2007－DPS－130 (3)
2007－CSEC－ 36 (3)
2007／3／1

A Distributed Coordination Protocol for a Heterogeneous Group of Peer Processes

Ailixier Aikebaier†, Naohiro Hayashibara†, Tomoya Enokido††, and Makoto Takizawa†
†*Tokyo Denki University, Japan*, ††*Rissho University, Japan*
†{*alixir, haya, taki*}*@takilab.k.dendai.ac.jp*, ††*eno@ris.ac.jp*

In distributed applications like computer supported cooperative work (CSCW), multiple peer processes are required to cooperate to make a global decision, e.g. fix a date for a meeting of multiple persons. We discuss how multiple peer processes make a decision to achieve some objectives in a peer-to-peer (P2P) overlay network. Here, every process is assumed to be peer and autonomous. A domain of a process is a collection of possible values which the process can take. An existentially dominant relation shows what values a process can take after taking a value. In addition, values are also ordered in the preferential relation. Based on the existential and preferential relations, each process takes the most preferable value in the domain, which is dominantly preceded by the value v. In this paper, we discuss how every process makes an agreement on a tuple of values while each process can change the value according to the existential and preferential relations. In this paper, we discuss a coordination protocol in a type of heterogeneous system where every pair of processes have the same domain but may have different existential and preferential relations.

異種ピアプロセスグループのための分散協調プロトコル

エリシール アクパール†、林原 尚浩†、榎戸 智也††、滝沢 誠†
†東京電機大学 理工学研究科 情報システム工学専攻、
††立正大学 経営学部

コンピュータ支援協調作業（CSCW）のような分散アプリケーションでは、複数のピアプロセス (ピア) が意思決定を行うために協調動作する必要がある。本論文では、ピアツーピア (P2P) オーバレイネットワーク上で、複数の自律的ピアが何らかの目的を実現するために、どのように協調動作を行うかについて議論する。各プロセスの領域 (domain) は、プロセスが決定できる値の集合とする。各プロセスは、まず、領域内の値 v を決定し、他のプロセスに通知する。各プロセスは、他のプロセスからの値をもとに、値 v を他の値 v' に変える。ここで、値 v から v' に支配関係 (dominant relation) がある場合のみ、値を変更できる。さらに、値は嗜好優先関係 (preferential relation) によっても順序づけられる。これら２つの関係に基づいて、各プロセスは領域内で最も望ましい値を決定する。本論文では、２つの関係に従って各プロセスが値を変更することができる状況下で、あらゆるプロセスが値の組においてどのように合意をどのように取るかについて議論する。さらに、各プロセスが同じ領域を持ち、異なった支配関係、嗜好優先関係を持つという異種システムにおける協調プロトコルについて議論する。

1. Introduction

In distributed applications like computer supported cooperative work (CSCW) [3], a group G of multiple peer processes $p_1, \ldots, p_n$ are cooperating to achieve some objectives by exchanging messages with each other in peer-to-peer (P2P) overlay networks. For example, multiple peer processes fix a schedule of next month in a project. Thus, multiple processes are required to make an agreement in a group. In order to make an agreement, each process p_i initially takes a value v_i and notifies the other processes of the value v_i. A domain D_i of a process p_i is a set of possible values which p_i can take. The process p_i in turn receives values $v_1, \ldots, v_n$ from other processes in a group G. From a tuple $\langle v_1, \ldots, v_n \rangle$ of the values, a process p_i obtains one value v. For example, a majority value v in a tuple $\langle v_1, \ldots, v_n \rangle$ of values is taken. Protocols for making an agreement on a value are discussed in papers [2, 4] where each process p_i does not change the value v_i.

In the atomic commitment control on multiple database systems [1], there are one client process p_0 and multiple server processes $p_1, \ldots, p_n$. A process

p_i can take one value which is 0 (abort) or 1 (commit) in a binary domain $D_i = \{0, 1\}$. One coordinator process, i.e. client process p_0 asks every server process p_i to notify of a value, i.e. 0 or 1. Only if every process takes 1, every process agrees on the value 1. Otherwise, every process takes 0 even if some of them notifies 1. A process which makes a decision on the value 0 unilaterally aborts. However, a process which notifies the value 1 takes 0, i.e. aborts if the global decision of the coordinator process p_0 is 0. Thus, 0 is more *dominant* than 1 because a process notifying 1 may change the value with 0 but a process notifying 0 cannot change the value with 1.

