join request to $L_i^{(j)}$.

3. $L_i^{(j)}$ Checks:

   (a) If $\delta(X, L_i^{(j)}) < \alpha_i^{(0)}$ then Calls join_sub(X, $S_i^{(j)}$); exit;

   (b) Otherwise,

      i. If $(j == 0$ && $\delta(X, L_i^{(j)}) > \alpha_i^{(0)})$ then forwards join request to the leader $L_u^{(j+1)}$ of the sub-community $S_u^{(j+1)}$ at the upper layer where, $L_i^{(j)} \in S_u^{(j+1)}$.

      ii. Otherwise, If $(j > 0$ && $\delta(X, L_i^{(j)}) > \alpha_i^{(0)})$ then forwards join request to the leader $L_d^{(j-1)}$ of the sub-community $S_d^{(j-1)}$ at the lower layer where, $L_i^{(j)} \in S_d^{(j-1)}$.

4. Set *j:= j+1* and then $L_u^{(j)}$ checks:

   (a) If $\delta(X, L_u^{(j)}) < \alpha_u^{(0)}$ then it forwards join request down.

   (b) Otherwise, it forwards join request to all members z_m belongs to $S_u^{(j)}$ except $L_u^{(j)}$.

5. Each node z_m checks:

   (a) If $\delta(X, z_m) < \alpha_i^{(0)}$ then z_m sends a reply message to $L_u^{(j)}$ that contains "*ok to join*", $\delta(X, z_m)$ and its join-rate ϕ(t) that has been defined in the previous section.

   (b) Otherwise, z_m does not send any reply message to $L_u^{(j)}$.

6. $L_u^{(j)}$ waits for a period of time γ to gather replies from all z_m nodes belong to the sub-community $S_u^{(j)}$. There exist two cases as follows:

   (a) $L_u^{(j)}$ receives some replies and then selects one that has the smallest latency to X. Then, it forwards the join request down (i.e. set *j:= j-1*) to the selected leader that calls join_sub() routine.

   (b) Otherwise, $L_u^{(j)}$ does not receive any reply within the time-out period γ. Thus, it forwards the join request to the upper layer (i.e. set *j:= j+1*) if j is smaller than the number of layers and then repeats steps 3-6, otherwise it forwards the join request to the lower layer (i.e. set *j:=j-1* and then repeats steps 3-6.

7. The join request is forwarded from bottom to up and then top to down until the LAC is satisfied. Otherwise, there is no sub-community satisfies the LAC then a new sub-community contains the joining node is created.

The join process is illustrated as a recursive function in Appendix A. It terminates at layer 0 when the joining node either finds a sub-community (e.g. $S_4^{(0)}$ in Figure 6.14) that satisfies the LAC or not. Therefore, the join overhead is $O(\beta_0)$ in terms of the number of nodes that must check the latency with the joining node. This joining process reflects that each node autonomously takes the decision based on its local information and there is no specific server that is responsible for membership management. Moreover, new nodes join the autonomous multi-layer community overlay network without stopping the community service. Thus, the proposed autonomous decentralized online expansion and construction technique is scalable for large number of nodes.

6.2.5 Maintenance and Fault-tolerance

When node X wishes to leave the community, it notifies its neighbors in the sub-community $S_i^{(0)}$. $L_i^{(0)}$ and $M_i^{(0)}$ are the leader and the disseminator of $S_i^{(0)}$ respectively. The leave algorithm is described as follows:

1. If (($X \neq L_i^{(0)}$) and ($X \neq M_i^{(0)}$)) then the neighbors of X remove their links to X. For example, nodes k and G remove their links to the leaved node N in Figure 6.10. Node X processes the leave() procedure that has been illustrated in the previous section.

2. If (($X == L_i^{(0)}$) and $L_i^{(0)} \in$ layers $l_0, ..., l_h$) then each disseminator $M_i^{(j)}$ becomes leader (i.e. $L_i^{(j)} = M_i^{(j)}$), where j = 0,..., h. Then, each leader selects a new node that has the smallest latency from its neighbors and assigns it as a new disseminator. This process is repeated on h-layers $l_0, ..., l_h$.

3. If (($X == M_i^{(0)}$) and $M_i^{(0)} \in$ layers $l_0, ..., l_h$) then each leader $L_i^{(j)}$ selects a new disseminator from its neighbors that has the smallest latency to $L_i^{(j)}$, where $j = 0$, ..., h.

Similar to the fault-tolerance technique in Flat-CON, it is also required to consider the difficult case of node failure in Multilayer-CON. In such case, failure should be detected locally as follows. The neighboring nodes periodically exchange keep-alive message with node X. If X is unresponsive for a period T, it is presumed failed. All neighbors of the failed node update their neighbor sets. This technology scales well: exchanging messages among small number of nodes does fault detection, and recovery from faults is local; only a small number of nodes are involved. In Multilayer-CON, if the leader of the sub-community fails and the disseminator is still working then the disseminator takes the leader responsibilities, connects to the leader's neighbors and selects another disseminator to take its responsibilities. Therefore, the failure of the leader does not affect the community service continuity of other nodes. Similarly, when disseminator fails, the leader is still working and can appoint new disseminator quickly. However, If both leader and disseminator simultaneously fail then all nodes in the sub-community elect new node as leader. Then this new leader assigns new disseminator.

6.2.6 Community Communication over Multilayer-CON

For an efficient community communication, we create a multi-layer connected control topology. The content delivery path is implicitly defined in the way the multi-layer is structured and no additional route computations are required. The disseminators in this multi-layer structure play important roles in this communication technology. Each node can send a message to its neighbors in its sub-community $S_i^{(0)}$ by using $1 \rightarrow N$ communication. Once the disseminator $M_i^{(0)}$ receives such message, it forwards the message to all disseminators belonging to the sub-community at the upper layer. Each disseminator forwards such message to all members in its sub-community. Each disseminator $M_i^{(j)}$ executes an instance of the following procedure.

*Hcomm*unity_comm($S_i^{(0)}$, rf) {

// $M_i^{(j)}$ forwards the message that received from rf.

If ($M_i^{(j)} \in$ layers l_0 ,..., l_m in sub-communities $S^{(0)}$, ..., $S^{(m)}$)

for ($p = 0$; $p \leq m$; p++)

If (rf $\notin S^{(p)}$)

ForwardMessageTo ($S^{(p)}$ - { $M^{(p)}$});

}

Consequently all nodes in the community will receive such message. Assume all $\alpha_i^{(j)} = \alpha_{i+1}^{(j)}$ at each layer j, where i=1,.., β_{j-1} and j=0, ..., k. Thus, the transmission time to forward a message from a community node to all nodes is bounded by

$$\alpha^{(k)} + \sum_{j=0}^{k-1} 2 * \alpha^{(j)} \tag{6.2}$$

For example, Figure 6.15 shows the message transmission initiated from node E. Node E sends a message to all members in the sub-community $S_1^{(0)}$, once the disseminator $M_1^{(0)}$ receives such message, it forwards the message to all members in the $S_1^{(1)}$. The disseminator $M_1^{(1)}$ forwards the message up to the layer 2 and so on. The required sequence to forward the message to all members in the community through the multi-layer structure is shown in Figure 6.15 by dotted arrows with index of order. In this figure, the transmission time is bounded by 90ms. Clearly, the multi-layer sub-community approach considers the heterogeneity of node-to-node latencies.

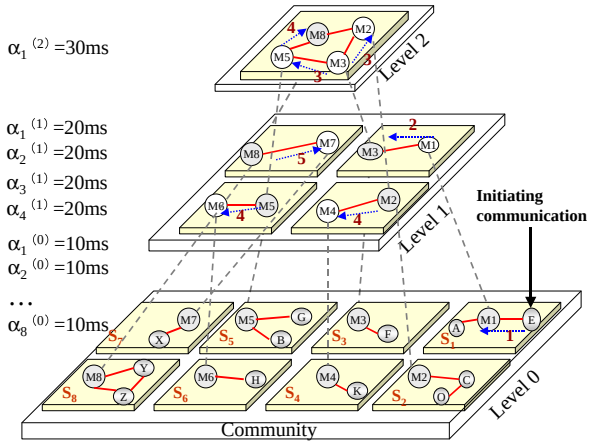

Figure 6.15. Community communication through Multilayer structure.

It results in a community network clustering of community nodes into homogenous sub-communities (*Flat-CON*) thereby reducing the communication delay.

6.2.7 Performance Verification

To evaluate the performance, we have developed a simulation over a random generated network with different communication cost between nodes. This simulation demonstrates that ADCCS-Multilayer architecture can perform quite well in realistic Internet settings. In this section, we present the performance metrics and then study the performance issues with large community size using simulation experiments.

Performance Metrics

To evaluate the communication system ADCCS over Multilayer-CON (so called *ADCCS-Multilayer*) and compare it with ADCCS over Flat-CON without considering the latency awareness and the conventional communication techniques; we used the following metrics.

- ***Latency.*** It measures the communication delay from a community node to all others community nodes.

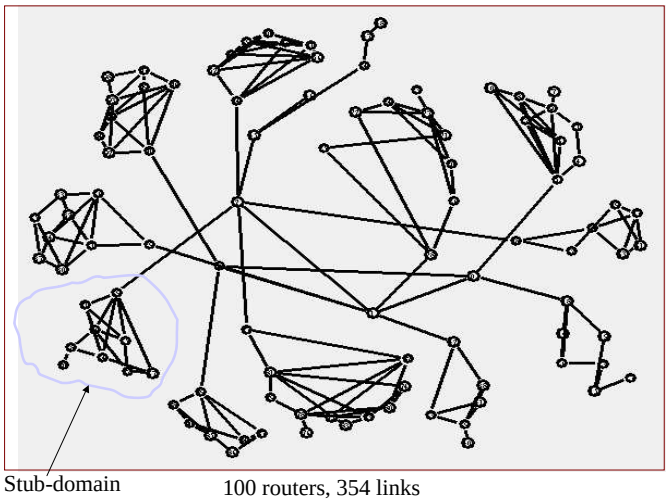

Figure 6.16. Simulation setup: transit-stub network model.

