SEIKEI UNIVERSITY

Efficient and Flexible Agreement Protocols Based on Trustworthiness Relation of Peers in Unstructured Peer-to-Peer Overlay Networks

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF SCIENCE AND TECHNOLOGY

in Computer and Information Science

by

Ailixier Aikebaier (**Alisher Akber**)

Dissertation committee:

Professor Makoto Takizawa, Committee Chair
Professor Leonard Barolli
Professor Shin-ich Kuribayashi
Professor Hitomi Murakami
Professor Kimio Oguchi

2011

Efficient and Flexible Agreement Protocols Based on Trustworthiness Relation of Peers in Unstructured Peer-to-Peer Overlay Networks

This dissertation of Ailixier Aikebaier (Alisher Akber)
is approved and is acceptable
in quality and form for publication:

Makoto Takizawa, Committee Chair

Leonard Barolli

Shin-ich Kuribayashi

Hitomi Murakami

Kimio Oguchi

Seikei University
2011

Contents

List of Figures **vi**

List of Tables **vii**

Acknowledgments **viii**

Curriculum Vitae **ix**

Abstract **x**

1 Introduction **1**
1.1 Peer-to-peer (P2P) overlay networks 1
1.1.1 Background 1
1.1.2 P2P overlay networks 2
1.1.3 Principles of the P2P paradigm 3
1.1.4 Classification 5
1.1.5 Applications 6
1.2 Agreement protocols 7
1.2.1 Background 7
1.2.2 Classification 7
1.3 Overview of this dissertation 8

2 Trustworthiness **9**
2.1 Acquaintances 10
2.2 Subjective trustworthiness 13
2.2.1 Direct communication 14
2.2.2 Acquainted communication 16
2.3 Objective trustworthiness 18

2.3.1 Types of objective trustworthiness 18
2.3.2 Computation of trustworthiness 19
2.4 Confidence on subjective trustworthiness 22

3 A Basic Agreement Protocol 25
3.1 Precedent relations 25
3.1.1 E-precedent relation 27
3.1.2 P-precedent relation 29
3.2 Coordination procedure 30
3.2.1 Agreement conditions 31
3.2.2 Global decision function 32
3.2.3 Local decision function 32
3.2.4 Initial value 34
3.2.5 Meta coordination 36
3.2.6 Types of coordination strategies 36
3.2.7 Inconsistent strategies 38
3.2.8 Resolution among different strategies 39
3.2.9 Behaviors of peers 44
3.3 A history of a peer 46
3.3.1 History 46
3.3.2 Methods on a history 47
3.3.3 Compensation 49
3.3.4 Constraints on values 50
3.4 Back-warding strategies 51
3.4.1 Cuts 51
3.4.2 Re-selectable values 57

4 Distributed Agreement Protocols 59
4.1 Value exchange schemes 59
4.1.1 Single value exchange scheme 59
4.1.2 Multi-value exchange (MVE) scheme 60
4.2 Multipoint relaying (MPR) scheme 65
4.2.1 Basic algorithm 65
4.2.2 Faults 68
4.3 Trustworthiness-based broadcast (TBB) scheme 70
4.3.1 Trustworthiness of peer 70
4.3.2 Trustworthiness - based broadcast (TBB) algorithm 71

5 Evaluation **75**
5.1 Assumptions 75
5.2 Scenarios 76
5.3 Results 78

6 Conclusions and Future Work **81**
6.1 Conclusions 81
6.2 Future work 83

Bibliography **84**

List of Figures

2.1 Acquaintance peer p_j. 12
2.2 Direct interaction. 13
2.3 Indirect interaction. 14
2.4 Acquainter. 17
2.5 Objective trustworthiness. 20
2.6 Objective trustworthiness ot_{01}. 21

3.1 Coordination protocol. 31
3.2 Lub and Glb.. 35
3.3 Coordination procedure of a peer. 37
3.4 Mining and backward strategies. 39
3.5 Resolution of multiple recoverable cuts. 42
3.6 Aggressive peer. 45
3.7 History methods. 48
3.8 Coordination procedure of a peer. 48
3.9 Most recently uncompensatable sequence. 50
3.10 Obtainable cut. 53
3.11 Cuts. 54
3.12 Multiple cuts. 56

4.1 Multi-value exchange. 62
4.2 Maximal-value exchange (XVE) scheme. 63
4.3 Single-value exchange (SVE) scheme. 63
4.4 Multi-value exchange (MVE) scheme. 64
4.5 Multipoint relays. 66
4.6 Failure in multipoint relays. 68
4.7 Trusted neighbors in multipoint relays. 73
4.8 Trustworthiness of peer. 74

5.1 Number of messages (F = 0.05). 79
5.2 Number of messages (F = 0.1). 79
5.3 Network coverage to fault ratio. 80
5.4 Number of messages to fault ratio. 80

List of Tables

3.1 Consistency among strategies 39
3.2 Consistency conditions. 43

Acknowledgments

The author have received tremendous amount of support from so many people upon completing the degree that he cannot enumerate all of them at this moment.

First of all, the author would like to express his endless appreciation to his supervisor, Professor Makoto Takizawa, for his kindness, support, and instruction. He is the one of the persons who has the most influence in authors life, the author had study many things not only about how to do research but also how to be a good person, and how to do things correctly. He has always gave the best support and help to the author when it needed.

Doctor Tomoya Enokido (Rissho University), Doctor Kenichi Watanabe (Tokyo Denki University), and Distributed Systems laboratory members of Seikei university and Tokyo Denki university, gave the author helpful discussions, instruction, and comments. The author would like to dedicate this work to all of them, because this work could not be done without all of them.

Most importantly, the author would like to acknowledge his uncle (Heyder Peyzulla) and aunt (Munire) for introduced him to Professor Makoto Takizawa and made it possible to him to came to Japan at the first place.

Lastly, the author would like to express his appreciation and love to his parents (Akber Peyzulla and Patigul Mijit), younger brother (Mirzat Akber), grandma (Kurbanem), uncle (Ablimit Mijit and Behtiyar Dawut), aunt (Rena Mijit and Mahire Mijit), niece (Berna Behtiyar and Nazile Ablimit) and all other relatives for giving him behind-the-scenes support over many years. Words cannot express author's love to them.

Curriculum Vitae

Ailixier Aikebaier (**Alisher Akber**)

2004 B.E. in Computers and Information Science, Xinjiang University, China.

2009 M.E. in Computers and Systems Engineering, Graduate School of Science and Engineering, Tokyo Denki University, Japan

2011 Research Fellowship, Japan Society for the Promotion of Science (JSPS)

2011 Ph.D. in Computer and Information Science, Seikei University, Japan

Field of Study

Peer-to-Peer systems, Fault tolerance, Agreement protocols, Consensus problems, and Distributed systems.

Abstract

Nowadays information systems are being shifted to distributed architectures to obtain the benefits like scalability, autonomy, and faulty-tolerance. Since peer-to-peer (P2P) systems are open world systems differently from other systems like cloud computing model, a huge number of computers and various types of computers with P2P application are interconnected in large-scale P2P overlay networks lying on the top of underlying physical computer networks like the Internet Protocol (IP) network. Except centralized or hybrid P2P systems, there is no centralized index server which controls the whole P2P system, and the peers which represent the individual computers in the P2P system, autonomously take actions and cooperate with each other to realize their purpose such as file sharing, building distributed storage, instant messaging, realizing distributed computation, contents delivery, cooperative work, and so forth. Because of the nature of the P2P systems, it is difficult for every peer to figure out what kinds of information are distributed to what peers, what kinds of peers exist in P2P overlay networks, and what kinds of relations among peers are. In addition, malicious peers and faulty peers like a crash-faulty peer can join and leave a P2P system without being authenticated and authorized. This rises a question on how each peer to trust a target peer in the P2P systems. Therefore efficient and reliable synchronization methods are required to be supported in order to achieve the cooperation among peers in the P2P systems. The P2P system is a disruptive technology for deploying applications that scale to millions of simultaneous participants. Because each user contributes computer and networking resources, it offers a low-barrier-of-entry platform with high scalability. Extensions to the basic model could offer different grades of service as well as address limitations of the basic model. These limitations are due to the decentralized character of the overlay and the unreliability of the peers. As disruptive technology, P2P systems raise important questions about the long-term impact on other approaches for video delivery, telephony, and other information delivery services. In addition, P2P applications to date have been pri-

marily adopted in the consumer space. Requirements for further growth such as manageability, security, or ability to generate revenue may in the near term require hybrid variations of the basic model. The ability to incorporate reliable and secure transactions is still nascent.

An agreement or consensus procedure is one of the most essential parts in our daily life. In our history, many astonishing achievements are done by the collaboration of many peoples, like the pyramids in Egypt. In order to achieve the collaboration, we need agreement procedures for a group of multiple participants to support it, and that is why it is essential in our daily life. Without exception in computer world, we can find many footprints of agreement procedures in basic and important parts of the information systems. For example, the two-phase commit protocol (2PC) in transaction processing, distributed database systems, and computer networks. The two-phase commit protocol (2PC) is a typical type of an atomic commitment protocol. It is a distributed algorithm that coordinates all the processes that participate in a distributed atomic transaction on whether to commit or abort (roll back) the transaction. The 2PC protocol is a specialized type of consensus protocol. Following the transformation of the information systems from the traditional centralized client-server models to the decentralized distributed models like P2P systems, how to achieve the agreement procedure in fully distributed environment become a question to us to be solve. In human societies, participants make an agreement in more flexible and efficient ways. For example, participants can change their mind in the agreement procedure. In this dissertation, we first introduce the novel relations among values which each peer can take from a given domain, existentially (E-) and preferentially (P-) precedent relations, which describe the relations between values in the domain of a peer. If a peer can take a value b after taking a value a, the value a E-precedents the value b. Suppose a peer can take a pair of values a and b after taking value c, if the peer prefers the value a to the value b, it denotes the value a P-precedents the value b. Based on the precedent relations, we discuss the flexible agreement protocol. Then, in order to improve the efficiency of the agreement protocol, we newly introduced the concept of obtainable cuts, which is a set of values which are exchanged by the participants during the agreement procedure and also satisfies the agreement condition. In addition, by defining the forward and backward strategies and history of values which each peer has so far taken, we introduces an efficient way to discover the obtainable cuts in the history of peers, ultimately improves the overall performance of the agreement protocol. By introducing the multi-value exchange (MVE) scheme, the time spent for a complete agreement procedure can be significantly reduced, therefore the efficiency of agreement protocol is improved.

In order to achieve the agreement procedure in a fully distributed system, many problems has to be solved, for example, how to exchange information among participants, how to detect the agreement condition being satisfied through out the network and so on. As one of the most important steps of the agreement procedure, the message exchange phase is in charge of delivering and collecting information from all participants in the group. To realized the distributed agreement procedures, reliable message exchange protocols among peers are required to be realized as the most important phase of the whole procedure. In order to achieve our goal which is required to efficiently and reliably realize agreement procedure in a fully distributed system, we newly proposed a trustworthiness-based broadcast (TBB) algorithms in addition to the multi-value exchange (MVE) scheme. In this dissertation, we show our approach to designing and realizing the agreement procedure in a fully distributed system. The evaluation results show that by using our proposed trustworthiness-based (TBB) scheme, totally 22 percentage of the unnecessary message broadcast can be reduced in the network compared with the multipoint relay algorithm and pure message flooding. Furthermore, a message can be delivered to every peer in presence of faulty peers. By improving the efficiency of the message exchange phase of the protocol, we improved the overall performance of the agreement protocol.

The concepts, algorithms, implementation, and evaluation of the agreement protocol discussed in this dissertation can be not only theoretical but also practical foundation to design and develop various of applications on P2P overlay networks.

Keywords: Peer-to-Peer (P2P) overlay network, Agreement protocol, Consensus problems, Trustworthiness, distributed systems.

Chapter 1

Introduction

1.1 Peer-to-peer (P2P) overlay networks

1.1.1 Background

Traditional information systems have been realized in client-server systems (CSSs). A CSS is composed of a server, a process which supports client with some service for applications, and a client which is a interface between applications and servers. During issuing requests to servers, application programs (APs) are performed on clients and application servers in 2-tier and 3-tier CSSs, respectively. On receipt of requests from APs on clients/application servers, the requests are performed on servers and then responses of the requests are sent back to the APs. Here, each computer can play on role of client, application server, and database server. In the CSS, all clients access a centralized serer like a database server since data is stored in the server. Consequently, the server might be performance bottleneck due to the heavy traffic and furthermore a single point of failure. Moreover, servers cannot meet every user's requirements since various types and a huge number of computers are interconnected in CSSs.

According to the advance of computer and network technologies and varieties of applications, information systems are now being shifted to *peer-to-peer* (P2P) systems from CSSs. Various types of applications and businesses can be cost-effectively realized in the P2P systems. Here, due to the fact that systems or applications are called "peer-to-peer" not because of their internal operation or architecture, but rather as the result of how they are perceived externally, there are number of different definitions on "peer-to-peer", that is there may not general agreement on what is and what is not "peer-to-peer". On the web [1], "peer-to-

peer" systems have been defined as a class of applications that takes advantage of resources-storage, cycles, content, human presence-available at the edges of the Internet. This definition includes systems which rely upon centralized servers and systems on the field of Grid computing [2, 3]. The difference between P2P and Grid computing is often discussed, but it is beyond the scope of this dissertation to discuss the difference.

In a P2P system, each process on computers is a peer process which can provide the same service. A group of peers on computers are cooperating to achieve some objectives by exchanging messages. Each peer is often called *servent* which is the compound word of SERVer and cliENT since the peer can play any role of client, application server, or database server. Resources, indices which indicate locations of the resources, and load on a server in a CSS are distributed to peers interconnected in a network of peers. The network is formed on the top of the underlying physical computer network and is thus referred to as a P2P "overlay" network [4, 5]. Connection between peers in a P2P overlay network is a virtual or logical link. Even if a source peer does not know an IP address of a destination peer, a message from the source peer can be delivered to the destination peer through a P2P overlay network.

A P2P overlay network is characterized by *scalability*, i.e. a huge number of peers are connected to the overlay network, *stateless infrastructure*, i.e. network topology is dynamically changed since every peer can join/leave the overly network whenever the peer would like to, *open world*, i.e. any kind of computer with a P2P application can join the overlay network, *robustness*, i.e. a P2P system does not have a single point of failure because the system does not depend on servers, and, *ad-hocracy*, i.e. every peer autonomously operates.

1.1.2 P2P overlay networks

Peers in P2P applications communicate with other peers using messages transmitted over the Internet or other types of networks. The protocol for a P2P application is the set of different message types and their semantics, which are understood by all peers. The protocols of various P2P applications have some common features. First, these protocols are constructed at the application layer of the network protocol stack. Second, in most designs peers have a unique identifier, which is the peer ID or peer address. Third, many of the message types defined in various P2P protocols are similar. Finally, the protocol supports some type of message-routing capability. That is, a message intended for one peer can be transmitted via intermediate peers to reach the destination peer.

To distinguish the operation of the P2P protocol at the application layer from the behaviour of the underlying physical network, the collection of peer connections in a P2P network is called a *P2P overlay*. While their host is connected to the overlay, each end user shares in the cost of operating the overlay. This cost sharing by the participants lowers the barrier of entry to overlay providers. The low barrier of entry means that little hardware or network investment is needed to launch a P2P application.

The practice of overlay networks predates the P2P application ear. For example, protocols used in Internet news servers and Internet mail servers are early examples of widely used overlay that implement important network services. These specialized overlay networks were developed for various reasons, such as enabling end-to-end network communication regardless of network boundaries caused by network address translation (NAT).

Another important reason for the use of overlays is to provide a network service that is not yet available within the network. For example, multicast routing is a network service that to date has been only partially adopted on the Internet. Multicast routing enables a message sent to a single multicast address to be routed to all receivers that are members of the multicast group. This is important for reducing network traffic for one-to-many applications such as video broadcasting or videoconferencing. Since multicast routing is not universally supported in Internet routers, researchers developed an application layer capability for multicast routing called *application layer multicast* (ALM) or *overlay multicast* (OM).

Finally, other examples of network services that can be supported using an overlay include secure delivery of packets, trust establishment between arbitrary endpoints, anonymous message delivery, and censorship-resistant communications. Such services are incompletely provided in today's Internet and can be more rapidly delivered using an overlay network because application layer features do not require network hardware upgrades.

1.1.3 Principles of the P2P paradigm

A peer-to-peer overlay is a distributed collection of autonomous end-system computing devices called peers that form a set of interconnections called an *overlay* to share resources of the peers such that peers have symmetric roles in the overlay for both message routing and resource sharing. The P2P overlays has following paradigm: self-organization, role symmetry, resource sharing, scalability, peer autonomy, and resiliency.

The peers self-organize the overlay. Self-organization is a characteristic of

many physical and social systems such that the organization of the system increases without being controlled by an encompassing agent or the environment. An overlay network design that is consistent with self-organization would not use a star topology or a broadcast topology to operate the peers or form the overlay. Self-organization means that peers cooperate in the formation and maintenance of the overly, with each peer using local state and partial information about the overlay.

The peers have symmetric roles. In contrast to client/server computing, where the roles of the endpoints are /textitasymmetric, peers are functionally equal. Any peer can store objects on behalf of other peers, support queries, and perform routing of messages.

Peer-to-peer overlay are highly scalable. Several P2P applications operate today with millions of peers participating. An important dimension of scalability is the ability to operate the P2P overlay as the size grows by 100 times or more. Scalability means that the network and computing resources used at each peer exhibit a growth rate as a function of overlay size that is less than linear.

Peers are autonomous. Each peer determines its capabilities based on its own resources. Each peer also determines when it joins the overlay, what requests it makes to the overlay, and when it leaves the overlay.

A P2P overlay provides a shared resource pool. The resources a peer contributes include compute cycles, disk storage, and network bandwidth. There are minimum resource contribution threshold for a peer to join the P2P overlay. Each peer's resources are used to support the operation of the overlay and provide application services to other peers.

Peer-to-peer overlays are resilient in the face of dynamic peer membership. Since peers have an incomplete view of the overlay topology and peer membership, the overlay depends on intermediate peers to forward messages to the correct region of the overlay. When peers leave or join the overlay, the routing paths are affected. The overlay graph structure or geometry contributes to resilience by enabling connectedness in the topology despite peer of endpoints.

The principles of P2P overlays are generally not completely satisfied in any single system. Hybrid P2P systems may relax one of more these design goals. Some systems use central servers to authenticate peers, after peers are authenticated, the overlay itself operates without the central server.

1.1.4 Classification

P2P architectures are categorized in terms of a level of overlay network centralization, and P2P overlay networks are categorized in terms of a level of overlay network structure, respectively [5]. There are three types of P2P architectures, i.e. *hybrid decentralized*, *purely decentralized*, and *partially centralized* ones, and there are three types of P2P overlay networks, i.e. *unstructured*, *structured*, and *loosely structured* ones, respectively.

- Overlay network centralization
  - **Hybrid decentralized:** A CSS (there is a centralized server managing the whole P2P system by maintaining directories of information of file locations) and a P2P system (files are transferred with end-to-end communication) are mixed.
  - **Purely decentralized:** There is no centralized server, and all peers provide the same service and act as both servers and clients. The peers are called servents.
  - **Partially centralized:** The basic concept is same as the purely decentralized architecture. However, several peers, called superpeers, have a more important role, e.g. a superpeer manages index information of its normal peers and acts as a bridge/gateway between the normal peers.
- Overlay network structure
  - **Unstructured:** A file location depends on a topology of an overlay network, so an efficient look-up protocol is needed, e.g. flooding algorithm.
  - **Structured:** Topology of an overlay network is controlled, and a file location is precisely specified.
  - **Loosely structured:** An overlay network structure is in between unstructured and structured networks, and a file location is not completely specified.

In this dissertations, we aim at discussing a efficient and flexible agreement protocol in a decentralized and unstructured P2P system.

1.1.5 Applications

A definition of a P2P application has been proposed by Dave Winer [6]. P2P applications have the following characteristics:

- User interfaces do not run in a web browser.
- Each computer can act as both servers and clients.
- It is easy for users to manipulate and implement a system.
- Tools for creating users' own content and additional functionality are included.
- Users can create or join a P2P community.
- A system does something new or exciting.
- Cross-network protocols such as XML-RPC and SOAP are supported.

We point to the following applications as an example of a P2P application:

- File sharing system: Napster [7], Gnutella [8], WinMx, LimeWire [9], Kazaa [10], Bearshare, Morpheus, eDonkey, BitTorrent, Ares Galaxy, iMesh, etc.
- Distributed storage: Freenet [11], Free Haven [12], Distributed Hash Tables (DHTs) [13, 14, 15, 16, 17, 18], etc.
- Instant messaging: P2P Messenger, ICQ [19], Jabber [20], Skype, etc.
- Distributed computation: OceanStore [21], SETI@home (search for extra terrestrial intelligence at home) [22], HyperBee, etc.
- Contents delivery service: Akamai, Kontiki, Peercast, Streamer P2P radio, Dijjer, etc.
- Network game: Diablo, Age of Empire, etc.
- Cooperative work: Groove Workspace, etc.

1.2 Agreement protocols

1.2.1 Background

The agreement procedure or consensus making are the most basic and essential process in our human society. Since people are living in a unit of group, from the ancient time the agreement procedure are needed in order to achieve some objectives. For example, in the ancient times when our ancestors go for hunting, before begin the hunt they have to make sure every ones position and role on the hunt and other things like who will attack first who is the leader of the team and how to bring the prey to home after catch it and so on. Each of these decisions are the outcome of a agreement procedure. In addition, some of the most stunning architectures in our world like the pyramid in Egypt also shows the importance of the agreement procedure, without collaboration of the million of people it is impossible to construct project like pyramid. In order to do collaboration, the agreement procedure are needed.

Nowadays information systems are being shifted to distributed architectures to obtain the benefits like scalability, autonomy, and faulty-tolerance. Following the transaction from centralized systems to the decentralized one, distributed agreement protocol are considered as a successor for the traditional centralized agreement protocols.

1.2.2 Classification

According to the structure of the systems, the agreement protocols can classify into following two groups:

- Centralized systems.
- Decentralized systems.

In centralized systems, like two-phase commit protocol [57] in transaction processing, databases, and computer networking. It is a distributed algorithm that coordinates all the processes that participate in a distributed atomic transaction on whether to commit or abort (roll back) the transaction. The characteristic of the system is that a coordinator in the system collects and make the final decision on the agreement value, so that whole system is centralized controlled by the coordinator.

In decentralized systems, there are no centralized control in the system. Therefore, this kind of systems can archive high reliability and scalability, but on the same time the problems like efficiency and trustworthiness has to be concern.

Nowadays, the traditional centralized systems are being shifted to decentralized one. Decentralized systems in systems theory are naturally occurring, usually self-regulating systems found which function without an organized center or authority. A system that is decentralized lacks a nuclear body or center of control, and is commonly composed of many components which work in unison, and together form a stable structure. Such systems can be found in society as well as in nature. For example, a market economy is a system formed by human trade and business. Therefore, the traditional centralized agreement protocols are transforming into decentralized agreement protocols. On the other hand, to improve and solve the problems rises with the decentralization like trustworthiness among peers and efficiency of the system in terms of the message broadcasting in the system has to be consider and new algorithms are needed.

1.3 Overview of this dissertation

The rest of the dissertation is organized as follows. In chapter 2, we present the trustworthiness concept on peer-to-peer (P2P) overlay networks. In chapter 3, we introduce the basic agreement procedure and different strategies to make agreement among peer processes. In chapter 4, we discuss distributed agreement protocols with difficulty to achieve the agreement within given group of peers. We also discuss two novel algorithms to improve the efficiency and reliability of the agreement protocol. In chapter 5, we show the evaluation result of the proposed algorithms. In chapter 6, we conclude this dissertation and suggest some areas for future research.

Chapter 2

Trustworthiness

In a fully distributed, unstructured peer-to-peer (P2P) overlay network, there is no centralized coordinator like centralized index [7] and super peer [10], A peer process (*peer*) p_i is cooperating with another peer p_j by not only exchanging messages but also remotely manipulating objects in p_j. There are many discussions on how to detect a target peer which holds an object like flooding algorithms [25, 28, 8, 35, 32, 33, 17, 38]. A peer has to manipulate a target object in addition to detecting the target object. Only a peer which is granted an access right (permission) is allowed to manipulate the target object in an authorized way. Peers are classified into $holder$ peers where an object o is stored, $manipulation$ peers which are allowed to manipulate the object o, and $authorization$ peers which can grant access rights of the object o to other peers [34, 35].

In a fully distributed P2P overlay network, each peer has to obtain service information on what peers support what types of service through communicating with its acquaintance peers. A peer may leave and join the network and obtain new service by downloading and removing files. Another peer might be faulty. Service changes of peers are propagated to peers through peer-to-acquaintance communications. It takes time to propagate the service change information in the network. Hence, a peer might hold obsolete service information. Here, it is critical for each peer to recognize which acquaintance is trustworthy on service information. There are *subjective* and *objective* types of the trustworthiness of each acquaintance. In the subjective approach, a peer obtains a trustworthiness opinion of an acquaintance by communicating with the acquaintance. A peer issues an access request to an acquaintance and then receives a reply from the acquaintance. If the reply satisfies the access request, the peer perceives the acquaintance to be more trustworthy. On the other hand, a peer obtains the trustworthiness opinions

of an acquaintance from other peers in the objective approach. The more trusted an acquaintance is, the more trustworthy the acquaintance is perceived to be. This is similar to the traditional reputation concept [37]. In this paper, we newly discuss the trustworthiness concepts based on the $confidence$ of each peer. The less confident of its own subjective trustworthiness the peer is, the more significant the objective trustworthiness opinions of other peers is. If the peer is more confident of its own opinion, the peer only takes trustworthiness opinions of acquaintances which the peer knows well and whose opinions are similar to its own opinion. A most confident peer takes only its own opinion. There are some varieties between them. We discuss types of the objective trustworthiness in this paper. In addition, we discuss how a peer takes the types of trustworthiness based on the confidence.

2.1 Acquaintances

In P2P overlay networks, applications have to not only detect target objects [24, 7, 32, 8, 33] but also manipulate the objects. Even if a target object is detected, the object cannot be manipulated if the requesting peer is not authorized. An $access\ right$ is specified in a form $\langle o, op\rangle$ for an object o and a method op [27]. An access request to manipulate an object o in a method op is also written in a form $\langle o, op\rangle$ as well. A peer is allowed to manipulate the object o in the method op only if an access right $\langle o, op\rangle$ is granted to the peer.

A pair of peers p_i and p_j are *requesting* and *requested* peers, respectively, if p_i issues an access request to the other peer p_j. A *holder* peer p holds an object o (written as $p \mid o$). A *manipulation* peer p can manipulate an object o in a method op ($p \models_{op} o$), i.e. p is granted an access right $\langle o, op\rangle$. An *authorization* peer p can grant an access right $\langle o, op\rangle$ to another peer ($p \vdash_{op} o$). A peer p is a $serving$ peer of an access request $\langle o, op\rangle$ ($p \,\square_{op}\, o$) iff $p \mid o$, $p \models_{op} o$, or $p \vdash_{op} o$. Service supported by a peer is specified in a form $\langle o, \square, op\rangle$. For example, a manipulation peer $p \models_{op} o$ supports a type of service ρ_i (= $\langle o, \models, op\rangle$). If a peer p receives a request $\langle o, op\rangle$ for manipulating an object o in a method op from an application, p issues an access request $\langle o, \square, op\rangle$ to an acquaintance p_i. For example, if p_i knows p_j holds an object o ($p_j \mid o$) and p is not granted an access right $\langle o, op\rangle$, p asks p_j to grant $\langle o, op\rangle$, i.e. issues $\langle o, \vdash, op\rangle$ to p_i. An *acquaintance* peer p_i of a peer p with respect to a service type ρ (= $\langle o, \square, op\rangle$) ($p \rightarrow (p_i \square_{op} o)$) is a peer which p knows about service ρ, i.e. $p_i \square_{op} o$ or p_i has an acquaintance p_i ($p_i \rightarrow (p_j \square_{op} o)$. Here, p_i is a *direct* acquaintance of p with respect to a service type $\langle o, \square, op\rangle$ iff $p_i \square_{op} o$. p_j is an *indirect* acquaintance of p iff p_i does not support the service $\langle o, \square, op\rangle$ but has

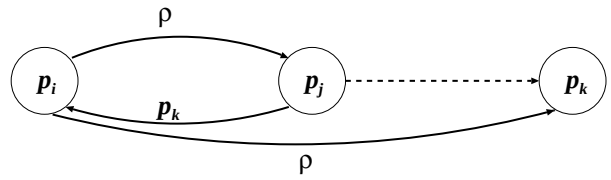
an acquaintance p_k ($p_k \rightarrow (p_j \square_{op} o)$). However, p may not be an acquaintance of p_i even if p_i is an acquaintance of p. A *friend* peer p_j of a peer p_i is an acquaintance of p_i with which p_i can directly communicate. If p_j is a friend of p_i, p_i is assumed to be a friend of p_j.

In order to get a friend of another peer p_j, a peer p_i has to not only know a type of service of p_j but also communicate with p_j. If p_j allows p_i to communicate with p_j, p_j is a friend of p_i. Let $V(p_i, \rho)$ be a set of acquaintances of a peer p_i with respect to a service type ρ (= $\langle o, \square, op \rangle$), i.e. $\{p_j \mid p_i \rightarrow (p_i \square_{op} o)\}$. A peer p is referred to as *directly satisfy* an access request $\langle o, \square, op \rangle$ if $p \square_{op} o$. p is referred to as *indirectly satisfy* $\langle o, \square, op \rangle$ if $p \rightarrow (p_i \square_{op} o)$. p ***satisfies*** an access request $\langle o, \square, op \rangle$ if p directly or indirectly satisfies $\langle o, \square, op \rangle$. Otherwise, p is not satisfiable for $\langle o, \square, op \rangle$. For example, suppose a peer p_i asks an acquaintance p_j to detect an object o, i.e. $\langle o, |, \_ \rangle$. If the acquaintance p_j holds o, p_j satisfies $\langle o, |, \_ \rangle$. Next, consider an access request $\rho = \langle o, \models, op \rangle$, i.e. p_i would like to manipulate an object o in a method op. $a_{ij}(\rho) = 1$ if an acquaintance p_j manipulates o in the method op. Otherwise, $a_{ij}(\rho) = 0$.