In agreement procedures of our life, a person often changes the opinion after saying something to others so that every process can make some agreement in a cooperative society. In addition, a person cannot arbitrarily change the opinion but can change the opinion with ones depending on the previous opinion. That is, a current opinion of a person depends on the previous opinion. We define an *existentially dominant relation* $\preceq_i^E$ [5] on a domain D_i of each process p_i. In the commitment protocol, $1 \preceq_i^P 0$ as presented here. Furthermore, a person takes a value out of more than two values 0 and 1, i.e. the domain includes multiple values. In addition, each peer has preference on values in the domain. For example, a peer p_i can take any one of v_2 and v_3 after taking a value v_1, i.e. v_2 and v_3 existentially dominating v_1. Here, if the peer p_i *prefers* v_2 to v_3 ($v_3 \preceq_i^P v_2$), the peer takes v_2. Values in a domain D_i of each process p_i are partially ordered in the relations $\preceq_i^P$ and $\preceq_i^E$. Each process p_i is characterized in terms of the domain D_i and the relations $\preceq_i^E$ and $\preceq_i^P$, $p_i = \langle D_i, \preceq_i^D, \preceq_i^P \rangle$.

In this paper, we discuss a coordination protocol for a group of multiple peer processes to make an agreement on some values notified by the processes. A system is a set of processes. A some pair of processes in a heterogeneous system have different domains or different relations. In this paper, we discuss a coordination protocol in a type of heterogeneous system where each process has the same domain but may have partially ordered relations different from another process. Initially, each process does not know anything about the relations of every other process. A process p_i can learn which value dominates others and is preferred to a value in another process p_j through exchanging values with p_j. A process p_i can take one of possible values which more processes may be able to take by taking usage of knowledge about the other processes.

In section 2, we discuss a model of distributed coordination of multiple peer processes. In section 3, we discuss a basic coordination protocol. In section 4, we present a coordination protocol for a heterogeneous system.

2. Dominant Relations

2.1 E-dominant relation

A system S is composed of n (≥ 1) peer processes $p_1, \ldots, p_n$. Let P be a set $\{p_1, \ldots, p_n\}$ of the processes in the system S. Each process p_i takes a value v_i and notifies the other processes of the value v_i in the coordination protocol of the processes in P. A *domain* D_i of a process p_i is a set of possible values which the process p_i can take.

In the coordination protocol, 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 ($j = 1, \ldots, n, j \neq n$). The process p_i takes another value v_i^1 from a tuple $\langle v_1^0, \ldots, v_n^0 \rangle$ of values which p_i receives from the other processes. This is the first round. Then, p_i notifies the other processes of the value v_i^1. Thus, at the t^{th} round, p_i collects a tuple $v^{t-1} = \langle v_1^{t-1}, \ldots, v_i^{t-1}, \ldots, v_n^{t-1} \rangle$ of the values obtained. If the tuple v^{t-1} does not satisfy the agreement condition, p_i takes one value from the tuple v^{t-1} as an agreement value. Otherwise, p_i takes a value v_i^t in the domain D_i and notifies the other processes of the value v_i^t as shown in Figure 1.

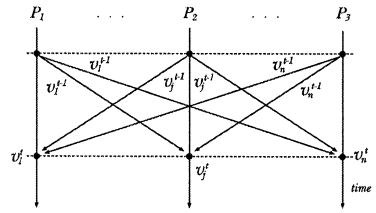

Figure 1. Round *t+1*.

In the commitment protocols [1,6], a process which notifies *commit* (1) may take *abort* (0) if the coordinator process indicates *abort*. Here, a process which notifies 0 cannot take 1. Thus, there are a subset $C(v)$ ($\subseteq D_i$) of values which each process p_i is allowed to take after taking a value v. Each process p_i can take a value v_i^t after v_i^{t-1} in the domain D_i if p_i can change the value v_i^{t-1} to v_i^t at round *t*. Here, p_i changes the opinion from the value v_i^{t-1} to v_i^t. If p_i cannot take any value from a value v_i^{t-1}, p_i still takes v_i^{t-1} as v_i^t. Here, a value v_{ik} is referred to as *existentially (E) precede* a value v_{ih} with respect to a process p_i ($v_{ik} \rightarrow_i v_{ih}$) if and only if (*iff*) the process p_i can change the value v_{ik} to v_{ih} ($\rightarrow_i \subseteq D_i^2$). In the commitment protocol, $1 \rightarrow 0$ but $0 \not\rightarrow 1$. In some agreement protocol, a process p_i cannot take any value which p_i has so far taken. Here, $v_{ik} \not\rightarrow_i v_{ih}$ if p_i had not taken v_{ih}.