- ***Relative Mean Delay Penalty*** (RMDP). It defines the ratio of the mean delay between two community nodes along the ADCCS-Multilayer, ADCCS and Unicast to the IP multicast delay between them. It is used to measure of the increase delay that applications perceive while using ADCCS-Multilayer and ADCCS.

- ***Stress.*** It measures the number of identical copies of a message carried by a physical link. Obviously, we would like to keep the link stress on all links low as possible.

Simulation Setup

The simulations ran on two underlying network models, *transit-stub* model and *Waxman* model, with 100 routers linked by core links. The Georigia Tech [75] random graph generator is used to create both network models. Figure 6.16 shows an example of the transit-stub network model. Random link delay of 4-12ms was assigned to each core link. The community end-nodes were randomly assigned to routers in the core with uniform probability. Each community end-node was directly attached by a LAN link to its assigned router. The delay of each LAN link was set to be 1ms. End-nodes join the community network with joining rate 100 nodes/Sec with equal

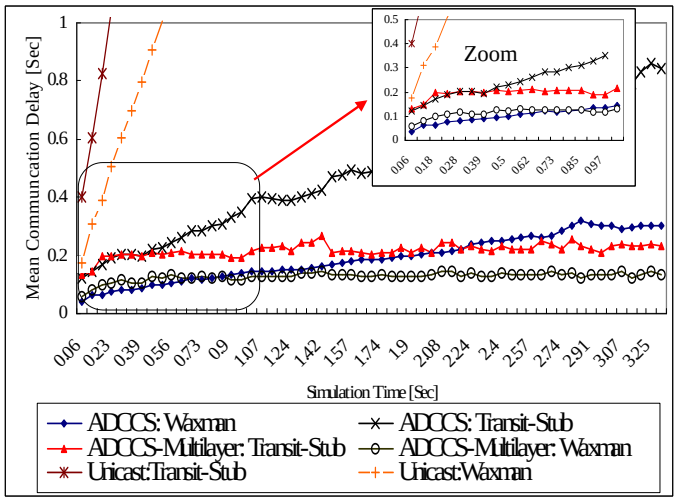

Figure 6.17. MCD: Comparison.

distribution. Members leave the community network with leaving rate 10 nodes/Sec with random distribution. We have conducted a simulation to compare ADCCS-Multilayer with Unicast and ADCCS. Unicast operates over underlying network (i.e. no overlay network). ADCCS operates over 4-regular graph overly network (Flat-CON) that spends 4-array connectivity for each node. ADCCS-Multilayer is organized with $\alpha^{(0)} = 5$ and $\alpha^{(j+1)} = 2 * \alpha^{(j)}$. The number of sub-communities and layers changes with α and the number of end-nodes. For example, 300 end-nodes are organized into 4 layers and 100 sub-communities at layer zero when $\alpha^{(0)} = 5$.

Multilayer-CON: Communication Delay

In each run of the simulation, one community node randomly is selected as a source. Then the communication delay the source node requires to send a message to all nodes is evaluated. In this simulation, only static latency including no process delay is evaluated. We ran this simulation for 20 minutes as a result the community network size becomes 108,000 members. For simplicity, Figure 6.17 shows only the simulation results of the first 3.5 Seconds from the simulation running time. It plots the variations of the ***Mean Communication Delay*** (MCD) that is required to send a message from a node to all nodes participated at each instance of time

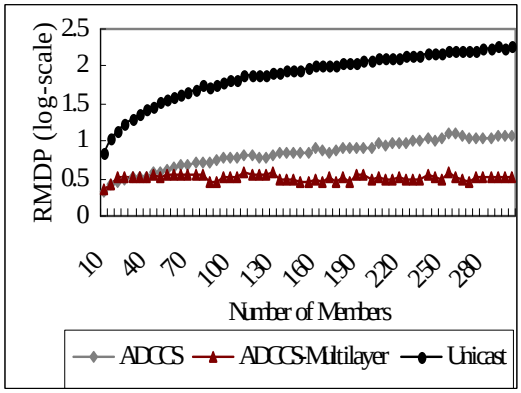

Figure 6.18. RMDP: Comparison.

during the experiment. It evaluates and compares ADCCS-Multilayer with ADCCS and Unicast over two underlying network models. ADCCS-Multilayer has shown about 39% improvement of the MCD compared to ADCCS and 93% compared to Unicast. Owing to the design of the latency-awareness Multilayer-CON structure, the ADCCS-Multilayer has about 39% imprecision over ADCCS. The MCD of the Unicast, ADCCS and ADCCS-Multilayer over Waxman topology is small than over transit-stub topology. That is to be expected because of the delay of the hierarchy of the transit-stub model. Moreover, the zoom part in Figure 6.17 shows that the ADCCS-Multilayer is not effective for small number of end-nodes compared to ADCCS. However, ADCCS-Multilayer is effective for large number of end-nodes compared to ADCCS. In addition, Figure 6.18 plots the variation of RMDP for sequential Unicast, ADCCS and ADCCS-Multilayer over transit-stub model. The vertical axis represents a given value of RMDP associated with the community network size in log-scale presentation. ADCCS-Multilayer shows about 94% improvement of the RMDP to Unicast and about 47% imprecision to ADCCS.

It is remained to study the proposed community communication system in a randomly joining/leaving network as follows. End-nodes join/leave the community overlay network with random distribution as shown in Figure 6.19. It shows the variations of the community network size with the simulation time. Figure 6.20 shows only the simulation results of the first 3.5 seconds from the simulation running time. It plots

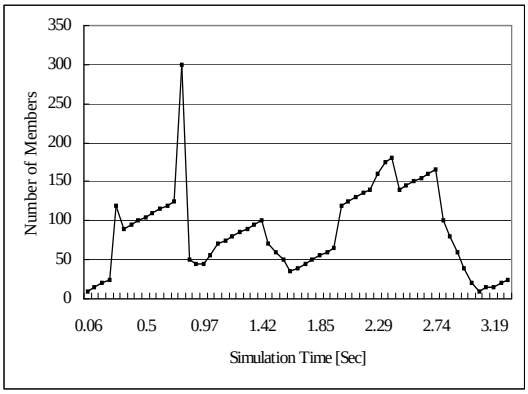

Figure 6.19. Variation of Community Overlay network size per time.

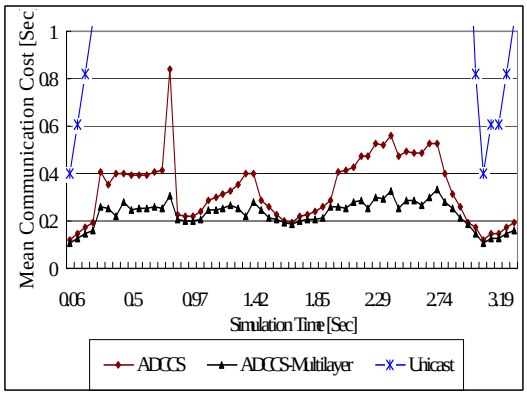

Figure 6.20. MCD: Comparison (Random community overlay network).

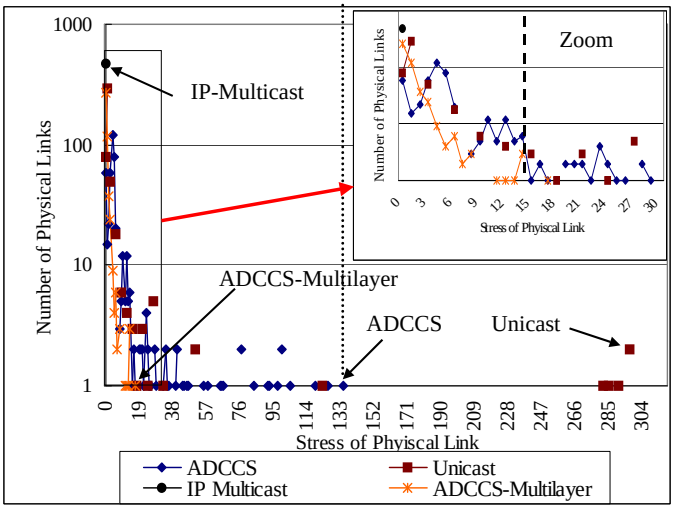

Figure 6.21. Physical link stress.

the variations of *MCD* at each instance of time during the experiment. Similarly, ADCCS-Multilayer has shown about 39% improvement of the MCD over ADCCS and 93% over unicast. From these results we conclude that the ADCCS-Multilayer enhances the community communication in realistic Internet settings compared to the ADCCS. Thus, the timeliness is achieved.

Multilayer-CON: Network Traffic

We have conducted our experiment with a community size 300 members. We randomly selected one member as a source and then evaluated the stress of each physical link. We study the variation of physical link stress for ADCCS-Multilayer, ADCCS, IP Multicast and naive Unicast as shown in Figure 6.21. The horizontal axis represents stress and the vertical axis represents the number of physical links with a given stress. The stress is at most one message per physical link for IP Multicast. Under ADCCS-Multilayer, ADCCS and naive Unicast, most links have a small stress-this to be expected. However, the significant lies in the tails of the plots. Under naive Unicast, one link has stress 299. This because that links near the source have high stress. However, ADCCS-Multilayer and ADCCS distribute the stress more evenly across the physical links. ADCCS-Multilayer has about 94% improvement over naive

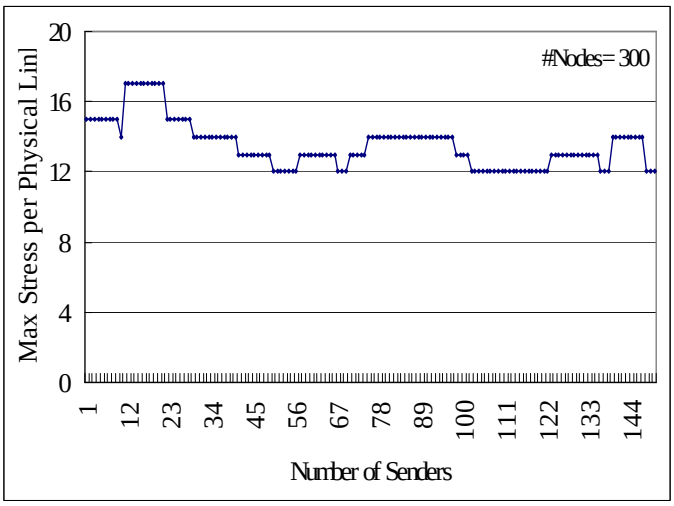

Figure 6.22. Multilayer-CON: Network traffic .