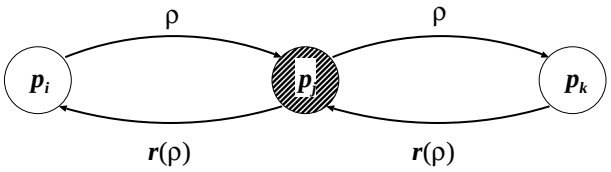
Each peer includes its service information in access requests and responses which the peer sends. A peer p sends a query request $\langle o, ?, op \rangle$ to an acquaintance p_i to get what type of service on o and op p_i can support to p. On receipt of the query, the acquaintance p_i sends an answer $\langle p_i, o, \square, op \rangle$ to p if $p_i \square_{op} o$. If $p_i \not\square_{op} o$ but $p_i \rightarrow (p_j \square_{op} o)$, p_i sends an answer $\langle p_j, o, \square, op \rangle$ to p. On receipt of the answer $\langle p_k, o \square, op \rangle$ from p_i, p stores $\langle p_k, o \square, op \rangle$ in the database DB_i Unless p_i supports $\langle o, \square, op \rangle$, p_i sends $\langle p_i, o, \_, op \rangle$ to p. Suppose a peer p_j loses service $\langle o, \square, op \rangle$. The peer p_j sends a *loss* message $\langle p_j, o, \square, op \rangle$ to its acquaintances. On receipt of the loss message $\langle p_k, o, \square, op \rangle$, the peer p removes $\langle p_k, o, \square, op \rangle$ from DB_i. Here, p sends a loss message $\langle p_k, o, \square, op \rangle$ to the acquaintance. Next, suppose that a peer p_j newly obtains service $\langle p_k, o, \square, op \rangle$. The peer p_j sends a *new* message $\langle p_k, o, \square, op \rangle$ to its acquaintances. On receipt of the new message $\langle p_k, o, \square, op \rangle$, p adds $\langle p_k, o, \square, op \rangle$ is DB_i. Thus, peers exchange service information of their acquaintances with each other. It takes time to propagate service change of a peer. Suppose a peer p_i holds service information $\langle p_k, o, \square, op \rangle$. If a peer p_k supports service $\langle o, \square, op \rangle$, p_i is ***proper***. Otherwise, p_i is faulty. For example, a peer p_i can ask its acquaintances about a service type $\langle o, \square, op \rangle$. On receipt of the request from p_i, an acquaintance p_j sends the service information $p_j \square_{op} o$ or $p_j \rightarrow (p_k \square_{op} o)$ to the peer p_i. If p_i receives the service information $p_j \rightarrow (p_k \square_{op} o)$ from p_j, p_k gets an acquaintance of p_i with respect to the service type $\langle o, \square, op \rangle$. The service information $\langle p_k, \square, op \rangle$ obtained from the acquaintances is stored in the database DB_i of p_i. The peer p_i informs another acquaintance p_k of the service

information $\langle p_k, \square, op\rangle$. Since the size of DB_i is finite, some service information might be lost to make space to store new service information. For example, the least recently used service information of a type of service $\langle p_k, \square, op\rangle$ is thrown away. Here, p_k still thinks p_i to be its acquaintance on $\langle o, \square, op\rangle$ but p_i loses the service information. If p_k asks p_i about $\langle o, \square, op\rangle$, p_i does not know anything about the service. Here, the information $p_k \rightarrow (p_i \square_{op} o)$ is obsolete.

Suppose a peer p_i issues a service request ρ (= $\langle o, \square, op\rangle$) to an acquaintance p_j, i.e. $p_i \rightarrow (p_j \square_{op} o)$. There are two cases. In one case, p_j supports the service type ρ. Here, p_j performs the access request ρ and then sends the reply $r(\rho)$ to p_i. In the other case, the acquaintance p_j does not support the service type ρ but knows an acquaintance p_k which supports ρ, i.e. $p_j \rightarrow (p_k \square_{op} o)$. There are two cases. First, the acquaintance p_j just informs the peer p_i of p_k. Then, p_i issues the access request ρ to p_k. Secondly, p_j forwards the access request ρ to p_k. On receipt of the reply $r(\rho)$ from p_k, p_j forwards the reply $r(\rho)$ to p_i. If p_k informs p_j of a peer p_k which supports the service type ρ, p_j forwards the access request ρ to p_k. If p_j receives the reply $r(\rho)$ from p_k, p_j forwards the reply $r(\rho)$ to p_i. Here, p_j is referred to as $acquaintance$ of the requesting peer p_i with respect to the service type ρ.



(1) Direct



(2) Agent

Figure 2.1: Acquaintance peer p_j.

2.2 Subjective trustworthiness

Let p_i be a peer and p_j be an acquaintance of the peer p_i. Let ρ be an access request $\langle o, \Box, op \rangle$. A peer p_i makes a decision on how much p_i can trust an acquaintance p_j with respect to an access request $\langle o, \Box, op \rangle$ by itself. There are two cases, *direct* and *indirect* interactions with an acquaintance. First, suppose that p_j is a direct acquaintance of p_i and $p_j \Box_{op} o$, i.e. $p_i \rightarrow (p_j \Box_{op} o)$. A peer p_i issues an access request ρ to an acquaintance p_j and receives a reply $r(\rho)$ from p_j as shown in Figure 2.2. The peer p_i measures the *satisfiability value* $s_{ij}(\rho)$ showing how much the reply $r(\rho)$ is satisfiable for the request ρ.

Next, suppose a peer p_i does not know to which acquaintance the peer p_i can issue an access request ρ but knows an acquaintance p_j which knows some serving peer of the access request ρ, i.e. $p_j \rightarrow (p_k \Box_{op} o)$. The peer p_i asks the acquaintance p_i to introduce some serving peer of the access request ρ. Then, the acquaintance p_j introduces a peer p_k to p_i if p_j knows an acquaintance p_k to be a serving peer, $p_k \Box_{op} o$. Here, p_k is an acquaintance of p_i with respect to the access request ρ. The peer p_i issues the access request ρ to p_k and then receives a reply $r(\rho)$ from p_k as shown in Figure 2.3. Here, the peer p_i calculates the subjective trustworthiness of p_k from the reply $r(\rho)$ as discussed later. In addition, p_i perceives the acquaintance p_j to be trustworthy if p_k returns the more satisfiable reply to p_i, because the acquaintance p_j introduces p_k to p_i. Otherwise, the trustworthiness of the acquaintance p_j is decreased in p_i.

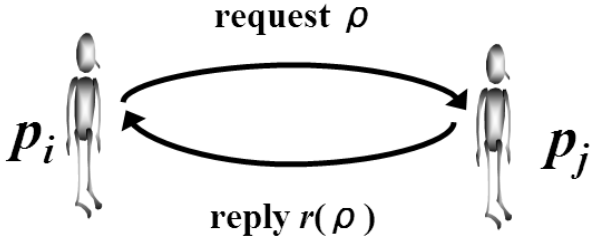


Figure 2.2: Direct interaction.

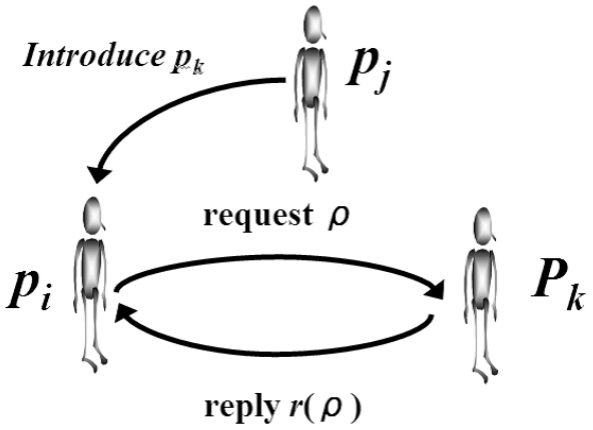


Figure 2.3: Indirect interaction.

2.2.1 Direct communication

A peer p_i issues an access request ρ to an acquaintance p_j. Then, p_i receives a reply $r(\rho)$ from p_j. The peer p_i obtains the satisfiability value $s_{ij}(\rho)$ of the acquaintance p_j from the reply $r(\rho)$. The satisfiability for each type of access request is discussed in papers [31, 36] by taking into account how many peers an access request passes to get to a target peer. In this paper, the satisfiability $s_{ij}(\rho)$ for an access request ρ issued to an acquaintance p_j is characterized in terms of whether or not the reply $r(\rho)$ satisfies ρ, how long it takes to get $r(\rho)$, and how much quality of service (QoS) the reply $r(\rho)$ supports. We consider another aspect of the satisfiability. First, the *answerability* $a_{ij}(\rho)$ is given as follows:

$$a_{ij}(\rho) = \begin{cases} 1 & \text{if } p_j \text{ satisfies } \rho. \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Suppose a peer p_i issues an access request ρ to a pair of acquaintances p_j and p_k. Here, suppose p_j supports a service type ρ while p_k does not support but knows another peer p_h supports the service type ρ. On receipt of the request ρ, p_j sends a reply $r_{ij}(\rho)$ to p_i. On the other hand, p_k forwards the access request ρ to p_h. The peer p_h sends a reply $r_k(\rho)$ to the peer p_k and p_k forwards the reply $r_k(\rho)$ to p_i. Here, suppose that both the replies $r_j(\rho)$ and $r_k(\rho)$ satisfy the access request ρ, i.e. $a_{ij}(\rho) = a_{ik}(\rho) = 1$. However, it takes a longer time to obtain the reply $r_k(\rho)$ than $r_j(\rho)$. The reply $r_j(\rho)$ more satisfies p_i than $r_k(\rho)$. Let $t_{ij}(\rho)$ show

the response time of an access request ρ issued by a peer p_i to an acquaintance p_j. The peer p_i is more satisfiable to receive the reply $r_{ij}(\rho)$ from the acquaintance p_j than p_k if $t_{ij}(\rho) < t_{ik}(\rho)$. In this paper, the response time $t_{ij}(\rho)$ is given an inverse of hop number, i.e. how many peer an access request ρ issued by p_i hops to get to a target peer p_j. For each request ρ, the allowable maximum time $maxt_\rho$ and the allowable minimum time $mint_\rho$ are defined. Suppose it takes τ time units to receive a reply $r_{ij}(\rho)$ from an acquaint p_j since a peer p_i sends a request ρ to p_j. $t_{ij}(\rho)$ = 1 if $\tau \leq mint_\rho$ and $t_{ij}(\rho)$ = 0 if $\tau \geq maxt_\rho$. $t_{ij}(\rho)$ = (τ - $mint_\rho$) / ($maxt_\rho$ - $mint_\rho$) otherwise,

In addition, a peer p_i is more satisfiable if the peer p_i receives a reply $r_{ij}(\rho)$ from an acquaintance p_j whose quality of service (QoS) $q_{ij}(\rho)$ like frame rate and number of columns is higher than the peer p_k. Thus, a replies $r_{ij}(\rho)$ from an acquaintance p_j to a requesting peer p_i is characterized in terms of answerability $a_{ij}(\rho)$, response time $t_{ij}(\rho)$, and QoS $q_{ij}(\rho)$.

A peer p_i records the satisfiability value $s_{ij}(\rho)$ obtained each time p_i issues an access request ρ. Then, the peer p_i obtains the subjective trustworthiness $st_{ij}(\rho)$ from satisfiability values obtained through the direct interactions with the acquaintance p_j. In one way, the average value of the satisfiability values is taken as the subjective trustworthiness $st_{ij}(\rho)$. Initially, $st_{ij}(\rho)$ = 0 for every acquaintance p_j in p_i. A counter $c_{ij}(\rho)$ is manipulated for p_j and ρ in p_i. Initially, $c_{ij}(\rho)$ = 0. Each time p_i obtains the satisfiability value $s_{ij}(\rho)$, $c_{ij}(\rho)$ is incremented by one. Here, let S_{ij} show the current subjective trustworthiness $st_{ij}(\rho)$. Then, the new subjective trustworthiness $st_{ij}(\rho)$ is obtained as the average value by the following function:

$$DS_0(S_{ij}, s_{ij}(\rho)) := (c_{ij}(\rho) \cdot S_{ij} + s_{ij}(\rho))/(c_{ij}(\rho) + 1). \tag{2.2}$$

The larger the counter $c_{ij}(\rho)$ is, the more shortly DS_0 changes for change of the satisfiability. In our life, one person recognizes another person p_j to be trustworthy only by observing the most recent behavior. That is, even if a person p_j had not been trustworthy, p_j is considered to be trustworthy just after p_j does the satisfiable job. On the other hand, a person may consider the person p_j to be trustworthy on the basis of long-term communications among them. This means, p_j is considered to be trustworthy if p_j has so far done satisfiable jobs even if p_j fails to do the current job. In order to take into account different views, we consider the following function DS_1:

$$DS_1(S_{ij}, s_{ij}(\rho), \alpha_i) := \alpha_i \cdot S_{ij} + (1 - \alpha_i) \cdot s_{ij}(\rho). \tag{2.3}$$

α_i is a direct subjective trustworthiness (DS) constant ($0 \leq \alpha_i \leq 1$) for a peer p_i. If α_i = 1, the subjective trustworthiness $st_{ij}(\rho)$ is not changed even if a new subjective trustworthiness st_{ij} is obtained. If α_i = 0, $\sigma_{ij}(\rho)$ is decided only by the current satisfiability value s_{ij}. If $\alpha_i = c_{ij}(\rho) / (c_{ij}(\rho) + 1)$, DS_1 is the same as DS_0. The smaller α_i is, the more the current satisfiability value s_{ij} dominates the subjective trustworthiness $st_{ij}(\rho)$.

2.2.2 Acquainted communication

Suppose a peer p_i issues an access request ρ (= $\langle o, \square, op \rangle$) to an acquaintance p_j but p_j does not support the service type ρ. Here, suppose the acquaintance p_j perceives that another peer p_k supports the service type ρ. On receipt of the service request ρ from the peer p_i, the acquaintance p_j informs p_i that p_k is a serving peer of the service type ρ. Here, p_k gets an acquaintance of p_i. The acquaintance p_j is referred to as *acquainter* of p_k in p_i. The peer p_i issues an access request ρ to p_k [Figure 2.4]. Then, p_i receives the reply $r(\rho)$ from p_k. Here, the satisfiability value $s_{ik}(\rho)$ is obtained as discussed in the preceding subsection. That is, the subjective trustworthiness $st_{ik}(\rho)$ is calculated by a direct subjective (DS) trustworthiness function, DS_0 or DS_1. In addition, the subjective trustworthiness $st_{ij}(\rho)$ to the acquainter p_j of p_k is changed. The larger the subjective trustworthiness $s_{ik}(\rho)$ of the servicing peer p_k is, the more $st_{ij}(\rho)$ to the acquainter p_j is increased. Let S_{ij} and S_{ik} be the current subjective trustworthiness values of the peer p_i to the acquainter p_j and to the serving peer p_k, respectively. Let s_{ik} be the satisfiability value which p_i obtained from p_k for the access request ρ. α_i is the DS constant which is used in the function (3). β_i is also a *indirect subjective trustworthiness* (IS) constant ($0 \leq \beta_i \leq 1$). The subjective trustworthiness $st_{ij}(\rho)$ is first calculated by the following function:

$$IS_1(S_{ij}, s_{ik}, \beta_i) := \beta_i \cdot S_{ij} + (1 - \beta_i) \cdot s_{ik}. \tag{2.4}$$

Usually, β_i is α_i. The IS function (4) is the same as $DS_1(S_{ij}, s_{ik}, \alpha_i)$ if $\beta_i = \alpha_i$.

The acquainter p_j may only know a serving peer p_k whose subjective trustworthiness $st_{jk}(\rho)$ is small. If the peer p_j introduces such a less trustworthy acquaintance p_k to the requesting peer p_i, the peer p_i decreases the subjective trustworthiness $st_{ij}(\rho)$ to the acquainter p_j by the formula (4). Hence, if a peer p_j knows only acquaintances whose subjective trustworthiness values are smaller, p_j is wondering if p_j loses the trustworthiness from p_i and does not acquaint p_i with any peer. In this paper, the acquaintance p_j informs p_i of not only a serving peer

p_k but also its subjective trustworthiness $st_{jk}(\rho)$. If the satisfiability value $s_{ik}(\rho)$ is closer to the subjective trustworthiness $st_{jk}(\rho)$, the subjective trustworthiness $st_{ij}(\rho)$ of p_i to the acquainter p_j is increased. Otherwise, $st_{ij}(\rho)$ is decreased.

$$IS_2(S_{ij}, S_{jk}, s_{ik}, \beta_i) = \beta_i \cdot S_{ij} + (1 - \beta_i) \cdot \delta(S_{jk}, s_{ik}).$$

$$\delta(S, s) = \begin{cases} 1 & \text{if } |S - s|/S \leq \epsilon_i \\ (1 - |S - s|/S) & \text{otherwise.} \end{cases} \tag{2.5}$$

ϵ_i is a constant ($0 \leq \epsilon_i \leq 1$). For ϵ_i = 0, $\delta(S_{jk}, s_{ik})$ = 1 if $S_{jk} = s_{ik}$.

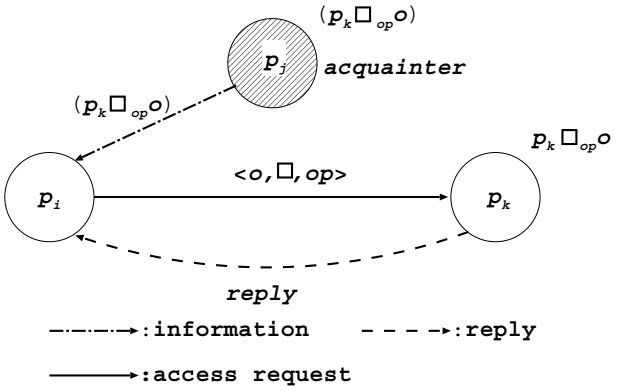


Figure 2.4: Acquainter.

For example, let us consider three peers p_i, p_j, and p_k as shown in Figure 2.4. Here, p_j is an acquainter of p_i and p_k in a serving peer p_k of an access request ρ. The peer p_i asks the acquaintance p_j to acquaint p_i with a serving peer for the access request ρ. Then, the acquaintance p_j acquaints p_i with a serving peer p_k and also informs p_i of the subjective trustworthiness $st_{jk}(\rho)$. The peer p_i issues an access request ρ to p_k. Suppose the subjective trustworthiness S_{ij} = 0.5 (= $st_{ij}(\rho)$) and S_{jk} = 0.4. Suppose the peer p_i receives a response $r(\rho)$ from the serving peer p_k and the satisfiability value s_{ik} = 0.8 is obtained. Suppose β_i = 0.8. The new $st_{ij}(\rho)$ is obtained as $IS_1(S_{ij}, s_{ik}, \beta_i)$ = 0.8 · 0.5 + (1 - 0.8) · 0.8 = 0.4 + 0.16 = 0.56. Since the acquaintance p_j introduces a more trustworthy peer p_k to p_i, the subjective trustworthiness $st_{ij}(\rho)$ is increased to 0.56 from 0.5. On the other hand, the subjective trustworthiness S_{jk} (= $st_{jk}(\rho)$) of the acquaintance p_j to p_k is 0.4

but the satisfiability $s_{ik}(\rho)$ which p_i just obtains from p_k is 0.8. The difference between S_{jk} and s_{ik} is not small. Here, $IS_2(S_{ij}, S_{jk}, s_{ik}, \beta_i) = \beta_i \cdot \delta_{ij} + (1 - \beta_i) \cdot |S_{jk} - s_{ik}| / S_{jk} = 0.8 \cdot 0.5 + (1 - 0.8) \cdot (1 - |0.4 - 0.8| / 0.4) = 0.4$.

Each peer p_i is similarly classified into *shortsighted*, *middlesighted*, and *longsighted* ones with respect to the IS constant β_i as discussed in the DS constant α_i.

2.3 Objective trustworthiness

2.3.1 Types of objective trustworthiness

A peer p_i listens to what trustworthiness opinions on an acquaintance p_j other peers have with respect to a service type ρ (= $\langle o, \square, op \rangle$). In the first way, p_i collects an opinion on the trustworthiness of the acquaintance p_j, i.e. the subjective trustworthiness $st_{kj}(\rho)$ of each peer p_k to the acquaintance p_j. Then, p_i takes the average of the subjective trustworthiness values obtained. This is the traditional *reputation* concept [37]. However, every opinion collected may not be correct. For example, since some peer p_k has not communicated with p_j for a long time, the peer p_k holds just obsolete subjective trustworthiness $st_{ij}(\rho)$ to p_j. We have to exclude such faulty trustworthiness opinions.

It is not easy to recognize a faulty acquaintance which informs the peer p_i of faulty subjective trustworthiness. In our approach to excluding faulty trustworthiness opinions, a peer p_i makes a decision on which an acquaintance p_j is faulty based on its own subjective trustworthiness $st_{ij}(\rho)$ depending on the confidence of p_i. If p_i is not confident of its own opinion $st_{ij}(\rho)$, p_i obeys the opinions of an acquaintance p_k on the trustworthiness st_{kj} of p_j. Here, p_i collects opinions of other peers which know about the peer p_j. If p_i is the most confident of its opinion, subjective trustworthiness $st_{ij}(\rho)$, p_i takes only its own trustworthiness on the acquaintance p_j. These two ways are at the extreme ends. There are some intermediate ways to obtain the objective trustworthiness:

1. A peer p_i collects the subjective trustworthiness $st_{kj}(\rho)$ from every acquaintance p_k of p_j.
2. p_i collects the subjective trustworthiness $st_{kj}(\rho)$ from every acquaintance p_k of p_i.
3. p_i collects the subjective trustworthiness $st_{kj}(\rho)$ from every trustworthy acquaintance p_k, where $st_{ik}(\rho) \geq \lambda_i$, i.e. an acquaintance p_k which p_i can trust.

4. p_i collects the subjective trustworthiness $st_{kj}(\rho)$ from every trustworthy acquaintance p_k, whose $st_{kj}(\rho)$ is similar to its own one $st_{ij}(\rho)$.

In the first way, the peer p_i takes the general public opinion on the trustworthiness of p_j. In the other ways, p_i takes the specific opinions of the peers which p_i can trust. In this paper, we postulate that peers which a peer p_i can trust are acquaintances of p_i. In the second way, p_i takes opinions of all of its acquaintances. In the third way, p_i does not consider all the acquaintances but takes only the opinions of the acquaintances which p_i can trust. λ_i is a *trustworthiness* constant ($0 \leq \tau_i \leq 1$). The peer p_i thinks an acquaintance $p_k R$ to be trusted if $st_{ik}(\rho) \geq \lambda_i$. Here, even a trustworthy acquaintance p_k shows a less trustworthiness opinion $st_{kj}(\rho)$. If p_i is confident of its own opinion $st_{ij}(\rho)$, p_i takes its own opinion $st_{ij}(\rho)$ and throws way the opinion of the acquaintance p_k. In the last way, p_i considers only the trustworthy acquaintances whose opinions are similar to p_i.

2.3.2 Computation of trustworthiness

The objective trustworthiness $ot_{ij}(\rho)$ of a requesting peer p_i to an acquaintance p_j shows the general public opinion on the trustworthiness of p_j, i.e. how much the acquaintance p_j is trusted by other peers. Let p_i be a requesting peer and p_j be its acquaintance. The *reputation* [26, 29] of the acquaintance p_j shows how much the acquaintance p_j is trusted by other peers. The reputation is influenced by faulty acquaintances which hold obsolete service information. Let ρ be an access request $\langle o, \Box, op \rangle$.

The reputation [26, 29] of an acquaintance p_j is obtained by the following function:

$$OT_0(p_i, p_j, \rho) := \frac{\sum_{\{p_k | p_j \in V(p_k, \rho)\}} st_{kj}(\rho)}{\mid \{p_k \mid p_j \in V(p_k, \rho)\} \mid}. \tag{2.6}$$

Here, $V(p_k, \rho)$ is a set of acquaintances of a peer p_k which supports with service ρ.

In order to exclude the subjective trustworthiness of every faulty peer, each requesting peer p_i first only considers every acquaintance p_k of both p_j and p_i to calculate the objective trustworthiness $ot_{ij}(\rho)$.

$$OT_1(p_i, p_j, \rho) := \frac{\sum_{p_k \in V(p_i, \rho)} st_{kj}(\rho)}{\mid V(p_i, \rho) \mid}. \tag{2.7}$$

Even an acquaintance p_k of a peer p_i might be faulty, i.e. p_k has obsolete service information on a peer p_j. In OT_1, the trustworthiness of faulty acquaintances

are still considered. Next, less trustworthy acquaintances of the requesting peer p_i are not considered to calculate the objective trustworthiness $ot_{ij}(\rho)$.

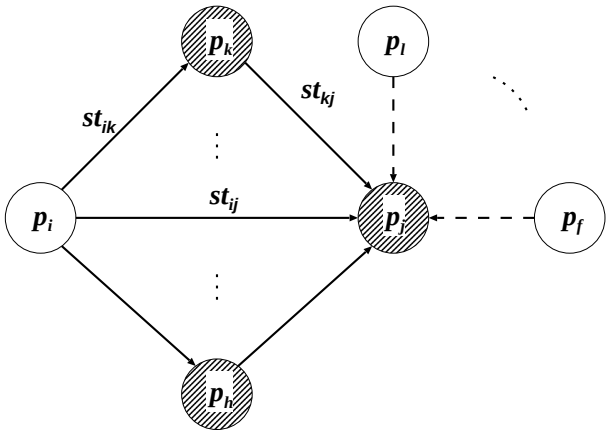


Figure 2.5: Objective trustworthiness.

Each peer p_i calculates the objective trustworthiness $ot_{ij}(\rho)$ by the following function:

$$OT_2(p_i, p_j, \rho) := \frac{\sum_{p_k \in V(p_i,\rho) \wedge st_{ik}(\rho) \geq \lambda_i} st_{kj}(\rho)}{\mid \{p_k \in V(p_i, \rho) \mid st_{ik}(\rho) \geq \lambda_i\} \mid}. \tag{2.8}$$

Hence, only the subjective trustworthiness $st_{ik}(\rho)$ of the trustworthy acquaintance p_k is considered to calculate the objective trustworthiness, $ot_{ij}(\rho)$ where $st_{ik}(\rho) \geq \lambda_i$ for a trustworthiness constant λ_i ($0 \leq \lambda_i \leq 1$). This means, the requesting peer p_i perceives that p_i can trust p_k if $st_{ik}(\rho) \geq \lambda_i$. The subjective trustworthiness $st_{kj}(\rho)$ of a less trustworthy acquaintance p_k to the peer p_j is removed in the function OT_2. If an acquaintance p_k is more trustworthy to the requesting peer p_i, p_i more trusts the opinion of p_k on p_j.

Let us consider an example where there are six peers p_0, p_1, p_2, p_3, p_4, and p_5. Here, suppose the $V(p_0, \rho) = \{p_1, p_2, p_3, p_4\}$ and $V(p_1, \rho) = \{p_0, p_2, p_3, p_4, p_5\}$ for an access request ρ. Suppose the subjective trustworthiness $st_{01}(\rho)$ of the peer p_0 is given as 0.7, $st_{11}(\rho) = 1.0$, $st_{02}(\rho) = 0.7$, $st_{03}(\rho) = 0.0$, $st_{04}(\rho) = 0.4$, $st_{21}(\rho) = 0.8$, $st_{31}(\rho) = 0.9$, $st_{41}(\rho) = 0.6$, and $st_{51}(\rho) = 0.5$ as shown in Figure 2.6. According to the traditional reputation concepts [26, 29], the objective

trustworthiness $ot_{01}(\rho)$ is given as $OT_0(p_0, p_1, \rho)$ = [$st_{01}(\rho)$ + $st_{21}(\rho)$ + $st_{31}(\rho)$ + $st_{41}(\rho)$ + $st_{51}(\rho)$] / 5 = 0.7. Next, only common acquaintances of p_0 and p_1, i.e. p_1, p_2, p_3, and p_4 are considered in OT_1, i.e. $OT_1(p_0, p_1, \rho)$ = [$st_{01}(\rho)$ + $st_{21}(\rho)$ + $st_{31}(\rho)$ + $st_{41}(\rho)$] / 4 = 0.75. Here, $st_{51}(\rho)$ is not calculated since p_i is not an acquaintance of p_0. In the function OT_1, p_3 is not trusted by p_0, i.e. $st_{03}(\rho)$ = 0.0. $st_{31}(\rho)$ is not considered in OT_2 p_3 trusts p_1 since p_0 does not trust p_3. In the function OT_2, only the subjective trustworthiness of a trustworthy acquaintance of p_0 is considered. The objective trustworthiness, $ot_{01}(\rho)$ is given by $OT_2(p_0, p_1, \rho)$ = [$st_{11}(\rho)$ + $st_{21}(\rho)$ + $st_{41}(\rho)$] / 3 = 0.8 for λ_i = 0.1.

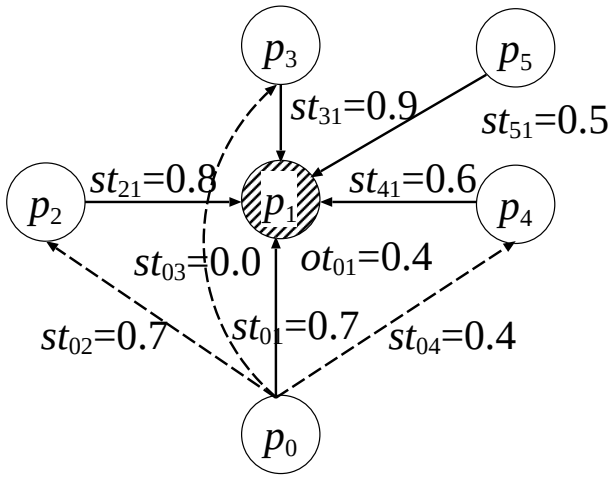


Figure 2.6: Objective trustworthiness ot_{01}.

In our life, each person finally makes a decision based on its own opinion even if other people show different opinions. A peer p_i first removes acquaintances' opinions quite different from its own opinion. *Watanabe et al.* discuss the ranking factor with the deviation based on this rule. We introduce the following function OT_3 to obtain the objective trustworthiness $ot_{ij}(\rho)$ based on the idea:

$$T_{ikj}(\rho) = \begin{cases} \sqrt{st_{ik}(\rho) \cdot st_{kj}(\rho)} \\ \text{if} \sqrt{\mid st_{ij}^2(\rho) - st_{ik}(\rho) \cdot st_{kj}(\rho) \mid} \leq \varphi_i. \\ 0 \qquad\qquad\qquad \text{otherwise}. \end{cases} \tag{2.9}$$

$$OT_3(p_i, p_j, \rho) := \frac{\sum_{p_k \in V(p_i,\rho)} T_{ikj}(\rho)}{\mid \{p_k \in V(p_i,\rho) \mid T_{ikj}(\rho) \neq 0\} \mid}. \tag{2.10}$$

Here, φ_i is a constant ($0 \leq \varphi_i \leq 1$). In Figure 2.6, $T_{011}(\rho) = 0$, $T_{021}(\rho) = \sqrt{ot_{02}(\rho) \cdot ot_{21}(\rho)} = \sqrt{0.7 \cdot 0.8} = 0.748$, and $T_{041}(\rho) = \sqrt{ot_{04}(\rho) \cdot ot_{41}(\rho)} = \sqrt{0.4 \cdot 0.6} = 0.490$.