There are two points on the transitivity of the existentially (E) precedent relation $\rightarrow_i$; $\rightarrow_i$ is transitive or not transitive. If the E-precedent relation $\rightarrow_i$ is not transitive, we introduce a transitively E-precedent relation $\Rightarrow_i$. A value v_1 *transitively E-precedes* another value v_2 in a domain D_i with respect to a process $p_i(v_1 \Rightarrow_i v_2)$ iff *a)* $v_1 \rightarrow_i v_2$ or *b)* $v_1 \Rightarrow_i v_3 \Rightarrow_i v_2$ but $v_1 \not\rightarrow_i v_2$ for some value v_3 in $D_i(\Rightarrow_i \subseteq D_i^2)$. Suppose that $v_1 \rightarrow_i v_2 \rightarrow_i v_3$ but $v_1 \not\rightarrow_i v_3$, i.e. $v_1 \Rightarrow_i v_3$. Here, the process p_i can take a value v_2 but cannot take a value v_3 just after p_i took a value v_1. In this paper, we assume the dominant relation $\rightarrow_i$ is transitive. Here, the process p_i can take v_3 only after taking v_2.

$v_1 \prec_i^E v_2$ if $v_1 \rightarrow_i v_2$ but $v_2 \not\rightarrow_i v_1$. A pair of values v_1 and v_2 are *existentially (E) equivalent* $(v_1 \equiv_i^E v_2)$ iff $v_1 \rightarrow_i v_2$ and $v_2 \rightarrow_i v_1$. A pair of values v_1 and v_2 are *existentially (E) independent* $(v_1 \|_i^E v_2)$ iff neither $v_1 \rightarrow_i v_2$ nor $v_2 \rightarrow_i v_1$.

[Definition] A value v_1 *dominates* a value v_2 in a process $p_i(v_2 \preceq_i^E v_1)$ iff $v_2 \prec_i^E v_1$ or $v_1 \equiv_i^E v_2$.

An E-dominant relation "$v_2 \preceq_i^E v_1$" means that a process p_i can take a value v_1 after taking a value v_2. A transitive E-dominant relation $\succeq_i^{E*}$ is defined as $v_2 \preceq_i^{E*} v_1$ iff $v_2 \preceq_i^E v_1$ or $v_1 \Rightarrow_i v_2$ for every pair of values v_1 and v_2 in D_i. If the E-precedent relation $\rightarrow_i$ is transitive, $\preceq_i^E = \preceq_i^{E*}$.

A domain D_i is partially ordered in the E-dominant relation $\preceq_i^E$. A *least upper bound* (*lub*) of values v_1 and v_2 $(v_1 \cup_i^E v_2)$ is a value v_3 in the domain D_i such that $v_1 \preceq_i^E v_3$, $v_2 \preceq_i^E v_3$, and there is no value v_4 such that $v_1 \preceq_i^E v_4 \preceq_i^E v_3$ and $v_2 \preceq_i^E v_4 \preceq_i^E v_3$. 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-dominant relation, $\preceq_i^E = \preceq_j^E = \preceq^E$ on the same domain, $D_1 = D_2 = D$. If there exists a *least upper bound(lub)* $v_3 = v_1 \cup^E v_2$, both the processes p_1 and p_2 can take the value v_3 after taking v_1 and v_2, respectively, i.e. make an agreement on v_3. A *greatest lower bound* (*glb*) of values v_1 and v_2 $(v_1 \cap_i^E v_2)$ is a value v_3 in D_i such that $v_3 \preceq_i^E v_1$, $v_3 \preceq_i^E v_2$, and there is no value v_4 such that $v_3 \preceq_i^E v_4 \preceq_i^E v_1$ and $v_3 \preceq_i^E v_4 \preceq_i^E v_{ih}$. The processes p_1 and p_2 can also take the *greatest lower bound* $v_4 = v_1 \cap^E v_2$ as an agreement value if the processes can take previous values again. In this paper, we assume there exist a pair of special values, *bottom* $\perp_i^E$ and *top* $\top_i^E$ where $\perp_i^E \preceq_i^E v$ and $v \preceq_i^E \top_i^E$ for every value v in the domain D_i. This means that a process p_i can take any value in D_i after taking the bottom value $\perp_i^E$. On the other hand, a process p_i taking the top $\top_i^E$ cannot change the value. In the commitment control protocol [6], each process p_i has a binary domain $D_i = \{0, 1\}$ where $1 \preceq_i^E 0$. Here, 0 is $\top_i^E$ and 1 is $\perp_i^E$.