Unicast. ADCCS has about 55% improvement over naive Unicast. The zoom part in Figure 6.21 shows that the ADCCS-Multilayer maximum stress per physical link is 15. Thus, the stress perceived while using ADCCS-Multilayer is close to IP-Multicast. We argue that ADCCS-Multilayer has about 87% improvement over ADCCS to the ADCCS-Multilayer structure that is aware with the latency among the community nodes.

We ran an additional experiment to evaluate the Multilayer-CON traffic when many nodes in the community send an identical message at once. Each node monitors the recent received messages and forward only one copy from the received messages. The simultaneous senders were selected randomly and their number increased from one sender to about 50% senders from the participated nodes in the community network. Figure 6.22 plots the variation of maximum stress per physical link with the number of simultaneous senders. It shows that the increase of the number of senders does not increase the maximum stress per physical link. This result indicates that Multilayer-CON does an efficient job in distributing loads over all links. We conclude that the Multilayer-CON's network traffic does not increase when multiple nodes send an identical message at once.

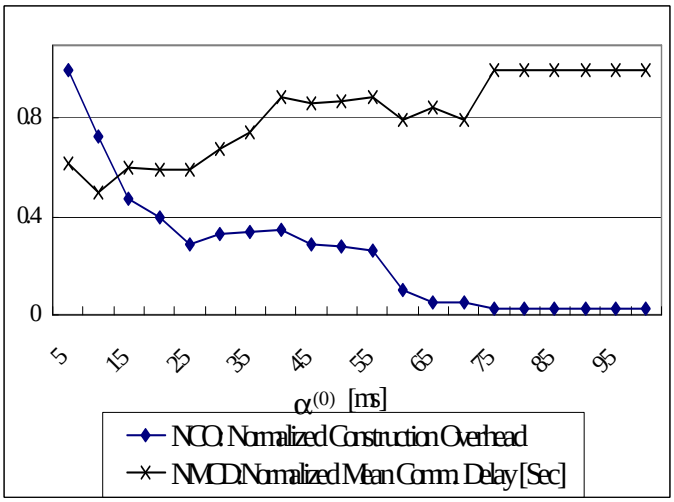

Figure 6.23. Trade-off.

Multilayer-CON: Construction Overhead

The construction overhead is measured as the average number of control messages per physical links that are required to join a new node to the ADCCS-Multilayer. Figure 6.23 shows the variation of $\alpha^{(0)}$ with the normalized mean communication delay (NMCD) and the normalized construction overhead (NCO). It verifies that the trade-off between the community communication delay and the construction overhead exists. The average number of control messages per physical links to add a new node to 300 end-nodes is 1.525 and 0.034, where $\alpha^{(0)}$ is 5 and 100 respectively. The results indicate that the proposed construction technique is scalable and does not show any practical problem.

Multilayer-CON: Fault-tolerance

We have conducted a simulation to verify the effectiveness of the proposed fault-tolerance process over 300 community nodes. Each node sends 20 keep alive-message per second to its neighbors to detect neighbors nodes failure. We define *Failure Detection Frequency*(FDF), the number of keep alive messages per second to detect neighbors nodes failure. Each message size is 20 bytes.

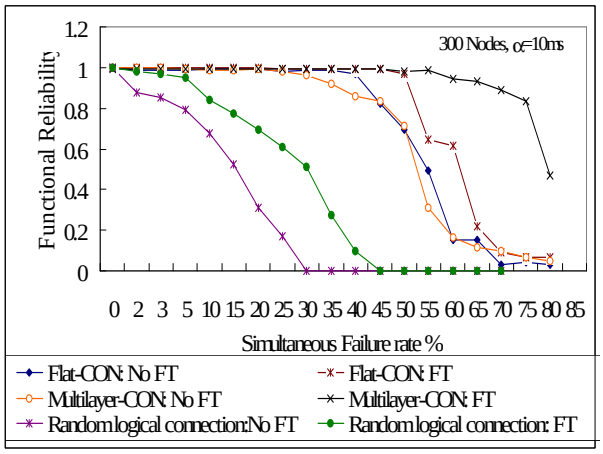

Figure 6.24. Comparison: Flat-CON, Multilayer-CON and Random logical overlay network.

To demonstrate the effectiveness of the proposed fault-tolerance technique we ran an experiment as follows. One community node is randomly selected as a source of a message and a number of nodes are randomly selected to act as failed nodes simultaneously. The failed nodes can be a sub-community leader, disseminator or normal member. Then the *Functional Reliability (FR)*, *fraction of mean number of nodes received a message* is evaluated. Similarly to the simulation environment for Flat-CON at previous section, we assume that the community node-node TCP connections provide data reliability on a hop-by-hop basis that implies that message losses due to network congestion and transmission errors are eliminated. Instead, the main reason for message losses in ADCCS are due to transient network link failures, or node failures. Figure 6.24 shows the variation of the *FR* with the simultaneous failure rate. It shows that Multilayer-CON improves the *FR* about 17% and 60% compared to Flat-CON with fault-tolerance and random logical connection overlay network with fault-tolerance respectively. Moreover it demonstrates that the catastrophic failure of Multilayer-CON starts when about 75% of random nodes simultaneously failed. In addition, Figure 6.25 shows that the Timeliness decreases along with the increasing of the number of Hamilton cycles. The reason is the increasing of Hamilton cycles increases the network traffic. Then the mean communication time increases. However, the Figure 6.25 also shows that the functional reliability FR increases along with the

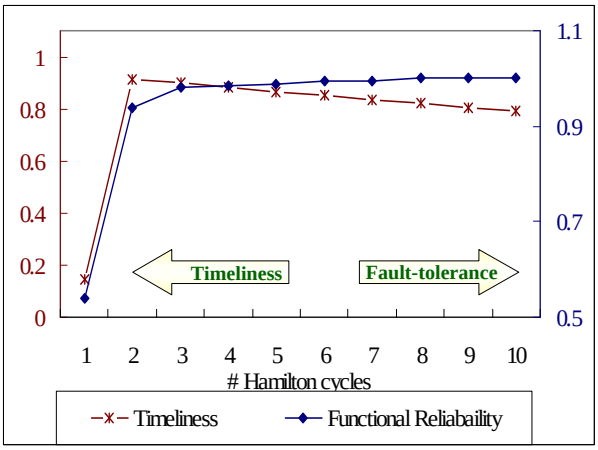

Figure 6.25. Timeliness and Fault-tolerance Tradeoff.

increasing of the number of the Hamilton cycles. Thus, Figure 6.25 gives an evidence of the existence of the trade-off relation between fault-tolerance and timeliness.

6.3 Community Overlay Network Reconstruction Techniques

6.3.1 Self-Adaptable Multilayer-CON

Major design issue of the Multilayer-CON is not only how to build, but also how to adapt the virtual layer that is crucial to the performance of the CON. The main goal of this section is to adapt the changes of the node-to-node latency. Each node periodically processes the following instance of code to monitor its latency to the leader of its sub-community.

*Node*_LatencyCheck()

{

// $MySelf \in S_i^{(0)}$

// Path = { $x_0 = MySelf, x_1, ..., x_m = L_i^{(0)}$ }

$\tau = \sum_{x_i \in Path} \delta(x_i, x_{i+1})$;

if ($\tau > \alpha_i^{(0)}$) {

```
        Wait(W_t);
        τ = Σ_{x_i∈Path} δ(x_i, x_{i+1}) ;
        if (τ > α_i^(0)) {
          Send(L_i^(0),"Ok: Movefrom(S_i^(0))",τ);
          Return 1;
        }
      }
      Else Return 0;
}
```

The Node_LatencyCheck() function is installed in the Neighbors module. It periodically monitors the latency of the node to the $L_i^{(0)}$ if it satisfies the LAC then return 0, otherwise it waits timeout W_t. After W_t, it checks the LAC again. As soon as $L_i^{(0)}$ receives the Movefrom request, it forwards this request Bottom-up and Top-down like join request until find the appropriate sub-community $S_j^{(0)}$ that satisfies the LAC. Then the node moves or migrates to the appropriate sub-community. Otherwise, a new sub-community and/or new level are created. Owing to the proposed adaptable Multilayer-CON, each node can adapt its location in the Multilayer-CON and as a result the community communication delay is improved. In the next sections, the evaluation and simulations give evidence that Multilayer-CON can scale to large number of members.

Adaptable Multilayer-CON: Communication Delay

To simulate the self-adaptable CON technique during the simulation time, we pick about 2% from core links randomly and increase their latencies by 50ms. Figure 6.26 evaluates and compares Multilayer-CON before the change of the latency of about 2% of the core links, after the change of the latency without adaptation and after the change of the latency with node migration technique. Clearly, it presents the effectiveness of the proposed migration (adaptable) technique. Owing to the adaptable Multilayer-CON architecture and migration technique, the adaptable Multilayer-

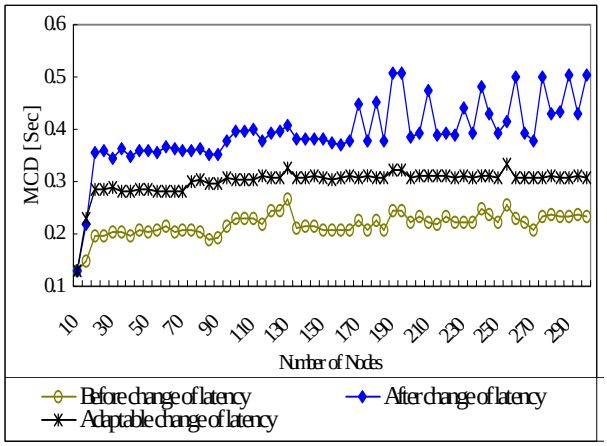

Figure 6.26. Comparison: Adaptable and non-adaptable Multilayer-CON.