Let φ_0 be 0.5. $\sqrt{\mid st_{02}(\rho) \cdot st_{21}(\rho) - st_{01}(\rho)^2 \mid} = \sqrt{\mid 0.56 - 0.49 \mid} = \sqrt{0.07} \leq 0.5$. $\sqrt{\mid st_{04}(\rho) \cdot st_{41}(\rho) - st_{01}(\rho)^2 \mid} = \sqrt{0.25} \leq 0.5$. The objective trustworthiness $ot_{01}(\rho)$ is $OT_3(p_0, p_1, \rho) = (\sqrt{st_{01}(\rho) \cdot st_{11}(\rho)} + \sqrt{st_{02}(\rho) \cdot st_{21}(\rho)} + \sqrt{st_{04}(\rho) \cdot st_{41}(\rho)}) / 3 = (\sqrt{0.7 \cdot 1.0} + \sqrt{0.8 \cdot 0.7} + \sqrt{0.6 \cdot 0.4}) / 3 = 0.692$. If $\varphi_0 = 0.3$, $OT_3(p_0, p_1, \rho) = \sqrt{st_{02}(\rho) \cdot st_{01}(\rho)} = \sqrt{0.8 \cdot 0.7} = 0.75$. Thus, only the acquaintance p_k where $\sqrt{st_{ik}(\rho) \cdot st_{kj}(\rho)}$ is closer to the subjective trustworthiness $st_{ij}(\rho)$ is taken into account if φ_0 is getting smaller. The constant φ_0 means that p_0 takes only its own opinion to p_1.

An objective trustworthiness function $OT(p_i, p_j, \rho)$ means some of $OT_h(p_i, p_j, \rho)$ ($h = 0, 1, 2, 3$). OT_h is *higher* than OT_k if $h > k$. The higher OT_h is, the more the objective trustworthiness $ot_{ij}(\rho)$ of an acquaintance p_j depends on the requesting peer p_i.

We discuss the trustworthiness $st_{ij}(\rho)$ and $ot_{ij}(\rho)$ with respect to a specific service type ρ. An acquaintance peer p_j supports multiple types p_{j1}, ..., p_{jl_j}. We define the aggregate trustworthiness st_{ij} and ot_{ij} as follows.

$$st_{ij} = \sum_{k=1,\ldots,l_j} st_{ij}(\rho_{jk}). \tag{2.11}$$

$$ot_{ij} = \sum_{k=1,\ldots,l_j} ot_{ij}(\rho_{jk}). \tag{2.12}$$

2.4 Confidence on subjective trustworthiness

As discussed in the preceding sections, a peer p_i obtains the subjective trustworthiness $st_{ij}(\rho)$ and objective trustworthiness $ot_{ij}(\rho)$ from the trustworthiness opinions of other peers on a peer p_j. Then, the peer p_i has to decide on how much the peer p_i can trust the acquaintance p_j. It depends on how much a peer p_i is confident of its own opinion $st_{ij}(\rho)$ on an acquaintance p_j. As discussed here, a most confident peer p_i takes the subjective trustworthiness $st_{ij}(\rho)$. On the other hand, a least confident peer p_i takes the objective trustworthiness $ot_{ij}(\rho)$ decided by the

lowest level function OT_0. Let $cf_{ij}(\rho)$ show the confidence of a peer p_i to an acquaintance p_j with respect to a service type ρ ($0 \leq cf_{ij}(\rho) \leq 1$). We discuss how to compute the confidence $cf_{ij}(\rho)$. There are two types of confidence, subjective confidence $sf_{ij}(\rho)$ and objective confidence of $of_{ij}(\rho)$ as discussed in the trustworthiness. First, we consider the subjective confidence $sf_{ij}(\rho)$ which a peer p_i obtains through issuing a service request ρ to an acquaintance. Suppose a peer p_i issues an access request ρ to an acquaintance p_j and receives a reply $r(\rho)$ from p_j. Then, p_i obtains the subjective trustworthiness $st_{ij}(\rho)$ as discussed. If p_i had not communicated with the acquaintance p_j for a long time, p_i is less confident of its own $st_{ij}(\rho)$ since the types and quality of service supported by p_j might be changed. The confidence also depends on how frequently p_i has communicated with p_j. Even if p_i often communicates with p_j, p_i might not be confident. For example, p_j may issue messages to p_i like DoS attacks [30]. The acquaintance p_j might have sent replies with different satisfiability values. In this paper, if the peer p_i receives replies from the acquaintance p_j whose satisfiability values are similar, p_i is more confident. Thus, we consider the following parameters to compute the subjective confidence $sf_{ij}(\rho)$:

1. $l_{ij}(\rho)$ = communication time, i.e. how long a peer p_i has communicated with an acquaintance p_j with respect to a service request ρ [sec].

2. $f_{ij}(\rho)$ = communication frequently, i.e. how frequently p_i has communicated with p_j with respect to ρ [req/sec].

3. $v_{ij}(\rho)$ = variance of satisfiability values of replies $r(\rho)$ which p_i has received from p_j.

The subjective confidence $sc_{ij}(\rho)$ is given in a tuple $\langle l_{ij}(\rho), f_{ij}(\rho), v_{ij}(\rho)\rangle$. Let $c_1 = \langle c_{11}, c_{12}, c_{13}\rangle$ and $c_2 = \langle c_{21}, c_{22}, c_{23}\rangle$ be subjective confidence values. Here, $c_1 \geq c_2$ iff $c_{11} \geq c_{21}$, $c_{12} \geq c_{22}$, and $c_{13} \geq c_{23}$.

Next, a peer p_i can obtain the confidence by comparing its opinion with other peers. If a peer p_i knows a more number of peers have similar opinions, On the other hand, a peer p_i can be confident if another peer p_j trusts p_i. The objective confidence $of_{ij}(\rho)$ of a peer p_i to an acquaintance p_j with respect to a service type ρ is obtained in terms of trustworthiness opinions of other peers. A person can be confident if more people think the person to be trustworthy. Thus, the more number of peers trust a peer p_i, the more the peer is confident. We take the following parameter: $\tau_{ij}(\rho)$ = number of acquaintances which trust a peer p_i, i.e. $\{\ p_k \mid p_k \in \mathrm{V}(p_i, \rho)$ and $st_{kj}(\rho) \geq \lambda_i\}$. The confidence $cf_{ij}(\rho)$ is given in a tuple

$\langle l_{ij}(\rho)$, $f_{ij}(\rho)$, $v_{ij}(\rho)$, $\tau_{ij}(\rho)\rangle$. Here, let c_k be a tuple $\langle c_{k1}, c_{k2}, c_{k3}, c_{k4}\rangle$. For a peer of tuple c_1 and c_2, $c_1 \geq c_2$ iff $c_{11} \geq c_{21}$, $c_{12} \geq c_{22}$, $c_{13} \geq c_{23}$, and $c_{14} \geq c_{24}$.

Chapter 3

A Basic Agreement Protocol

3.1 Precedent relations

At each round, a peer p_i takes a value v_i^t (= LD_i ($v_1^{t-1}, \ldots, v_n^{t-1}$)) at round t. The value v_i^t may depend on the previous value v_i^{t-1}. We define the *existentially (E-) precedent* relation $\rightarrow_i^E$ ($\subseteq D_i^2$) and *preferentially (P-) precedent* relation $\rightarrow_i^P$ ($\subseteq D_i^2$) on the domain D_i to show with which value a peer p_i can take after a current value at each round.

[Definition] For every pair of values v_1 and v_2 in a domain D_i of a peer p_i,

1. v_1 *E-precedes* v_2 in the peer p_i ($v_1 \rightarrow_i^E v_2$) if and only if (*iff*) the peer p_i is allowed to take v_1 after v_2.

2. v_1 *P-precedes* v_2 in p_i ($v_1 \rightarrow_i^P v_2$) iff p_i prefers v_1 to v_2.

3. $v_1 \rightarrow_i^E v_2$ and $v_1 \rightarrow_i^P v_2$ if $v_1 \rightarrow_i^E v_3 \rightarrow_i^E v_2$ and $v_1 \rightarrow_i^P v_3 \rightarrow_i^P v_2$ for some value v_3, respectively.

In the commitment protocols [47, 54], a peer which sends *commit* may *abort* if the coordinator peer indicates *abort*. However, a peer which notifies other processes of *abort* unilaterally aborts, i.e. cannot take *commit*. That is, a peer can change *commit* with *abort* but cannot change *abort* with any value, *commit* $\rightarrow_i^E$ *abort* but *abort* $\not\rightarrow_i^E$ *commit*. In another example of distributed auction system [47], each person cannot show a cheaper value v_2 than a previous value v_1. Here, $v_1 \rightarrow_i^E v_2$ where $v_2 > v_1$.

Suppose a peer p_i can take a pair of values v_1 and v_2 after taking the value v in the E-dominant relation $\rightarrow_i^E$, i.e. $v \rightarrow_i^E v_1$ and $v \rightarrow_i^E v_2$. Suppose neither

$v_1 \rightarrow_i^E v_2$ nor $v_2 \rightarrow_i^E v_1$, i.e. the peer p_i is allowed to take any value of v_1 and v_2 after taking the value v. Here, the peer p_i has to take one of the values v_1 and v_3. For example, if a peer p_i prefers a value v_1 to another value v_2 ($v_2 \rightarrow_i^P v_1$), the peer p_i would like to take the value v_1. It is noted that the peer p_i may take the value v_2 even if the value v_1 is more preferable than the value v_2.

The precedent relations $\rightarrow_i^E$ and $\rightarrow_i^P$ with the domain D_i are assumed to be *a priori* specified when each peer p_i is initiated. In a homogeneous system, every peer p_i has the same relations $\rightarrow_i^E$ and $\rightarrow_i^P$ on the same domain D_i. In a heterogeneous system, some pair of peers p_i and p_j have different relations $\rightarrow_i^E \neq \rightarrow_j^E$ or $\rightarrow_i^P \neq \rightarrow_j^P$ on different domains, $D_i \neq D_j$.

There are the following relations between a pair of values v_1 and v_2 in the domain D_i of a peer p_i:

1. v_1 is *E-equivalent* with v_2 in p_i ($v_1 \equiv_i^E v_2$) iff $v_1 \rightarrow_i^E v_2$ and $v_2 \rightarrow_i^E v_1$.
2. v_2 is more *E-significant* than v_1 in p_i ($v_1 \prec_i^E v_2$) iff $v_1 \rightarrow_i^E v_2$ but $v_2 \not\rightarrow_i^E v_1$.
3. v_1 *E-dominates* v_2 in p_i ($v_1 \preceq_i^E v_2$) iff $v_1 \prec_i^E v_2$ or $v_1 \equiv_i^E v_2$.
4. v_1 is *E-incomparable* with v_2 in p_i ($v_1 \mid_i^E v_2$) iff neither $v_1 \rightarrow_i^E v_2$ nor $v_2 \rightarrow_i^E v_1$.
5. v_1 is *P-equivalent* with v_2 in p_i ($v_1 \equiv_i^P v_2$) iff $v_1 \rightarrow_i^P v_2$ and $v_2 \rightarrow_i^P v_1$.
6. v_1 is more *P-significant* than v_2 in p_i ($v_2 \prec_i^P v_1$) iff $v_1 \rightarrow_i^P v_2$ but $v_2 \not\rightarrow_i^P v_1$.
7. v_1 *P-dominates* v_2 in p_i ($v_2 \preceq_i^P v_1$) iff $v_2 \prec_i^P v_1$ and $v_2 \equiv_i^P v_1$.
8. v_1 and v_2 are *P-incomparable* in p_i ($v_1 \mid_i^P v_2$) iff neither $v_1 \rightarrow_i^P v_2$ nor $v_2 \rightarrow_i^P v_1$.

A value v_1 is referred to as *maximal* and *minimal* iff there is no value v_2 such that $v_1 \rightarrow_i^E v_2$ and $v_2 \rightarrow_i^E v_1$ in the domain D_i, respectively. For example, *abort* is maximal and *commit* is minimal in the commitment protocol. If a peer p_i takes a maximal value v in the domain D_i, the peer p_i cannot take any new value. On the other hand, a peer p_i can take a value after taking a minimal value in the domain D_i. A value v_1 is referred to as *top* iff $v_2 \rightarrow_i^E v_1$ for every value v_2 in D_i. A value v_1 is referred to as *bottom* iff $v_1 \rightarrow_i^E v_2$ for every value v_2 in D_i. Let $Corn_i(x)$ be a set of values which a peer p_i can take after taking a value x in a domain D_i, $Corn_i(x) = \{ y \mid x \rightarrow_i^E y \} \subseteq D_i$. If a value x is maximal, $Corn_i(x) = \phi$. If there

are multiple values which a peer p_i can take after a value x, i.e. $|Corn_i(x)| \geq 2$, the value x is referred to as *branchable* in the domain D_i.

A *least upper bound* (*lub*) of values v_1 and v_2 ($v_1 \sqcup_i^E v_2$) is a value v_3 in the domain D_i such that $v_1 \rightarrow_i^E v_3$, $v_2 \rightarrow_i^E v_3$, and there is no value v_4 such that $v_1 \rightarrow_i^E v_4 \rightarrow_i^E v_3$ and $v_2 \rightarrow_i^E v_4 \rightarrow_i^E v_3$ in a peer p_i. For example a peer p_i takes a value v_i and another peer p_j takes a value v_j at round t. If $\rightarrow_i^E = \rightarrow_j^E$, the peer p_i and p_j can take $v_i \sqcup_i^E v_j$ to make an agreement. A *greatest lower bound* (*glb*) of values v_1 and v_2 ($v_1 \sqcap_i^E v_2$) is a value v_3 in the domain D_i such that $v_3 \rightarrow_i^E v_1$, $v_3 \rightarrow_i^E v_2$, and there is no value v_4 such that $v_3 \rightarrow_i^E v_4 \rightarrow_i^E v_1$ and $v_3 \rightarrow_i^E v_4 \rightarrow_i^E v_2$. A least upper bound (*lub*) $\sqcup_i^P$ and greatest lower bound (*glb*) $\sqcap_i^P$ are defined for the P-dominant relation $\rightarrow_i^P$ in the same way as $\sqcup_i^E$ and $\sqcap_i^E$.

3.1.1 E-precedent relation

A system S is composed of *n* (≥ 1) peer processes $p_1, \ldots, p_n$. A *domain* D_i of a process p_i is a set of possible values which the process p_i can take. Each process p_i initially takes a value v_i^0 in the domain D_i and notifies the other processes of the value v_i^0. A process p_i receives a value v_j^0 from every other process p_j ($j = 1, \ldots, n$). The process p_i takes another value v_i^1 from a tuple $\langle v_1^0, \ldots, v_n^0 \rangle$. This is the first round. Then, the process p_i notifies the other processes of the value v_i^1. Thus, at the t^{th} round, the process p_i collects a tuple v^{t-1} = $\langle v_1^{t-1}, \ldots, v_i^{t-1}, \ldots, v_n^{t-1} \rangle$ where p_i takes a value v_i^{t-1} and receives a value v_j^{t-1} from each other process p_j ($j \neq i$). If v^{t-1} satisfies the agreement condition AC_i, the process p_i obtains one value v from v^{t-1} as an agreement value and terminates. Otherwise, p_i takes a value v_i^t in the domain D_i and notifies the other processes of v_i^t. Here, p_i changes the opinion from the value v_i^{t-1} to v_i^t. Here, $v_i^0, \ldots, v_i^{t-1}$ are referred to as *previous* values and v_i^t is a *current* value.

In the commitment protocols [47, 54], a process which notifies *commit* may *abort* if the coordinator process indicates *abort* to the process after receiving the values from the processes. Here, a process which notifies *abort* cannot take *commit*. Each process p_i can take a value v_i^t after taking a value v_i^{t-1} in the domain D_i at round *t* if p_i can change v_i^{t-1} to v_i^t. Here, p_i changes the opinion from v_i^{t-1} to v_i^t. If p_i cannot take any value from v_i^{t-1}, p_i still takes the value v_i^{t-1} as the current value v_i^t or backs to the previous value v_i^{t-2} and tries to take another value from v_i^{t-2}.

[Definition] A value v_1 is referred to as *existentially (E) precede* a value v_2 with respect to a process p_i ($v_1 \rightarrow_i^E v_2$) if and only if (*iff*) the process p_i can take v_1 after taking v_2 in the domain D_i ($\rightarrow_i^E \subseteq D_i^2$).

We assume the relation $\rightarrow_i^E$ to be transitive. A value v_1 is *E-equivalent* with a value v_2 in a process p_i ($v_1 \equiv_i^E v_2$) iff $v_1 \rightarrow_i^E v_2$ and $v_2 \rightarrow_i^E v_1$. A value v_1 is *E-uncomparable* with a value v_2 in a process p_i ($v_1 \mid_i^E v_2$) iff neither $v_1 \rightarrow_i^E v_2$ nor $v_2 \rightarrow_i^E v_1$. A value v_1 *E-dominates* a value v_2 in p_i ($v_1 \prec_i^E v_2$) iff $v_1 \rightarrow_i^E v_2$ but $v_2 \not\rightarrow_i^E v_1$. $v_1 \preceq_i^E v_2$ iff $v_1 \prec_i^E v_2$ or $v_1 \equiv_i^E v_2$. In the commitment protocol, *commit* $\rightarrow_i^E$ *abort* but *abort* $\not\rightarrow_i^E$ *commit* for every process p_i as presented here. Hence, *commit* $\prec_i^E$ *abort*.

For every pair of values v_1 and v_2, v_1 *E-precedes* v_2 ($v_1 \rightarrow^E v_2$), v_1 is *E-equivalent* with v_2 ($v_1 \equiv^E v_2$), v_2 is more *E-significant* than v_1 ($v_1 \prec^E v_2$), v_2 *E-dominates* v_1 ($v_1 \preceq^E v_2$), and v_1 is *E-uncomparable* with v_2 ($v_1 \mid^E v_2$) iff $v_1 \rightarrow_i^E v_2$, $v_1 \equiv_i^E v_2$, $v_1 \prec_i^E v_2$, $v_1 \preceq_i^E v_2$, and $v_1 \mid_i^E v_2$ for every process p_i, respectively.

A process p_i can take a value v_2 after taking another value v_1 if $v_1 \rightarrow_i^E v_2$. Here, suppose that the process p_i had taken v_2 before v_1. Question is whether or not p_i can take again a value which p_i has previously taken.

[Definition] A value v_1 *acyclically E-precedes* (*AE-precedes*) a value v_2 in a process p_i ($v_1 \Rightarrow_i^E v_2$) iff p_i can take v_2 after taking v_1 and p_i had previously not taken v_2.

A *least upper bound* (*lub*) of values v_1 and v_2 ($v_1 \sqcup_i^E v_2$) is a value v_3 in the domain D_i such that $v_1 \rightarrow_i^E v_3$, $v_2 \rightarrow_i^E v_3$, and there is no value v_4 such that $v_1 \rightarrow_i^E v_4 \rightarrow_i^E v_3$ and $v_2 \rightarrow_i^E v_4 \rightarrow_i^E v_3$. Suppose there are a pair of processes p_1 and p_2 notifying one another of values v_1 and v_2, respectively. Suppose the processes p_1 and p_2 have the same E-precedent relation, $\rightarrow_1^E = \rightarrow_2^E = \rightarrow^E$ on the same domain D, $D_1 = D_2 = D$. Here, $\sqcup_1^E = \sqcup_2^E = \sqcup^E$ and $\sqcap_1^E = \sqcap_2^E = \sqcap^E$. If there exists an *lub* $v_3 = v_1 \sqcup^E v_2$, both p_1 and p_2 can take v_3 after taking v_1 and v_2, respectively. A *greatest lower bound* (*glb*) of values v_1 and v_2 ($v_1 \sqcap_i^E v_2$) is a value v_3 in D_i such that $v_3 \rightarrow_i^E v_1$, $v_3 \rightarrow_i^E v_2$, and there is no value v_4 such that $v_3 \rightarrow_i^E v_4 \rightarrow_i^E v_1$ and $v_3 \rightarrow_i^E v_4 \rightarrow_i^E v_2$. The processes p_1 and p_2 can also take the *glb* $v_4 = v_1 \sqcap^E v_2$ as an agreement value if the processes can take previous values again. In this paper, there exist a pair of special values, *bottom* value $\bot_i^E$ and *top* value $\top_i^E$ where $\bot_i^E \rightarrow_i^E v$ and $v \rightarrow_i^E \top_i^E$ for every value v in D_i. This means that a process p_i can take any value in D_i after taking the bottom $\bot_i^E$. On the other hand, p_i taking the top $\top_i^E$ cannot change the value. In the commitment protocol [42, 43, 44], each process p_i has a binary domain $D_i = \{$*abort*, *commit*$\}$ where *commit* $\rightarrow_i^E$ *abort*. Here, *abort* is the top $\top_i^E$ and *commit* is the bottom $\bot_i^E$. The value *abort* E-dominates *commit* (*commit* $\prec_i^E$ *abort*).

[Definition] Let $\rightarrow_i^E$ and $\rightarrow_j^E$ be E-precedent relations of processes p_i and p_j, respectively. A pair of precedent relations $\rightarrow_i^E$ and $\rightarrow_j^E$ are *existentially (E) con-*

sistent ($\rightarrow_i^E \cong^E \rightarrow_j^E$) iff for every pair of values v_1 and v_2 in $D_i \cap D_j$, $v_1 \rightarrow_i^E v_2$ iff $v_1 \rightarrow_j^E v_2$.

A pair of E-precedent relations $\rightarrow_i^E$ and $\rightarrow_j^E$ are *E-inconsistent* ($\rightarrow_i \ncong^E \rightarrow_j$) iff $\rightarrow_i^E$ and $\rightarrow_j^E$ are not consistent. Let us consider a pair of processes p_1 and p_2. Here, D_1 = { *a, b, c* } and D_2 = { *a, b, d, e* }. In the process p_1, *a* $\rightarrow_1^E$ *b* and *a* $\rightarrow_1^E$ *c*. In the process p_2, *a* $\rightarrow_2^E$ *b* $\rightarrow_2^E$ *d* and *b* $\rightarrow_2^E$ *e*. Here, $\rightarrow_1^E \neq \rightarrow_2^E$ but $\rightarrow_1^E$ and $\rightarrow_2^E$ are E-consistent ($\rightarrow_1^E \cong \rightarrow_2^E$) since $D_1 \cap D_2$ = { *a, b* } and *a* $\rightarrow_1^E$ *b* and *a* $\rightarrow_2^E$ *b*. Another process p_3 has a domain D_3 = { *a, b, e* } where *b* $\rightarrow_3^E$ *a*. Here, the E-precedent relation $\rightarrow_3^E$ is not E-consistent with $\rightarrow_1^E$ ($\rightarrow_3^E \ncong \rightarrow_1^E$) since *a* $\rightarrow_1^E$ *b* but *b* $\rightarrow_3^E$ *a* for $D_1 \cap D_3$ = { *a, b* }.

In the E-precedent relation $\rightarrow_i^E$ ($\subseteq D_i^2$), a process p_i makes a decision on a value v_2 which p_i notifies to the other processes depending on a value v_1 most recently taken. That is, p_i takes a value v_2 where $v_1 \rightarrow_i^E v_2$. Let $Next_i^E(v_1)$ be $\{v_2 \mid v_1 \rightarrow_i^E v_2\}$ of values which p_i can take next from a value v_1. For example, $Next_i^E$(*commit*) = {*commit, abort*} and $Next_i^E$(*abort*) = {*abort*}.

3.1.2 P-precedent relation

Suppose a process p_i can take a pair of values v_1 and v_2 after taking a value v_3 in the E-dominant relation $\rightarrow_i^E$, i.e. $v_3 \rightarrow_i^E v_1$ and $v_3 \rightarrow_i^E v_2$. Here, the process p_i has to take one of the values v_1 and v_3. If p_i prefers v_1 to v_2 ($v_2 \rightarrow_i^P v_1$), p_i can first take v_1. A partially ordered relation $\rightarrow_i^P \subseteq D_i^2$ is a *preferentially (P) precedent* relation on the domain D_i.

[Definition] A value v_1 *P-precedes* a value v_2 in a process p_i ($v_1 \rightarrow_i^P v_2$) iff p_i prefers v_1 to v_2.

A value v_1 is *P-equivalent* with a value v_2 in a process p_i ($v_1 \equiv_i^P v_2$) iff $v_1 \rightarrow_i^P v_2$ and $v_2 \rightarrow_i^P v_1$. v_1 is more *P-significant* than v_2 in p_i ($v_2 \prec_i^P v_1$) iff $v_1 \rightarrow_i^P v_2$ but $v_2 \not\rightarrow_i^P v_1$. v_1 *P-dominates* v_2 in p_i ($v_2 \preceq_i^P v_1$) iff $v_2 \prec_i^P v_1$ and $v_2 \equiv_i^P v_1$. A pair of values v_1 and v_2 are *P-uncomparable* in p_i ($v_1 \mid_i^P v_2$) iff neither $v_1 \rightarrow_i^P v_2$ nor $v_2 \rightarrow_i^P v_1$. In addition, $v_1 \rightarrow^P v_2$, $v_1 \equiv^P v_2$, $v_1 \prec^P v_2$, $v_1 \preceq^P v_2$, and $v_1 \mid^E v_2$ iff $v_1 \rightarrow_i^P v_2$, $v_1 \equiv_i^P v_2$, $v_1 \prec_i^P v_2$, $v_1 \preceq_i^P v_2$, and $v_1 \mid_i^E v_2$ for every process p_i, respectively.

The *least upper bound* $\sqcup_i^P$ and *greatest lower bound* $\sqcap_i^P$ are defined for the P-dominant relation $\rightarrow_i^P$ in the same way as $\sqcup_i^E$ and $\sqcap_i^E$. There are special values, *top* $\top_i^P$ and *bottom* $\bot_i^P$ with respect to the P-precedent relation $\rightarrow_i^P$ in the same way as the E-precedent relation $\rightarrow_i^E$. We assume the P-precedent relation $\rightarrow_i^P$ is transitive.

3.2 Coordination procedure

In fully distributed peer-to-peer (P2P) applications, there is no centralized coordinator and every peer makes a decision by itself through communicating with the other peers. In addition, it is a common function of most P2P applications for multiple peers $p_1, \ldots, p_n$ to make an agreement, for example, to fix a date for a meeting of members in a society. A *domain* D_i of a peer p_i is a set of possible values which the peer p_i can take in an agreement procedure. We assume every pair of peers can reliably communicate with one another in the underlying network. We also assume that every peer is reliable, i.e. every peer is not faulty in this paper.

Figure 3.1 shows the overview of the coordination protocol to make an agreement. Each peer p_i initially takes a value v_i^0 in the domain D_i and sends the value v_i^0 to the other processes $p_1, \ldots, p_n$. The peer p_i in turn receives values $v_1^0, \ldots, v_n^0$ from the other peers $p_1, \ldots, p_n$, respectively. The agreement condition AC_i is checked for the tuple $\langle v_1^0, \ldots, v_i^0, \ldots, v_n^0 \rangle$ of the received values. Each peer p_i checks if the tuple $\langle v_1^0, \ldots, v_n^0 \rangle$ satisfies the agreement condition AC_i. There are *agree*, *all*, *majority*, *weighted majority*, and *consonance* types of agreement conditions [44, 45]. For example, the *all*-condition AC_i is satisfied if every peer p_i takes the same value v, i.e. $AC_i(v_1^0, \ldots, v_n^0)$ is true. Every peer p_i is assumed to have the same agreement condition $AC_i = AC$. If the agreement condition AC_i is not satisfied, a peer p_i takes another value v_i^1 in the domain D_i, which is obtained by performing a local decision function LD_i on a tuple $\langle v_1^0, \ldots, v_n^0 \rangle$ of the values, i.e. $v_i^1 = LD_i(v_1^0, \ldots, v_n^0)$. This is the first round. Each peer p_i sends the value v_i^1 to the other peers and receives values from the other peers. Thus, at each round t, each peer p_i collects a tuple $\langle v_1^{t-1}, \ldots, v_n^{t-1} \rangle$ of values received from the other peers. If the tuple $\langle v_1^{t-1}, \ldots, v_i^{t-1}, \ldots, v_n^{t-1} \rangle$ satisfies the agreement condition AC_i, the peer p_i obtains one agreement value $v = GD_i(v_1^{t-1}, \ldots, v_n^{t-1})$ by performing a global decision function GD_i and then terminates. For example, if there is such a value v that more than half of the values are the same in the tuple, the majority agreement condition AC_i is satisfied and then each peer p_i takes the value v as the agreement value. Every peer p_i is assumed to have the same global decision function $GD_i = GD$.

If the agreement condition AC_i is not satisfied, a peer p_i takes a value $v_i^t = LD_i(v_1^{t-1}, \ldots, v_n^{t-1})$ which may be different from the previous value v_i^{t-1}. Then, the peer p_i notifies the other peers of the selected value v_i^t. Thus, the peer p_i changes the opinion from the value v_i^{t-1} to the value v_i^t at each round t. Here, the values $v_i^0, v_i^1, \ldots, v_i^{t-1}$ which the peer p_i has so far taken are referred to as

previous values at round t. The value v_i^t is a *current* value c_i^t at round t.

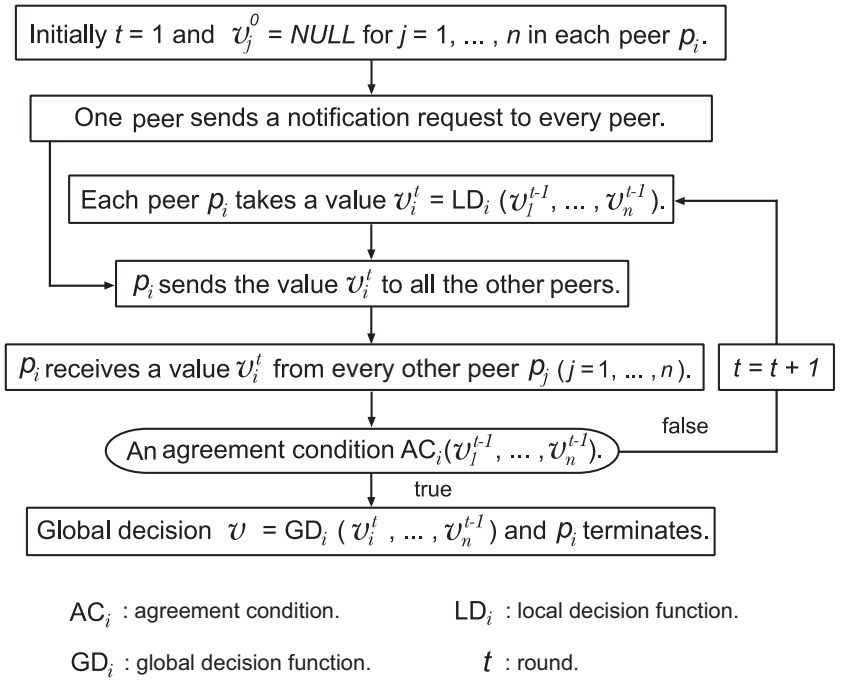


Figure 3.1: Coordination protocol.