A lattice $L_i = \langle D_i, \preceq_i^E, \cup_i^E, \cap_i^E \rangle$ is thus defined for each process p_i $(i = 1, \ldots, n)$. Figure 2 shows a Hasse diagram of the E-dominant relation $\preceq_i^E$ of a binary domain $D_i = \{0, 1\}$ in the commitment control. Here, a directed edge $\alpha \rightarrow \beta$ shows $\alpha \preceq_i^E \beta$. The value 0 E-dominates the value 1 in the domain $D_i = \{0, 1\}$.

[Definition] Let $\preceq_i^E$ and $\preceq_j^E$ be E-dominant relations of processes p_i and p_j, respectively. $\preceq_i^E$ and $\preceq_j^E$ are *existentially (E) consistent* $(\preceq_i^E \cong^E \preceq_j^E)$ iff for every pair of values v_1 and v_2 in $D_i \cap^E D_j$, $v_2 \prec_i^E v_1$ does not hold iff $v_1 \preceq_j^E v_2$.

$\preceq_i^E$ and $\preceq_j^E$ are *E-inconsistent* $(\preceq_i \ncong^E \preceq_j)$ iff $\preceq_i^E$ and $\preceq_j^E$ are not consistent. $\preceq_i^E$ is more *E-restricted* than $\preceq_j^E$ iff $\preceq_i^E$ and $\preceq_j^E$ are E-consistent $(\preceq_i \cong^E \preceq_j)$ and $\preceq_i^E \supseteq \preceq_j^E$.

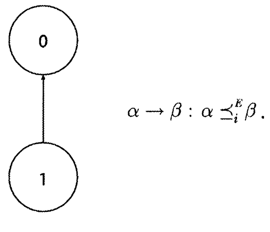

Figure 2. E-dominant relation.

Let us consider an agent-based auction system as an example. A system is composed of multiple processes each of which plays a role of an agent of a person. A process first shows proposed price for some goods. Then, each process obtains the maximum price among the price values from the other processes. A process showing the maximum price is referred to as *leading* process. The other processes are *secondary* processes. If a secondary process still would like to get the goods, the process *bids* more higher price than the maximum one. If a process would not like to get the goods, the process notifies the other process of *quit*. Here, if a process quits the auction, the process cannot join it again. If a process would like to just observe the auction for some time units and join it later, the process notifies the other processes of *listen*. If a *leading* process p_i showing the maximum price still shows "*bid*" and every other process *listens* or *quits*, the process p_i *bought* the goods. The domain D includes values { *bid, quit, listen, bought* }.

In the E-dominant relation $\preceq_i^E$ $(\subseteq D_i^2)$, a process p_i makes a decision on a value v' which p_i notifies to the other processes depending on a value v most recently taken. That is, p_i takes a value v' where $v \preceq_i^E v'$. Let $Corn_i^E(v_1)$ be a set $\{v_2 \mid v_1 \preceq_i^{E*} v_2\}$ of values which a process p_i can eventually take from a value v_1. Let $Next_i^E(v_1)$ be $\{v_2 \mid v_1 \preceq_i^E v_2\}$ of values which p_i can take next from v_1. $Next_i^E(v_1) \subseteq Corn_i^E(v_1)$ for every value v_1 in D_1. If $\preceq_i^E$ is transitive, $Next_i^E(v_1) = Corn_i^E(v)$. A *universal* domain D of a system S is a union of domains $D_1, \ldots, D_n$ of the processes $p_1, \ldots, p_n$, $D = D_1 \cup \ldots, \cup D_n$.