CON has shown about 22% improvement of the MCD compared with non-adaptable Multilayer-CON.

6.3.2 Sub-Communities Division/Integration

Why Division/Integration?

This section presents discussion of the tradeoff between join overhead and the communication delay that sustains the needs for a division and integration technique of sub-communities. For the sake of simplicity, we assume that each sub-community at layer j has an equal $\alpha^{(j)} = \alpha_i^{(j)}$ for i=1,..., β_j and $\alpha^{(j+1)} = C \times \alpha^{(j)}$; $C > 1$. Thus the transmission time to forward a message from one node to all nodes is bounded by

$$\alpha^{(k)} + \sum_{j=0}^{k-1} 2 \times \alpha^{(j)} \tag{6.3}$$

We also assume that the maximum communication latency between any two nodes in the community network is τ. Thus, $\tau \leq \beta_k \times \alpha^{(k)} \rightarrow \tau \leq \beta_k * C \times \alpha^{(k-1)} \rightarrow \tau \leq \beta_k * C^2 \times \alpha^{(k-2)} \rightarrow \tau \leq \beta_k * C^k \times \alpha^{(0)}$. Then, the number of layers, k, can be determined as follows:

$$k \geq \log_C(\frac{\tau}{\beta_k \times \alpha^{(0)}}) \tag{6.4}$$

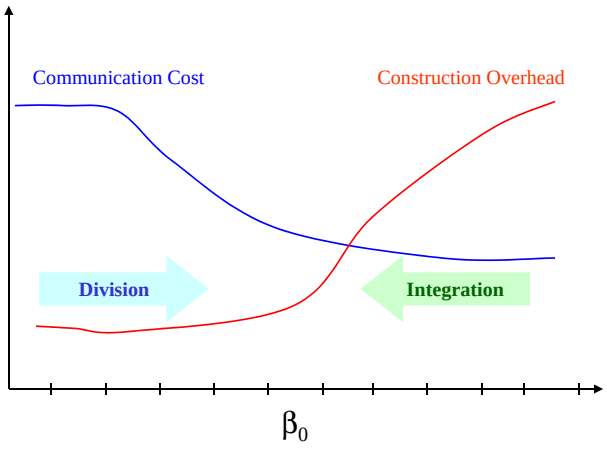

Figure 6.27. Relationship between Communication Delay and Construction overhead with various β_0.

For example, assume τ =120ms, β_k=2, $\alpha^{(0)}$ =10 and C=2, then the number of layers, $k \approx 3$. The join overhead is $O(\beta_0)$ in terms of the number of nodes to contact. It satisfies the following relation.

$$\beta_0 \propto \frac{1}{\alpha^0} \tag{6.5}$$

where $\tau \leq \beta_0 \times \alpha^{(0)}$. In addition, the average number of control messages that are required to add new node to the Multilayer-CON is proportional to β_0. If $\alpha^{(0)}$ increases then the join overhead decreases. Equation 6.3 shows the upper bound of the community communication delay on the proposed multi-layer structure. It can be written as follows:

$$\alpha^0 \times (C^k + 2 \times C^{k-1} + ... + 2) \tag{6.6}$$

where $\alpha^{(j+1)} = C \times \alpha^{(j)}$; C is constant and $C > 1$. Equation 6.6 shows that if $\alpha^{(0)}$ increases then the number of layers, k decreases and consequently the communication delay among community nodes increases. Thus, the increasing of β_0 leads to decrease the communication delay and increase the join overhead. This relation presents a tradeoff between the construction overhead and the community communication delay as shown in Figures 6.23 and 6.27. Consequently, if the number of sub-communities β_0 at layer 0 is small and there exist sub-communities with size $|S_i^{(0)}|$ larger than a threshold value, Thr_u, then this sub-community $S_i^{(0)}$ should be

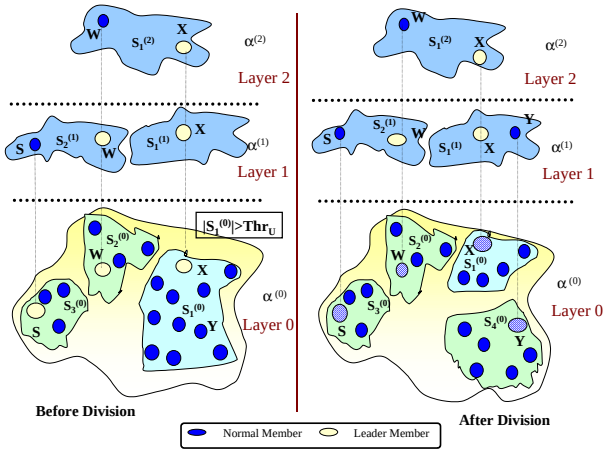

Figure 6.28. Sub-Community division.

divided. In contrast, if the number of sub-communities β_0 at layer 0 is large and there exist sub-communities with size $|S_i^{(0)}|$ and $|S_j^{(0)}|$ smaller than Thr_d then these sub-communities $S_i^{(0)}$ and $S_j^{(0)}$ should be integrated. Next subsections will discusses the sub-community division/integration technologies.

Sub-Community Division Technology

Due to the frequent joins and leaves of community members, the size of sub-communities may increase and the number of sub-communities decrease. Thus, the communication delay will be increased and the join overhead will be decreased. This section clarifies the sub-community division technology to divide the sub-community whose size is larger than threshold value Thr_u. The scenario of division process of the sub-community $S_i^{(0)}$ is as follows. When the leader ($L_i^{(0)}$) detects that $|S_i^{(0)}| > Thr_u$ then it calls an instance of code *Leader_DivProcess()* and autonomously takes a decision either to divide its sub-community or not. Also, the sub-community members process the instance of code *node_DivProc()* when they received the divide request message from the leader. The detailed description of the *Leader_DivProcess()* and *node_DivProc()* subroutines are illustrated in *Appendix B*. Each member then autonomously takes a decision either to move to the new created sub-community or to keep in its current sub-community depend on the LAC. Thus, each member sends

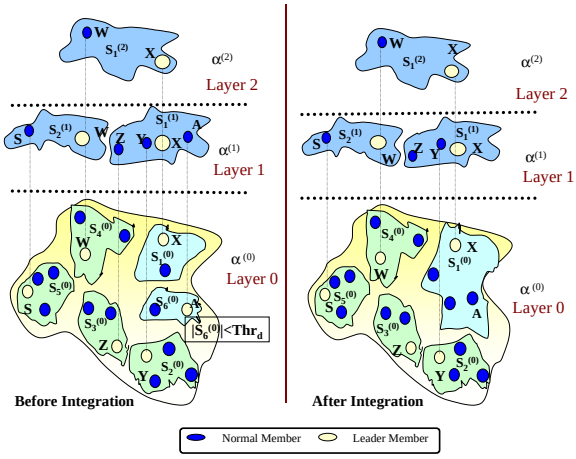

Figure 6.29. Sub-Community integration.

either "*LAC is satisfied*" or "*LAC is not satisfied*" message to $L_i^{(0)}$. $L_i^{(0)}$ sends a *promotion message* to the first replier. The first replier then becomes the leader of the new sub-community $S_{\beta_0+1}^{(0)}$. The members of $S_i^{(0)}$ who did not satisfy the LAC, leave the $S_i^{(0)}$ and join the new sub-community $S_{\beta_0+1}^{(0)}$. Figure 6.28 shows an example of the division process of the sub-community $S_1^{(0)}$ into two sub-communities $S_1^{(0)}$ and $S_{\beta_0+1}^{(0)}$ where $\beta_0 = 3$.

Sub-Communities Integration Technology

Similarly, due to the frequent joins and leaves of community members, the size of sub-communities may decrease and the number of sub-communities may increase. Thus, the communication delay will be decreased and the join overhead will be increased. This section clarifies the sub-community integration technology to integrate two sub-communities $S_i^{(0)}$ and $S_j^{(0)}$, where the size of both sub-communities is less than threshold value Thr_d. For example, Figure 6.29 shows the integration process of $S_1^{(0)}$and $S_6^{(0)}$ sub-communities, where ($|S_1^{(0)}| < Thr_d$ and $|S_6^{(0)}| < Thr_d$). The scenario of the integration process is as follows. When the leader $L_i^{(0)}$ detects that $|S_i^{(0)}| < Thr_d$ then it calls an instance of code *Leader_InitInteg()* to initiate the integration process. $L_i^{(0)}$ autonomously takes a decision either to integrate with the neighbor's sub-community or not. The *Leader_InitInteg()* subroutine is illustrated in *Appendix*

B. If integration is required then the $L_i^{(0)}$ sends integration requests to its neighbors. As soon as a leader $L_j^{(0)}$ receives integration requests it calls the instance of code *Sub_IntegProcess()* that is also illustrated in *Appendix B*. If the leaders $L_i^{(0)}$ and $L_j^{(0)}$ take a decision to integrate their sub-communities then ($L_i^{(0)}$) one of them becomes a leader of the integrated sub-communities. The leader $L_i^{(0)}$ then sets $\alpha_i^{(0)} = \alpha_i^{(0)} + \alpha_j^{(0)}$. $L_j^{(0)}$ then sends a message to all nodes in $S_j^{(0)}$ to change their sub-community identifier, becomes normal node and then leaves the sub-community at the upper layer. Thus, the number of sub-communities becomes $\beta_0 - 1$. Figure 6.29 shows that node X calls *Leader_InitInteg()* subroutine and node A is the first replier to node X. Node A calls the *Leader_IntegProcess()* subroutine. Then node X sends acknowledgement of the integration between $S_1^{(0)}$ and $S_6^{(0)}$. When node A receives the acknowledgement, it forward the $S_1^{(0)}$ identifer to all members in $S_6^{(0)}$. Node A then becomes normal node and leaves form the sub-community of leader at the upper layer.