3.2.1 Agreement conditions

A predicate AC_i: $D_1 \times \cdots \times D_n \rightarrow \{true, false\}$ is the *agreement condition* of a process p_i on a tuple of values $v_1, \ldots, v_n$. For a tuple of values $\langle v_1, \ldots, v_n \rangle$, $AC_i(v_1, \ldots, v_n)$ = *true* if the processes $p_1, \ldots, p_n$ can make an agreement. We assume every process p_i has the same agreement condition AC in this paper. At each round t, each prcess p_i holds a tuple $\langle v_1^t, \ldots, v_n^t \rangle$ of values notified by the processes $p_1, \ldots, p_n$, respectively. Here, if AC_i $(v_1^t, \ldots, v_n^t)$ is *true*, the coordination protocol terminates. There are following types of agreement conditions:

1. *All condition*: AC_i $(v_1^t, \ldots, v_n^t)$ = *true* if $v_1^t = \ldots = v_n^t$.
2. *Majority condition*: AC_i $(v_1^t, \ldots, v_n^t)$ = *true* if $| \{ p_j \mid v_j^t = v \} | > n/2$.

3. *Maximal condition*: AC_i ($v_1^t, \ldots, v_n^t$) = *true* if $v = v_1^t \sqcup_i \ldots \sqcup_i v_n^t$ in D_i.
4. *Minimal condition*: AC_i ($v_1^t, \ldots, v_n^t$) = *true* if $v = v_i^t \sqcap_i \ldots \sqcap_i v_n^t$ in D_i.
5. *Consonance condition*: AC_i ($v_1^t, \ldots, v_n^t$) = *true* if $v_j^t \neq v_k^t$ for every pair of different processes p_j and p_k.

The conditions 3 and 4 are only used in homogeneous systems.

3.2.2 Global decision function

A function GD_i: $D_1 \times \cdots \times D_n \rightarrow D_i$ is a *global decision function* of a process p_i which gives a value v_i which p_i takes as the global decision. GD_i depends on the agreement condition AC_i. For example, GD_i ($v_1^t, \ldots, v_n^t$) takes a majority value in a set $\{v_1^t, \ldots, v_n^t\}$ if the majority agreement condition AC_i ($v_1^t, \ldots, v_n^t$) is *true*. There are the following types of the agreement conditions:

1. *All condition*: $v = GD_i$ ($v_1^t, \ldots, v_n^t$) if $v_1^t = \ldots = v_n^t = v$.
2. *Majority condition*: $v = GD_i$ ($v_1^t, \ldots, v_n^t$) if $\mid \{v_j^t \mid v_j^t = v\} \mid > n/2$.
3. *Maximal condition*: $v = GD_i$ ($v_1^t, \ldots, v_n^t$) if $v = v_1^t \sqcup_i \ldots \sqcup_i v_n^t$.
4. *Minimal condition*: $v = GD_i$ ($v_1^t, \ldots, v_n^t$) if $v = v_1^t \sqcap_i \ldots \sqcap_i v_n^t$.
5. *Consonance condition*: $v_i^t = GD_i$ ($v_1^t, \ldots, v_n^t$) if $v_j^t \neq v_k^t$ for every pair of different processes p_j and p_k.

3.2.3 Local decision function

A function LD_i: $D_1 \times \cdots \times D_n \rightarrow D_i$ is a *local decision function* of a process p_i which gives a value v_i^{t+1} in the domain D_i from a tuple $\langle v_1^t, \ldots, v_n^t \rangle$. Here, a value v_i^t has to E-precede a value v_i^{t+1} ($v_i^t \rightarrow_i^E v_i^{t+1}$). If there are multiple values which E-precedes v_i^t, p_i takes one of them. $Next_i(v_i^t)$ is a set $\{v \mid v_i^t \rightarrow_i^E v$ in $D_i\}$ of values which E-precede a value v. One strategy to obtain a value which the process p_i to take is that p_i takes the least preferable value in the set $Next_i(v_i^t)$. That is, $\sqcap_{i,v \in E_i}^P v$ is taken by p_i. In another strategy, p_i takes the most preferable value, i.e. $\sqcap_{i,v \in E_i}^P v$.

First, suppose that each process p_i receives a tuple of values $\langle v_1^t, \ldots, v_n^t \rangle$ at round t, where each value v_j^t is received from a process p_j ($j = 1, \ldots, n, j \neq i$) and v_i^t is a value which p_i takes. A process p_i takes one value v_i^{t+1} such that $v_i^t \rightarrow_i v_i^{t+1}$, i.e. $v_i^{t+1} = LD_i(v_1^t, \ldots, v_n^t)$ if the agreement condition $AC_i(v_1^t, \ldots, v_n^t)$ is not satisfied. The process p_i finds a value v_i^{t+1} for a tuple $\langle v_1^t, \ldots, v_i^t \ldots, v_{i+1}^t \rangle$ by the following function LD_i:

LD$_i$**(**$v_1^t, \ldots, v_n^t$**)**
{

/* forwarding */
v = **Fsrch**$_i$($v_1^t, \ldots, v_n^t$);
if $v \neq$ **NULL**, **return** (v);
else
/* backwarding */
return (**Bsrch**$_i$ ($v_1^t, \ldots, v_n^t$));

}

Fsrch$_i$**(**$v_1^t, \ldots, v_n^t$**)**

1. **if** $Next_i(v_i^t) = \emptyset$, **return** (**NULL**);
2. take a value v in $Next_i$ (v_i^t) such that $|\ \{\ v_j^t \mid v \rightarrow_{ij} v_j^t$ for every $p_j\ \}\ |$ ($\geq n/2$) is the largest;
   if v exists, **return** (v);
3. take a value v in $Next_i$ (v_i^t) such that $|\ \{\ v_j^t \mid v \rightarrow_i v_j^t$ for every $p_j\ \}\ |$ ($\neq$ **0**) is the largest;
   If v exists, **return** (v);
4. take a value v in $Next_i$ (v_i^t) such that $|\ Corn_i(v)\ |$ is the largest;
   If v exists, **return** (v).

The forwarding procedure **Fsrch**$_i$($v_1^t, \ldots, v_n^t$) takes a value v_i^{t+1} preceding the current value v_i^t ($v_i^t \rightarrow_i v_i^{t+1}$). This is a *forwarding* strategy since we are always going up to upper bounds of current values in the lattice $L_i = \langle\ D_i, \rightarrow_i, \sqcap_i, \sqcup_i\ \rangle$ of the process p_i.

If a process p_i could not find a value, i.e. **Fsrch**$_i$($v_1^t, \ldots, v_n^t$) = **NULL** in the forwarding strategy, p_i takes a *backward* strategy, i.e. backs to the previous value. Suppose a process p_i takes a value v_i^t and another process p_j takes a value v_j^t at round t. Suppose the process p_i could not find a *least upper bound*(*lub*) $v_i^t \sqcup_i v_j^t$. Here, p_i finds the *greatest lower bound*(*glb*) $v_i^t \sqcap_i v_j^t$. If a value $v = v_i^t \sqcap_i v_j^t$ is found in the domain D_i, p_i takes the value v, i.e. backwards to the value v in the lattice L_i of the process p_i [Figure 2]. Then, the forwarding strategy is adopted as follows:

Bsrch$_i$**(**$v_1^t, \ldots, v_n^t$**)**

1. **if** there is a value $v = v_1^t \sqcap_i, \ldots, \sqcap_i v_n^t$ in D_i,
   {

v = **Fsrch**$_i$($v_1^t, \ldots, v_{i-1}^t, v, v_{i+1}^t, \ldots, v_n^t$);
if $v \neq$ **NULL**, **return** (v);
}
else {
v = **Bsrch**$_i$($v_1^t, \ldots, v_{i-1}^t, v, v_{i+1}^t, \ldots, v_n^t$);
if $v \neq$ **NULL**, **return** (v);
else return ($v_1^t \sqcap_i \cdots \sqcap_i v_n^t$);
}

2. Otherwise, v = **NULL**,
{
find a value v such that $v \rightarrow_i^E v_i^t$ and $\mid \{p_j \mid v \rightarrow_{ij} v_j^t\} \mid$ is the largest;
v = **Fsrch**$_i$($v_1^t, \ldots, v_{i-1}^t, v, v_{i+1}^t, \ldots, v_n^t$);
if $v \neq$ **NULL**, **return** (v);
else {
v = **Bsrch**$_i$($v_1^t, \ldots, v_{i-1}^t, v, v_{i+1}^t, \ldots, v_n^t$);
if $v \neq$ **NULL**, **return** (v);
else return ($v_1^t \sqcap_i \cdots \sqcap_i v_n^t$);
}
}

On receipt of a value v_j^t from another process p_j, a process p_i stores a tuple $\langle v_j^{t-1}, v_j^t \rangle$ showing a precedent relation $v_j^{t-1} \rightarrow_{ij} v_j^t$ in the local database DB_i. Here, DB_{ij} shows a part of the local database DB_i where a precedent relation $\rightarrow_{ij}$ on another process p_j is stored ($j \neq i$). DB_{ii} includes the precedent relation $\rightarrow_i$ of the process p_i. $DB_i = DB_{i1} \cup \cdots \cup DB_{in}$. The size of the local database DB_i is limited. Every precedent relation $\rightarrow_{ij}$ obtained from each process p_j cannot be stored in DB_{ij}. The process p_i has to forget some tuples in DB_{ij}. We take a *least-recently-used (LRU)* replacement strategy where a tuple *least recently used* is withdrawn from DB_{ij} if DB_{ij} is full. Then, a new tuple on $\rightarrow_{ij}$ is stored.

3.2.4 Initial value

A process p_i initially takes a value v_i^0 and then sends the value v_i^0 to all the processes $p_1, \ldots, p_n$. Question is which value the process p_i initially takes in the domain D_i. Each process p_i has a value v_1 which p_i would like to take and precedes any value v_2 ($v_1 \rightarrow_i v_2$) which p_i would not like to take. The value v_1

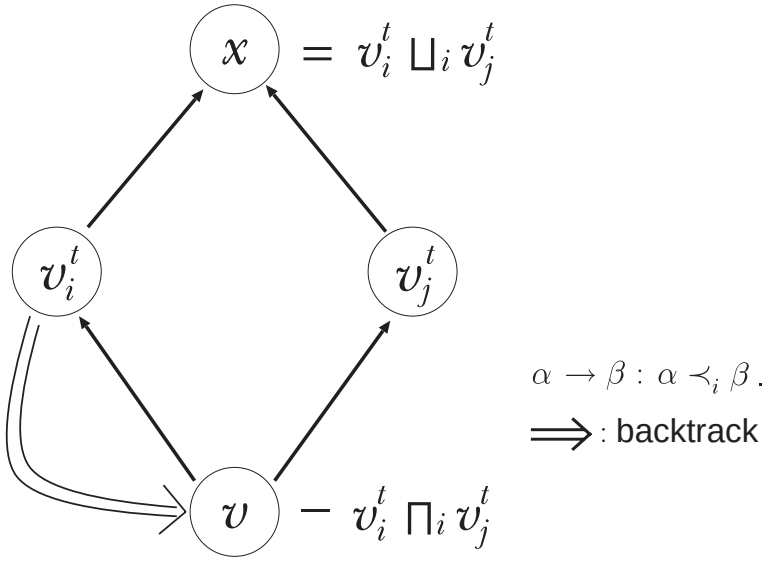


Figure 3.2: Lub and Glb..

is referred to as *maximal* target value. There is another value v_1 which the process p_i would like to take but is preceded by any value which would not like to take. The value v_1 is referred to as *minimal* target value. There are two strategies, *minimal-start* and *maximal-start* ones. In the minimal start strategy, a process p_i initially takes a minimal target value v in D_i. Then, the process p_i upgrades values. In the maximal strategy, a process p_i initially takes a maximal target value v. The process p_i insists of taking the maximal value v. This is the most aggressive strategy.

Initially, every process p_i does not know anything about the precedent relation $\rightarrow_j$ of another process p_j ($j \neq i$). In the coordination protocol, the processes exchange values with each other at each round. If a process p_i receives a value v_2 after taking another value v_1 from another process p_j, the process p_i perceives that the value v_1 precedes v_2 ($v_1 \rightarrow_j v_2$) in the process p_j. Thus, the process p_i learns the precedent relation $\rightarrow_j$ of another process p_j through communicating with the process p_j. The precedent relations of the other processes which a process p_i obtains through communication are stored in the local database DB_i of the process p_i. Let $\rightarrow_{ij}$ be a part of the precedent relation $\rightarrow_j$ which a process p_i

knows, $\rightarrow_{ij} \subseteq \rightarrow_j$. That is, if a process p_i receives a value v_2 after receiving a value v_1 from another process p_j, a precedent relation "$v_1 \rightarrow_{ij} v_2$" holds in the process p_i. The least upper bound $v_1 \sqcup_{ij} v_2$ and the greatest lower bound $v_1 \sqcap_{ij} v_2$ are defined for the precedent relation $\rightarrow_{ij}$ in the process p_i.

3.2.5 Meta coordination

Suppose a process p_i takes a value x and then y and another process p_j takes y and then x. The process p_i takes x to make an agreement with p_j by using the backward strategy. However, p_j takes y. The processes p_i and p_j can take x or y as an agreement value but connot take the same value. This is a kind of live-lock. In order to resolve the difficulties, we introduce following meta coordination commands:

1. *freeze*: a process p_i does not change the current value v_i^{t-1} at the succeeding round t.
2. *back*: a process p_i takes a previous value v_i^u which p_i has taken before.

Each process p_i takes one of the *meta* action, freeze or back. If p_i is cooperative, p_i does not change the opinion by *freeze* and wait for change of values of other peers. If p_i is selfish, p_i just takes a value without issuing *freeze* and *back*. Thus, processes are classified with respect to how each peer cares other peers.

3.2.6 Types of coordination strategies

At each round t_i, a peer p_i shows a value $v_i^{t_i}$ to other peers. Here, each local history $H_i^{t_i}$ is given a sequence $\langle v_i^0, v_i^1, \ldots, v_i^{t_i-1} \rangle$. There are the following strategies for each peer p_i to take a value $v_i^{t_i}$ at each round t_i:

1. *Forward* (f) strategy: the peer p_i takes a value $v_i^{t_i}$ which is preceded by values in the local history $H_i^{t_i} = \langle v_i^0, \ldots, v_i^{t_i-1} \rangle$.
2. *Backward* (b) strategy: the peer p_i backs to the previous branchable round u ($< t_i$) and takes a new value v from the local history $H_i^u = \langle v_i^0, \ldots, v_i^{u-1} \rangle$.
3. *Mining* (m) strategy: the peer p_i finds a recoverable cut $ct = [v_1^{u_1}, \ldots, v_n^{u_n}]$ in the local history H_i^t and proposes the other pees to make an agreement on the cut ct. If every peer agrees on the cut ct, the peer p_i takes an agreement value on the values $v_1^{u_1}, \ldots, v_n^{u_n}$ and terminates.
4. *Observation* (o) strategy: the peer p_i takes the same value $v_i^{t_i}$ as the previous one $v_i^{t_i-1}$ at round t_i.

Each peer p_i autonomously takes one of the forward, backward, mining, and observation strategies at each round t_i as shown in Figure 3.8. Each round is composed of two phases, *strategy decision* (SD) and *value exchange* (VE) phases. In the strategy decision phase, every peer makes a decision on the coordination strategy. Then, each peer exchanges values in the value exchange (VE) phase according to the strategy.

We classify peers into *aggressive*, *passive*, *cooperative*, and *fancy* types depending on which coordination strategy each peer takes at each round. An aggressive peer p_i more frequently takes the *forward* strategy. That is, the peer p_i is trying to take a new value based on the precedent relations. *Passive* and *cooperative* peers do not prefer the forward strategy. A peer carefully observes what the other peers have so far done. A peer p_i first takes the mining strategy to back to the previous round. Unless successful, the passive and cooperative peers take the forward and backward strategies, respectively. A *fancy* peer arbitrarily takes one of the strategies.

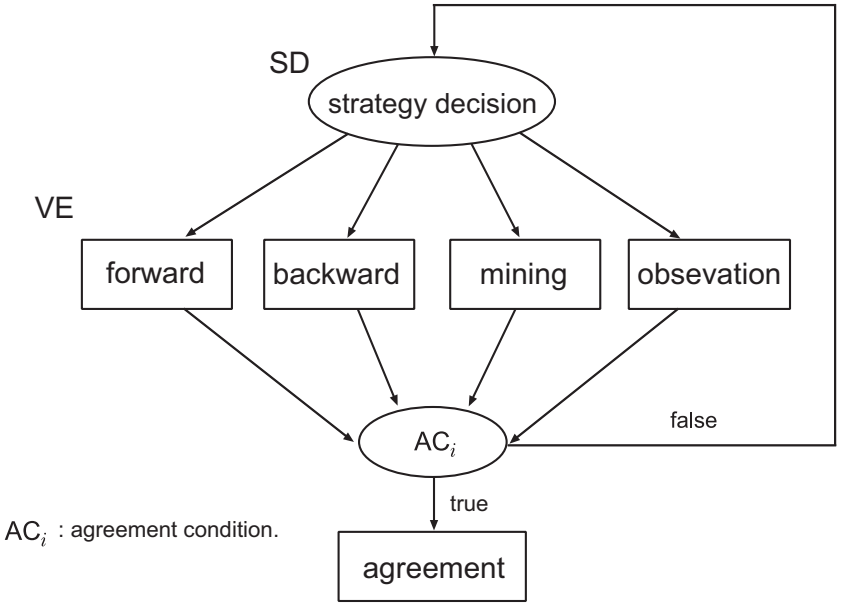


Figure 3.3: Coordination procedure of a peer.

3.2.7 Inconsistent strategies

Each peer p_i first proposes a strategy $ps_i^{t_i}$ and informs the other peers of the proposed strategy $ps_i^{t_i}$ at each round t_i with a local history $H_i^{t_i} = \langle v_i^0, \ldots, v_i^{t_i-1} \rangle$.

At each round, different peers may propose different coordination strategies. A pair of proposed strategies $ps_i^{t_i}$ and $ps_j^{t_j}$ are *consistent* with one another iff a pair of different peers p_i and p_j can take the strategies $ps_i^{t_i}$ and $ps_j^{t_j}$, respectively. For example, suppose a pair of peers p_i and p_j propose the forward strategies $\langle f, v_i \rangle$ and $\langle f, v_j \rangle$, respectively. Since a pair of the peers p_i and p_j can take the values v_i and v_j, respectively, the forward strategies $\langle f, v_i \rangle$ and $\langle f, v_j \rangle$ are consistent with one another.

Suppose a peer p_i proposes the mining strategy $\langle m, rc_i = [u_1, \ldots, u_n] \rangle$ but another peer p_j takes a different strategy, say the forward strategy $\langle f, v_i \rangle$. In order for the peer p_i to take the mining strategy, every other peer has to agree on the mining strategy. Thus, the mining strategy is inconsistent with every other strategy. Hence, every peer has to make a decision on whether or not the mining strategy is taken on receipt of the proposed mining strategy.

Next, suppose a peer p_i takes a forward strategy $\langle f, v_i \rangle$ and another peer p_j takes a backward strategy $\langle b, u_j \rangle$. If p_j compensates previous values in the local history $H_j^{t_j}$ by backing to the previous round u_j, $H_j^{t_j} = \langle v_j^0, \ldots, v_j^{u-1}, \ldots, v_j^{t_j-1} \rangle$ stored in every peer p_i is also reduced to the prefix $H_j^u = \langle v_i^0, \ldots, v_i^{u-1} \rangle$ ($u < t_j$). If the value v_i to be taken by p_i depends on a value v_j^s ($s \geq u$) to be compensated, p_i cannot take the value v_i after p_j takes the backward strategy $\langle b, u_j \rangle$. Thus, the forward strategy $\langle f, v_i \rangle$ and the backward strategy $\langle b, u_j \rangle$ are inconsistent. On the other hand, if v_i does not depend on any value to be compensated, the peers p_i and p_j can take the forward and backward strategies independently of one another. Thus, the strategies $\langle f, v_i \rangle$ and $\langle b, u_j \rangle$ is *conditionally consistent*.

Next, suppose a pair of peers p_i and p_j propose the backward strategies $\langle b, u_i \rangle$ and $\langle b, u_j \rangle$, respectively. The values $\langle v_i^{u_i}, \ldots, v_i^{t_i-1} \rangle$ and $\langle v_j^{u_j}, \ldots, v_j^{t_j-1} \rangle$ are compensated and the local histories $H_i^{u_i} = \langle v_i^0, \ldots, v_i^{u_i-1} \rangle$ and $H_j^{u_j} = \langle v_j^0, \ldots, v_j^{u_j-1} \rangle$ are then obtained in the peers p_i and p_j, respectively. If a cut $\left[u_i, u_j\right]$ is consistent, i.e. for every value v_i^s ($s \geq u_i$) in $H_i^{t_i}$, there is no value v_j in $H_j^{u_j}$ such that $v_i^s \rightarrow v_j$ and for every value v_j^s ($s \geq u_j$) in $H_j^{t_j}$, there is no value v_i in $H_i^{u_i}$ such that $v_j^s \rightarrow v_i$, the peers p_i and p_j can back to the rounds u_i and u_j, respectively. Hence, the backward strategies $\langle b, u_i \rangle$ and $\langle b, u_j \rangle$ are consistent. Otherwise, the backward strategies are inconsistent. Thus, a pair of the backward strategies are conditionally inconsistent.

Table 3.1: Consistency among strategies

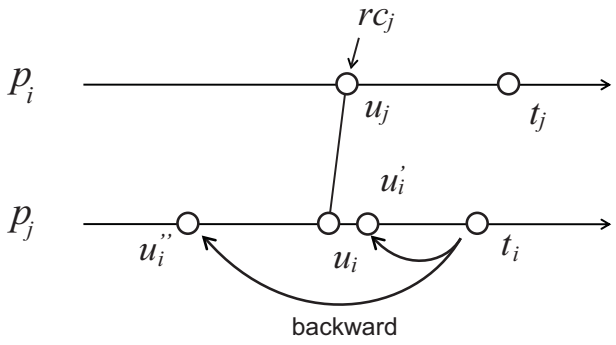


Figure 3.4: Mining and backward strategies.

Table 1 summaries the consistency among the coordination strategies. The conditional consistency among the strategies $ps_i^{t_i}$ and $ps_j^{t_j}$ means that $ps_i^{t_i}$ and $ps_j^{t_j}$ are consistent if some conditions hold. Table 2 shows the conditions. For example, a pair of peers p_i and p_j propose the applicable mining strategies $\langle m, [u_1, \ldots, u_n]\rangle$ and $\langle m, [s_1, \ldots, s_n]\rangle$, respectively. If $\big[u_1, \ldots, u_n\big]$ = $\big[s_1, \ldots, s_n\big]$, the mining strategies proposed by p_i and p_j are consistent. Next, suppose a peer p_j proposes the mining strategy $\langle m, rc_j = [u_1, \ldots, u_n]\rangle$. Suppose the peer p_i proposes a backward strategy $\langle b, u_i'\rangle$. Here, suppose $u_i' \leq u_i$ as shown in Figure 3.4. Here, even if the peer p_i backs to the round v_i' the peer p_i can take the mining strategy $\langle m, rc_j\rangle$. Next, suppose p_i proposes a backward strategy $\langle b, u_i''\rangle$ where $u_i < u_i''$. If the peer p_i back to the round u_i'', p_i cannot take the mining strategy as shown in Figure 3.4. Here, the mining strategy $\langle m, [u_1, \ldots, u_n]\rangle$ is inconsistent with the backward strategy $\langle b, u_i''\rangle$.

3.2.8 Resolution among different strategies

At each round, different peers may take different coordination strategies since the peers are autonomous. Suppose one peer p_i takes the mining (m) strategy but another peer p_j takes a different strategy from the mining one. In order for the

Table 1. Consistency among strategies.

p_j \ p_i	f	○	b	m
f	○	○	△	×
○	○	○	△	×
b	△	△	△	△
m	×	×	△	△

○ : consistent. × : inconsistent. △ : conditionally consistent.

peer p_i to take the mining strategy, every other peer agrees on the mining one. Hence, every peer has to make a decision on whether or not the mining strategy is taken. Each peer p_i takes the mining strategy only if every peer could take the mining strategy. Next, suppose a peer p_i takes a forward strategy and another peer p_j takes a backward strategy. If p_j compensates previous values in the local history $H_j^{t_j}$ by backing to the previous round u, the local history $H_j^{t_j}$ in p_i is also changed with H_j^u ($u < t_j$). Then, the peer p_i selects a new value from the local histories $H_i^{t_i}$ of p_i and H_j^u of p_j.

At the strategy decision (SD) phase, each peer p_i first takes a proposing strategy $ps_i^{t_i}$ and informs the other peers of the strategy $ps_i^{t_i}$ at each round t_i. There are the following proposing strategies:

- Forward strategy $\langle f, v \rangle$: the peer p_i takes a new value v based on the precedent relations.
- Backward strategy $\langle b, u_i, v \rangle$: p_i backs to the previous round u_i and then takes a value v based on the precedent relation.
- Mining strategy $\langle m, rc_i = [u_1, \ldots, u_n] \rangle$: p_i finds a recoverable cut $ct_i = [v_1^{u_1}, \ldots, v_n^{u_n}]$. If every peer agrees on the cut ct_i, the peer p_i takes a global agreement value on the cut ct_i and terminates.

- Observation strategy $\langle o, - \rangle$: p_i takes the same values as the previous value $v_i^{t_i - 1}$.

Every peer p_i sends a proposing strategy $ps_i^{t_i}$ and in turns receives a proposing strategy $ps_j^{t_j}$ from every other peer p_j. Then, the peer p_i selects a strategy $s_i^{t_i}$ for the proposing strategies $ps_1^{t_1}, \ldots, ps_n^{t_n}$ which p_i receives. First, suppose a peer p_i takes the forward strategy $\langle f, v_i \rangle$ at the strategy decision phase. Another peer p_j takes one of the strategies, *forward* (f), *backward* (b), *mining* (m), and *observation* (o) ones. If the peer p_j takes the forward strategy $ps_j^{t_j} = \langle f, v_j \rangle$, the peer p_i takes a new value from the local histories $H_1^{t_1}, \ldots, H_n^{t_n}$. Next, if p_j takes the backward strategy $\langle b, u_j, v_j \rangle$, the local history $H_j^{t_j}$ in p_i is compensated with $H_j^{u_j}$, i.e. values $v_j^{t_j - 1}$, $\ldots$, $v_j^{u_j}$ are compensated. The peer p_i takes a value on the local histories $H_i^{t_i}$ and H_j^u. Next, if the peer p_j takes the mining strategy $\langle m, rc_j = [u_1, \ldots, u_n] \rangle$, p_i has to make a decision on whether p_i takes the mining strategy on the cut ct_j if the cut ct_j is obtainable in p_i. If ct_j in not obtainable in p_i, p_j takes a new value in the forward strategy.

Secondly, a peer p_i takes the backward (b) strategy if there is a branchable round u. The peer p_i compensates the values to the round u and selects a value v in $P_i^u(v_i^{u-1})$. Then, p_i sends the backward strategy $\langle b, u, v \rangle$. If the peer p_i receives the forward, backward, or observation strategy from another peer p_j, the peer p_i behaves in the same way as taking the forward strategy. Suppose the peer p_i receives the mining strategy $\langle m, rc_j = [u_1, \ldots, u_n] \rangle$ from p_j. If the cut ct_j = $[v_1^{u_1}, \ldots, v_n^{u_n}]$ is not obtainable in p_i, the peer p_i does not take the mining strategy. There are two cases, $u \leq u_i$ and $u > u_i$ if the cut ct_j is obtainable in p_i. If $u \leq u_i$, the peer p_i changes the strategy with the mining strategy $\langle m, ct_j \rangle$. Otherwise, the peer p_i backs to the previous round u in the backward strategy. Here, the peer p_j has to give up taking the mining strategy.

Next, a peer p_i takes a mining strategy $\langle m, rc_i = [u_1, \ldots, u_n] \rangle$ for a cut ct_i = $[v_1^{u_1}, \ldots, v_n^{u_n}]$. Only if every other peer takes the same mining strategy $\langle m, rc_i \rangle$, the peer p_i backs to the cut ct_i and takes an agreement value. Suppose the peer p_i receives the mining strategy $\langle m, rc_j = [s_1, \ldots, s_n] \rangle$ from another peer p_j. If $rc_i \neq rc_j$ and the ct_j = $[v_1^{s_1}, \ldots, v_n^{s_n}]$ is obtainable in p_i, p_i has to decide on which cut ct_i or ct_j to take. Thus, multiple peers may find different recoverable cuts. Suppose a pair of peers p_i and p_j find different recoverable cuts ct_i and ct_j ($ct_i \neq ct_j$), respectively. The peers p_i and p_j send cut requests ct_i and ct_j to every other peer, respectively, as presented here. Now, suppose the peer p_i receives ACK from every peer and sends ACK to p_j. Here, since p_j may receive ACK from

every other peer, p_i sends a confirmation message of the cut ct_i to p_j. If p_j receives NAK from some peer, p_j sends NAK of ct_j to p_i. Here, the peer p_i sends *Agree* of the cut ct_i to every peer and every peer p_j obtains a value v from the cut ct_i. If p_j receives ACK of ct_j from every peer, p_j also sends a confirmation message of the cut ct_j to the peer p_i. Here, each of the peers p_i and p_j takes either the cut ct_i or ct_j. If ct_i is smaller than ct_j ($ct_i < ct_j$), both of the peers p_i and p_j take the smaller cut ct_i. The peer p_i sends *Agree* of ct_i to every peer. Then, every peer p_j makes an agreement on a value v for the cut ct_i [Figure 3.5].