2.2 P-dominant relation

Suppose a process p_i can take a pair of values v_1 and v_2 after v_3 in the E-dominant relation $\preceq_i^E$, i.e. $v_3 \preceq_i^E v_1$ and $v_3 \preceq_i^E v_2$. The process p_i has to take one of the values v_1 and v_3. Here, if the process p_i prefers v_1 to v_2 ($v_2 \preceq_i^P v_1$), the process p_i first takes v_1. $\preceq_i^P$ is a *preferentially (P) dominant* relation on the domain D_i, $\preceq_i^P \subseteq D_i^2$. The least upper bound $\cup_i^P$ and greatest lower bound $\cap_i^P$ are defined for the P-dominant relation $\preceq_i^P$. There are special values, top $\top^P$ and bottom $\bot^P$ with respect to the P-dominant relation $\preceq_i^P$ in the same way as the E-dominant relation $\preceq_i^E$.

Let D be a domain $\{J$ (Japanese), C (Cantonese), S (Sichuan), U (Uyghur), I (Italian), F (French)$\}$, showing types of meals. A process p_i has the P-dominant relation $\preceq_i^P$ as shown in Figure 3. For example, $C \preceq_i^P S$, $C \cup_i^P F = I$, and $S \cap_i^P I = C$. Suppose a process p_i takes C and p_j takes F. The processes p_i and p_j have the same P-dominant relations $\preceq_i^P = \preceq_j^P = \preceq^P$. Here, $C \cup_i^P F = I$. The processes p_i and p_j can take I as an agreement value.

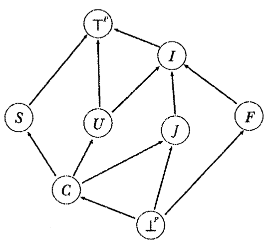

Figure 3. Hasse diagram of P-dominant relation.

As discussed, values are partially ordered in E- and P- dominant relations $\preceq_i^E$ and $\preceq_i^P$ in a domain D_i. A value v_1 *dominates* a value v_2 in a process p_i ($v_1 \succeq_i v_2$) iff the following conditions hold:

1. $v_2 \preceq_i^E v_1$.
2. $v_2 \preceq_i^P v_1$ if $v_1 \equiv_i^E v_2$.

The least upper bound $\cup_i$ and greatest lower bound $\cap_i$ are defined for the dominant relation $\preceq_i$.

2.3 Types of systems

Systems of processes $p_1, \ldots, p_n$ are classified into homogeneous and heterogeneous ones in terms of domains and relations of the processes. A system S of processes $p_1, \ldots, p_n$ is referred to as fully *homogeneous* iff $D_i = D_j$, $\preceq_i^E = \preceq_j^E$, and $\preceq_i^P = \preceq_j^P$ for every pair of processes p_i and p_j in S. A system is *homogeneous* iff $D_i = D_j$ and $\preceq_i^E = \preceq_j^E$ for every pair of processes p_i and p_j in S. Here, some pair of processes p_i and p_j may have different P-dominant relation $\preceq_i^P \neq \preceq_j^P$. Every process p_i can make the same decision $v_1 \cup_i \cdots \cup_i v_n$ on a tuple of values $v_1, \ldots, v_n$ received from the other processes. A system in the commitment control is *homogeneous*.

A system S is referred to as *heterogeneous* if $D_i \neq D_j$ or $\preceq_i \neq \preceq_j$ for some pair of processes p_i and p_j. For example, suppose a system S is composed of a pair of processes p_i and p_j. Here, the process p_i has a domain $D_i = \{a, b, c\}$ when $c \succeq_i b \succeq_i a$ and p_j has a domain $D_j = \{a, b, c\}$ where $b \succeq_j a$ and $c \succeq_j a$ in the system S. Here, the system S is heterogeneous since $D_i = D_j$ but $\preceq_i$ and $\preceq_j$ are inconsistent ($\preceq_i \not\equiv \preceq_j$). *Heterogeneous* systems are furthermore classified into *domain-homogeneous* (*DH*) and *order-homogeneous* (*OH*) heterogeneous types of systems. A system S is *domain-homogeneous* (*DH*) heterogeneous iff S is heterogeneous and $D_i = D_j$ for every pair of processes p_i and p_j. In the DH heterogeneous system, an existential (E-) dominant relation $\preceq_i^E$ may be inconsistent with another relation $\preceq_j^E$ even if $D_i = D_j$ for every pair of precesses p_i and p_j. The system S is DH heterogeneous since $D_i = D_j$. A system S is *order-homogeneous* (*OH*) heterogeneous iff S is heterogeneous but $\preceq_i^E = \preceq_j^E$ and $\preceq_i^P = \preceq_j^P$ for every pair of different processes p_i and p_j. A system S is *fully heterogeneous* iff $D_i \neq D_j$ and $\preceq_i \neq \preceq_j$ for some pair of different processes p_i and p_j.

