6F－6

A Decentralized Coordinator for Committee Coordination Problem*

Tongjun HUANG† Zixue CHENG†† Norio SHIRATORI†††

† Information Systems and Technology Center, The University of Aizu
†† Department of Computer Software, The University of Aizu
††† Research Institute of Electrical Communication, Tohoku University

Abstract

In this paper, we present a decentralized algorithm for extended committee coordination problem. Our algorithm can guarantees synchronous, exclusion, progress, individual preference and stable assignment properties. This algorithm can be used in group scheduling.

1 Introduction

Computer networks provide a variety of supports to the human society. On the other hand, activities of the human society is strongly dependent on the computer networks. This trend is advancing toward a new environment called *Autonomous Decentralized Social Environment*. An agent acts autonomously and follows social rules (norms).

The committee coordination problem is described by K.M.Chandy and J.Misra in [1]. In this paper, we consider an extended committee coordination problem in a decentralized social environment and present a solution to this problem. We extend the problem in the following way.

1. We relax the restriction as follows:
   Given a quorum for a committee, a committee meeting may be started only if the quorum is satisfied.

2. We introduce individual preference for every professor.

To solve this problem is not a trivial problem. The problem is to devise a decentralized social algorithm which decides which meetings should be started in order to guarantee synchronous, exclusion properties, as well as individual preference properties. The decision made by the algorithm should be stable and make progress. We will discuss these properties in detail in the next section.

*"A Decentralized Coordinator for Committee Coordination Problem" by Tongjun HUANG(The University of Aizu), Zixue CHENG(The University of Aizu), Norio SHIRATORI(Tohoku University)

2 Problem Description

2.1 The Description of the Problem

Let $\mathbf{P} = \{p_1, p_2, ..., p_n\}$ be a set of **n** professors and $\mathbf{C} = \{c_1, c_2, ..., c_m\}$ be a set of **m** committees. Each professor participates in one or more committees. The set of committees in which a professor p_i participates is called his/her *committee-set* and is referred to as C_i. Thus $C_i \subseteq \mathbf{C}$. A totally ordering "$\leq$" is defined on the elements of C_i, which represents p_i's preference of committees. Each committee, say c_k, is associated with a set of one or more professors; this set is called the *professor-set* of the committee and is referred to as P_k. Thus $P_k \subseteq \mathbf{P}$. The professor-set of a committee and the committee-set of a professor are both static sets.

A professor is either idle or busy. A busy professor autonomously makes a transition to become idle. An idle professor p_i starts waiting and remains waiting until a meeting of a committee in C_i is started.

Let q_k be *quorum* of committee c_k. The conditions on meetings are as follows: (1) a committee meeting may be started only if at least q_k members of that committee are waiting and (2) no two committees may meet simultaneously if they have a common member and the first condition cannot be satisfied without participation of the common member.

Every professor in the system satisfies the following conditions:

(a) An idle professor p_i remains idle until a committee meeting in his/her C_i is started.

(b) An idle professor p_i becomes busy if a committee meeting in his/her C_i is started.

A committee c_k is either enabled or disabled. Committee c_k is enabled if and only if at least q_k professors belonging to P_k are idle; c_k is disabled if there exist more than $|P_k| - q_k$ professors that belong to P_k are busy.

We use the phrase "a committee meeting c_k is started" to mean that each professor that belongs to P_k is idle and has decided to attend c_k. We are required to devise an algorithm which allows an idle professor to attend an enabled committee such that the following properties are satisfied.

[Property 1] Synchronous property:
A disabled committee can not be started.

[Property 2] Exclusion property:
No two committees may meet simultaneously if they have a common member and condition (1) above cannot be satisfied without the participation of the common member.

[Property 3] Progress property:
If there are enabled committees, at least one c_k of them will be started by not less than q_k professors of its professor-set.

[Property 4] Individual preference property:
If a professor has more than one enabled committee and whatever the professor selects, the progress property can be guaranteed, then the professor can select one enabled committee according to his/her preference.