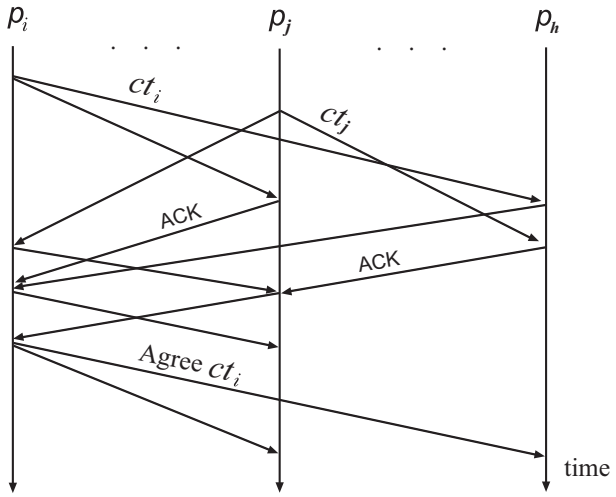


Figure 3.5: Resolution of multiple recoverable cuts.

Finally, suppose that a peer p_i takes the observation (o) strategy $\langle o, - \rangle$. If p_i receives the forward, backward, or observation strategy from another peer p_j, the peer p_i behaves in a same way as discussed here. Suppose the peer p_i receives the mining strategy $\langle m, rc_j = [u_1, \ldots, u_n] \rangle$ from another peer p_j. If the cut ct_j $[v_1^{u_1}, \ldots, v_n^{u_n}]$ is obtainable in the peer p_i, the peer p_j takes the mining strategy on the cut ct_j.

Every peer p_i proposes a coordination strategy $ps_i^{t_i}$ and exchanges the proposed strategies with the other peers. Then, the peer p_i selects a strategy $s_i^{t_i}$ for the proposed strategies $ps_1^{t_1}, \ldots, ps_n^{t_n}$ which p_i receives. The tuple $\langle ps_1^{t_i}, \ldots, ps_n^{t_i} \rangle$ of the strategies may be inconsistent. For example, if every proposed strategy $ps_i^{t_i}$ is the forward strategy $\langle f, v_i \rangle$, the strategies are consistent, i.e. every peer

$ps_i^{t_i}$	$ps_j^{t_j}$	conditions
$\langle f, u_i \rangle$ or $\langle o, v_i \rangle$	$\langle b, u_j \rangle$	1. $\langle b, u_j \rangle$ is applicable. 2. v_i does not depend on a value v_j^s ($s \geq u_j$) ($v_j^s \not\rightarrow_i v_i$).
$\langle b, u_i \rangle$	$\langle b, u_j \rangle$	1. $\langle b, u_j \rangle$ and $\langle b, u_i \rangle$ are applicable. 2. $[u_i, u_j]$ is consistent.
$\langle b, u_i \rangle$	$\langle m, rc_j = [s_1, \ldots, s_n] \rangle$	1. $\langle b, u_i \rangle$ and rc_j are applicable. 2. $s_i \leq u_i$. 3. $\langle b, u_i \rangle$ and $\langle b, s_k \rangle$ are consistent for every peer p_k.
$\langle m, rc_i = [u_1, \ldots, u_n] \rangle$	$\langle m, rc_j = [s_1, \ldots, s_n] \rangle$	1. rc_i and rc_j are applicable. 2. $u_h = s_h$ for every peer p_h.

Table 3.2: Consistency conditions.

p_i can send the value v_i to the other peers. On the other hand, if some peer p_i proposes the mining strategy $ps_i^{t_i} = \langle m, rc_i \rangle$ while the other peers propose the forward strategies, the tuple of the proposed strategies are inconsistent, i.e. no peer p_i can take the proposed strategy $ps_i^{t_i}$. If a pair of strategies $ps_i^{t_i}$ and $ps_j^{t_j}$ proposed by peers p_i and p_j, respectively, are inconsistent, either one of the strategies can be taken.

Now, suppose the proposed strategies $\langle ps_1^{t_1}, \ldots, ps_n^{t_n} \rangle$ are applicable in every peer. If the strategies are consistent, each peer p_i takes the proposed strategy $ps_i^{t_i}$. Otherwise, the peers have to resolve the inconsistency of the proposed strategies. We take the following approach toward resolving the inconsistency of the proposed strategies $\langle ps_1^{t_1}, \ldots, ps_n^{t_n} \rangle$ in the paper:

1. If a mining strategy $ps_j^{t_j} = \langle m, rc_j \rangle$ is proposed by some peer p_j, each peer p_i takes it.
2. If multiple mining strategies are proposed, each peer p_i takes one of the

mining strategies with the smallest cut. Suppose a peer p_i proposes a mining strategy $\langle m, rc_i = [s_1, \ldots, s_n]\rangle$ for a cut $ct_i = [v_1^{s_1}, \ldots, v_n^{s_n}]$ and another peer p_j proposes $\langle m, rc_j = [u_1, \ldots, u_n]\rangle$ with a cut $ct_j = [v_1^{u_1}, \ldots, v_n^{u_n}]$. If $rc_i = rc_j$, the peer p_i agrees on taking the mining strategy $\langle m, rc_j\rangle$. If $rc_i \neq rc_j$ and the cut $ct_j = [v_1^{u_1}, \ldots, v_n^{u_n}]$ is obtainable in p_i, p_i has to decide on which cut ct_i or ct_j to take. If ct_i precedes ct_j, the peers p_i and p_j take the cut ct_j. Amount of rounds to be compensated for the cut ct_j is smaller than ct_i. This means, it takes earliest to compensate the local history in every peer.

3. If a backward strategy $\langle b, u_j\rangle$ is proposed by a peer p_j, p_j backs to the round u_j. A local history $H_j^{t_j}$ is compensated in every peer. Here, let G be a global history $\langle H_1^{t_1}, \ldots, H_n^{t_n}\rangle$ obtained by compensation of p_i. Then, every peer p_i which proposes a *forward* or *backward* strategy takes a new value v_i on the global history G, so that the precedent relation $\rightarrow_i^E$ is satisfied. A peer p_i which proposes an observation strategy takes the same value $v_i^{t_i-1}$ as one taken at the round t_i - 1. Then, every peer proposes a strategy again.

3.2.9 Behaviors of peers

At each round t_i, a peer p_i applies the following functions to a global history $G = \langle H_1^{t_1}, \ldots, H_n^{t_n}\rangle$ where each local history $H_j^{t_j}$ is $\langle v_j^0, v_j^1, \ldots, v_j^{t_j-1}\rangle$ ($j = 1, \ldots, m$).

- $v = LD_i(v_1, \ldots, v_n)$: local decision function which gives such a value v that satisfies the precedent relation on the values $v_1, \ldots, v_n$ in the peer p_i.
- $v = nextLD_i(v_1, \ldots, v_n)$: a next value which satisfies the precedent relations for the values $v_1, \ldots, v_n$ is taken in p_i.
- $u_i = findBR_i(H_i^{t_i})$: a branchable round u_i is found in the local history $H_i^{t_i}$.
- $ct_i = findRC_i(H_1^{t_1}, \ldots, H_n^{t_n})$: a recoverable cut ct_i in the local histories $H_1^{t_1}, \ldots, H_n^{t_n}$ is found in p_i.
- $ct_i = nextRC_i(H_1^{t_1}, \ldots, H_n^{t_n})$: a next recoverable cut ct_i for the local histories $H_1^{t_1}, \ldots, H_n^{t_n}$ is found in p_i.
- $AC_i(v_1, \ldots, v_n) = T$ if the values $v_1, \ldots, v_n$ satisfy the agreement condition AC_i.

An aggressive peer p_i first tries to take a new value $v_i^{t_i} = LD_i(v_1^{t_1-1}, \ldots, v_n^{t_n-1})$ in the forward (f) strategy. Then, the peer p_i sends a forward strategy $\langle f, v_i^{t_i} \rangle$ to every other peer. If not found, p_i finds a branchable round $u_i = findBR_i(H_i^{t_i})$ in the local history $H_i^{t_i}$. If found, the peer p_i sends a backward strategy $\langle b, u_i \rangle$ to every other peer. If not found, p_i finds a recoverable cut $ct_i = findRC_i(H_1^{t_1}, \ldots, H_n^{t_n})$. The peer p_i sends the mining strategy $\langle m, [u_1, \ldots, u_n] \rangle$ to every other peer if a recoverable cut $ct_i = [v_1^{u_1}, \ldots, v_n^{u_n}]$ could be found. If not found, the peer p_i sends the observation strategy $\langle o, - \rangle$. Thus, an aggressive peer takes the strategies in the order $\langle f, b, m, p \rangle$. Figure 3.6 shows the strategy decision phase of an aggressive peer p_i at round t_i where $H_i^{t_i} = \langle v_i^0, \ldots, v_i^{t_i-1} \rangle$.

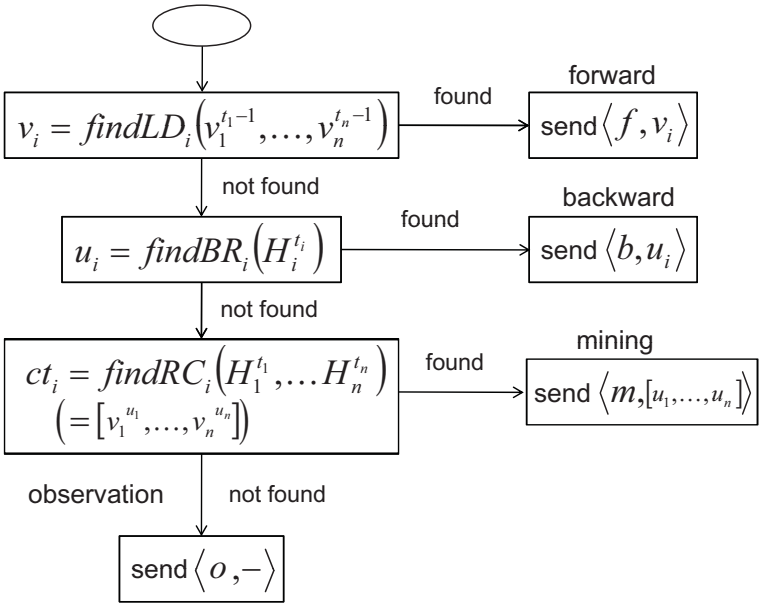


Figure 3.6: Aggressive peer.

A cooperative peer p_i first tries to find a recoverable cut $ct_i = findRC_i(H_1^{t_1}, \ldots, H_n^{t_n})$. If a recoverable cut $ct_i = [v_1^{u_1}, \ldots, v_n^{u_n}]$ is found, the peer p_i sends a mining strategy $\langle m, [u_1, \ldots, u_n] \rangle$ to every other peer. If not found , the peer p_i takes a new value $v_i^{t_i} = LD_i(v_1^{t_1-1}, \ldots, v_n^{t_n-1})$. If found, p_i sends the forward strategy $\langle f, v_i^{t_i} \rangle$. If not found, the peer p_i finds a branchable round $u_i = findBR_i(H_i^{t_i})$. If found, p_i sends the backward strategy $\langle b, u_i, v \rangle$ where $v = findLD_i(v_1^{t_1-1}, \ldots,$

$v_i^{u_i-1}$, ..., $v_n^{t_n-1}$). If not found, p_i sends the observation strategy $\langle o, - \rangle$. Thus, a cooperative peer p_i takes these strategies in the order $\langle m, f, b, p \rangle$.

A passive peer p_i takes the strategies in the order $\langle m, b, p, f \rangle$. A fancy peer takes arbitrarily a strategy.

3.3 A history of a peer

3.3.1 History

Each peer p_i takes a value while exchanging values with the other peers at each round as discussed in the basic coordination protocol. A *history* H_i^t of a peer p_i is a collection of local histories $\langle H_{i1}^t, \ldots, H_{in}^t \rangle$ at round t. A local history H_{ii}^t is a sequence $\langle v_i^0, v_i^1, \ldots, v_i^{t-1} \rangle$ of values which a peer p_i has taken until round t from the initial round 0. A local history H_{ij}^t is a sequence of values $\langle v_j^0, v_j^1, \ldots, v_j^{t-1} \rangle$ which a peer p_i has received from another peer p_j until round t (j = 1, ... n, $i \neq j$). Here, $H_{ij}^t = H_{jj}^t$ since we assume the network and every peer to be reliable and every peer surely receives every value sent in the sending order by another peer. Initially, $H_{ij}^0 = \phi$ (j = 1, ... n). A notation $H_{ij}^t|_u$ shows a value v_j^u which a peer p_i receives from a peer p_j at round u ($u \leq t$). $H_{ii}^t|_u$ shows a value v_i^u which the peer p_i takes at the round u. Suppose a peer p_i receives values *a*, *b*, *c*, *d*, and *e* at rounds 0, 1, 2, 3, and 4, respectively from another peer p_j. Here, $H_{ij}^5 = \langle a, b, c, d, e \rangle$. $H_{ij}^5|_2 = c$.

Let H be a sequence $\langle x_1, \ldots, x_m \rangle$ ($m \geq 1$) of values. Here, a value x_l is referred to as *precede* another value x_h ($x_l \Rightarrow x_h$) if $l < h$ in the sequence H. A notation "$H + x$" shows a sequence $\langle x_1, \ldots, x_m, x \rangle$ of values obtained by adding a value x to the sequence H. A subsequence $\langle x_1, \ldots, x_l \rangle$ ($l \leq m$) is a *prefix* of the sequence H. A subsequence $\langle x_k, \ldots, x_m \rangle$ ($1 < k$) is a *postfix* of the sequence H. For example, let H be a sequence $\langle a, b, c, d, e \rangle$ of values. A pair of subsequences $\langle a, b, c, d \rangle$ and $\langle c, d, e \rangle$ are a prefix and postfix of the sequence H, respectively. $H + \langle y_1, \ldots, y_l \rangle$ = ((... ((H + y_1) + y_2) ...) + y_{l-1}) + y_l = $\langle x_1, \ldots, x_m, y_1, \ldots, y_l \rangle$. "H - x_l" gives a prefix $\langle x_1, \ldots, x_{l-1} \rangle$ of a sequence $H = \langle x_1, \ldots, x_l \rangle$. H - $\langle x_l, \ldots, x_m \rangle$ = (... ((H - x_m) - x_{m-1}) ...) - x_l for a sequence $H = \langle x_1, \ldots, x_m \rangle$ and $l \leq m$. For example, H + $\langle f, a \rangle$ = $\langle$ a, b, c, d, e, f, α $\rangle$ and H - $\langle d, e \rangle$ = $\langle a, b, c \rangle$

A value x may occur multiple times in a sequence H of values. Let $H_{ij}^t[x]$ show a subsequence $\langle x, \ldots, x \rangle$ of instances of a value x in a local history H_{ij}^t. $|H_{ij}^t[x]|$ is the number of instances of a value x in a local history H_{ij}^t. For example,

let H^7_{ij} be a sequence $\langle a, b, x, c, d, x, e, f \rangle$ of values. $H^7_{ij}[x] = \langle x, x \rangle$, $H^7_{ij}[c] = \langle c \rangle$, $|H^7_{ij}[x]| = 2$, and $|H^7_{ij}[c]| = 1$.

A peer p_i takes a current value v_i^t after obtaining a tuple $\langle v_1^{t-1}, \ldots, v_n^{t-1} \rangle$ of values from the other peers p_1, ..., p_n, respectively, at round t. Let c^t_{ii} indicate the current value v_i^t. Then, the peer p_i sends the value v_i^t and receives a value v_j^t from another peer p_j. Let c^t_{ij} be a value v_j^t which the peer p_i receives from the peer p_j at round t. At round $t + 1$, the local history H^{t+1}_{ij} of a peer p_i is obtained as $H^t_{ij} + c^t_{ij}$ (= v_j^t) = $\langle v_j^0, v_j^1, \ldots, v_j^{t-1}, v_j^t \rangle$ ($j = 1, \ldots, n$).

For a pair of values v_1 and v_2 in a local history H^t_{ij}, v_1 *precedes* v_2 ($v_1 \Rightarrow_j v_2$) if a peer p_i receives the value v_2 after the value v_1 from a peer p_j ($j = 1, \ldots, n$). Suppose there are a pair of values v_1 and v_2 in a local history H^t_{ij}. If a pair of values v_1 and v_2 are in the local history H^t_{ij} and the value v_1 E-precedes the value v_2 in a peer p_j ($v_1 \rightarrow^E_j v_2$), the value v_1 precedes the value v_2 ($v_1 \Rightarrow_j v_2$) in the local history H^t_{ij}. However, even if $v_1 \rightarrow^P_j v_2$, $v_2 \Rightarrow_j v_1$ might hold in the local history H^t_{ij}. A peer p_j may take a less preferable value at some round.

3.3.2 Methods on a history

A peer p_i takes a value and receives values based on the history H_i^t at each round t. Let D_i^* show a set of possible sequences of values obtained from the domain D_i. Here, $H^t_{ii} \in D_i^*$ for every local history H^t_{ii}.

We introduce the following coordination methods on the history H_i^t [Figure 3.7]:

1. *forward*: $D_i^* \rightarrow D_i^*$. For a sequence H_1 in D_i^*, H_1 is a prefix of *forward*(H_1).
2. *compensate*: $D_i^* \rightarrow D_i^*$. For a sequence H_1 in D_i^*, *compensate*(H_1) is a prefix of H_1.
3. *null*: $D_i^* \rightarrow D_i^*$. For a sequence H_1 in D_i^*, *null*(H_1) = H_1.

Let H_1 be a sequence $\langle\ v_1, \ldots, v_m\ \rangle$ of values. *forward*(H_1) = $\langle v_1, \ldots, v_m, v_{m+1}, \ldots, v_l \rangle$. In the *forward* method, a sequence $\langle v_{m+1}, \ldots, v_l \rangle$ is added to the sequence H_1, i.e. $H_1 + \langle v_{m+1}, \ldots, v_l \rangle$. *compensate*($H_1$) gives a prefix $\langle v_1, \ldots, v_k \rangle$ ($k \leq m$) of the sequence H_1. This means, a peer p_i backs to the previous state

forward: $\langle v_1, \ldots, v_m \rangle \rightarrow \langle v_1, \ldots, v_m, v_{m+1}, \ldots, v_l \rangle$

compensate: $\langle v_1, \ldots, v_k, \ldots, v_m \rangle \rightarrow \langle v_1, \ldots, v_k \rangle$

null: $\langle v_1, \ldots, v_m \rangle \rightarrow \langle v_1, \ldots, v_m \rangle$

Figure 3.7: History methods.

with a history H_2 by withdrawing the values $v_{k+1}, \ldots, v_m$. In the *null* method, no new value is taken at round t, *null* (H_1) = H_1.

Each peer p_i takes one of the coordination methods at each round as shown in Figure 3.8. If a peer p_i takes the *forward* method, the peer p_i selects a new value v_i^t at round t as presented in the basic coordination procedure. In the *forward* method, a value v is taken and added to the history H_1, i.e. $\langle v_1, \ldots, v_m, v \rangle$. If a peer p_i takes the *null* method, the peer p_i does not select a new value at round t. If a peer p_i takes the *compensate* method, the peer p_i backs to a previous round $k+1$ where values taken from the round $k+1$ to the present round are withdrawn.

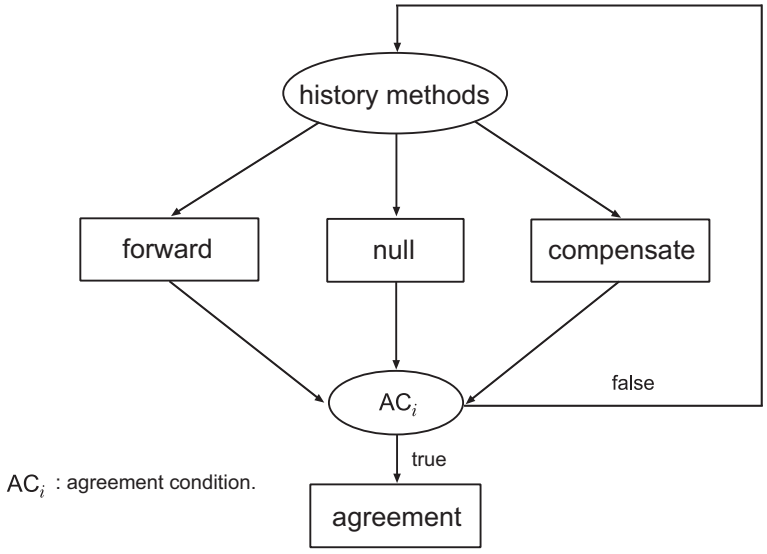


Figure 3.8: Coordination procedure of a peer.

3.3.3 Compensation

A peer p_i can back to the previous round u by compensating a history H_i^t at current round t ($u < t$). In some meeting of multiple persons, there may be some rule that each person can withdraw his remark but cannot withdraw a special remark "no". Thus, there is some value v which a person cannot withdraw after the person shows the value v to others. A value x is referred to as *primarily uncompensatable* in a peer p_i iff the peer p_i cannot withdraw the value x after showing the value x to the other peers, i.e. the value x cannot be withdrawn in the history. Otherwise, a value is *primarily compensatable* in a peer p_i. Let us consider a local history $H_{ii}^4 = \langle a, b, c, d, e\rangle$ of a peer p_i. Suppose a value c is primarily uncompensatable and the other values are primarily compensatable in the peer p_i. Here, the values d and e can be compensated but the value c cannot be compensated. Although the values a and b are primarily compensatable, neither the value a nor the value b can be withdrawn because the value c preceded by the values a and b in the local history H_{ii}^4 is primarily uncompensatable.

[Definition] In a local history $H_{ii}^t = \langle v_i^0, \ldots, v_i^{u-1}, \ldots, v_i^{t-1}\rangle$ of a peer p_i, a value v_i^{u-1} is referred to as *uncompensatable* iff the value v_i^{u-1} is primarily uncompensatable or some value v_i ($v_i^{u-1} \Rightarrow v_i$) preceded by the value v_i^{u-1} is uncompensatable. A value u_i^{u-1} is *compensatable* iff v_i^{u-1} is not uncompensatable in the local history H_{ii}^t.

A sequence $\langle v_i^0, v_i^1, \ldots, v_i^{t-1}\rangle$ is referred to as *uncompensatable* iff the value v_i^{t-1} is primarily uncompensatable or the subsequence $\langle v_i^0, \ldots, v_i^{t-2}\rangle$ is uncompensatable. A peer p_i cannot back to the previous round u at round t ($u < t$) if a value v_i^s ($u < s < t$) is uncompensatable in the local history H_{ii}^t.

In the example, the local history $H_{ii}^4 = \langle a, b, c, d, e\rangle$ is uncompensatable, because the value c is primarily uncompensatable. The subsequence $\langle d, e\rangle$ is compensatable. In a local history $H_{ii}^t = \langle v_i^0, v_i^1, \ldots, v_i^{t-1}\rangle$, an uncompensatable value v_i^u is a *most recently uncompensatable* value iff a subsequence $\langle v_i^{u+1}, \ldots, v_i^{t-1}\rangle$ is compensatable as shown in Figure 3.9. A compensatable postfix $\langle v_i^{u+1}, \ldots, v_i^{t-1}\rangle$ is referred to as *maximally compensatable* subsequence of a local history $H_{ii}^t = \langle v_i^0, v_i^1, \ldots, v_i^{t-1}\rangle$ iff a prefix $\langle v_i^0, v_i^1, \ldots, v_i^u\rangle$ is uncompensatable. In the local history $H_{ii}^4 = \langle a, b, c, d, e\rangle$, the value c is the most recently uncompensatable value. A postfix $\langle d, e\rangle$ is the maximally compensatable subsequence of H_{ii}^4. A postfix $\langle e\rangle$ is compensatable but not maximally compensatable.

At round t, a peer p_i takes a value v_i^t from the tuple $\langle v_1^{t-1}, \ldots, v_n^{t-1}\rangle$. Here,

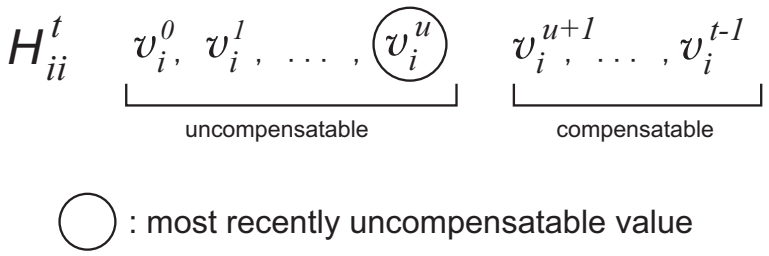


Figure 3.9: Most recently uncompensatable sequence.

$v_i^{t-1} \rightarrow_i^E v_i^t$. Suppose $v_j^{t-1} \rightarrow_i^E v_i^t$, i.e. $v_i^t = v_i^{t-1} \sqcap_i^E v_j^{t-1}$. Suppose the peer p_j withdraws the value v_j^{t-1}. If a peer p_j takes another value v ($\neq v_j^{t-1}$), the peer p_i may take a different value from the value v_i^t depending on the value v. Hence, if the peer p_j compensates the value v_j^{t-1}, the peer p_i has to compensate the value v_i^t since the peer p_i takes the value v_i^t which is obtained by applying the local decision function LD_i to the value v_j^{t-1}.

[Definition] For each value v_i^t at round t, a *minimal domain* $MD(v_i^t)$ is defined to be a subset of values in the tuple $\langle v_1^{t-1}, \ldots, v_n^{t-1} \rangle$ such that $v_i^t = \sqcap_{i\ x \in MD(v_i^t)}^E x$ and $v_i^t \neq \sqcap_{i\ x \in MD(v_i^t)-y}^E x$ for every value y in $MD(v_i^t)$.

If any value in $MD(v_i^t)$ is omitted, a least upper bound (*lub*) of values in $MD_i(v_i^t)$ is not the value v_i^t. A value v_i^t is referred to as *depend on* a value v_j^{t-1} in a peer p_i ($v_j^{t-1} \vdash v_i^t$) iff $v_j^{t-1} \in MD(v_i^t)$.

It is straightforward for the following theorem to hold from the definitions.

[Theorem 1] A value v_i^t is required to be compensated in a peer p_i if at least one value in the minimal domain $MD(v_i^t)$ is compensated.

[Theorem 2] If a value v_i^t is uncompensatable in a peer p_i, every value in the minimal domain $MD(v_i^t)$ is uncompensated.

[Proof] Let v be a value in the minimal domain $MD(v_i^t)$, $v \vdash_i v_i^t$. Suppose the value v is compensated in some peer p_j. The peer p_j might take another value v' ($\neq v$) after compensating the value v. The value v_i^t might not be the least upper bound of $MD(v_i^t)$ since the value v' is not in $MD(v_i^t)$.

3.3.4 Constraints on values

In some meeting, there is a rule on how many times each person can say a remark. For example, each person can say "no" at most once in some meeting. Thus, each value v in a domain D_i is characterized in terms of the maximum occurrence

$MO_i(v)$. The maximum occurrence $MO_i(v)$ shows how many times a peer p_i can take in an agreement procedure. If $MO_i(v)$ = 1, a peer p_i can take a value v at most once. If a peer p_i had so far taken a value v or fever times than $MO_i(v)$, i.e. $|H_{ii}^t[v]| < MO_i(v)$, the peer p_i can take a value v again at round t. If $MO_i(v) = \infty$, a peer p_i can take a value v as many times as the peer p_i would like to take.

At round t, a peer p_i takes a value v_i^t in the forward method on a history H_i^t. Here, the value v_i^t has to satisfy the following conditions:

[Conditions of possible values]

1. For every value x in the local history H_{ii}^t, $x \rightarrow_i^E v_i^t$.
2. $|H_{ii}^t[v_i^t]| < MO_i(v_i^t)$.

Let $P_i^t(v_i^{t-1})$ show a set of possible values which a peer p_i can take at round t. The set $P_i^t(v_i^{t-1})$ is defined from the conditions as follows:

$P_i^t(v_i^{t-1})$ = { v | $|H_{ii}^t[v]| < MO_i(v)$ and for every value x in H_{ii}^t, $v_i^{t-1} \rightarrow_i^E v$ }.

A value v is not included in the possible value set $P_i^t(v_i^{t-1})$ if $|H_{ii}^t\ [v]| = MO_i(v)$. Then, the peer p_i takes a value v_i^t in the set $P_i^t(v_i^{t-1})$ if $P_i^t(v_i^{t-1}) \neq \phi$. Then, the peer p_i sends the value v_i^t to the other peers. If $P_i^t(v_i^{t-1})$ = ϕ, there is no value which the peer p_i can take at round t in the forward method after taking a value v_i^{t-1} at round t - 1. Here the peer p_i has to take another coordination method, *null* or *compensate*. If $|P_i^t(v_i^{t-1})| \geq 2$, the value v_i^{t-1} is referred to as *branchable* at round t. Even if a value v_i^{t-1} is branchable in the domain D_i, i.e. $|Corn_i(v_i^{t-1})| \geq 2$, the value v_i^{t-1} may be taken in previous rounds of the local history H_{ii}^t.

3.4 Back-warding strategies

3.4.1 Cuts

Let δ_i show the current round of a peer p_i. A peer p_i has a local history $H_{ii}^{\delta_i} = \langle v_i^0, v_i^1, \ldots, v_i^{\delta_i - 1} \rangle$ at round δ_i. Since a peer p_i may compensate a local history $H_{ii}^{\delta_i}$, "$\delta_i = \delta_j$" does not always hold for every pair of peers p_i and p_j. A history $H_i^{\delta_i}$ is a tuple $\langle H_{i1}^{\delta_1}, \ldots, H_{ii}^{\delta_i}, \ldots, H_{in}^{\delta_n} \rangle$ of local histories.

[Definition] A *cut* of a history $H_i^{\delta_i}$ is a tuple of values $\langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ where $t_j < \delta_j$ for each $j = 1, \ldots, n$.

A cut $\langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ is referred to as *current* iff $t_j = \delta_j$ for every peer p_j. A cut $\langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ is referred to as *concurrent* iff each value $v_i^{t_i}$ is taken at the same round. A current cut is concurrent.

[Definition] A cut $\langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ of a history $H_i^{\delta_i}$ is *satisfiable* in a peer p_i iff the cut $\langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ satisfies the agreement condition AC_i.

Since every peer p_i is assumed to have the same agreement condition $AC_i = AC$ in this paper, a satisfiable cut in some peer is also satisfiable in every peer.