6.4 Summary

This chapter elucidated the construction technologies of the community overlay network. These technologies confronts the challenges that mentioned in chapter 4. The construction and maintenance techniques of both Flat-CON and Multilayer-CON achieve the following requirements: *scalable online-expansion* and *fault-tolerance*. Flat-CON is a *self-organized* overlay network that is constructed and maintained with low complexity. We studied the efficiency of the proposed community communication and the autonomous community overlay network construction techniques by evaluating the communication delay, fault-tolerance overhead and network traffic associated with Flat-CON. The results show that Flat-CON can achieve the *timeliness scalable Online-expansion* and *fault-tolerance* of the large-scale information-dissemination systems under the homogenous end-node to end-node latency assumption. However, in the Internet environment the end-node to end-node latency is heterogeneous. The question then is: can ADCCS support large number of members with different communication delay? The main concern of Multilayer-CON is to answer this question. Multilayer-CON considered latency between community nodes

as an important criterion that need to be optimized. It organized the community overlay network into a number of homogenous sub-communities. To reduce both the communication delay among community nodes and the join overhead, this chapter presented a novel multi-layer structure, ADCCS-Multilayer, of sub-communities. Furthermore, this chapter described how to step-step construct, maintain and adapt sub-communities and the multi-layer structure as well. For adapting the network latency changes, this chapter presented the self-adapting Multilayer-CON technology. Then this chapter demonstrated the Sub-Community division/integration technology to cope with the dynamic change of the network. Finally, this chapter presented the simulation results that show the effectiveness of the proposed techniques. The results confirm that Multilayer-CON achieves the *timeliness*, *scalable online-expansion* and *fault-tolerance* of the large-scale information-dissemination systems in real Internet settings with heterogeneous end-node to end-node latency assumption with maintaining the community nodes autonomy. Moreover, the results give an evidence of the existence of the trade-off relation between fault-tolerance and timeliness.

7 EVALUATION

7.1 Qualitative Analysis

The advantages of autonomous decentralized community communication system are compared hereafter with some application level multicast communication systems.

7.1.1 Approaches

A literature review of related works that tackles the problem of constructing overlay networks for efficient information-dissemination of the application level multicast systems is illustrated as follows. In fact all overlay networks organize the group members into two topologies. First, *control topology* is the communication environment among members for membership management. Second, *data topology* is the communication media for data dissemination to all members in the overlay network. Depending on the sequence of construction of the control and data topologies, this section classifies the different overlay networks into different three approaches: *tree-first*, *mesh-first* and *implicit* approaches as shown in Figure 7.1. The tree-first approach includes ALMI [50], YOID [76] and Overcast [63]. It constructs a shared data delivery tree first directly. A drawback of using a shared tree is that failure of a single application may cause a partition of the overlay topology. Moreover, each member knows a few other members of the group that are not its neighbors on the overlay tree and establishes and maintains additional links to these members. Thus, the data delivery tree with these additional control links forms the control topology. In contrast, overlay networks based on the mesh-first approach firstly organize the group members into the overlay mesh topology. This approach includes Narada [51] and Scatercast [67]. Each member in this control topology computes unique overlay paths to every other member. Thus, a source specific data tree rooted at any member can be created. A drawback of the mesh-first approach is that the calculation of spanning trees requires running a multicast routing protocol (e.g., distance vector multicast routing protocol DVMRP [52]) within the overlay that adds complexity to

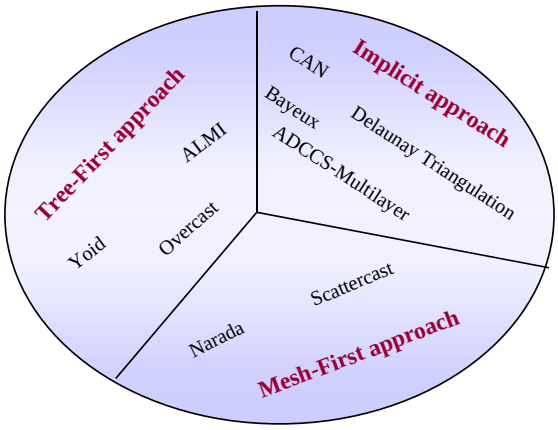

Figure 7.1. Application level multicast: Overlay Construction approaches.

the overlay network. Protocols [64,77,78] using the implicit approach creates a control topology that implicitly defines the data delivery path with some specific properties. For example, the overlay in CAN multicast [77] the logical addresses are obtained from n-dimensional *Cartesian* coordinates on an *n-torus*. An advantage of building overlay networks with logical addresses is that, for good choices of the address space and the topology, next hop routing information for unicast and multicast transmissions can be encoded in the logical addresses. A disadvantage of building overlay networks with logical addresses is that, the overlay network may not be a good match for the network topology. Some researchers have focused only on the performance of the application level multicast communication systems that could be improved if the overlay network topology is congruent with the underlying IP-level topology [73], but this is not the approach we have decided to follow. To our knowledge, Multilayer-CON is the first overlay network that considers the latency between nodes in order to optimize the end-end communication delay and take advantage from the node-node latency heterogeneity. The problem of constructing an optimal overlay network is known to be NP-hard [67, 74].

Table 7.1 compares different approaches of the current application level multicast (e.g. ALMI, Narada, Scatercast and CAN) that eases the data dissemination-oriented applications. It compares the membership management techniques, overlay network

Table 7.1
A comparison of data dissemination systems

	ALMI	Narada	Scattercast	CAN	ADCCS
Membership Management	Centralized	Decentralized	Distributed (Proxies)	Decentralized	
Approach	Tree-first	Mesh-first		Implicit	
				Coordinate Space	Latency Awareness
Tree-type	Shared	Source-specific			Shared
Communication	Address-based			Content-based	
Protocol	Unilateral unicast				Multilateral $1 \rightarrow N$

construction approaches, and communication based and protocols. Each node in the ADCCS considers the node to node communication delay during the community overlay network construction. In contrast, each node in ALMI, Narada, Scatercast and CAN does not consider the node to node communication during the construction of their overly networks.

7.1.2 Discussions

ADCCS-Multilayer, *ADCCS*, like *Overcast* [63], *Narada* [51] and *ALMI* [50], implement multicast, uses a self-organized overlay network and assume only Unicast support from the underlying network layer. Narada and ALMI target collaborative applications with a small number of group members. However, ADCCS-Multilayer and ADCCS are frameworks for collaborative applications with a large number of group members. ALMI is centralized overlay construction protocol that uses the tree-first approach. In this approach, a shared content delivery tree is constructed. It relies on a recursive algorithm to enhance the tree. Clearly, it constitutes a single point of failure for all control operations related to the group. Narada is distributed overlay construction protocol that uses the mesh-first approach. In this approach, ev-

ery member should keep a full list of all other members. Therefore, both ALMI and Narada approaches do not scale well to the large group sizes. In contrast, ADCCS-Multilayer takes a decentralized approach: no node knows the total system as shown in previous chapters. In addition, ADCCS-Multilayer creates a control multi-layer topology considering the latency awareness condition. The content delivery path is implicitly defined on this multi-layer topology. Thus, the ADCCS-Multilayer is scalable for large number of members. *Scattercast* [67] and *Overlay Multicast Network Infrastructure* OMNI [66] are designed for global content distribution. They argue for infrastructure support, where proxies are deployed in the Internet to support large number of users. For large-scale data distributions, such as live web casts, a single source exists. In contrast in the ADCCS, the nodes are considered to be equal peers and are organized in the community network. The community concept is a "real" end-system multicast approach. The end-systems (autonomous members) work cooperatively to deliver the data on the whole community members. ADCCS is dedicated for multi-sender applications with large number of participants. It does not depend on the multicast support by the routers (e.g. IP multicast) and does not depend on the multicast service nodes MSNs (e.g. Scattercast and OMNI). A rapid and sharp surge in the volume of requests arriving at MSN often leads to a flash crowd. Clearly, MSN constitutes a single point of failure for information provisions to the group. Scattercast, like Narada takes a mesh-based approach to the tree creation problem. Therefore, Scattercast does not scale well to the large group sizes. In the other side, the ADCCS scales well to the large number of members because each member is required to know a small number of other members (neighbors). The proposed ADCCS-Multilayer and ADCCS are framework for both information sharing and large-scale data distribution applications. Some other recent projects like CAN [62] have also addressed the scalability issue in creating the overlay network. CAN defines a virtual d-dimensional cartesian coordinate space, and each node owns a part of this space. Contrary to ADCCS-Multilayer, CAN did not consider the node-node latency as an important criteria that should be optimized. Therefore, CAN may suffer from performance degradation if some intermediate nodes slow down entire network branches.

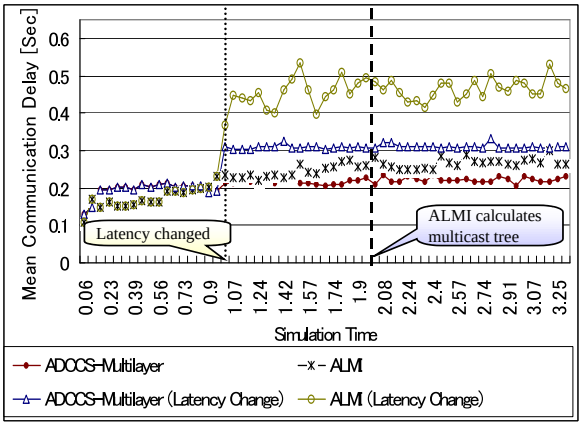

Figure 7.2. Comparison: ALMI and ADCCS-Multilayer.

Both ADCCS and CAN nodes maintain constant state for other members and as a result exchange a constant number of periodic messages. However, this overhead in Bayeux is logarithmic.

7.2 Quantitative Analysis

The goal of this section is to quantitatively study the ADCCS-Multilayer and one of the application level multicast such as ALMI. For comparison, we have implemented the entire, ALMI protocol from the description given in [50]. The ALMI protocol is a *tree-first* application-level multicast approach. In this approach, each node monitors a constant number of neighboring nodes from the total number of nodes. In this simulation each node monitors 2% nodes from the total number of nodes in the system. Periodically, these nodes send a report message to the session controller. Thus, the controller can build a sub-optimal *Minimal Spanning Tree* (MST) for efficient communication. In this simulation, ALMI controller calculates the multicast tree every 2 minutes. Moreover, we have conducted a simulation over transit-stub network model with the same setup parameters as used in chapter 6. After one minute from the simulation time, we randomly selected about 2% from core links and increased their latencies by 50ms. Figure 7.2 shows the variations of the Mean Communication Delay for both ALMI and ADCCS-Multilayer with simulation

time. For small number of members ADCCS-Multilayer is not effective compared to ALMI but ADCCS-Multilayer reveals meaningful results when the total number of members in the community increases sharply. After one minute of the simulation, 2% from core links increase their latencies by 50ms. ADCCS-Multilayer adapts these changes quickly however ALMI periodically (2 Minutes) monitors the changes and then constructs new multicast tree. ADCCS-Multilayer has shown about 19% improvement of the MCD compared to ALMI.