[Property 5] Stable assignment property:
If p_1 prefers c_1 to c_2, p_2 prefers c_2 to $c_3, \cdots, p_{h-1}$ prefers c_{h-1} to c_h, and p_h prefers c_h to c_1, it never happens that c_2 is assigned to p_1, c_3 is assigned to p_2, c_h is assigned to p_{h-1}, and c_1 is assigned to p_h.

3 Outline of the Algorithm

The *decentralized social algorithm* solves the extended committee coordination problem, defined in Section 2, in an autonomous decentralized social environment.

3.1 Basic ideas in the algorithm

Each committee, say c_k, is associated with a set of professors P_k, which participates in c_k. We assume that for every professor-set there is a coordinator.

An idle professor in a professor-set P_k sends a message WAIT to the coordinator of committee c_k before t_1. Here we assume t_1 to be a deadline. The coordinator judges whether at least q_k professors in the professor-set P_k have sent WAIT messages, i.e. the coordinator checks whether the committee is enabled at $t_1 + t_0$. Here, t_0 is the upper bound of a transmission delay on a link. Then the coordinator informs all professors in the professor-set of the judgement by sending messages ENABLE or DISABLE.

When there is a professor participating in more than one committee, the professor can select the most preferable committee from the enable committees.

To maintain exclusion property, after selecting a committee, a professor needs to let other professors in the committees, of which the professor is a member, know of his/her selection. Thus, the professor sends a massage SELECTED to the coordinator of the selected committee. The SELECTED have to be sent before t_2, where t_2 is another deadline and is larger than $t_1 + 2t_0$.

A coordinator checks whether the committee has been selected by at least q_k professors and sends DO, UNDO or NOTMEET to the professors of the committee at $t_2 + t_0$.

If a message DO is received, a professor sends a message UNSELECTED to every coordinator of the committees which are not selected and then attends the committee meeting. If a message UNDO is received, a professor selects the second most preferred committee from the enable committees and tries again. This process is repeated until a DO message is received or all enable committees are tried out.

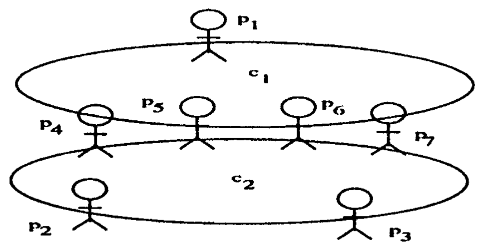

Figure 1: An example of deadlock

If every professor selects a committee according to his/her preference, then a deadlock may happen. An example of deadlock is shown in Fig. 1. There are seven professors organized into two committees. Suppose that the q_1 and q_2 are 4 and 5, respectively. If professor p_4 and p_5 prefers c_1 to c_2, professor p_6 and p_7 prefers c_2 to c_1, and all of them select committees according to their preference, then both the meetings cannot be started, since no committee can have enough members to attend.

In the example, p_4, p_5, p_6, and p_7 will receive an UNDO message. If all these professors reselect the second most preferred committees, the deadlock cannot be recovered. Thus,we use the *"committee identifier"* to break the tie. Let every committee have a unique identifier. Professor p_i will reselect the same committee c_k if committee c_k first selected has the smallest identifier in C_i.

4 Conclusion

In this paper, We first extend the committee coordination problem by introducing some social properties, such as individual preference and stable assignment and give a decentralized social algorithm as a solution to this extended problem. The algorithm guarantees not only the synchronous and exclusion properties but also individual preference and stable assignment properties.

The extended problem models many paradigms of synchronization and mutual exclusion among groups of people and/or agents. More applications such as group scheduling support and multi-group decision support are required in real life. Our solution will be useful for this purpose.

References

[1] K.M.Chandy and J.Misra, Parallel Program Design:A Foundation, Reading, MA: Addison-Wesley, 1988.