In the coordination protocol presented in the preceding section, a peer p_i takes a *forward* function, i.e. takes a new value. However, even if the current cut is not satisfiable, there might be a satisfiable cut in a history $H_i^{\delta_i}$. Suppose the current cut $\langle v_1^{\delta_1}, \ldots, v_n^{\delta_n} \rangle$ is not satisfiable but another cut $\langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ is satisfiable in a history $H_i^{\delta_i}$. Here, if every peer p_i backs to the previous round $t_i + 1$ by compensating the local history $H_{ii}^{\delta_i}$ ($i = 1, \ldots, n$), all the peers $p_1, \ldots, p_n$ can make an agreement on a value $v = GD_i(v_1^{t_1}, \ldots, v_n^{t_n})$ for the cut $\langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$.

[Definition] Let $H_{ii}^{\delta_i}$ be a local history $\langle v_i^0, v_i^1, \ldots, v_i^{\delta_i - 1} \rangle$ of each peer p_i ($i = 1, \ldots, n$). A cut $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ is *obtainable* in a peer p_i iff a postfix $\langle v_i^{t_i+1}, \ldots, v_i^{\delta_i - 1} \rangle$ of the local history $H_{ii}^{\delta_i}$ is compensatable in the peer p_i.

A peer p_i can back to the previous value $v_i^{t_i}$ if there is an obtainable cut $\langle v_1^{t_1}, \ldots, v_i^{t_1}, \ldots, v_n^{t_1} \rangle$. A cut $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ is *obtainable* iff the cut ct is obtainable in every peer. A cut $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ is *maximally obtainable* in a peer p_i iff the cut ct is obtainable in the peer p_i^t, i.e. a postfix $\langle\ v_i^{t_i+1}, \ldots, v_i^{\delta_i - 1}\ \rangle$ is compensatable, but a postfix $\langle\ v_i^{t_i}, v_i^{t_i+1}, \ldots, v_i^{\delta_i - 1}\ \rangle$ of the local history $H_{ii}^{\delta_i}$ is not compensatable. A cut $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ is *maximally obtainable* iff the cut ct is maximally obtainable in every peer p_i.

Let ct be a cut $\langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ in a history $H_i^{\delta_i}$. The cut ct is obtainable if the following conditions hold:

[Obtainability conditions]

1. Let mru_j be the most recently uncompensatable value in a local history $H_{ij}^{\delta_j}$. A value $v_j^{t_j}$ in the cut ct precedes the value mru_j in $H_{ij}^{\delta_j}$ ($mru_j \Rightarrow v_j^{t_j}$).

2. Let v_k mean a value from a peer p_k in the minimal domain $MD_j(v_j^{t_j})$, i.e. $v_j^{t_j}$ depends on v_k ($v_k \vdash_j v_j^{t_j}$). From each value $v_j^{t_j}$ in the cut ct, every value v_k in $MD_j(v_j^{t_j})$ precedes a value $v_k^{t_k}$ in the local history $H_{ij}^{\delta_j}$ ($v_k \Rightarrow v_j^{t_j}$) as shown in Figure 3.10.

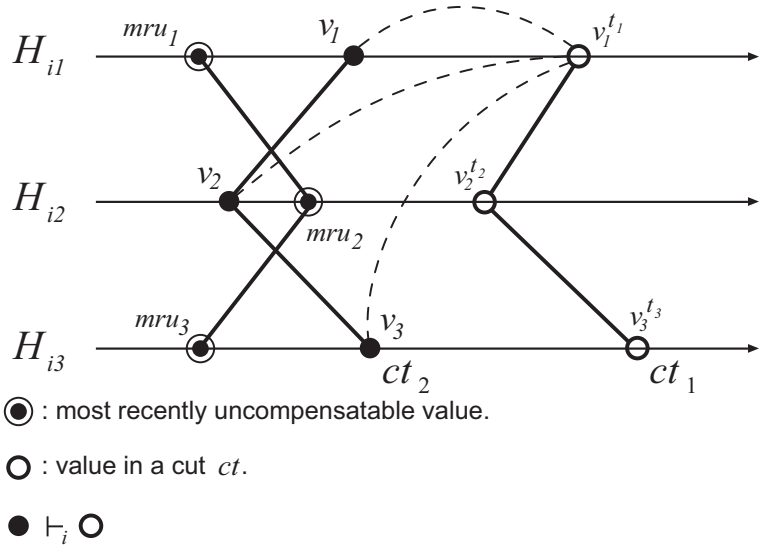


Figure 3.10: Obtainable cut.

Even if a cut $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ satisfies the agreement condition AC_i, the previous value $v_i^{t_i}$ may not be obtainable in some peer p_i. We have to find a cut which is not only satisfiable but also obtainable in each peer p_i.

[Definition] A $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ is referred to as *recoverable* in a history $H_i^{\delta_i}$ iff the cut ct is satisfiable and obtainable in a peer p_i. A cut ct is *recoverable* iff the cut ct is recoverable in every peer p_i.

[Theorem] If there is a recoverable cut $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ in a history $H_i^{\delta_i}$, every peer p_i can make an agreement on the cut ct by backing to the previous round t_i + 1.

[Proof] The cut $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ satisfies the agreement condition from the assumption. Each peer p_i can back to the previous round t_i + 1 since the cut ct is obtainable in the peer p_i.

In a history $H_i^{\delta_i}$, there might be multiple recoverable cuts $ct_1, \ldots, ct_m$ ($m >$ 1). Every peer p_i has to take the same cut ct_l out of the possible cuts $ct_1, \ldots, ct_n$. Let ct_1 and ct_2 be a pair of recoverable cuts $\langle v_{11}^{t_{11}}, \ldots, v_{1n}^{t_{1n}} \rangle$ and $\langle c_{21}^{t_{21}}, \ldots, c_{2n}^{t_{2n}} \rangle$ of a history $H_i^{\delta_i}$, respectively. First, the cut ct_1 is referred to as *precede* the other cut ct_2 in the history $H_i^{\delta_i}$ ($ct_1 \rightarrow ct_2$) if $t_{1j} \leq t_{2j}$ for every peer p_j (= 1, ..., n). Otherwise, a pair of the cuts ct_1 and ct_2 are referred to as *intersect*. Figure 3.11 shows a history $H_i^{\delta_i}$ and three cuts ct_1, ct_2 and ct_3. Here, the cut ct_1 precedes the

other cut ct_2 ($ct_1 \rightarrow ct_2$). The cuts ct_1 and ct_2 intersect and the cuts ct_2 and ct_3 also intersect. Suppose there are a pair of recoverable cuts ct_1 and ct_2 in a history $H_i^{\delta_i}$. Here, each peer p_i has to make a decision on which cut ct_1 or ct_2 to be taken. In this paper, each peer p_i takes the cut ct_1 if the cut ct_2 precedes the cut ct_1 in the history $H_i^{\delta_i}$ ($ct_2 \rightarrow ct_1$). Next, suppose a pair of the cuts ct_1 and ct_2 intersect in the history $H_i^{\delta_i}$. Here, we introduce the weight $|ct|$ for a cut $ct = \langle v_1^{t_1}, \ldots, v_n^{t_n} \rangle$ in a history $\langle H_{i1}^{\delta_1}, \ldots, H_{in}^{\delta_n} \rangle$ as $|ct| = \sum_{j=1,\ldots,n} (\delta_j - t_j)$. A cut ct_1 is referred to as *smaller* than another cut ct_2 ($ct_1 < ct_2$) if $|ct_1| < |ct_2|$. The cut ct_1 is taken if the cuts ct_1 and ct_2 intersect and the ct_1 is smaller than the cut ct_2 ($|ct_1| < |ct_2|$). Let CT be a set of recoverable cuts in a history $H_i^{\delta_i}$. A cut ct is referred to as *maximal* in the history $H_i^{\delta_i}$ iff there is no cut ct' in the history $H_i^{\delta_i}$ where ct precedes the cut ct' ($ct \rightarrow ct'$). Each peer p_i selects a cut ct in the set CT as follows:

[Select (CT)]

1. Let MCT be a set of maximal cuts in the set CT with respect to the precedent relation $\rightarrow$.

2. A smallest cut ct is selected in the minimal cut set MCT.

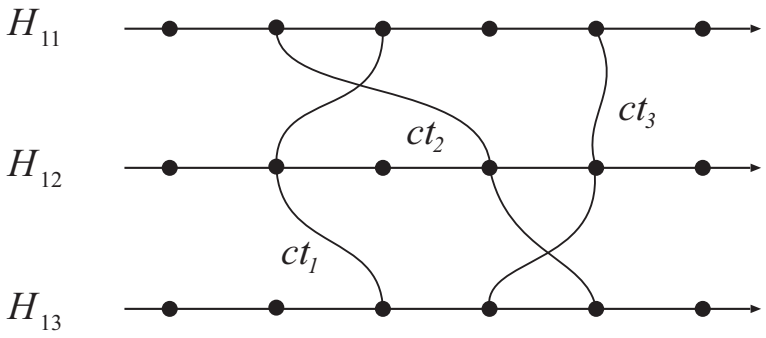


Figure 3.11: Cuts.

This selection rule means each peer takes a more recent cut to reduce the number of values which to be withdrawn.

The cuts can be also ordered in the preference of each peer. A cut $ct_1 = \langle v_{11}^{t_{11}}, \ldots, v_{1n}^{t_{1n}} \rangle$ is referred to as *more preferable* than another cut $ct_2 = \langle v_{21}^{t_{21}}, \ldots, v_{2n}^{t_{2n}} \rangle$ ($ct_1 \succeq ct_2$) iff $v_{2j}^{t_{2j}} \preceq_j^P v_{1j}^{t_{1j}}$ or $v_{2j}^{t_{2j}} \mid_j^P v_{1j}^{t_{1j}}$ for every peer p_j ($j = 1, \ldots, n$). The

cuts ct_1 and ct_2 are *preferentially independent* ($ct_1 \mid ct_2$) iff neither $ct_1 \preceq ct_2$ nor $ct_1 \succeq ct_2$. A cut ct_1 is referred to as *preferentially superior* to another cut ct_2 ($ct_1 \Rightarrow ct_2$) iff $ct_1 \mid ct_2$ and $|\{v_{1j} \mid v_{2j}^{t_{2j}} \preceq_j^P v_{1j}^{t_{1j}}\}| \geq |\{v_{2k} \mid v_{1k}^{t_{2k}} \preceq_k^P v_{2k}^{t_{1k}}\}|$. The cut ct_1 includes more preferable values than the other cut ct_2. A cut ct is taken by every peer in the selection rule Mselection(CT):

[Selection rules: Mselect(CT)]

1. Let MPC be a set of cuts which are maximally preferable in the cut set CT.

2. If $MPC \neq \phi$, one cut ct is selected in the set MPC, i.e. where ct is the smallest in the set MPC.

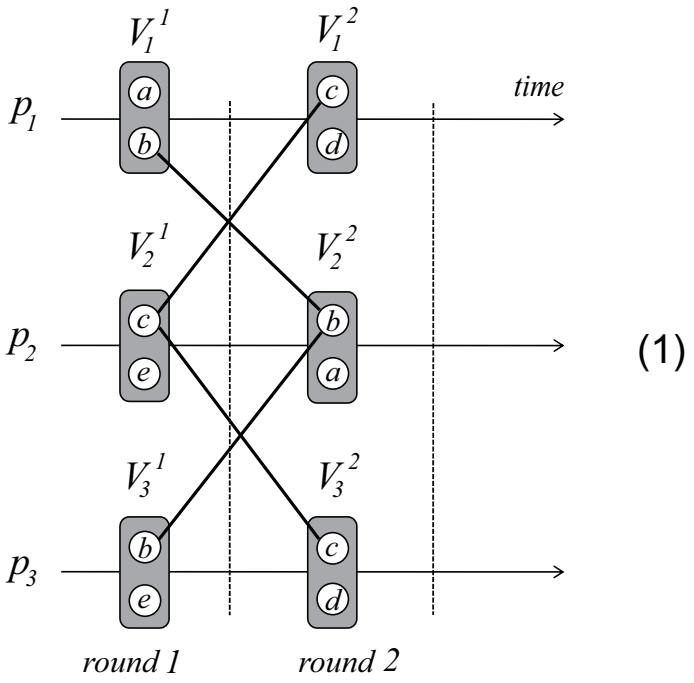
3. If $MPC = \phi$, ct = Select(CT).

A local history $H_j^{t_j}$ of a peer p_j at round t_j is a sequence $\langle v_j^0, v_j^1, \ldots, v_j^{t_j-1} \rangle$ of values which each peer p_j takes until round t_j ($j = 1, \ldots, n$). v_j^s *precedes* v_j^u iff $s < u$. The value $v_j^{t_j-1}$ is *current* and the other values are *previous* in $H_j^{t_j}$. $\langle v_j^0, \ldots, v_j^u \rangle$ and $\langle v_j^u, \ldots, v_j^{t_j-1} \rangle$ ($0 \leq u \leq t_j - 1$) are prefix and postfix of $H_j^{t_j}$, respectively. Suppose a peer p_i receives values *a*, *b*, *c*, *d*, and *e* from another peer p_j. Here, $H_j^5 = \langle a, b, c, a, d, e \rangle$. $\langle a, b, c \rangle$ is a prefix and $\langle d, e \rangle$ is a postfix of H_j^5.

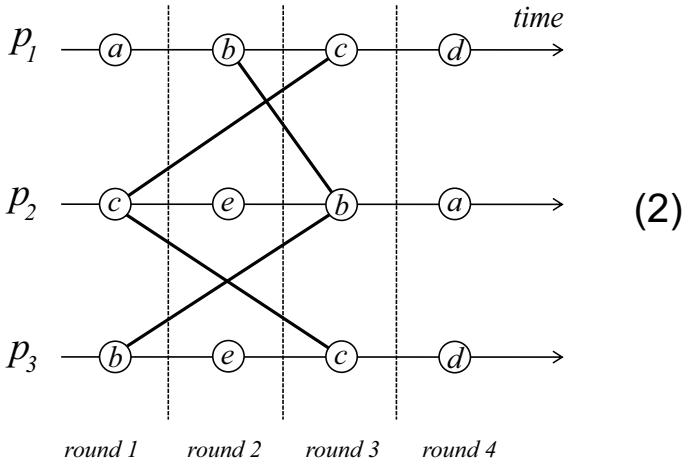
For each value v_j^u in the history $H_j^{t_j}$, let $V_j^u(v_j^u)$ show a *package* of v_j^u. That means, a peer p_j sends the package $V_j^u(v_j^u)$ and takes the primary value v_j^u at round u.

A *global history* G is a collection of local histories $\langle H_1^{t_1}, \ldots, H_n^{t_n} \rangle$. In a global history $G = \langle H_1^{t_1}, \ldots, H_n^{t_n} \rangle$ where $H_i^{t_i} = \langle v_i^0, \ldots, v_i^{t_i-1} \rangle$ ($i = 1, \ldots, n$), a tuple $[x_1^{u_1}, \ldots, x_n^{u_n}]$ of values is referred to as *cut*, where $u_k \leq t_k$ and each value $x_k^{u_k}$ is in a package $V_k^{u_k}$ of a local history $H_k^{t_k}$ for $k = 1, \ldots, n$. A cut $[x_1^{u_1}, \ldots, x_n^{u_n}]$ is *satisfiable* if the values $x_1^{u_1}, \ldots, x_n^{v_n}$ satisfy the agreement condition AC.

Let $cu = [x_1^{u_1}, \ldots, x_n^{u_n}]$ and $cs = [y_1^{s_1}, \ldots, y_n^{s_n}]$ be a pair of satisfiable cuts in a global history $G = \langle H_1^{t_1}, \ldots, H_n^{t_n} \rangle$ where $v_i \leq t_i$ and $s_i \leq t_i$ for $i = 1, \ldots, n$. The size $|cu|$ of the cut cu is given as $\sum_{i=1}^{n}(t_i - u_i)$. A cut cu is *smaller* than another cut cs ($cu < cs$) iff $|cu| < |cs|$. The cut cu *precedes* another cut ct iff $u_i \leq s_i$ for $i = 1, \ldots, n$. A cut $cu = [y_1^{u_1}, \ldots, y_n^{u_n}]$ is *maximally satisfiable* iff ct is satisfiable and there is no satisfiable cut ct' which precedes ct in a global history G. A cut cu is maximally satisfiable iff cu is satisfiable and there is no satisfiable cut cu' which is smaller than cu.



(1)



(2)

Figure 3.12: Multiple cuts.

Figure 3.12 (1) shows the history of three peers p_1, p_2, and p_3 after exchanging

packages with each other. At each round k, each peer p_i sends a package V_i^k which including a pair of values. For example, the peer p_1 sends a package $V_1^1 = \langle a, b \rangle$ where a value a is primary and b is secondary. In Figure 3.12 (2), each peer takes the single-value exchange scheme to send the same values as Figure 3.12 (1). In this example, each peer has five different values in the value domain $D = \langle a, b, c, d, e \rangle$. The E-precedent relations between values in each peer p_i as follows:

p_1: $a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$
p_2: $c \rightarrow e \rightarrow b \rightarrow a \rightarrow d$
p_3: $b \rightarrow e \rightarrow c \rightarrow d \rightarrow a$

In the multi-value exchange scheme, according to the precedent relation between values, the peers p_1, p_2 and p_3 send packages $V_1^1 = \langle a, b \rangle$, $V_2^1 = \langle c, e \rangle$, and $V_3^1 = \langle b, e \rangle$ to the other peers at round 1, respectively. On the other hand, in the single-value exchange scheme, the peers p_1, p_2 and p_3 send values a, c, and b to each other at round 1, respectively. In the multi-value exchange scheme, as shown in the Figure 3.12 (1) after two rounds, the peers detect two satisfiable cuts ct_1 = [$c, c, c,$] and ct_2 = [b, b, b] in the history, respectively. Therefore, the peers makes agreement on the cut ct_2, because ct_2 is the smallest cut among satisfiable cuts. In the single-value exchange scheme, as shown in the Figure 3.12 (2), it takes three rounds to detect the same satisfiable cuts. Following the example, it is obvious that, by using the multi-value exchange scheme, we can significantly reduce the overall time consumption.

In this paper, we consider the binary package V_i^i only contains two values. After evaluating the scheme, we would like to extend it to a multi-ary package which can include more than two values.

[Definition] A cut $\left[v_1^{u_1-1}, \ldots, v_n^{u_n-1}\right]$ is *consistent* in a global history $G = \langle H_1^{t_1}, \ldots, H_n^{t_n} \rangle$ iff there is no value v_j in a package $V_j^{sj}(v_j^{sj})$ ($s_j \leq u_j - 1$) such that $v_i^{r_i} \rightarrow_i v_j$ and $u_i - 1 \leq r_i$.

3.4.2 Re-selectable values

Suppose a peer p_i takes a maximal value v_i^t in the domain D_i in the forward method at round t. At round t + 1, the peer p_i cannot take another value since the value v_i^t is maximal in the domain D_i. Here, the peer p_i has to go back to the previous round u by the compensation method and takes another value. We discuss to which previous round the peer p_i can compensate the history H_i^t at round t.

Suppose a peer p_i takes values $v_i^0, v_i^1, \ldots, v_i^u, \ldots, v_i^{\delta_i - 1}$ in the local history $H_{ii}^{\delta_i}$. A peer p_i takes a value v_i^{u+1} after taking a value v_i^u. If there is only one value v_i^{u+1} which follows the value v_i^u, i.e. $v_i^u \rightarrow_i^E v_i^{u+1}$, the peer p_i cannot take another value different from the value v_i^{u+1} at round u. Hence, it is meaningless to compensate a subsequence $\langle v_i^{u+1}, \ldots, v_i^{\delta_i - 1} \rangle$ in the local history $H_{ii}^{\delta_i}$, i.e. goes back to the previous round u + 1. On the other hand, suppose there are multiple values $v_1, \ldots, v_m$ ($m \geq 2$) which the value v_i^u precedes, i.e. $v_i^u \rightarrow_i^E v_l$ (l = 1, $\ldots, m$). Suppose the peer p_i takes a value v_l as v_i^{u+1} in the values $v_1, \ldots, v_m$. If the postfix $\langle v_i^{u+1}, \ldots, v_i^{\delta_i - 1} \rangle$ in the local history $H_{ii}^{\delta_i}$ is compensated, the peer p_i takes another value v_k ($\neq v_l$) where $v_i^u \rightarrow_i^E v_k$ by backing to the previous round u + 1.

Each time the peer p_i backs to the round u + 1, the peer p_i has to take a value in $Corn_i(v_i^u)$ which has not been so far taken. For each branchable value v_i^u, let $Used_i(v_i^u)$ indicate a set of values in $Corn_i(v_i^u)$ which the peer p_i has taken until round u + 1. If a value v_i^u is first taken at round u + 1, $Used_i(v_i^u) = \phi$. The peer p_i takes the forward method and eventually backs to the round t + 1. Then, a value v in $Corn_i(v_i^u)$ is taken. Here, $Used_i(v_i^u) = \{v\}$. Suppose the peer p_i backs to the previous round u + 1. Here, the peer p_i takes a value v in $Corn_i(v_i^u)$ but not in $Used_i(v_i^u)$ which satisfies the possible value condition, i.e. $v \in Corn_i(v_i^u)$ - $Used_i(v_i^u)$. The set $P_i^{u+1}(v_i^u)$ defined in the previous subsection is redefined as follows:

$$P_i^{u+1}(v_i^u) = \{\ v \mid v \in Corn_i(v_i^u) \text{ - } Used_i(v_i^u) \text{ and } |H_{ii}^{u+1}[v]| < MO_i(v)\ \}.$$

Then, the value v is added to the set $Used_i(v_i^u)$. A value v_i^u is referred to as *branchable* in the history $H_i^{\delta_i}$ iff $P_i^{u+1}(v_i^u) \neq \phi$.

[Definition] Let $H_{ii}^{\delta_i}$ be a local history $\langle v_i^0, v_i^1, \ldots, v_i^u, \ldots, v_i^{\delta_i - 1} \rangle$ of values which a peer p_i has taken until round δ_i. A value v_i^u is referred to as *reselectable* in the history H_{ii}^{δ} iff v_i^u is branchable in H_{ii}^{δ} and a subsequence $\langle v_i^{u+1}, \ldots, v_i^{\delta_i - 1} \rangle$ is compensatable.

By compensation, a peer p_i can back to a re-selectable value v_i^u taken at the previous round u + 1. Then, the peer p_i takes a new value in the possible value set $P_i^{u+1}(v_i^u) = Corn_i(v_i^u)$ - $Used_i(v_i^u)$.

Chapter 4

Distributed Agreement Protocols

4.1 Value exchange schemes

4.1.1 Single value exchange scheme

A *group* is composed of reliable peers interconnected in a reliable network. A *domain* D_i of a peer p_i is a set of possible values which p_i can take. In this paper, we assume every peer p_i has the same domain D_i (= D). Each peer p_i takes a value $v_i^{t_i-1}$ in D_i and sends $v_i^{t_i-1}$ to the other peers $p_1, \ldots, p_n$ at each round t_i. Unless the tuple $\langle v_1^{t_1-1}, \ldots, v_n^{t_n-1} \rangle$ satisfies the agreement condition AC; *all*, *majority*, *weighted majority*, *some*, and *consonance* ones [43, 44], a peer p_i sends another value $v_i^{t_i}$. Until AC is satisfied, this procedure is iterated.

A value should be more expensive than the previous values in auction systems. Thus, some values can be taken but the other values cannot be taken after a value is taken. A value v_1 *existentially* (*E*-) *precedes* another value v_2 in a peer p_i ($v_1 \rightarrow_i^E v_2$) if and only if (*iff*) p_i is allowed to take v_1 after v_2. We assume the precedent relation $\rightarrow_i^E$ is transitive. v_1 and v_2 are *E- incomparable* in p_i ($v_1 |_i^E v_2$) iff neither $v_1 \rightarrow_i^E v_2$ nor $v_2 \rightarrow_i^E v_1$. The *preferentially* (*P*-) *precedent* relation $\rightarrow_i^P$ [42, 43, 44] is also defined. In this paper, we consider only the E-precedent relation $\rightarrow_i^E$ for simplicity.

A value v_1 is *maximal* and *minimal* with respect to the relation $\rightarrow_i^E$ iff there is no value v_2 such that $v_1 \rightarrow_i^E v_2$ and $v_2 \rightarrow_i^E v_1$ in D_i. A value v_1 is *top* and *bottom* with respect to the relation $\rightarrow_i^E$ iff $v_2 \rightarrow_i^E v_1$ and $v_1 \rightarrow_i^E v_2$ for every value v_2 in D_i. Let $Corn_i(x)$ be a set of values $\{ y \mid x \rightarrow_i^E y \}$ which p_i can take after a value x in D_i. A *least upper bound* (*lub*) of values v_1 and v_2 ($v_1 \sqcup_i^E v_2$) is a value v_3 in D_i such that $v_1 \rightarrow_i^E v_3$, $v_2 \rightarrow_i^E v_3$, and there is no value v_4 such that

$v_1 \rightarrow_i^E v_4 \rightarrow_i^E v_3$ and $v_2 \rightarrow_i^E v_4 \rightarrow_i^E v_3$ in a peer p_i. For example, a pair of peers p_i and p_j take v_i and v_j at round t, respectively. A *greatest lower bound* (*glb*) of v_1 and v_2 ($v_1 \sqcap_i^E v_2$) is similarly defined.

At round t_i, a peer p_i takes a value $v_i^{t_i}$ from the tuple $\langle v_1^{t_i-1}, \ldots, v_n^{t_i-1} \rangle$ where $v_i^t = v_i^{t-1} \sqcap_i^E v_j^{t-1}$ for some peer p_j. Here, suppose the peer p_j compensates the value v_j^{t-1}. If p_j takes another value v ($\neq v_j^{t_i-1}$), p_i may take a different value from v_i^t depending on the value v. Hence, the peer p_i has to compensate the value v_i^t since p_i takes v_i^t from the value $v_j^{t_i-1}$ by using the precedent relations. For a value $v_i^{t_i}$ at each round t_i, a *minimal dominant domain* $MD(v_i^{t_i})$ is a subset of values in the tuple $\langle v_1^{t_1-1}, \ldots, v_n^{t_n-1} \rangle$ such that $v_i^{t_i} = \sqcap_{i\ x \in MD(v_i^{t_i})}^E x$ and $v_i^t \neq \sqcap_{i\ x \in MD(v_i^{t_i})-y}^E x$ for every value y in $MD(v_i^{t_i})$.

[Definition] A value $v_i^{t_i}$ *depends on* a value $v_j^{t_i-1}$ in a peer p_i ($v_j^{t_i-1} \Rightarrow_i v_i^{t_i}$) iff $v_j^{t_j-1} \in MD(v_i^{t_i})$.

4.1.2 Multi-value exchange (MVE) scheme

In an agreement protocol, each peer sends one value to the other peers at each round [42, 43, 44, 53]. To more efficiently make an agreement among peers, we newly consider a *multi-value exchange* scheme. Here, at each round where peers exchange the proposing values with each other, each peer p_i sends a package of values to the other peers. In the package, not only a proposing value but also additional candidate values are included. In previous works [42, 43, 44], we mainly discuss the *single-value exchange* schemes where each peer sends only one value to the other peers at each round. By using the multi-value exchange scheme, we can more efficiently detect a value which satisfies the agreement conditions. We can significantly reduce the overall time overhead of the agreement procedure.

In the multi-value exchange scheme, each peer p_i sends a set V_i^t of values to the other peers at round t. The set V_i^t is referred to as *package* of values. The values in the package V_i^t is totally ordered in the preference as $\langle v_i^{t1}, \ldots, v_i^{tm_i} \rangle$ ($m_i \geq 1$) where v_i^{ti} is referred to as primary, i.e. most preferable value and v_i^{tk} is the k^{th} preferable value. For every value v_i^{tk} in the package V_i^t, $v_i^{t-1} \rightarrow_i^E v_i^{tk}$.

At each round t, a peer p_i receives the packages $V_1^t, \ldots, V_n^t$ from the peers $p_1, \ldots, p_n$ as shown in Figure 4.1. Here, if there is a tuple $\langle v_1, \ldots, v_n \rangle \in \langle V_1^t \times \cdots \times V_n^t \rangle$ of values which satisfy the agreement condition AC, every peer p_i makes an agreement on the tuple $\langle v_1, \ldots, v_n \rangle$ and then take an agreement value. There may be multiple tuples in $V_1^t \times \cdots \times V_n^t$ which satisfy agreement condition AC. Here, let $ord(v_j)$ denote the preference order of a value v_j in a package V_j^t.

For example, $ord(v_j^{tk})$ is k in a package $V_i^t = \langle v_j^{t1}, \ldots, v_j^{tm_j} \rangle$. Let $\langle x_1, \ldots, x_n \rangle$ and $\langle y_1, \ldots, y_n \rangle$ be a pair of tuples in the direct product $V_1^t \times \cdots \times V_n^t$. Here, $\langle x_1, \ldots, x_n \rangle$ is more preferable to $\langle y_1, \ldots, y_n \rangle$ if $\sum_{i=1}^{n} ord(x_i) < \sum_{j=1}^{n} ord(y_j)$. Each peer p_i takes the most preferable tuple which satisfies the agreement condition AC.

If there is no tuple satisfying the agreement condition AC, each peer p_i finds values which is E-preceded by the primary value v_i^{t1} in the package V_i^t. At round t + 1, each peer p_i sends package V_i^{t+1} where every value is E-preceded by the value v_i^{t1}. In this paper, we assume each package V_i^t can include at most two values, primary value $v_{i\alpha}^t$ and secondary value $v_{i\beta}^t$ for simplicity.

The application layer of each individual peer makes a decision on what value the peer can take at the next round. In addition, the agreement condition of the group is decided according to the purpose of the group, like majority decision and so on. If the peer could not change the primary value after the current round, for example, the peer p_i takes the primary value $v_{i\alpha}$ and sends the value package $\langle v_{i\alpha}, v_{i\alpha} \rangle$ to the other peers. By analyzing the value package which receives from each other, it is not difficult for each peer p_j to find that, the peer p_i will not change its primary value $v_{i\alpha}$ from now on. In traditional single-value exchange schemes, it takes one more round to find out that, the individual peer has reached the final decision value.

Let us consider a group G of multiple peers $p_1, \ldots, p_n$ ($n > 1$). The domain D_i is a set of possible values which a peer p_i can take. In this paper, we assume every domain D_i is the same D ($i = 1, \ldots, n$). First, each peer p_i shows a value v_1 in D to the other peers. If the peers do not make an agreement on the values, each peer p_i takes another value v_2 in D. Here, ther are values which p_i can take. A value v_1 *existentially* (*E*-) *precedes* another value v_2 in a peer p_i ($v_1 \rightarrow_i^E v_2$) if and only if (*iff*) p_i is allowed to take v_1 after v_2 [42, 43, 44]. v_1 and v_2 are *E-incomparable* in p_i ($v_1 |_i^E v_2$) iff neither $v_1 \rightarrow_i^E v_2$ nor $v_2 \rightarrow_i^E v_1$. Let $Corn_i(x)$ be a set of values which a peer p_i can take after a value x, i.e. $\{y \mid x \rightarrow_i^E y\}$. The *preferentially* (*P*-) *precedent* relation $v_1 \rightarrow_i^P v_2$ [42, 43, 44] is also defined to show that p_i prefers v_1 to v_2 if p_i can take any of v_1 and v_2. In this paper, we consider a static group where each peer p_i does not change the domain D_i and the precedent relations $\rightarrow_i^E$ and $\rightarrow_i^P$.