Systems are also classified into *static* and *dynamic* types of systems. In a static system, each process p_i cannot change the domain D_i and E- and P-dominant relations $\preceq_i^E$ and $\preceq_i^P$. In a dynamic system, each process can change the domain and dominant relations. For example, a process p_i adds some values in the domain D_i and a new relation in a dominant relation. In this paper, we discuss homogeneous and DH heterogeneous systems which are static.

3. A Basic Coordination (BCoRD) Protocol

We discuss a basic coordination (BCoRD) protocol for multiple peer processes $p_1, \ldots, p_n$ to make an agreement. Let P be a set of processes $p_1, \ldots, p_n$. Each process p_i is characterized in terms of a lattice $L_i = \langle D_i, \preceq_i, \cup_i, \cap_i \rangle$ as discussed.

[Coordination Protocol BCoRD(P)]

1. Initially, $t = 0$ and $V_i = \langle \bot, \ldots, \bot \rangle$ for every process p_i.

2. One process sends a notification request (*val-req*) to every process p_i in the process set P.

3. On receipt of the notification request (*val-req*) from a process p_j, each process p_i takes a value v_i^t in the domain D_i, where v_i^t = $GD_i(\langle \bot, \ldots, \bot \rangle)$. Then, the process p_i sends the value v_i^t to all the other processes in the set P.

4. On receipt of a value v_j^t from a process p_j, a process p_i stores v_i^t in buffer V_i. If the process p_i receives values from all the processes $p_1, \ldots, p_n$, the process p_i does the following steps for a tuple $\langle v_1^t, \ldots, v_n^t \rangle$ in the buffer V_i:

   (a) If the agreement condition $AC_i(\langle v_1^t, \ldots, v_n^t \rangle)$ is satisfied, v_i^{t+1} = $GD_i(\langle v_1^t, \ldots, v_n^t \rangle)$. Here, the value v^{t+1} is a global decision. The process p_i terminates.

   (b) Otherwise, the process p_i takes a value v_i^{t+1} = $LD_i(\langle v_1^t, \ldots, v_n^t \rangle)$.
      - $t = t + 1$.
      - The process p_i sends the value v_i^{t+1} to all the other processes and goto step 3.

A predicate AC_i is the agreement condition on a tuple of values $v_1, \ldots, v_n$, AC_i: $D_1 \times \cdots \times D_n \rightarrow \{True, False\}$. Every process p_i has the same agreement condition $AC_i = AC$ 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 $p_1, \ldots, p_n$. Here, if the agreement condition AC_i $(\langle v_1^t, \ldots, v_n^t \rangle)$ is true, the coordination protocol terminates for a tuple $\langle v_1^t, \ldots, v_n^t \rangle$. For example, in the majority agreement, if there is a majority value v in the tuple $\langle v_1^t, \ldots, v_n^t \rangle$, the agreement condition AC_i $(\langle v_1^t, \ldots, v_n^t \rangle)$ is true.

A function LD_i is a local decision function which gives a value v_i^{t+1} in the domain D_i from a tuple $\langle v_1^t, \ldots, v_n^t \rangle$ of values, LD_i: $D_1 \times \cdots \times D_n \rightarrow D_i$. Here, $v_i^t \preceq_i^E v_i^{t+1}$. If there are multiple values which existentially dominates v_i^t, the process p_i takes one of them. One strategy is p_i takes the least preferable value in them. In a *homogeneous* system, $\preceq = \preceq_i$, i.e. $\preceq^E = \preceq_i^E$, $\preceq^P = \preceq_i^P$, and $D = D_i$ for every process p_i in the process set P. v_i^{t+1} is given a *least upper bound(lub)* $v_1^t \cup^E \cdots \cup^E v_n^t$ in every process p_i. In the order-homogeneous (OH) heterogeneous system, each process p_i can also take a *least upper bound* $v_1^t \cup_i^E \cdots \cup_i^E v_n^t$ since every process has the same dominant relation $\preceq$. Here, if there are multiple lubs $l_1, \ldots, l_m$ $(m > 1)$, $l_1 \cup_i^P, \ldots, \cup_i^P l_m$ is taken. On the other hand, in the other types of heterogeneous systems, $v_1^t \cup_i^E \cdots \cup_i^E v_n^t \neq v_1^t \cup_j^E \cdots \cup_j^E v_n^t$ for some pair of processes p_i and p_j since p_i and p_j have different E-dominant relations, $\preceq_i = \preceq_j$.