7.3 Summary

This chapter presented a review of related works and study their approaches for construction the overlay networks. In this chapter, we discussed the other approaches to demonstrate the strength and weakness in compare with the ADCCS. Finally this chapter demonstrated the effectiveness of the ADCCS compared to ALMI, one of the best application level multicast system. Thus, we conclude that the ADCCS achieves the timeliness.

8 CONCLUSIONS

8.1 Summary

The accelerating progress in both economy and new information and communication technology are radically modifying the ways in which we use communication. Over the last few years, there have been dramatic changes in the computing and communication landscape. The combination of these changes is bringing forth a number of new and interesting challenges forcing us to re-evaluate how to design and implement large-scale decentralized information-dissemination systems.

This thesis focuses on clarifying a new paradigm for information-dissemination systems in a heterogeneous and highly dynamic environment such as *Internet*. The dissertation illustrates the concept, architecture and real-time and fault-tolerance oriented technologies of *Autonomous Decentralized Community Communication System* ADCCS. The main contributions of this research are:

- Proposition of *ADCCS* concept in information-dissemination services (right me, right time and right location) bases on the leading *Autonomous Decentralized Concept*. In contrast to the traditional anyone-anywhere-anytime model, ADCCS's concept is to provide specific users with information at specific place and time.

- Proposition of autonomous decentralized community communication architecture. It is loosely connected and controlled mass of nodes. Each node autonomously controls itself and coordinates with the others nodes to disseminate information timely and reliably. The architecture is fully decentralized model, the community membership management doesn't rely on any centralized authority. This is key to a scalable system for a large number of members.

- Proposition of service-oriented and multilateral autonomous decentralized community communication technology. It makes a shift from traditional unilateral and address-oriented dissemination systems to multilateral and service-oriented

dissemination systems. In addition, it realizes a flexible, a timely and a productive cooperation among members.

- Proposition of autonomous self-organized and self-adaptable *Community Overlay Network* (CON) construction, reconstruction and maintenance techniques. These techniques allow users to join with low complexity and with considering node-node latency to enhance the communication delay. Moreover, they allow members to detect, recover a node/link failure with low complexity and to adapt the network changes. Thus these techniques achieve:
  - Scalable Online-expansion,
  - Fault-tolerance,
  - Timeliness and adaptability,

  under dynamic and heterogeneous environment without requiring any knowledge of the total system structure. Then, CON enables the efficient information-dissemination at the application layer without any support from routers.

- Quantitative comparison of autonomous decentralized community communication system (ADCCS) with unicst, IP Multicast and ALMI.

- Qualitative comparison of ADCCS with the other conventional information-dissemination systems such as Narada, ALMI and others.

In accordance with the assumption stated in section 6.1 (homogenous node-node latency), a time analysis was elucidated. This analysis manifests that the mean communication time introduced by the ADCCS's communication technology over Flat-CON is improved about 93 percent compared to unicast and 87 percent compared to unicast using caching proxies with hit rate 50%. However, section 6.2 reveals the time analysis according to the realistic assumption (heterogeneous node-node latency). This analysis manifests that the mean communication time introduced by the ADCCS's communication technology over Multilayer-CON is improved about 93 percent compared to unicast and 39 compared to ADCCS over Flat-CON. Moreover,

a study of stress per physical links was conducted to evaluate the load per link during the running of ADCCS. The result of this study shows that under ADCCS performed over Multilayer-CON, the maximum stress per physical link of is 15. In contrast, it is 299 under naive Unicast. Thus, the stress perceived while using ADCCS is close and converges to IP-Multicast. From these results we conclude that the ADCCS's over Multilayer-CON enhances the community communication in realistic Internet settings compared to the ADCCS over FLAT-CON. Thus, the timeliness is achieved. In addition, we analyze the fault-tolerance and Timeliness and then show the existence of the tradeoff relation between timeliness and fault-tolerance. Moreover, this analysis manifests that the Functional Reliability FR introduced by the ADCCS's fault-tolerance process over Multilayer-CON is improved about 17 and 60 percent compared to Flat-CON with fault-tolerance and random logical overlay network with fault-tolerance respectively.

Beyond the promising performance results, the major fulfillments of the autonomous decentralized communication system are tenant in the community concept, communication technology and the construction, reconstruction and maintenance technologies of the community overlay network. These proposed technologies achieve scalable online-expansion, timeliness, adaptability and fault-tolerance to the system while maintaining the autonomy of the system's components.

8.2 Future Work

We believe that the proposed technologies and simulations that are realized during this thesis have been a nucleus for further research and development of large-scale information-dissemination systems. We thought this thesis is the base stone of this research direction and raised some important trends and challenges for new generations of students to follow this direction and enlarge it. Due to the time scale of a Ph.D., this thesis has been limited to disseminate non-multimedia data. Streaming media delivery applications have motivated increasing interests. In large-scale and dynamic environments, they require more difficult real-time requirements that cur-

rently cannot fulfil. Moreover, the wireless and mobility are added dimensions to a streaming media network.

Internet-based telephony - Voice-over-IP - (VoIP) has been around for years but has not reached the mainstream market. These systems uses centralized technologies to route calls through firewalls or *Network Address Translations*. The result is that companies operating such services typically allocate very little resources on their servers per user which seriously degrades the call quality. *Skype* [79] is the first P2P telephony system. Utilizing our experience in this thesis to leverage all of the available resources in a network without the need for costly centralized resources with achieving service level agreement and maintaining the autonomy of each subsystem.

Security is also an important issue that should be handled with maintaining the autonomy of the system's components. Future work is to develop secure autonomous decentralized information-dissemination systems that realizing the following general goals: trustworthy encryption, data integrity, data confidentiality, access control and robustness against denial-of-service attacks.

Publications

I. Journals & Magazines

1. K. Ragab, N. Kaji and K. Mori, "Scalable Multilateral Autonomous Decentralized Community Communication Technique for Large-Scale Information Systems," IEICE TRANS. COMMUN. Vol. E87-B, No. 3, pp. 660-670, March 2004.

2. K. Ragab, N. Kaji and K. Mori, "ACIS: A large-scale Autonomous Decentralized Community Communication Infrastructure," IEICE TRANS. INFO.& SYST. Vol. E87-D, No. 4, pp. 936-946, April 2004.

3. K. Ragab and K. Mori, "ACIS-Hierarchy: Enhancing Community Communication Delay for Large-Scale Information Systems," IEICE Trans. COMMUN. Vol. E87-B, No. 7, pp. 1797-1805, July 2004.

4. K. Ragab N. Kaji, Y. Horikoshi, H. Kuriyama and K. Mori, "Autonomous Decentralized Community Communication for information dissemination," IEEE Internet Computing Magazine Vol. 8 No. 3, pp. 29-36, May/June 2004.

II. International Conferences

First Author

1. K. Ragab, T. Ono, N. Kaji, and K. Mori, "Community Communication Technology for achieving Timeliness in Autonomous Decentralized Community Systems," Proc. IEEE CS of the 2nd International Workshop on Autonomous Decentralized System (IWADS 2002), pp. 56-60, Beijing, China.

2. K. Ragab, T. Ono, N. Kaji, K. Mori, " Autonomous Decentralized Community Concept and Architecture for a Complex Adaptive Information System," Proc. IEEE CS of FTDCS, pp. 9-15, Puerto Rico, May 2003.

3. K. Ragab, T. Ono, N. Kaji, K. Mori, "An Efficient Communication Technology for Autonomous Decentralized Community Information System," Supplement Proc. IEEE CS of ISADS, pp. 7-8, Pisa, Italy, April 2003. (Fast abstract)

4. K. Ragab, N. Kaji, K. Mori, "Scalable Multilateral Communication Technique for Large-Scale Information Systems," Proc. IEEE of COMPSAC, pp. 222-227, Dallas, USA, Nov. 2003.

5. K. Ragab, N. Kaji, K. Mori, "Service-Oriented Autonomous Decentralized Community Communication Technique for a Complex Adaptive Information System," Proc. IEEE/WIC CS of WI'2003, pp. 323-329, Halifax, Canada, October 2003.

6. K.Ragab, N.Kaji, K. Anwar, Y. Hirokoshi, H. Kuriyama and K. Mori, "A Novel Multilayer Community Architecture With End-End Delay Awareness for Communication Delay Enhancement," Proc. IEEE CS of SAINT'2004, pp. 43-49, Tokyo, Japan, Jan., 2004.

7. K. Ragab, Y. Horikoshi, H. Kuriyama and K. Mori "Multilayer Autonomous Community Overlay Network for Enhancing Communication Delay ", Proc. IEEE CS of ISCC, pp. 987-992, Alexandria, Egypt, June 2004.

Coauthor

1. T. Ono, K. Ragab, N. Kaji, and K. Mori, "Service-oriented Communication Technology for Achieving Assurance", Proc. IEEE CS of the 22nd International Conference on Distributed Computing Systems Workshops, pp. 69-74, Vienna, Austria 2002.

2. T. Ono, K. Ragab, N. Kaji, and K. Mori, "Autonomous Cooperation Technique to Achieve Fault tolerance in Service oriented Community System", Proc. IEEE CS of the 2nd International Workshop on Autonomous Decentralized System (IWADS 2002), pp. 84-89, Beijing, China.

3. N. Kaji, K. Ragab, T. Ono, and K. Mori, "Autonomous Synchronization Technology for Achieving Real Time Property in Service Oriented Community System", Proc. IEEE CS of the 2nd International Workshop on Autonomous Decentralized System (IWADS 2002), pp. 16-21, Beijing, China.