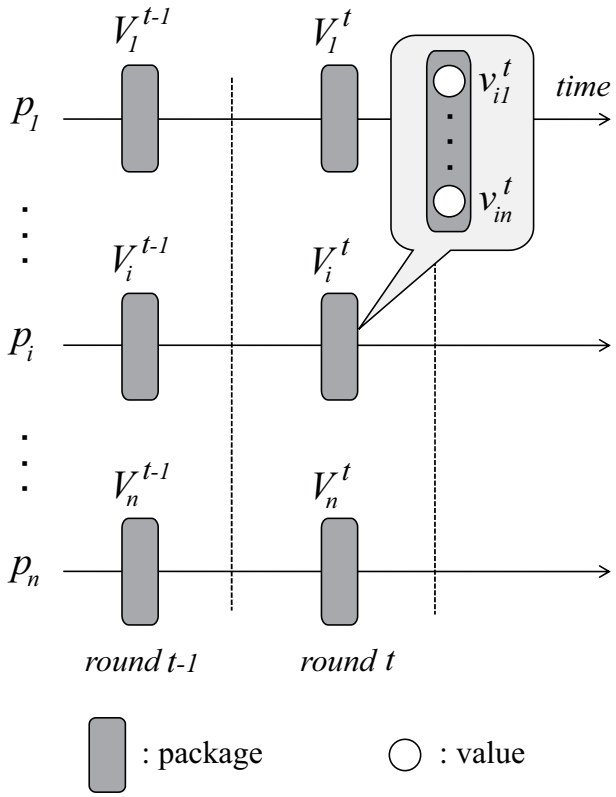


Figure 4.1: Multi-value exchange.

Suppose each peer p_i can have a subset I_i of initial values ($I_i \subseteq D_i$) which p_i would like to take in the agreement procedure. Let PV_i be a set of values $\cup_{x \in I_i} Corn_i(x)$, which shows a subset of possible values which a peer p_i can take at the initial round. If there is a satisfiable tuple $\langle v_1, \ldots, v_n \rangle \in PV_1 \times \cdots \times PV_n$ which satisfies the agreement condition AC, every peer can make an agreement on the tuple. Here, the group G of the peers are *agreeable* for the agreement condition AC. Suppose there are a pair of satisfiable tuples $\langle x_1, \ldots, x_n \rangle$ and $\langle y_1, \ldots, y_n \rangle$. If $x_i \rightarrow_i^E y_i$ or $x_i \mid_i^E y_i$ for $i = 1, \ldots, n$, the tuple $\langle x_1, \ldots, x_n \rangle$ *precedes* the tuple $\langle y_1, \ldots, y_n \rangle$. Suppose a pair of satisfiable tuples $\langle x_1, \ldots, x_n \rangle$ and $\langle y_1, \ldots, y_n \rangle$ are not preceded. If $x_i \rightarrow_i^P y_i$ or $x_i \mid_i^P y_i$ for $i = 1, \ldots, n$, the tuple $\langle x_1, \ldots, x_n \rangle$ is more *preferable* than the tuple $\langle y_1, \ldots, y_n \rangle$.

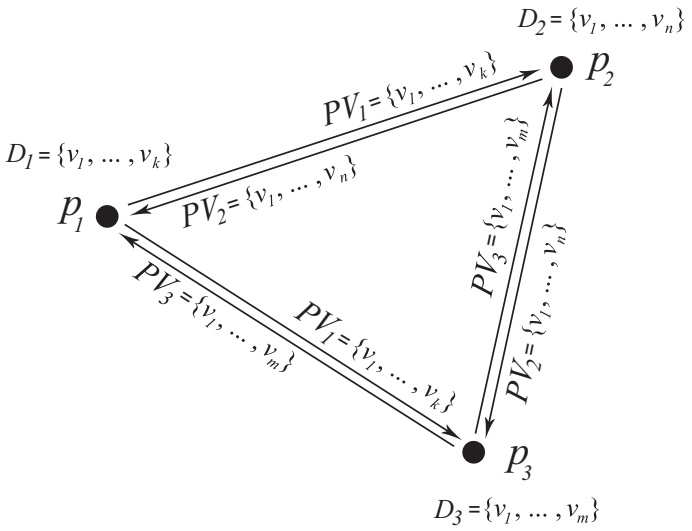


Figure 4.2: Maximal-value exchange (XVE) scheme.

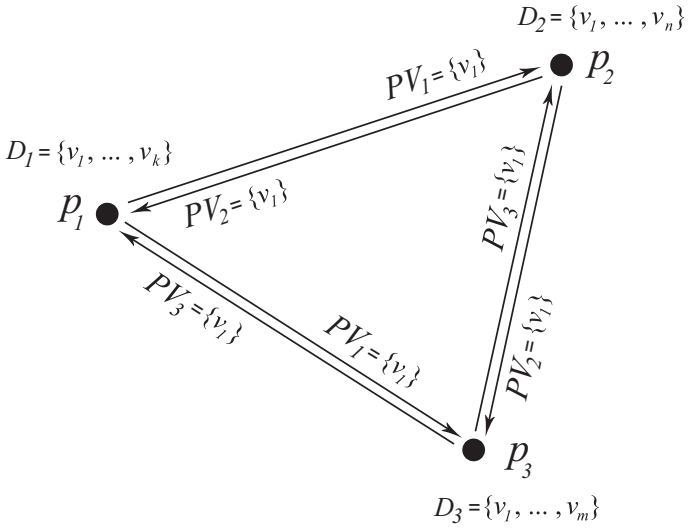


Figure 4.3: Single-value exchange (SVE) scheme.

In the basic agreement protocol, each peer p_i exchanges the value set PV_i with the other peers. Then, each peer p_i finds the most preceded, preferable tuple in the direct product $PV_1 \times \cdots \times PV_n$. It takes just one round to make an agreement. This is a *maximal value exchange* (XVE) scheme [Figure 4.2]. At the

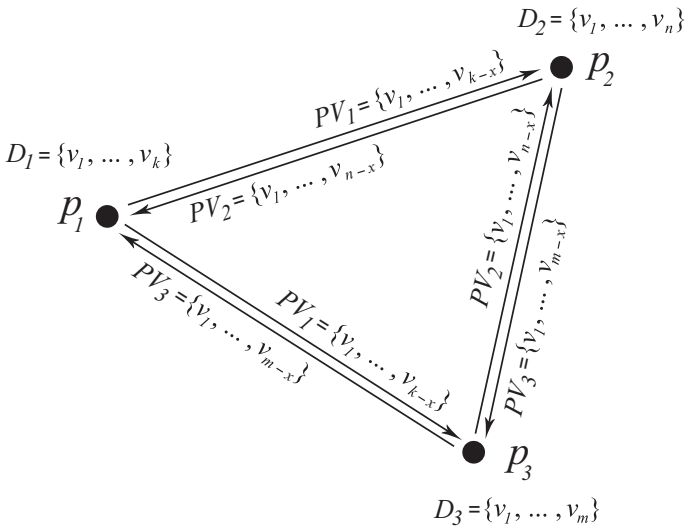


Figure 4.4: Multi-value exchange (MVE) scheme.

other extreme, each peer sends only one value in PV_i like the simple protocols [42, 43, 44, 53]. Each peer p_i has to show a value x after y where $y \rightarrow_i^E x$. This is a *single value exchange* (SVE) scheme [Figure 4.3]. There is a *multi-value exchange* (MVE) [Figure 4.4] scheme in between XVE and SVE. Here, each peer p_i sends a subset V_i of PV_i to the other peers. At each round t, each peer p_i sends a *package* V_i of possible values to the other peers. Values in V_i are ordered in the preference. The top value of the package is the most preferable value named *primary* one. The others are *secondary* ones. On receipt of the package V_j from every peer p_j, each peer p_i finds a satisfiable tuple of values in a collection of the packages $V_1, \ldots, V_n$.

Suppose a pair of peers p_1 and p_2 have possible values a and b and possible values b and c, respectively. If p_1 and p_2 show values a and c, respectively, the peers show different values a and c in the SVE scheme. Here, the peers p_1 and p_2 cannot make an agreement even if the peers have the satisfiable value b. It takes more than one round to show multiple possible values to the other peers. Furthermore, depending on an order in which each peer shows values to the other peers, the peers may not make an agreement. The peer p_1 sends a package $V_1 = \{a, b\}$ and p_2 sends $V_2 = \{b, c\}$ in the MVE scheme. On receipt of the package V_2 from p_2, the peer p_1 finds that the other peer p_2 can also take the value b. Then, the peers p_1 and p_2 agree on the value b. Thus, by taking advantage of the MVE scheme, each peer p_i obtains one or more than one possible value from every other

peer at one round. Then, each peer p_i can find a satisfiable tuple of values in a collection of the packages $V_1, \ldots, V_n$ which p_i has received from the other peers. The more number of values are exchanged at each round, the shorter it takes to make an agreement and the higher possibility every peer makes an agreement but the more communication overhead and processing overhead might be implied. There is a trade off point between the size of a package and the overhead time and availability.

If there is no satisfiable tuple, each peer p_i finds values which is E-preceded by the primary value v_i^{t1} in the package V_i^t. At round t + 1, each peer p_i sends a package V_i^{t+1} where every value is E-preceded by the primary value v_i^{t1} in V_i^t. In this paper, we assume each package V_i^t can include at most some number K ($\geq$1) of the possible values; the primary value v_i^{t1} and secondary values $v_i^{t2}, \ldots, v_i^{tK}$ in order to increase the performance and make the implementation simple.

The application layer of each individual peer makes a decision on what value the peer can take at the next round. In addition, the agreement condition AC is decided according to the purpose of the group like majority decision.

4.2 Multipoint relaying (MPR) scheme

4.2.1 Basic algorithm

A group G is composed of multiple peers processes (peers) $p_1, \ldots, p_n$ ($n > 1$) which are interconnected in P2P overlay networks [58]. In a scalable P2P overlay network, each peer cannot directly send a message to every other peer of a group. Each peer can only send a message to its neighbor *acquaintance* peers [36]. In one approach to broadcasting a message, a peer p_i first sends a message to every neighbor peer p_j. On receipt of a message, the peer p_j forwards the message to the neighbor peers. This is a pure flooding scheme [60]. However, the pure flooding scheme implies the huge network overhead due to the message explosion.

The concept of "multipoint relaying (MPR)" scheme is developed to efficiently broadcast messages [59]. Here, on receipt of a message, a peer forwards the message to all the neighbor peers but only some of the neighbor peers forward the message to other peers. Each peer is assumed to know not only the first neighbor peers but also the second neighbor peers. First neighbor peers are acquaintance peers with which the peer p_i can directly communicate. The peer p_i is assumed to know every second neighbor peer, but cannot directly communicate with it. By taking into consideration the second neighbor peers in addition

to the first neighbor peers, each peer selects a subset of the first neighbor peers only which forward the message. The selected neighbor peers are referred to as *relay* peers. The other neighbor peers which just receive the message and do not forward the message are *leaf* peers. Since the number of messages transmitted can be significantly reduced, the MPR scheme provides an adequate solution to reduce the overhead to broadcast messages in P2P overlay networks. Every leaf peer just receives a message from a relay peer while every relay peer forwards the message to the neighbor peers.

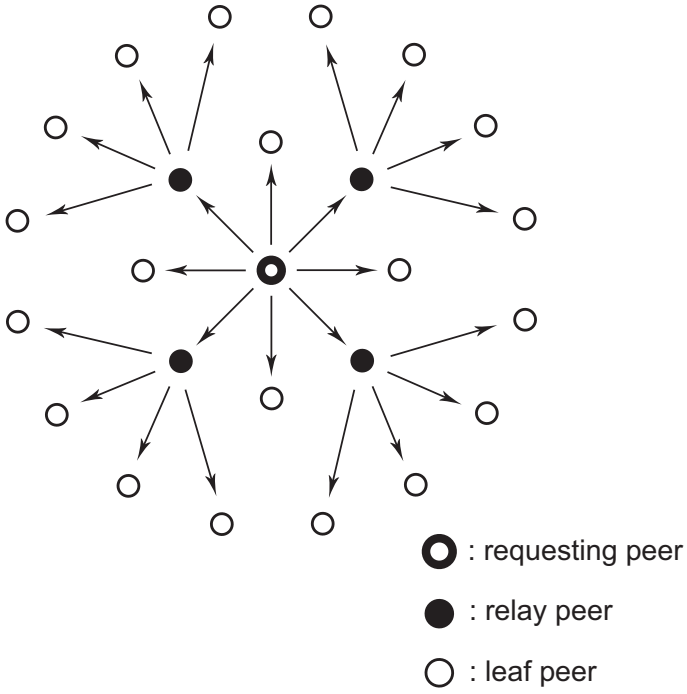


Figure 4.5: Multipoint relays.

Let $N(p_i)$ be a set of first neighbor peers of a peer p_i. A set of the second neighbor peers of a peer p_i is denoted by $N^2(p_i)$. $N^2(p_i) = \cup_{p_j \in N(p_i)} N(p_j) - N(p_i)$. Let $R(p_i)$ and $L(p_i)$ be collections of replay peers and leaf peers of a peer p_i, respectively. Here, $N(p_i) = R(p_i) \cup L(p_i)$ and $R(p_i) \cap L(p_i) = \phi$. The following condition is required to hold:

- $N^2(p_i) = \cup_{p_j \in R(p_i)} N(p_j)$.

A message sent by a peer p_i can be delivered to every second neighbor peer of p_i where only the relay neighbor peers of p_i forward the message to second neighbor peers of p_i. It is noted $N(p_i) \cap N(p_j)$ might not be ϕ for some pair of relay peers p_i and p_j of a peer. If $N(p_i) \cap N(p_j) \neq \phi$, there are multiple ways to deliver a message to a message to a common peer in $N(p_i) \cap N(p_j)$. Here, we define the *coverage* of a peer p_i:

- A peer p_j is referred to as *covered* by a peer p_i iff $p_j \in N(p_i)$ or p_j is covered by some relay peer $p_k \in R(p_i)$.

A collection of peers covered by a peer p_i is referred to as subnetwork *covered* by the peer p_i. An algorithm $MPR(p_i, N(p_i))$ for selecting $R(p_i)$ [59] in $N(p_i)$ is shown as follows:

[**MPR**(p_i, ***C***(p_i))] /* $C(p_i)$ is a subset of the first neighbor peers of a peer p_i. A collection $R(p_i)$ of relay peers are selected in $C(p_i)$ and each relay peer p_j in $R(p_i)$ is assigned with a set $C(p_j)$. */

1. Start with an empty multipoint relay set $R(p_i)$;
   $R(p_i) = \phi$. $S = N^2(p_i)$. $F = C(p_i)$.
2. While $F \neq \phi$, do the following steps:
   (a) select a neighbor peer p_j in F where $N(p_j) \cap N(p_k) = \phi$ for every other first neighbor peer p_k in F.
   (b) if found, $R(p_i) = R(p_i) \cup \{p_j\}$, $S = S - N(p_j)$, $F = F - \{p_j\}$.
   (c) if not found, go to step 3.
3. If $F = \phi$, terminate;
4. While $S \neq \phi$, do the following steps:
   (a) for each peer p_j in F, obtain a subset $U(p_j)$ of peers which p_j covers in the set S, $U(p_j) = N(p_j) \cap S$.
   (b) select a peer p_j where $|U(p_j)|$ is the maximum, $R(p_i) = R(p_i) \cup \{p_j\}$. $S = S - U(p_j)$, $F = F - \{p_j\}$, $C(p_j) = U(p_j)$.
5. For each peer p_j in F, $C(p_j) = \phi$, i.e. p_j is a leaf.
6. For each relay peer p_j in $R(p_i)$, $MPR(p_j, C(p_j))$.

Here, for each neighbor peer p_j in $N(p_i)$, $C(p_j)$ is obtained as a set of neighbor peers of p_j. If p_j is a leaf peer, $C(p_j) = \phi$. For each neighbor peer p_j in $C(p_i)$, the algorithm is recursively applied to obtain a set $R(p_j)$ of relay peers of p_j.

As shown in Figure 4.5, a *directed acyclic graph* (DAG) $D(p_i)$ if $D(p_i)$ is obtained by applying the algorithm MPR to the peer p_i in a group G. Here, p_i is referred to as a *root* peer in $D(p_i)$. Through the DAG $D(p_i)$ of p_i, the peer p_i can deliver a message to every peer in the group G.

4.2.2 Faults

In a DAG obtained by the MPR algorithm, a parent node p_i shows a relay peer which forwards values to the child peers on receipt of the values. A collection of the child peers of a peer p_i is shown as $C(p_i)$. $R(p_i)$ indicates a set of relay peers of a peer p_i obtained by the MPR algorithm. $U(p_i)$ is a set of leaf peers of p_i. Here, $C(p_i) = R(p_i) \cup U(p_i)$ and $R(p_i) \cap U(p_i) = \phi$. Peers colored black and white show relay and leaf peers, respectively, in Figure 4.6.

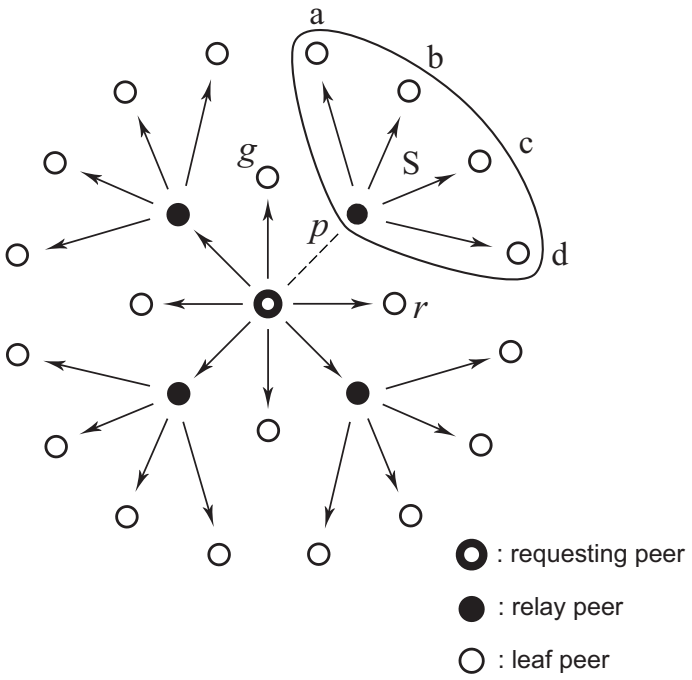


Figure 4.6: Failure in multipoint relays.

A peer which is chosen as a relay peer plays a critical role for delivering messages to other peers. If a relay peer p_i is faulty, every peer covered by the faulty peer p_i cannot receive a message from p_i. Let us consider a subnetwork S of a peer p shown in Figure 4.6, which is circled by the line. A peer p is a root of the subnetwork S which is also a DAG. Suppose the peer p is faulty. Here, every peer in the subnetwork S cannot receive messages which the peer p receives from

the parent. Thus, if a relay peer p_i is faulty, every peer p_j in a sub-network of the peer p_i may not receive messages. If p_i has only one parent p_j, p_i does not receive any messages. Here, p_j is isolated. If p_j has more than one parent, p_j may receive message from another parent, which is not isolated. If every parent of p_i is faulty or isolated, p_i does not receive any messages. Thus, a peer p_i is referred to as *isolated* iff every parent peer of p_i is faulty or each parent of p_i is faulty or isolated. An isolated peer does not receive any message while a faulty peer receives messages but does not send messages. In order to increase the robustness for broadcasting messages, we newly introduce the trustworthiness of a neighbor peer. A *trustworthy* peer is a peer which can send only correct messages to child peers if the peer is a relay type. The higher trustworthy a peer is, the more higher probability the peer can forward message. A peer p_i selects more trustworthy neighbor peers as relay peers. Then, the peer p_i sends a message to the neighbor peers and only the trustworthy neighbor peers forward the message to their neighbor peers. Suppose a second neighbor peer p_k in $N^2(p_i)$ has multiple first neighbor peers $p_{k1}, \ldots, p_{kl_k}$ in $N(p_i)$ which are parents of p_k. Hence, the most trustworthy neighbor peer p_{kh} is selected as a relay peer. Here, the peer p_{kh} has the highest possibility to deliver a message from p_i to p_k.

Let us consider Figure 4.7 (a) as an example. Here, let T_i show the trustworthiness value of a peer p_i. In Figure 4.7, suppose $T_g > T_r > T_p$ for three peers g, r, and p. Here, we select the most trustworthy peer g as a relay peer. Then, the peer g forwards a message to every peer in the subnetwork S. This is an ideal case, that is, the subnetwork S which is originally covered by the peer p can be also covered by the peer g. However, the peer g might not be able to cover every peer in the subnetwork S as shown in Figure 4.7 (b). Therefore, another peer has to be selected to cover the peers which the peer g does not cover. In Figure 4.7 (b), the peers c and d uncovered by the peer g are covered by the second most trustworthy peer r. The overall idea is that every subnetwork is covered by a most trustworthy relay peer. It depends on the overlay topology among peers how many number of relay peers are required to cover all the peers in a subnetwork. In Figure 4.7 (b), one more relay peer is required to cover the same subnetwork S as Figure 4.6. If we use more number of trustworthy neighbor peers to transmit messages to others, we can increase the overall fault-tolerance of the MPR mechanism.

4.3 Trustworthiness-based broadcast (TBB) scheme

4.3.1 Trustworthiness of peer

In P2P systems, each peer has to obtain information of other peers and propagate the information to other peers through neighbor peers. A neighbor peer p_j of a peer p_i means an acquaintance with which p_i can directly communicate. Thus, it is significant for each peer to have some number of neighbor peers. Moreover, it is more significant to discuss if each peer has trustworthy neighbor peers. In reality, each peer might be faulty or might send obsolete, even incorrect information to the other peers. If some peer p_j is faulty, other peers which receive incorrect information on the faulty peer p_j might reach a wrong decision. It is critical to discuss how a peer can trust each of its neighbor peers [36]. In this paper, we newly introduce a *trustworthiness-based broadcast* (*TBB*) algorithm by introducing the trustworthiness concept to the MPR algorithm.

Suppose a requesting peer p_r would like to select a neighbor peer p_i as a relay peer for broadcasting a message to the other peers. Let T_{ri} show the trustworthiness of a neighbor peer p_i for a peer p_r, which the peer p_r holds. $N(p_r)$ shows a collection of neighbor peers of the requesting peer p_r. The peer p_r calculates the trustworthiness T_{ri} of a neighbor peer p_i by collecting information on the peer p_i from every neighbor peer p_k in $N(p_r)$ which can communicate with both p_i and p_r, i.e. $p_k \in N(p_r) \cap N(p_i)$. There is some possibility that the peer p_i is faulty or sends incorrect information. Hence, the peer p_r does not consider the information from the target peer p_i to calculate the trustworthiness $T_r(p_i)$.

A peer p_k sends a trustworthiness request to the peer p_i and receives a reply from p_i. This interaction is referred to as *transaction*. If p_k receives a successful reply, the transaction is successful. Otherwise, it is unsuccessful. The peer p_k considers the neighbor peer p_i to be more trustworthy if p_k had more number of successful transactions for p_i. Let BT_{ki} be the *subjective* trustworthiness [36] T_{ki} on the target peer p_i which a peer p_k obtains through communicating with the peer p_i. Let TT_{ki} show the total number of transactions which p_k issues to p_i. Let ST_{ki} ($\leq TT_{ki}$) be the number of successful transactions which p_k issues to p_i. Here, the subjective trustworthiness BT_{ki} is calculated as follows:

$$BT_{ki} = \frac{ST_{ki}}{TT_{ki}} \tag{4.1}$$

If the peer p_i is not a neighbor peer p_k, $p_i \notin N(p_k)$, the peer p_k cannot obtain the subjective trustworthiness BT_{ki}. In addition, if the peer p_k had not issued any

transaction to the peer p_i even if $p_i \in N(p_k)$, $BT_{ki} = \perp$. Thus, according to communication with each neighbor peer p_k, each peer p_r obtains the subject trustworthiness BT_{ki} for the neighbor peer p_i. The subjective trustworthiness BT_{ki} shows how reliably a peer p_i is recognized by a peer p_k. Therefore, if a peer p_r would like to get the trustworthiness of a target peer p_i, the peer p_r asks each neighbor peer p_k to send the subjective trustworthiness BT_{ki} of the peer p_i. Each neighbor peer p_k keeps in record of the subject trustworthiness BT_{ki} in the log. Here, let S be a collection of neighbor peers which send the subjective trustworthiness on p_i which is not $\perp$ to the peer p_r. After collecting the subjective trustworthiness BT_{ki} of the target peer p_i from each neighbor peer p_k, the requesting peer p_r calculates the trustworthiness T_{ri} of the peer p_i by the following formula:

$$T_{ri} = \frac{\sum_{p_k \in \{p_k \in S | BT_{ki} \neq \perp\}} BT_{ki}}{|\{p_k \in S | BT_{ki} \neq \perp\}|} \tag{4.2}$$

Let us consider Figure 4.8 as an example. Here, a requesting peer p_r would like to know the trustworthiness T_{ri} of a neighbor peer p_i. The peer p_r has five neighbor peers, p_1, p_2, p_3, p_4, and p_i. Here, $N(p_r)$ = $\{p_1, p_2, p_3, p_4, p_i\}$. A collection of neighbor peers of the peer p_r which excludes the peer p_i is indicated by a collection S = $N(p_r)$ - $\{p_i\}$ = $\{p_1, p_2, p_3, p_4\}$. Here, the requesting peer p_r requests each neighbor peer p_k in the neighbor set S to send the subjective trustworthiness BT_{ki} of the peer p_i (k = 1, 2, 3, 4). After receiving the subjective trustworthiness of the peer p_i from all the four neighbors in S, the peer p_r calculates the trustworthiness T_{ri} of the peer p_i by using the formula (4.2), T_{ri} = (BT_{1i} + BT_{2i} + BT_{3i} + BT_{4i}) / 4.

4.3.2 Trustworthiness - based broadcast (TBB) algorithm

By using the trustworthiness of each neighbor peer, the original MPR algorithm is modified to the trustworthiness-based broadcast (TBB) algorithm. In order to select relay peers of a peer p_r, the following procedure $TBB(p_r, N(cp_r))$ is applied to a DAG whose root is p_r:

$\mathbf{TBB}(\mathbf{p_r}, \mathbf{C}(\mathbf{p_r}))$

1. Start with an empty relay set $R(p_r)$, $R(p_r) = \phi$. Let S be a set of trustworthy neighbors of p_r, i.e. $\{p_j \in C(p_r) \mid T_{rj} \geq \alpha\}$ where $0 \leq \alpha \leq 1$. α gives a threshold value on the trustworthiness. If $T_{ri} \geq \alpha$, the peer p_r recognizes

the neighbor peer p_i to be trustworthy. Otherwise, p_i is considered to be untrustworthy.

2. While $TF \neq \phi$, do the following steps:
   (a) select a trustworthy neighbor peer p_i in TF such that $N(p_i) \cap N(p_j)$ = ϕ for every trustworthy peer p_j in TF ($p_j \neq p_i$).
   (b) if found, F = F - $\{p_i\}$, TF = TF - $\{p_i\}$, S = S - $N(p_i)$, $R(p_r)$ = $R(p_r)$ $\cup$ $\{p_i\}$.
   (c) if not found, go to step 3.
3. While $TF \neq \phi$, do the following steps:
   (a) $U(p_j)$ = $N(p_j) \cap S$ for each p_j in TF.
   (b) select a trustworthy neighbor peer p_i in TF such that $|U(p_i)|$ is the maximum, i.e. the number of neighbor peers which are not covered is the maximum.
   (c) F = F - $\{p_i\}$, TF = TF - $\{p_i\}$, SS = S, S = S - $N(p_i)$, $R(p_r)$ = $R(p_r)$ $\cup$ $\{p_i\}$, $C(p_i)$ = $N(p_i) \cap SS$.
4. While $F \neq \phi$, /* TF = ϕ */ do the following steps:
   (a) select a peer p_j in F such that $|N(p_j) \cap S|$ is the minimum.
   (b) F = F - $\{p_j\}$, SS = S, S = S - $N(p_j)$, $R(p_r)$ = $R(p_r) \cup \{p_j\}$, $C(p_i)$ = $N(p_i) \cap SS$.
5. For each relay neighbor peer p_i in $R(p_r)$, $TBB(p_i, C(p_i))$.

For each neighbor peer p_i, $C(p_i)$ gives a collection of neighbor peers to which p_i forwards a message, $C(p_i) \subseteq N(p_i)$. If p_i is not a relay peer, $C(p_i)$ = ϕ. $C(p_i)$ = $R(p_i) \cup U(p_i)$ and $R(p_i) \cap U(p_i)$ = ϕ. In step 4, each untrustworthy neighbor peer p_i is assigned with as small number of neighbors as possible. Even if the peer p_i is faulty, only a smaller number of peers are damaged.

Let $MT(p_r)$ be a directed acyclic graph (DAG) of a peer p_r obtained by the algorithm $TBB(p_r, N(p_r))$. Here, p_r is a root peer of the DAG $MT(p_r)$. Here, a DAG $MT(p_r)$ is referred to as ***fault-isolated*** iff every relay peer p_i in $R(p_r)$ is trustworthy and a subDAG $MT(p_i)$ is also fault-isolated. In the fault-isolated DAG, every untrustworthy peer is a leaf peer. Hence, even if an untrustworthy peer p_i is faulty, no other peer is isolated.