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

1. Atomic condition: AC_i $(\langle v_1^t, \ldots, v_n^t \rangle)$ = *True* and $v = GD_i$ $(\langle v_1^t, \ldots, v_n^t \rangle)$ if $v_1^t = \ldots = v_n^t = v$.
2. Majority condition: AC_i $(\langle v_1^t, \ldots, v_n^t \rangle)$ = *True* and $v = GD_i$ $(\langle v_1^t, \ldots, v_n^t \rangle)$ if $| \{v_i^t \mid v_i^t = v\} | > n/2$.
3. Consonance condition: AC_i $(\langle v_1^t, \ldots, v_n^t \rangle)$ = *True* and $v = GD_i$ $(\langle v_1^t, \ldots, v_n^t \rangle)$ if $v_j^t \neq v_k^t$ for every pair of different values v_j^t and v_k^t.
4. General condition: AC_i $(\langle v_1^t, \ldots, v_n^t \rangle)$ = *True*, if some condition defined by an application is satisfied for $\langle v_1^t, \ldots, v_n^t \rangle$.

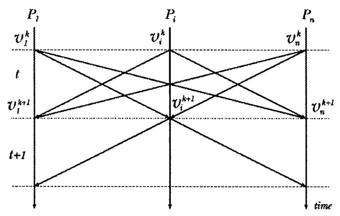

Figure 4. Rounds.

4. A Heterogeneous (HCoRD) Protocol

We consider a coordination protocol in a domain-homogeneous (DH) heterogeneous system S, where there are n (≥ 2) processes $p_1, \ldots, p_n$. In the paper, we assume every process p_i has no P-dominant relation $\preceq_i^P$ for simplicity, $\preceq_i$ means $\preceq_i^P$. The protocol is referred to as *heterogeneous coordination* (HCoRD) protocol. Here, each process p_i has a same domain D_i = $D = \{v_1, \ldots, v_m\}$ $(m \geq 2)$ while the dominant relation is not the same, $\preceq_k \neq \preceq_h$ for some pair of different processes p_k and p_h. Differently from a homogeneous system, a process p_i cannot take as an agreement value, the *least upper bound* $v_1^t \cup_i \cdots \cup_i v_n^t$ for a tuple $\langle v_1^t, \ldots, v_n^t \rangle$ of values received at round t because $v_1^t \cup_i \cdots \cup_i v_n^t \neq v_1^t \cup_j \cdots \cup_j v_n^t$ due to the difference of the dominant relations $\preceq_i^E \neq \preceq_j^E$ and $\preceq_i^P \neq \preceq_j^P$ for some pair of different processes p_i and p_j. We also assume that the system S is static, i.e. the domain D_i and the dominant relation $\preceq_i$ and $\preceq_i$ of each process p_i are invariant.

Initially, every process p_i does not know anything about the dominant relation $\preceq_j$ of another process p_j $(j \neq i)$. In the coordination protocol, the processes exchange values with each other. If a process p_i receives a value v_1 after another value v_2 from another process p_j, the process p_i perceives that $v_2 \preceq_j v_1$ in p_j. Thus, the process p_i learns the dominant relation $\preceq_j$ of another process p_j through communicating with p_j. The dominant 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 $\preceq_{ij}$ be a part of the dominant relation $\preceq_j$ which a process p_i knows, $\preceq_{ij} \subseteq \preceq_j$. That is, if a process p_i receives a value v_2 after v_1 from another process p_j, $v_1 \preceq_{ij} v_2$ in the process p_i.