4. N. Kaji, K. Ragab, T. Ono, and K. Mori, "Service Oriented Community Systems for Mobile Commerce", International Conference on Advances in Infrastructure for e-Business on the Internet (SSGRR 2002s), Italy, 2002.

5. N. Kaji, K. Ragab, T. Ono, and K. Mori, "Autonomous Cooperation Technologies for Achieving Real Time Property and Fault Tolerance in Service Oriented Community System", Proc. IEEE CS of the 23nd International Conference on Distributed Computing Systems, pp. 36-41, Rhode Island USA May, 2003.

6. Y. Hirokoshi, K. Ragab, N. Kaji, K. Mori, "Service Discovery Technology in Autonomous Decentralized Community System", Proc. IEEE/IEICE of the APSITT'2003, New Caledonia, Nov. 2003.

7. T. Ono, N. Kaji, Y. Hirokoshi,H. Kuriyama, K. Ragab and K. Mori "Autonomous Decentralized Community Construction Technology to Assure Quality of Services ", Proc. IEEE CS of FTDCS, pp. 299-305, China, May 2004.

III. Domestic Conferences

1. K. Ragab, T. Ono, K. Mori, "High Assurance Communication Technique for Autonomous File Sharing Community", Assurance Kenkyukai'02, pp. 9-16, Tokyo Metro. University, Tokyo, Japan.

2. Y. Hirokoshi, K. Ragab, T. Ono, N. Kaji, K. Mori, "Autonomous Service Discovery Technology To achieve Assurance in Community", Assurance Kenkyukai'03, pp. 59-66, Hiroshima City University, Japan.

3. T. Ono, K. Ragab, N. Kaji, Y. Hirokoshi, K. Mori, "Autonomous Construction Technology in Community to Assure the Quality of Services", Assurance Kenkyukai'03, pp. 51-58, Hiroshima City University, Japan.

REFERENCES

[1] L. G. Roberts. A Beyond Moore's law: Internet Growth Trend. *IEEE Computer*, 33:117, 2000.

[2] Peter Lyman and Hal R. Varian. How Much Information. *Journal of Electronic Publishing*, 6:25, 2000.

[3] David R. Cheriton. Dissemination-oriented Communication Systems. Technical report, Computer Science Dept., Stanford University, 1992.

[4] S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transaction on Computer Systems*, 8(2):85–110, May 1990.

[5] S. Deering. *Multicast Routing in Datagram Internetworks*. Ph.d., Stanford University, May 1991.

[6] K. Almeroth. The Evolution of Multicast: From the MBone to Inter-domain Multicast to Internet2 Deployment. *IEEE Network*, 14:10–20, January/February 2000.

[7] N. Kaji T. Ono, K. Ragab and K. Mori. Service-oriented Communication Technology for Achieving Assurance. In *IEEE CS Proc. 22nd ICDCS (ADSN workshop)*, pages 69–74, 2002.

[8] T. Ono N. Kaji, K. Ragab and K. Mori. Autonomous Cooperation Technologies for Achieving Real Time Property and Fault Tolerance in Service Oriented Community System. In *IEEE CS Proc. 23rd ICDCS (ADSN workshop)*, pages 36–41, 2003.

[9] Shaun Terry. *Enterprise JMS Programming*. John Wiley Publication, 2002.

[10] K. Mori Et. al. Proposition of Autonomous Decentralization Concept. *Journal of IEE Japan*, 104(12):303–310, 1994. (Japanese).

[11] K. Mori. Autonomous Decentralized Systems: Concept, Data Field Architecture and Future Trends. In *IEEE CS Proc. ISADS'93*, pages 28–34, 1993.

[12] Ray-Shang Lo. *The Embedding of Hamiltonian Paths in Faulty Arrangement Graphs*. Ph.d., National Taiwan University, 2000.

[13] I. Harrington. *Organizational Structure and Information Technology*. Prentice-Hall London, UK, 1991.

[14] M. Solman and J. Kramer. *Distributed Systems and Computer Network*. Prentice-Hall, 1987.

[15] H. Zimmerman. OSI Reference Model-the ISO Model of Architecture for Open Systems Interconnection. *IEEE Trans. on Communications*, 28(4):425–432, 1980.

[16] W. Stallings. *Networking Standards: A Guide to OSI, ISDN, LAN and MAN Standard*. Addison-Weseley, 1993.

[17] M.K. Buckland. What is a document? *Journal of the American Society of Information Science*, 48(9):804–809, 1997.

[18] ISC internet domain survey. http://www.isc.org/index.pl?/ops/ds/, January 2003.

[19] $33 Billion in Online Advertising by 2004. Forrester Research, Meta Group - http://WWW.forrester.com, August 1999.

[20] J. Han and M. Kamber. *Data Mining: Concepts and Techniques,*. Morgan Kaufmann, August 2000.

[21] A. Ankolekar. DAML-S: Web Service Description for the Semantic Web. In *Proc. Semantic Web Conf. ISWC'02*, 2002.

[22] David Gourley and Brian Totty. *HTTP: The Definitive Guide*. O'Reilly & Associates, 2002.

[23] T. Lindholm and F. Yellin. *The Java(TM) Virtual Machine Specification*. (Addison Wesley), 1997.

[24] A. D. Birrel and B. J. Neslon. Implementing Remote Procedure Calls. *ACM Trans. on Computer Systems*, 2:39–59, 1984.

[25] Errol Simon. *Distributed Information Systems: From Client/Server to Distributed Multimedia*. McGraw-Hill publishing Company, England, 1996.

[26] Andy Oram. *Peer-to-Peer Harnessing the Power of Disruptive Technologies, PUBLISHER =*.

[27] A Iamnitchi M, Ripeanu and I. Foster. Mapping the Gnutella Network. *IEEE Internet Computing*, 6:50–57, January 2002.

[28] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *ACM Proc. SIGCOMM'01*, pages 149–160, 2001.

[29] A. Susarla M. Parameswaran and A.B. Whinston. P2P Networking: An Information-Sharing Alternative. *IEEE Computer*, 43(7):31–38, 2001.

[30] Joe Malcolm Kurt J. Lidl, Josh Osborne. Drinking from the Firehose: Multicast USENET News. In *USENIX Conference Proceedings*, San Francisco, CA, 1994. USENIX.

[31] A. S. Tanenbaum. *Distributed Operating Systems*. Prentice-Hall, Inc., 1995.

[32] Tai Jin Martin Arlitt. Workload Characterization of the 1998 World Cup Web Site. Hewlett Packard Co., 1999.

[33] J.Jung and Et al. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *Proc. WWW'02*, Hawaii, USA, 2002.

[34] S. K. Shrivastava David B. Ingham and F. Panzieri. Constructing Dependable Web Services. *IEEE Internet Computing*, 4(1):25–33, 2000.

[35] H. Nakanishi K. Ohmachi K. Mori, S. Yamashita and Y. Hori. Service Accelerator (SEA) System for Supplying Demand Oriented Information Services. In *IEEE CS Proc. ISADS'97*, pages 129–136, 1997.

[36] R. Paul I.L. Yen and K. Mori. Towards Integrated Methods for High Assurance Systems. *IEEE Computer*, 31(4):32–34, 1998.

[37] K. Mori. Applications in Rapidly Changing Environments. *IEEE Computer*, 31(4):42–44, April 1998.

[38] Sape Mullender. *Distributed Systems*. Addison-Wesley, 1995.

[39] Steve Steinke. Middleware Meets the Network. *LAN: The Network Solutions Magazine*, 10(13):56, 1995.

[40] H. Eriksson. Mbone: The multicast backbone. *Communications of the ACM*, 37(8):54–60, 1994.

[41] S. Casner and S. Deering. First IETF Internet Audiocast. *ACM Computer Communication Review*, 22(3):92–97, July 1992.

[42] Injong Rhee, Nallathambi Ballaguru, and George N. Rouskas. MTCP: Scalable TCP-like Congestion Control for Reliable Multicast. In *IEEE CS Proc. INFOCOM*, March 1999.

[43] Luigi Rizzo. PGMCC: A TCP-friendly Single-rate Multicast. In *ACM Proc. SIGCOMM*, pages 17–28, 2000.

[44] J. C. Lin and S. Paul. RMTP: A Reliable Multicast Transport Protocol. In *IEEE CS Proc. INFOCOM'96*, pages 1414–1424, San Francisco, CA, March 1996.

[45] J. W. Wong and M. H. Aremar. Analysis of Broadcast Delivery in a Videotex System. *IEEE Transactions on Communications,*, 34(9):863–866, 1985.

[46] K. C. Lee G. E. Herman, G. Gopal and A. Weinrib. The Datacycle Architecture for Very High Throughput Database Systems. In *ACM SIGMOD*, pages 97–103, San Francisco, CA, May 1987.

[47] H. Garcia-Molina T. Yan. SIFT- A Tool for Wide-area Information Dissemination. In *Proc. USENIX Technical Conf.*, pages 177–186., 1995.

[48] Alex Siegel B. Oki, M. Pfluegl and Dale Skeen. The Information Bus: An Architecture for Extensible Distributed Systems. In *Proc. 14th SOSP*, December 1993.

[49] D. S. Roseablum A. Carzaniga and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transaction on Computer Systems*, 13(3):332–383, August 2001.

[50] D. Verma D. Pendarakis, S. Shi and M. Waldvogel. ALMI: An Application Level Multicast Infrastructure. In *Proc. of 3rd Usnex Symp. on Internet Technologies and Systems*, pages 49–60, San Francisco, CA, USA, March 2001.

[51] S. Seshan Y. Chu, S. G. Rao and H. Zhang. A Case for End System Multicast. *IEEE Journal on Selected Areas in Communication (JSAC)*, 20(8):1456–1471, October 2002.

[52] C. Partridge D. Waitzman and S. Deering. Distance Vector Multicast Routing Protocol. Technical report, RFC 1075, November 1998.

[53] M. Koizumi and K. Mori. Autonomous Coordinability of Decentralized System considering Subsystems Failures. In *Proc. 7th Int. Conf. On Multiple Criteria Decision Making*, Kyoto, Japan, 1986.

