A Framework for Highly Reliable Mobile Agents

Kenji Shinbori

School of Information Science,
Japan Advanced Institute of Science and Technology

February 15, 1999

Keywords: Mobile Agent,fault recovery,non-Byzantine fault,Event,CheckPoint.

1 Purpose and background

The purpose of this study is to construct a framework for highly reliable mobile agents. In this paper, we use the term *reliability* to denote the characteristics of mobile agents that they can cope with physical faults in the computer environments. When we use mobile agent technologies in real-world applications, we must consider both security and dependability of agents. In this paper, we handle the dependability. The mobile agents migrate to the hosts, which are connected to the network, and do work in the various computer environments. As for such the mobile agents must cope with the communication faults and the hosts down. However, the mobile agents, who made by the existing mobile agent systems, have not considered the design about the computer environment faults. When the faults occur in the computer environments, the mobile agents can't complete their own purpose. For example, 1) When the host down by the battery cutting, the mobile agents become extinct. 2) When the mobile agents can't get the computer resources that should be necessary, the mobile agents can't continue to processes. As seen from these examples, dependability is necessary and important. When the faults occur in the computer environments, we must propose a framework for highly reliable mobile agents, which can detect the faults and cope with it, to solve such those problems. In this research, the resource starvation problem of the mobile agents is handled as a fault, too. The security of agents should not be ignored, but it is omitted in this research.

2 Coop

We designed a framework for highly reliable mobile agents called *Coop*. Coop is a framework that the mobile agents can detect the non-Byzantine physical faults and cope with it. We show the faults that Coop handles in the following. 1) When the hosts down happened. 2) When the mobile agents couldn't use the computer resources (CPU, memory and so on) which being expected in the place of the migration. 3) When the hosts disconnect from the network. 4) When the mobile agents can't get the result in the time. The mobile agents based on Coop cope with those faults in the following. In the case of one, we are thinking about two kinds of the expectably host down or the non-expectably host down. When the mobile agents could expect it, it is in such cases as the battery cutting, UPS (uninterruptive power supply) doing, detachable shutdown and so on. At this time, the mobile agents migrate to the hosts or permanent in the secondary storage. When the mobile agents couldn't expect it, it is in such cases as the host down by the blackout and so on. At this time, the mobile agents cope with the fault by the checkpointing. The mobile agents do the checkpointing to create a representation of the agent's state in the secondary storage at an interval of the constant time. After the fault recovery of the host, the mobile agents can resume from the checkpoint (the recorded execution states). In the case of two, the mobile agents migrate to the another host or waits for the release of the computer resources and so on. In the case of three, we are thinking about two kinds of the communication faults; it is before the migration or migrating. The former of the fault recovery is that the mobile agents change the host of the migration or wait until the communication fault is recovered. The latter of the fault recovery is that the mobile agents make the replica in the place of sender and migrate to the host repeatedly. In the case of four, the mobile agents can't resume permanently; the reason of the host down isn't recovered, the execution states are broken and so on. At first, the mobile agents make replica in the place of the host of the agent's owner, after that the mobile agents migrate to another host again.

The architecture of Coop is composed by two of a) the recovery mechanism and b) the control module. Each mobile agent has the fault recovery mechanism and migrates to each host. The fault recovery mechanism is divided into the fault recovery methods and the event module. The fault recovery methods cope with the fault that these are defined in the agents. The event module watches the environment of the mobile agent. When an event (fault) occurs, the event module calls a fault recovery method corresponding to the fault. We design the fault recovery mechanism as the architecture that flexibility customization is possible. Actually, the fault recovery mechanism, which is only added to the existential mobile agent systems, can't cope with all the faults. For example, the mobile agents can't activate themselves from the permanent agents that recorded in the secondary storages. Therefore, it is necessary for mobile agents to activate from the permanent agents. The control module, which handles more than one mobile agent, is the stationary agent and exists toward one node.

3 implement

In this research, we designed Coop, which is a Java class library, and implemented it. The exercises, which we actually made, are the system information's acquire agent and the circulating notice agent. The system information's acquire agent acquires each hosts system information and informs the system administrator it. The circulating notice agent realizes the circulating notice of the actual world. In addition, we designed a simple distributed algorithm, which can cope with the crash fault, only for the cyclic mobile agents. We implement this distributed algorithm for the components. The programmer can choose whether this distributed algorithm is used freely. This algorithm is being used with the circulating notice agent. The experiment platform of Coop uses Satou's AgentSpace, which is the mobile agent system of the Java based. Coop is implemented by using Java, but we uesed the native method at the point of the event detection, so the executive environment is only Windows 95,98,NT4.0. Actually, Coop itself is platform independence. We used Java on Win32 for the experiments in this research.

4 Summary

We designed Coop — a framework for highly reliable mobile agents —and implemented it as a Java class library. Advantages of Coop are the following. 1) The architecture of Coop doesn't depend on the platform of the mobile agent systems. 2) It is the architecture that the customization of the fault recovery methods and the event modules is easy. 3) Because the recovery mechanism is independent, it can be handled as a module. About the problem of the mobile agents based on Coop, the load of event watching is high; each mobile agent has an event module and watches the events. However, the mobile agents with the event module can customize suitable for their own purpose. About this problem, We consider that it is trade-off of the flexibility of Coop.