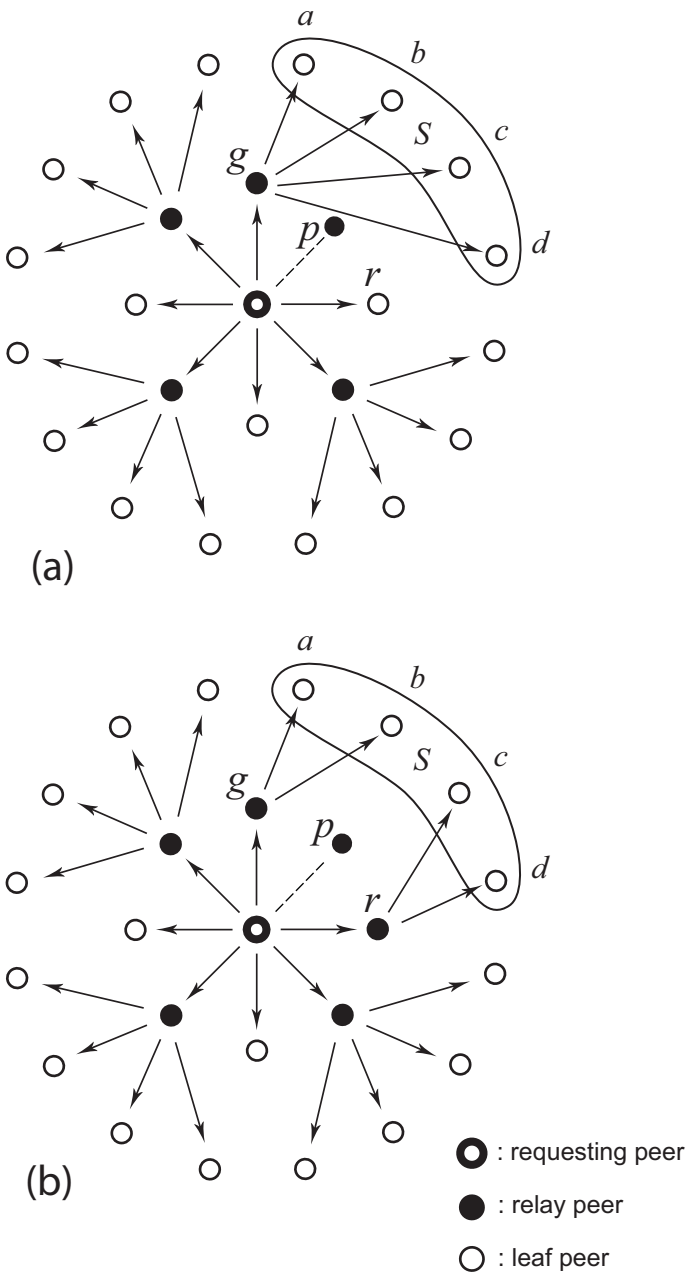


Figure 4.7: Trusted neighbors in multipoint relays.

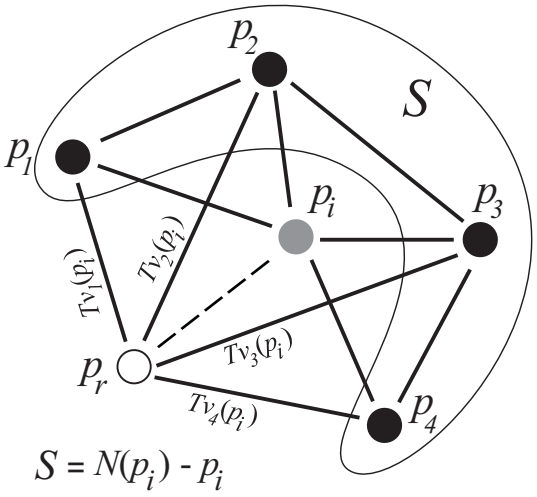


Figure 4.8: Trustworthiness of peer.

Chapter 5

Evaluation

5.1 Assumptions

Compared with the original MPR algorithm and pure flooding algorithm, we evaluate the proposed trustworthiness-based broadcast (TBB) algorithm in terms of the number of messages transmitted to broadcast a message in a network. In this evaluation, we consider an L * L grid structured overlay network for simplicity. In this evaluation, L shows the length of the grid which means how many peers on each point of the grid. The total number n of peers in the network is $L * L$. Since both of the MPR algorithm and the TBB algorithm aim at reducing the number of unnecessary messages, we measure the number of messages which are sent in each algorithm. As we mentioned in the preceding section, peers are isolated due to faults of relay peers in the MPR algorithm under a constraint that every peer receives a message sent by a root peer. Hence, we evaluate the algorithms in terms of the number of messages transmitted in presence of faulty peers.

In this paper, a faulty peer is assumed to receive a message but is not able to forward the message to other peers. An algorithm is referred to as *sound* iff a message can be delivered to all the peers in the network. Since in the agreement procedures the opinions of the participant peers are significant to the outcome of the agreement procedure, the protocol which we consider should work in sound way with fewer number of messages exchanged in the network.

In the evaluation, some number of peers are randomly selected to be faulty. F shows the ratio of the faulty peers to the total number n (= L^2) of peers in the network. For example, "F = 0.05" means that five percentages of the peers are faulty. T_i shows the trustworthiness of a peer p_i, which is randomly assigned to

each peer p_i. T_i is a value randomly chosen in range of 0.1 to 1.0. The higher T_i is, the more trustworthy the peer p_i is. First, the trustworthiness T_i is given to each peer p_i. Then, each peer p_i is decided whether p_i is faulty or not based on the faulty ratio F. Depending on the trustworthiness value T_i of each peer p_i, we select a peer which has the smallest T_i value to be faulty. If we found multiple peers which have the same lowest T_i value, we take a peer whose peer ID is the biggest. That is, the lower trustworthiness T_i a peer p_i has, the more frequently p_i is faulty.

5.2 Scenarios

In the MPR algorithm, no trustworthiness concept is considered while aiming at reducing the number of relay peers in the network. The basic procedure of the MPR algorithm is shown as follows:

1. Initiate the procedure from initiator peer.
2. Obtain a list $NP1$ of first neighbors. First neighbor peers of the initiator peer mean peers which have direct connections with the initiator peer.
3. After obtaining $NP1$, calculate a list $NP2$ of the second neighbor peers of the initiator peer by obtaining the first neighbor peers of the peers in $NP1$.
4. Calculate the MPR peers which can cover the peers in $NP2$ as follows:
   (a) Find a peer in $NP1$ which has the largest number of connections with other peers.
   (b) If multiple peers have the same number of connections, take a peer whose peer ID is the biggest.
   (c) Mark peers in $NP2$ through which the selected peer can pass the message, then put the selected peer into the MPR list.
5. After each MPR calculation, the network is checked, if all peers have received the message. If so, then terminate the procedure.
6. If all peers are not covered yet, return to step 2 with the MPR list and apply the same procedure to each MPR peer in the list.

7. By repeating this procedure, we can cover the $NP2$ set by using the peers who have the largest number of connections as MPR nodes.

8. Finally, we can deliver the message to each peer in the network by passing message only through MPR nodes.

9. We calculate the total number of messages sent to cover all peers in the network.

In the TBB algorithm, we consider how to more reliably deliver messages to other peers in presence of faulty peers in the network. The basic procedure of the TBB algorithm is shown as follows:

1. Assign the trustworthiness value T_i to each peer p_i in the network.

2. According to the faulty ratio F and trustworthiness, select faulty peers in the network. Since the trust value is considered, the peer p_i which has the higher trust value T_i is not be easily fail.

3. Initiate the procedure from initiator peer.

4. Calculates a list $NP1$ of first neighbor peers of the initiator peer.

5. Obtain a list $NP2$ of second neighbor peers according to the $NP1$ peers.

6. Calculate the MPR peers which can cover the $NP2$ neighbors as follows:

   (a) Find a peer in $NP1$ which has the highest trustworthiness value.

   (b) If multiple peers have the same trustworthiness value, take a peer whose peer ID is the largest.

   (c) Mark the peer in the $NP2$ set which the selected peer can pass the message thorough, then put the selected peer into the MPR list.

7. After each MPR calculation, the network is checked, if all peers received the message. If so, terminate the procedure.

8. Otherwise, return to step 4 with the MPR list previously calculated and apply the same procedure to each MPR peer in the list.

9. By repeating this procedure, we can cover the $NP2$ set by using peers who have higher trustworthiness value as MPR nodes.

10. After obtaining the MPR list for the initiator peer, we apply the same procedure to each MPR peer again.

11. Finally, we can deliver the message to each peer in the network by passing the message only through MPR peers.

12. We calculate the total number of messages sent to cover all peers in the network.

5.3 Results

We evaluate the algorithms for different faulty ratios F in the network. Figures 5.1 and 5.2 show the numbers of messages with total number n of peers for F = 0.05 and F = 0.1, respectively. Here, in absence of faulty peers in the network, i.e. F = 0 and with F = 0.05, a message can be delivered to all the peers by using fewer number of messages in the MPR algorithm than the TBB algorithm. In the pure flooding scheme, the largest number of messages are transmitted to deliver messages as shown in Figures 5.1 and 5.2. However, if ten percentages of the peers are faulty in the network (F = 0.1), a message cannot be delivered to all the peers in the MPR algorithm, i.e. MPR is not sound. On the other hand, the TBB algorithm is sound, i.e. a message can be delivered to all the peers with fewer number of messages than the pure flooding as shown in Figure 5.2. Thus, the TBB algorithm is more sound, i.e. more reliable and more efficient, i.e. fewer number of messages are transmitted.

Figure 5.3 shows the average value of network coverage of each algorithms to the faulty ratio F of the network where number of peers n taken from 100 to 10000, how many peers in the network. In the MPR algorithm, messages cannot be delivered to all the peers for larger than about six percentages of the faulty peers in the network (F = 0.06). For F = 0.1, about 40 percentage of the peer cannot receive messages. On the other hand, in the TBB algorithm, messages cannot be delivered to all the peers for $F > 0.18$. For F = 0.27, more than 90 percentages of the peers can receive messages. Figure 5.4 shows the average value of number of messages for the faulty ratio F where n taken from 100 to 10000. As shown in the Figure 5.4, the TBB algorithm can cover the same network with the less number of messages than the pure flooding and MPR ones. In addition, in reality, the situation like about 20 percentage of the peers are faulty in a network is unlikely happens.

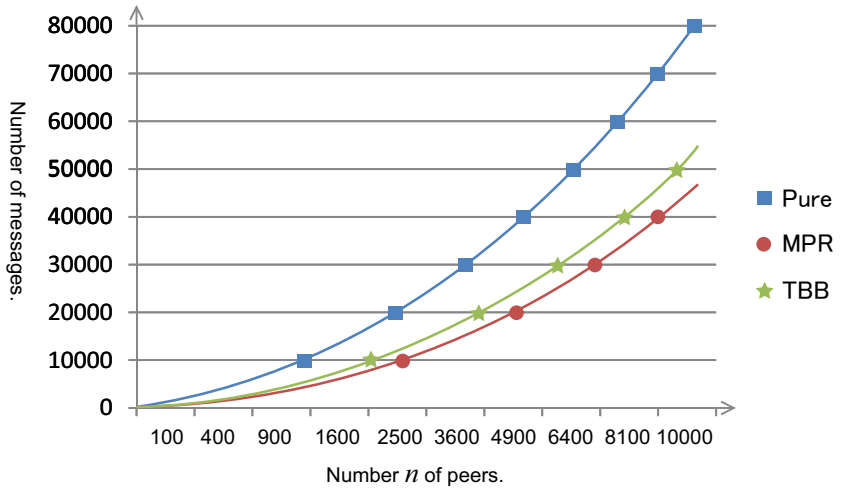


Figure 5.1: Number of messages (F = 0.05).

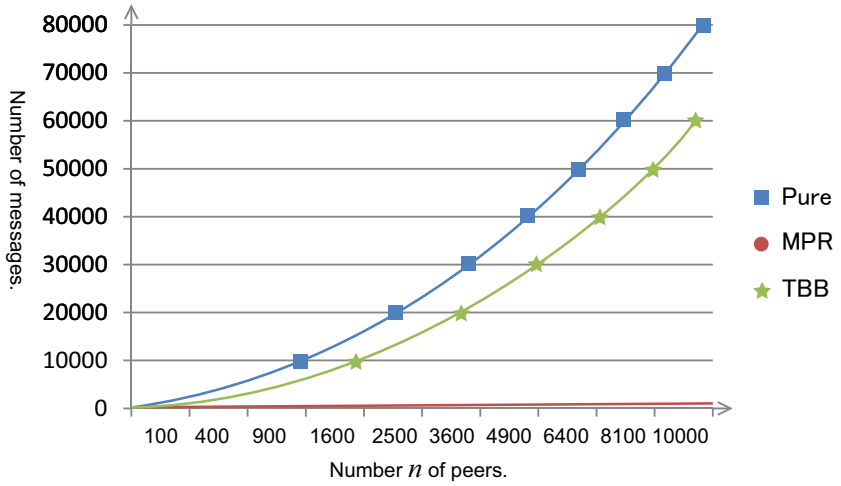


Figure 5.2: Number of messages (F = 0.1).

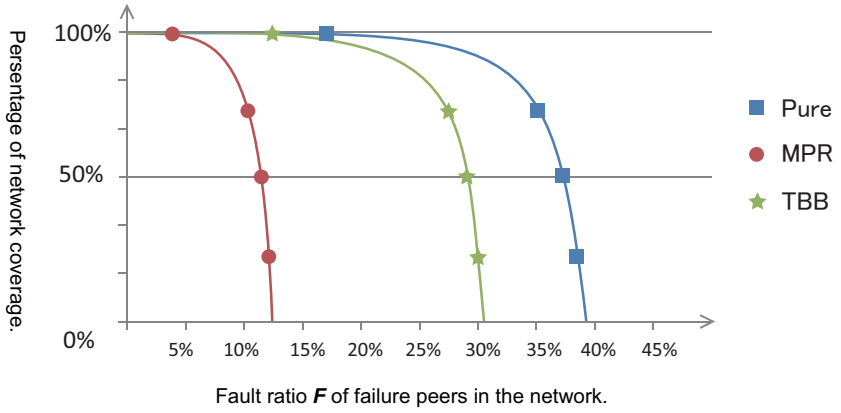


Figure 5.3: Network coverage to fault ratio.

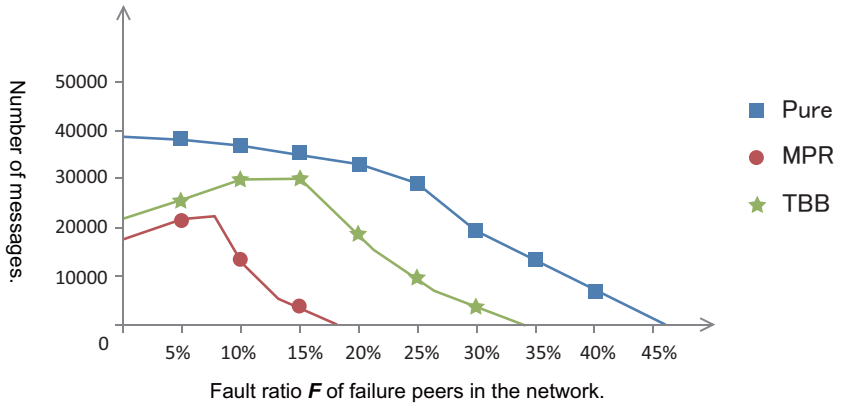


Figure 5.4: Number of messages to fault ratio.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In chapter 1, we discussed background and objectives of this research and the problems exists in the Peer-to-Peer (P2P) overlay networks and Distributed Agreement Protocols. We also discussed related work and contribution of this dissertation. It shows that variety of factors effect the agreement procedure in distributed systems but most importantly the way to reliably and efficiently exchange information among peers are the most critical one, in order to improve this issue and improve over all performance of the agreement protocol we introduced trustworthiness based broadcast (TBB) algorithm.

In chapter 2, We discussed how each peer trusts acquaintance peers in a fully distributed P2P overlay network. First, we defined the subjective trustworthiness $st_{ij}(\rho)$ of a peer p_i to an acquaintance p_j for an access request ρ issued to an acquaintance peer. If the acquaintance p_j returns a more satisfiable reply to the requesting peer p_{ij} the subjective trustworthiness $st_{ij}(\rho)$ is increased. Next, the objective trustworthiness ot_{ij} is introduced to show how much the acquaintance peer p_j is trusted by trustworthy acquaintance peers of the peer p_i. We defined four levels of the functions OT_0, OT_1, OT_2, and OT_3 to calculate the objective trustworthiness ot_{ij} of a requesting peer p_i to an acquaintance p_j. OT_0 stands for the traditional reputation [26, 29] where messages are flooding in the network. The higher the function is, the more the objective trustworthiness ot_{ij} is dominated by the trustworthiness opinion of the peer p_i to the acquaintance peer p_j. We showed that faulty service information from acquaintances can be removed to calculate the objective trustworthiness in the higher level OT functions through

the evaluation. We discussed the confidence of each peer on its own opinion of trustworthiness of another peer. A peer p_i takes the subjective trustworthiness to an acquaintance p_j if p_i is the most confident. If the peer p_i is the least confident, p_i takes the lowest level of the objective trustworthiness. The confidence of a peer depends on communication time, frequently, stableness, and number of peers trusting the peer.

In chapter 3, we mainly discussed the topic of basic agreement protocol. We introduced the basic procedure of an agreement protocol among multiple peers. We discussed the problems to appear in the distributed agreement protocols and showed our proposed algorithms to solve the problems. In human societies, each person may change its opinion in the agreement procedure. By abstracting the human behaviors in social agreement procedure, we discussed the flexible agreement protocol in a society of peers. First, values in a domain D_i are partially ordered in existentially (E-) and preferentially (P-) precedent relations $\rightarrow_i^E$ and $\rightarrow_i^P$ in each peer p_i. Each peer just autonomously takes a value by using the E- and P-precedent relations at each round. The peers may not make an agreement since a value from a peer might cross a value from another peer even if the values satisfy the agreement condition. In order to flexibly make an agreement, we need coordination mechanisms of multiple peers. We proposed four types of coordination strategies, *forward*, *backward*, *mining*, and *observation* strategies. If values taken by the peers do not satisfy the agreement condition, each peer takes a new value in the *forward* strategy. After some rounds, some collection of values which the peers have so far taken may satisfy the agreement condition. We defined a *satisfiable cut* which is a tuple of previous values satisfying the agreement condition which is taken by peers. There may be some values which a peer cannot withdraw. We defined *uncompensatable* values which a peer cannot withdraw after showing to other peers. We defined a *recoverable* cut of the previous values not only which is satisfiable but also to which every peer can back. If there is a recoverable cut where each peer can back to the previous round, every peer can make an agreement by backing to a previous round. Every peer first proposes a coordination strategy to the other peers. If proposed strategies are consistent, each peer applies its strategy. Strategies proposed by peers might be inconsistent, i.e. each peer cannot apply its proposed strategy. We defined the consistent, inconsistent, and conditionally consistent relations among the strategies. We discuss how to resolve the inconsistency among the strategies.

In chapter 4, we discussed the Distributed Agreement Protocols. A pair of novel algorithms, Multi-Value Exchange (MVE) and Trustworthiness-Based Broadcast (TBB) algorithms, respectively. By taking usage of the MVE and TBB algo-

rithms, we improved the efficiency of the most significant part of the agreement procedure, the value exchange phase. In the TBB algorithm, an efficient and reliable way to broadcast messages to all the peers in a group to make an agreement is discussed. We introduced the novel trustworthiness concept of neighbor peers and discussed the trustworthiness-based broadcast (TBB) algorithm to broadcast messages. Here, only more trustworthy peers forward messages and less trustworthy peers do not forward messages. By making trustworthy peers forward messages to other peers, we can remove effect of faulty peers to deliver message to all the peers.

In chapter 5, we evaluated the proposed TBB algorithms. In order to show the reliability and efficiency of the algorithm we compared the proposed TBB algorithm with the multipoint relay (MPR) algorithm and pure flooding. The evaluation result shows that, with more than five percentage faulty peers in the network, the MPR algorithm is not able to deliver the message to all peers in the network, i.e. not sound. On the other hand, the TBB algorithm can still deliver the message to the all peers in the network. Furthermore, about 22 percentages fewer number of messages are transmitted to deliver a message to all the peers than the traditional pure message flooding.

The concepts, algorithms, implementation, and evaluation of the agreement protocol discussed in this dissertation can be not only theoretical but also practical foundation to design and develop various of applications on P2P overlay networks.

6.2 Future work

In this dissertation, we evaluated the proposed Trust-based Broadcast (TBB) algorithm in the simulation which was discussed. To gather more real world data further more evaluation is suggested, like in the NS3 [41], Neko [39, 40] network simulators. Therefore, implementation of our TBB algorithm for large-scale P2P environment is the issue for our future work.

Bibliography

[1] C. Shirky, "What is p2p ... and what isn't", http://openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html, 2000.

[2] I. Foster and C. Kesselman, *The Grid2: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 2003.

[3] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *International Journal of High Performance Computing Applications (IJHPCA)*, 15(3), 2001, pp.200-222.

[4] S. Androutsellis-Theotokis and D. Spinellis, "A Survey of Peer-to-Peer Content Distribution Technologies", *ACM Computing Surveys*, 36(4), 2004, pp.335-371.

[5] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "A Distributed Approach to Solving Overlay Mismatching Problem", *Proceedings of the 24th IEEE International Conference on Distributed Computing System (ICDCS2004)*, 2004, pp.132-139.

[6] D. Winer, "DaveNet: What is p2p?", http://scripting.com/davenet/2000/09/20/whatisp2p.html, 2000.

[7] Napster, http://www.napster.com/.

[8] M. Ripeanu, "Peer-to-Peer Architecture Case Study: Gnutella Network", *Proceedings of the International Conference on Peer-to-Peer Computing (P2P2001)*, 2001, pp.99-100.

[9] LimeWire, http://www.limewire.com/.

[10] Kazaa, http://www.kazaa.com/.

[11] Freenet, http://www.freenetproject.org/.

[12] R. Dingledine, M. J. Freedman, and D. Molnar, “The Free Haven Project: Distributed Anonymous Storage Service”, *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability (DIAU200)*, 2000, pp.67-95.

[13] K. Aberer, “P-Grid: A Self-Organizing Access Structure for P2P Information Systems”, *Proceedings of the 9th International Conference on Cooperative Information Systems (CoopIS)*, 2001, pp.179-194.

[14] P. Maymounkov and D. Mazieres, “Kademlia: A Peer-to-peer Information System Based on the Xor Metric”, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS’02)*, 2002, pp.53-65.

[15] S. Ratnasamy, P. Francis, and S. Handley, “A Scalable Content-Addressable”, *ACM SIGCOMM2001*, 2001, pp.161-172.

[16] A.I.T Rowstron and P. Druschel, “Mapping the Gnutella Network”, *IEEE Internet Computing*, 6(1), 2002, pp.50-57.

[17] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications”, *IEEE/ACM Transactions on Networking (TON)*, 11(1), 2003, pp.17-32.

[18] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, “Tapestry: a fault-tolerant wide-area application infrastructure”, *Computer Communication Review (ACM SIGCOMM)*, 32(1), 2002, pp.81.

[19] ICQ, http://www.icq.com.

[20] Jabber, http://www.jabber.org.

[21] J. Kubiatowicz, D. Bindel, Y. Chen, S. E. Czerwinski, P. R. Eaton, D. Geels, R. Gummadi, S. C. Rhea, H. Watherspoon, W. Weimer, C. Wells, and B. Y. Zhao, “OceanStore: An Architecture for Global-Scale Persistent”, *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX 2000)*, 2000, pp.190-201.

[22] D. P. Anderson, J. Cobb, E. Korpella, M. Lebofsky, and D. Werthimer, "SETI@home: An Experiment in Public-Resource Computing", *Communications of the ACM (CACM)*, 45(11), 2002, pp.56-61.

[23] JXTA, http://www.jxta.org/.

[24] I. Clark, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System", *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, 2000, pp.311-320.

[25] A. Crespo and H. Garcia-Molina, "Routing Indices for Peer-to-Peer Systems", *Proceedings of the 22nd IEEE ICDCS*, 2002, pp.23-32.

[26] F. M. Cuenca-Acuna, R. P. Martin, and T. D. Nguyen, "PlanetP: Using Gossiping and Random Replication to Support Reliable Peer-to-Peer Content Search and Retrieval", *Technical Report DCS-TR-494*, Rutgers University, 2002.

[27] D. E. Denning and P. J. Denning, "Data Security", *In ACM Computing Surveys*, 1979, pp.227-249.

[28] T. Egemen, N. Deepa, and S. Hanan, "An Efficient Nearest Neighbor Algorithm for P2P Settings", *Proceedings of the 2005 national conference on Digital government research*, 2005, pp.21-28.

[29] D. S. Kamvar, T. M. Schlosser, and H. Garcia-Molina, "The Eigentrust Algorithm for Reputation Management in P2P Networks", *Proceedings of the 12th IEEE International Conference on World Wide Web*, 2003, pp.640-651.

[30] F. Lau, S. Rubin, M. Smith, and L. Trajkovic, "Distributed denial of service attacks", *In Proceedings of 2000 IEEE International Conference on Systems, Man, and Cybernetics*, 2000, pp.2275-2280.

[31] Y. Nakajima, K. Watanabe, N. Hayashibara, T. Enokido, M. Takizawa, and S. M. Deen, "Trustworthiness in peer-to-peer overlay networks", *In Proceedings of the IEEE International Conference on Sensor Network, Ubiquitous, and Trustworthy Computing (SUTC 2006)*, 2006, pp.86-93.

[32] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A Scalabel Content-Addressable Network", *In Proceedings of the 2001 conference on*

Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '01), 2001, pp.161-172.

[33] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systmes", *In Proceedings of IFIP/ACM International Conference on Distirubuted Systems Platforms (Middleware)*, 2001.

[34] K. Watanabe, T. Enokido, M. Takizawa, and K. Kim, "Charge-based Flooding Algorithm for Detecting Multimedia Objects in Peer-to-Peer Overlay Networks", *In Proceedings of the 19th IEEE Conference on Advanced Information Networking and Applications (AINA 2005)*, 2005, 1, pp.165-170.

[35] K. Watanabe, N. Hayashibara, and M. Takizawa, "CBF: Look-up Protocol for Distributed Multimedia Objects in Peer-to-Peer Overlay Networks", *Journal of Interconnection Networks (JOIN)*, 2005, 6(3), pp.323-344.

[36] K. Watanabe, Y. Nakajima, T. Enokido, M. Takizawa, "Ranking Factors in Peer-to-Peer Overlay Networks", *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2007, 2(3).

[37] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation Based Trust for Peer-to-Peer Electronic Communities", *IEEE Transactions on Knowledge and Data Engineering*, 2004, 16(7), pp.843-857.

[38] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. "Tapestry: An Infrastructure for Fault-resilient Wide-area Location and Routing", *Technical Report UCB/CSD-01-1141*, University of California, Berkeley, 2001.

[39] P. Urban, X. Defago, and A. Schiper, "Neko: A Single Environment to Simulate and Prototype Distributed Algorithms", *Journal of Information Science and Engineering (JISE)*, 18(6), 2002, pp.981-997.

[40] P. Urban, X. Defago, and A. Schiper, "Neko: A Single Environment to Simulate and Prototype Distributed Algorithms", *In Proceedings of the 15th International Conference on Information Networking (ICONIN2001)*, 2001, pp.503-511.

[41] The NS-3 network simulator, http://www.nsnam.org/.

[42] Aikebaier, A., Enokido, T., Takizawa, M.: Checkpointing in a Distributed Coordination Protocol for Multiple Peer Processes. *In: Proc. of the 2nd International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2008)*, pp. 48–54. (2008)

[43] Aikebaier, A., Hayashibara, N., Enokido, T., Takizawa, M.: A Distributed Coordination Protocol for a Heterogeneous Group of Peer Processes. *In: Proc. of the IEEE 21th Conference on Advanced Information Networking and Applications(AINA 2007)*, pp. 565–572. (2007)

[44] Aikebaier, A., Hayashibara, N., Enokido, T., Takizawa, M.: Making an Agreement in an Order-Heterogeneous Group by using a Distributed Coordination Protocol. *In: Proc. of the 2nd International Workshop on Advanced Distributed and Parallel Network Applications (ADPNA 2007)*, CD-ROM. (2007)

[45] Corman, A.B., Schachte, P., Teague, V.: A Secure Group Agreement (SGA) Protocol for Peer-to-Peer Applications. *In: Proc. of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, pp. 24–29. (2007)

[46] Ezhilchelvan, P., Morgan, G.: A Dependable Distributed Auction System: Architecture and an Implementation Framework. *In: Proc. of the IEEE 5th International Symposium on Autonomous Decentralized Systems (ISADS)*, pp. 3–7. (2001)

[47] Gray, J., Lamport, L.: Consensus on Transaction Commit. *ACM Transactions on Database Systems (TODS) archive*, vol. 31(1), pp. 133–160. (2006)

[48] Hurfin, M., Raynal, M., Tronel, F., Macedo, R.: A General Framework to Solve Agreement Problems. *In: Proc. of the 18th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pp. 56–65. (1999)

[49] Kling, R.: Cooperation, Coordination and Control in Computer-supported Work. *Communications of the ACM*, vol. 34(12), pp. 83–88. (1991)

[50] Lamport, L., Shostak, R., Pease, M.: The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, vol. 4(3), pp. 382–401. (1982)

[51] Lee, P., Lui, J., Yau, D.: Distributed Collaborative Key Agreement Protocols for Dynamic Peer Groups. *In: Proc. of the 10th IEEE International Conference on Network Protocols*, pp. 322–331. (2002)

[52] Sabater, J., Sierra, C.: Reputation and Social Network Analysis in Multiagent Systems. *In: Proc. of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, part 1, pp. 475–482. (2002)

[53] Shimojo, I., Tachikawa, T., Takizawa, M.: M-ary Commitment Protocol with Partially Ordered Domain. *In: Proc. of the 8th International Conference on Database and Expert Systems Applications (DEXA)*, pp. 397–408. (1997)

[54] Skeen, D.: NonBlocking Commit Protocols. *Proc. of the ACMSIGMOD International Conference on Management of Data*, pp. 133–142. (1981)

[55] Upadrashta, Y., Vassileva, J., Grassmann, W.: Social Networks in Peer-to-Peer Systems. *In: Proc. of the 38th Hawaii International Conference on System Sciences(HICSS-38 2005)*, CD-ROM. (2005)

[56] Montresor, A.: A robust protocol for building superpeer overlay topologies. *In: Proc. of the 4th International Conference on Peer-to-Peer Computing*, pp. 202–209. (2004)

[57] Two-phase commit protocol. http://en.wikipedia.org/wiki/Two-phase_commit_protocol.

[58] Upadrashta, Y., Vassileva, J., Grassmann, W., "Social Networks in Peer-to-Peer Systems", *In Proceedings of the 38th Hawaii International Conference on System Sciences(HICSS-38 2005)*, 2005.

[59] Qayyum, A., Viennot, L., Laouiti, A., "Multipoint relaying for flooding broadcast messages in mobile wireless networks", *In Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, 2002, pp.3866-3875.

[60] Ripeanu, M. and Foster, I., "Mapping Gnutella Network", *IEEE Internet Computing*, 2002, pp.50-57.