[54] M. Matsumoto and Et. al. Development of the Autonomous Decentralized Train Control System. *IEICE Trans. Communication*, E84-D(10):1333–1340, October 2001.

[55] G. H. Hillery. Defination of Community: Areas of Agreement. Rural Sociology, 1955.

[56] R. M. MacIver. Community. Macmillan, 1917.

[57] M. Jarke F. Matthes J. Mylopoulos M. P. Papazoglou K. Pohl J. Schmidt C. Woo E. Yu G. De Michelis, E. Dubois. *Cooperative Information Systems: A Manifesto*. Cooperative Information Systems: Trends & Directions, Academic-Press, New York, 1998.

[58] Toru Ishida, editor. *Community Computing: Collaboration over Global Information Networks*. John Wiley & Sons Ltd., 1998.

[59] H. Ihara Et al. K. Mori. Autonomous Decentralized Software Structure and its Application. In *IEEE CS Proc. FJCC'86*, November 1986.

[60] D. Reed J. Saltzer and D. Clark. End-to-End Arguments in System Design. *ACM Trans. on Computer Systems*, 2(4):195–206, 1984.

[61] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965.

[62] S. Ratnasamy and Et al. Scalable Content-Addressable Network. In *Proc. Of SIGCOMM'01*, California, USA., 2001.

[63] K. Johnson M. Kaashoek J. Jannotti, D. Gifford and Jr. J. O'Toole. Overcast: Reliable Multicasting with as n Overlay Network. In *Proc. 4th Sym. OSDI*, pages 197–212, October 2000.

[64] A. D. Joseph R. H. Katz S. Q. Zhuang, B. Y. Zhao and J. D. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In *Proceedings of NOSSDAV*, Port Jefferson, NY, USA, June 2001.

[65] Secure hash standard. Technical Report FIPS 180-1, National Inst. Of Standards and Technology, US Dept of Commerce, Washington D.C., April 1995.

[66] K. Kor B. Bobb S. Banerjee, C. Kommareddy and S. Khuller. Construction of an Efficient Overlay Multicast Infrastructure for Realtime Applications. In *IEEE Proc. of INFOCOM*, 2003.

[67] Y. Chawathe. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*. PhD thesis, University of California, Berkeley, Dec. 2000.

[68] S. H. Strogatz M. E. J. Newman and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev.*, E64, 2001.

[69] B. Jackson and H. Li. Hamilton cycles in 2-connected k-regular bipartite graphs. *Comb. Theory Series A*, 44(2):177–186, 1998.

[70] J.A. Bondy and U.S. R. Murty. *Graph Theory with Applications*. Macmilliam Press Ltd., 1976.

[71] Granbaum B. and Malkevitch J. Pairs of Edge-disjoint Hamiltonian Circuits. *Aequationes Math*, 14(1/2):191–196, 1976.

[72] Stephen Williams. Caching Proxies: Limitations and Potentials. In *Proc. 4th Int. World Wide Web Conference*, Dec. 1995.

[73] S. Ratnasamy and Et al. Topologically-aware Overlay Construction and Server Selection. In *IEEE Proc. INFOCOM*, New York, 2002.

[74] M.R. Garrey and Et al. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, San Francisco, CA, 1979.

[75] E. W. Zegura and Et al. How to model an Internetwork. In *IEEE CS Proc. INFOCOM*, San Francisco, 1996.

[76] P. Francis. Yoid: Extending the Internet Multicast Architecture. Technical report, AT&T Center for Internet Research at ICSI (ACIRI), April 2000.

[77] S. Ratnasamy and Et al. Application-level Multicast using Content-Addressable Networks. In *Proc. 3rd Int. Workshop on Networked Group Communication (NGC'01)*, London, U.K., 2001.

[78] Michael Nahas Jorg Liebeherr and Weisheng Si. Application-Layer Multicasting With Delaunay Triangulation Overlays. *IEEE Journal on Selected Areas in Communications*, 20(8):1472–1488, October 2002.

[79] http://www.skype.com/home.html.

APPENDICES

A MULTILAYER-CON: RECURSIVE JOINING AND RECONSTRUCTION

A.1 Multilayer-CON: Join Recursive Function

Join(X, FN, $S_i^{(j)}$)

{

// Global variables: First_Contact = 1, CHECK_OUT=0.

// CHECK_OUT: Number of sub-communities have been checked out

If ((!First_Contact) && (CHECK_OUT==β_0)) then

{

//No sub-community satisfied the LAC.

Create_Sub(X, $S_{\beta_0+1}^{(0)}$); // Set $L_{\beta_0+1}^{(0)}$ =X

Return ($S_{\beta_0+1}^{(0)}$);

}

If ((First_Contact) && ((δ(X, FN) + δ(FN, $L_i^{(j)}$)) < $\alpha_i^{(j)}$)) then

{

Create_link(X, FN);

Return ($S_i^{(0)}$);

}

If ((j == 0) && (δ(X, $L_i^{(j)}$) < $\alpha_i^{(0)}$)) then

{

Join_Sub(X, $S_i^{(0)}$);

Return ($S_i^{(0)}$);

}

// Set First_Contact=0 for next recursive call.

First_Contact =0;

If ((j==0) && (δ(X, $L_i^{(j)}$) > $\alpha_i^{(0)}$))

{ // u: Up

j = j+1;

CHECK_OUT++;

Return (Join(X, $L_u^{(j)}$, $S_u^{(j)}$));

}

If ((j>0) && δ(X, $L_i^{(j)}$) < $\alpha_i^{(0)}$) then

{ // d: down

j = j-1;

CHECK_OUT++;

Return (Join(X, $L_d^{(j)}$, $S_d^{(j)}$));

}

If ((j>0) && (δ(X, $L_i^{(j}$)) > $\alpha_i^{(0)}$)) then

{

// $L_i^{(j)}$ forwards a request to check LAC to its neighbors

// Each neighbor forwards that request to its neighbors

// until $\forall z_m \epsilon S_i^{(j)}$ received that request.

For (k=0; k < $L_i^{(j)}$.nu_neighbors; k++)

$L_i^{(j)}$.neighbor[k].Node_Join_Check(X, $L_i^{(j)}$);

$L_i^{(j)}$.Wait(γ); //γ is timeout $L_i^{(j)}$ wait for replies

CHECK_OUT = CHECK_OUT + $S_i^{(j)}$.Sub_Size -1;

If ($L_i^{(j)}$.received) then

{

//Selected: SubID with minimum latency from the repliers

Selected =SubID_MinLatency (Repliers);

j = j-1;

Return (Join(X, $L_{Selected}^{(j)}$, $S_{Selected}^{(j)}$));

}

Else

{

// i.e. No reply within γ : $L_i^{(j)}$.received = 0.

If (j < Number_layers) then

{

j = j+1;

Return (Join(X, $L_u^{(j)}$, $S_u^{(j)}$));

}

el se

{

j = j-1;

Return (Join(X, $L_d^{(j)}$, $S_d^{(j)}$));

}

} }

A.2 Multilayer-CON: Sub-Community Division and Integration Technology

A.2.1 Sub-Community Division Technology

*Leader*_DivProcess() // Leader calls this routine to initiate the division process

{

if (($Thr_d < |S_i^{(0)}|$) && ($|S_i^{(0)}| < Thr_u$))

return(0); // No need to divide

else

if ($Thr_u < |S_i^{(0)}|$)

{

$\alpha_i^{(0)} = \alpha_i^{(0)}$ - ω; // Leader autonomously determines ω

$\alpha_{\beta_0+1}^{(0)} = \omega$;

// Send division request to all members in $S_i^{(0)}$ with new α ($\alpha_i^{(0)}$)

Send_Divreq($\alpha_i^{(0)}$, $\alpha_{\beta_0+1}^{(0)}$, $\beta_0 + 1$);

while (Not Not_satified_reply); //First reply is received from node_id.

Ł$_{\beta_0+1}^{(0)}$ = node_id;

//Node with node_id becomes the leader of new sub-community $S_{\beta_0+1}^{(0)}$.

Send_promotion(node_id);

}

return(1);

}

// Each member calls this routine when it received a division request from leader

*node*_DivProcess(α_{new})

{ // α_{new} is new α received from leader

subid = myself.sub_id; //myself: Object contains node's information

if (δ(myself.id, $L^{(0)}_{subid}$) < α_{new})

{

$myself.\alpha_{subid} = \alpha_{new}$;

myself.send($L^{(0)}_{subid}$, "*LAC is satisfied*");

}

else

{

myself.send($L^{(0)}_{subid}$, "*LAC is not satisfied*");

myself.leave();

join_sub(myself, $S^{(0)}_{\beta_0+1}$);

}

}

A.2.2 Sub-Community Integration Technology

*Check*_Integ() // This routine checks if integration is required or not.

{

if (($Thr_d < |S_i^{(0)}|$) && ($|S_i^{(0)}| < Thr_u$))

return(0); // No need to Integrate

else

if ($Thr_d > |S_i^{(0)}|$)

return (1); // Integration is required

}

*Leader*_InitInteg() // Leader calls this routine to initiate the integration process

{

if (Check_Integ())

{

sub_id= myself.subid;

// Send integration request to its neighbors in sub-community of leaders at upper layer

Send_IntgRequest(Neighbors_nodes);

while(no_reply);

// Wait until receive a first reply from a *replier* that wants to integrate too

$\alpha^{(0)}_{sub\_id} = \alpha^{(0)}_{sub\_id} + \alpha^{(0)}_{replier}$;

Send_IntegAck(replier); // Send integration acknowledge to *replier*

}

}

//Leader calls this routine when it received an integration request from neighbor's leader

*Leader*_IntegProcess()

{

if (received_IntegRequest)

if (Check_Integ())

{

Send(ToRequester, "*Ok Integrate*");

WaitIntegAck(); // Wait integration acknowledgement

// Send a message to all sub-community members to change

// their sub-community identifer to InitInteg_SubID.

myself.SendSubMembers(InitInteg_SubID);

myself.leader = 0;

myself.normal = 1; // The leader becomes normal member

myself.subid = InitInteg_SubID;

}

}