First, 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 ($j = 1, \ldots, n$). A process p_i takes one value v_i^{t+1} such that $v_i^t \preceq_i^E v_i^{t+1}$, i.e. $v_i^{t+1} = LD_i(\langle v_1^t, \ldots, v_n^t \rangle)$ if the agreement condition $AC_i(\langle v_1^t, \ldots, v_n^t \rangle)$ 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 procedure $FFind_i$:

$FFind_i(\langle v_1^t, \ldots, v_n^t \rangle)$

1. If $Next_i(v_i^t) = \emptyset$, return (NULL).
2. Let I_i be a set $\{v \mid v \in Next_i(v_i^t)$, i.e. $v \succeq_i v_i^t$ and $v_j^t \preceq_j^E v$ for every value $v_j^t\}$ for a tuple $\langle v_1^t, \ldots, v_n^t \rangle$. Take a value v where $v' \preceq_i v$ for every value v' in I_i, return (v).
3. Otherwise, let J_i be a set $\{v \mid v \in Next_i(v_i^t)$ and $\mid \{p_j \mid v_j^t \preceq_{ij} v_i^{t+1}\} \mid$ is the largest$\}$. Take a value v where $v' \preceq_i v$ for every value v' in the set J_i, return (v). Find a value v_i^{t+1} in $Next_i(v_i^t)$ such that $\mid \{p_j \mid v_j^t \preceq_{ij} v_i^{t+1}\} \mid$ is the largest. If found, return (v_i^{t+1}).
4. Otherwise, the process p_i takes one value v_i^{t+1} in $Next_i(v_i^t)$, for example, such that $\mid Corn_i(v_i^{t+1})$ $\mid$ is the largest. return (v_i^{t+1}).

The procedure $FFind_i(\langle v_1^t, \ldots, v_n^t \rangle)$ takes a value v_i^{t+1} dominating the current value v_i^t. This is a *forwarding* strategy since we are always going up to upper bounds in the lattice L_i.

If a process p_i could not find a value $v_i^{t+1} = FFind_i(\langle v_1^t, \ldots, v_n^t \rangle)$, the process p_i takes a *backward* strategy. 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)* $v_i^t \cup_i v_j^t$. Here, the process p_i finds the *greatest lower bound(glb)* $v_i^t \cap_i v_j^t$. If a value $v = v_i^t \cap_i v_j^t$ is found in the domain D_i, the process p_i takes the value v, i.e. backwards to the value v in the lattice L_i. Then, the forwarding strategy is adopted as follows:

$BFind_i(\langle v_1^t, \ldots, v_n^t \rangle)$

1. If there is a value $v = v_1^t \cap_i, \ldots, \cap_i v_n^t$ in the domain D_i, return($FFind_i(\langle v_1^t, \ldots, v_{i-1}^t, v, v_{i+1}^t, \ldots, v_n^t \rangle)$).
2. Otherwise, find a value v such that $v \preceq_i^E v_i^t$ and $\mid \{p_j \mid v \preceq_{ij} v_j^t\} \mid$ is the largest. return($FFind_i(v_1, \ldots, v_n)$) where $v_j = v$ if $v \preceq_{ij} v_j^t$, else $v_j = v_j^t$.

5. Concluding Remarks

In this paper, we discussed coordination protocols for a group of multiple peer processes to make an agreement on a domain of multiple values. Each process p_i has a domain which is a set of possible values which p_i can take. Values in a domain are partially ordered in a pair of dominant relations, (E-) and (P-) ones. A process p_i can take a value v_1 after v_2 if and only if v_1 E-dominates v_2 ($v_2 \preceq_i^E v_1$). In addition, values are ordered in a preference of a process p_i. A process takes a preferable value. In a heterogeneous system, each process p_i has the same domain but different dominant relations than other processes. We discuss the heterogeneous coordination (HCoRD) protocol for a heterogeneous system of multiple processes. The coordination protocol can be adopted for agreement of multiple processes in various types of distributed applications.

Acknowledgment

This research is partially supported by Research Institute for Science and Technology [Q05J-04] and Academic Frontier Research and Department Center [16-J-6], Tokyo Denki University.

References

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

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

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

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

[5] I. Shimojo, T. Tachikawa, and M. Takizawa. M-ary Commitment Protocol with Partially Ordered Domain. *Proc. of DEXA '97*, pages 397–408, 1997.

[6] D. Skeen. NonBlocking Commit Protocols. *Proceedings of the ACMSIGMOD International Conference on Management of Data*, pages 133–142, 1981.