ABSTRACT
Effective fault-handling in emerging complex applications in large-scale MAS (Multi-agent Systems) requires the ability to dynamically adapt resource allocation and fault tolerance policies in response to changes in environment, user or application requirements, and available resources. This adaptation process incorporates an observation mechanism that transparently monitors the application’s behaviors as well as the availability of resources, and adaptively reconfigures the system resources. This process is realized by a specific module which exploits the information resulting from monitoring. In this paper, we present an approach for adaptive replication. This approach uses an observation mechanism and a feedback control system within an adaptive replication infrastructure to support adaptive fault tolerance in multi-agent organizations. The main strategy used in our approach is to insert control theory methodology and analysis to adaptive replication. Thus, our approach provides a systematic and scientific method for implementing adaptive fault tolerance policies in MAS.

Categories and Subject Descriptors
I.2.11 [Distributed Artificial Intelligence]: Multi-agent systems

General Terms: Reliability, Tools, Experimentation.

Keywords

1. INTRODUCTION
Agents are distributed on the network, and operate in open environments where available resources, load and failures are difficult to predict. Monitoring and feedback control are needed to acquire information from the system level and agent level, and exploit it to achieve the required performance level for fault tolerance of multi-agent organizations.

In this paper, we propose an approach that uses an observation mechanism which acquires the information related to the criticality of agents that can be gathered from application level such as roles of agents and system level such as communication load, failure rates. Then, this information for each agent is processed in a feedback control system in which P (Proportional) is chosen as the control function. The output of the feedback control system which is a normalized criticality value for each agent is sent in a content of FIPA message to each agent in a certain period. An agent applying the adaptive fault tolerance policy decides about its replication degree based on this information by executing the adaptive fault tolerance plan structure.

The observation mechanism and the feedback control mechanism were implemented in Agent Communication Channel (ACC) module of SEAGENT (A Semantic Web Enabled Multi-Agent Framework) agent development framework which is developed based on the FIPA’s agent architecture [1]. Adaptive fault tolerance policy is applied to the replication group by simply including the implemented observation and feedback control mechanism.

In the wider context of fault tolerance, fault tolerance approaches based on replication techniques have been applied in our approach. In order to implement replication techniques in our work, we have investigated several replication based fault tolerant distributed systems. The Advanced Automation System (AAS) [4] is a distributed real-time system that integrates all the services of the US air traffic control network. The high availability requirements were fulfilled by introducing both hardware and software redundancy. Critical services are replicated using either the active or the passive approach, according to the application semantics and the hardware configuration. Consul [5] is a collection of protocols that provides low-level mechanisms required for the development of fault tolerant distributed applications according to the State Machine approach. Psync is a reliable broadcast protocol that maintains the causal order of messages exchanged between group members and supports Consul. The Manetho protocol [6] is a combination of a rollback-recovery protocol and a process replication protocol based on the primary backup approach. Simple ad hoc broadcast and membership algorithms are used for this purpose. Manetho
performs very well in environments with low failure probability. It introduces minimal delays due to replication. However, the price for this is a considerable overhead in the control information.

There are also some other works done to design adaptive fault tolerance systems. Bondavalli et al. propose a framework for adaptive fault tolerance in a real-time context [8]. Their work explicitly addresses the real time constraints and employs a flexible and adaptable control strategy for managing redundancy within application software modules. Programmers can specify fault tolerance strategies for the application modules, including adaptive strategies taking into account available resources, task importance, deadlines and observed faults. Hiltunen and Schlichting have introduced a general model for adaptive systems and presented examples of how this model can be applied for different scenarios in the context of distributed system [9]. AFTM [7] is an adaptive fault tolerant middleware, which uses a CORBA-compliant object request broker. In AFTM the most suitable fault tolerant and resource allocation scheme is selected dynamically through user requests and the parameters in its three databases. Its services provide automatic reconfiguration of the groups, transparently masking faults from the users.

All related works mentioned above, provide either static fault tolerance or adaptive fault tolerance to the distributed or real time systems. The main difference presented work in this paper from those works is that the components of the infrastructure that provides fault tolerance to a system are implemented in a multi-agent development framework to improve fault tolerance in multi-agent system organizations. In addition to this, comparing to the systems that apply adaptive fault tolerance, the adaptation mechanism in our approach is based on the feedback control which is very similar to the mechanism used in feedback control real-time scheduling proposed by Stankovic et. al [10].

There are also well-known fault tolerance approaches based on replication techniques for multi agent systems. In order to increase fault tolerance in MAS, Fedoruk and Deters implemented transparent replication via proxies [11]. The proxy as an interface handles all communication between replicas and other agents in the MAS. The proxy also controls execution in a replica group and state management of a replica group. Although this proxy approach handles fault tolerance issues in a multi-agent system, proxy itself is a single point of failure. There is no recovery mechanism introduced in this work when the proxy fails. They chose FIPA-OS agent toolkit as a platform for their implementation. Since FIPA-OS does not provide any replication mechanism, the replication server is implemented as a standard FIPA-OS agent. Moreover, this approach does not support the idea of changing fault tolerance policies at run-time. Therefore, replication is realized by a programmer before the application starts.

Guessoum et al. present an adaptive multi-agent architecture with both agent level and organization level adaptation [12, 13]. The organization’s adaptation is based on the monitoring of the system’s behavior. The architecture was implemented with the DIMA [14] platform and the DarX middleware [15]. In DarX, software components can be either replicated or un-replicated, and it is possible to change the replication strategy at run time. Although we use the same techniques to implement fault tolerance policies within the organization, the main difference of our approach from this work is that we try to improve adaptive fault tolerance policies by using the feedback control mechanism within the adaptation infrastructure.

The remainder of this paper is structured as follows: Section 2 presents an abstract architecture for adaptive replication for MAS organizations, Section 3 briefly introduces SEAGENT overall architecture and describes how to support the development of fault tolerant multi-agent system in SEAGENT; Section 4 presents how to embed the observation mechanism; into SEAGENT; Section 5 presents the implementation details of the feedback control mechanism and how to realize adaptive replication; Section 6 gives the case study and finally Section 7 gives the conclusion.

2. AN ABSTRACT ARCHITECTURE FOR ADAPTIVE REPLICATION

In this section, we briefly describe the proposed abstract architecture for adaptive replication to support adaptive fault tolerance in a MAS architecture. This architecture is illustrated in Figure 1 and build on the FIPA based MAS architecture.

Effective fault-handling in large-scale MAS requires the ability to dynamically adapt resource allocation and fault tolerance policies in response to changes in environment, and available resources. In order to achieve effective fault handling, we have to be aware of the environment and collect the data to adapt the agent organizations.

In order to collect the data, we associate an observation mechanism to ACC (Agent Communication Channel) module which is defined in FIPA specifications since all FIPA messages are received by ACC module and then forwarded to the receiver agents. The collected data will then be processed in a feedback control mechanism in order to compute the agent’s criticality. The feedback control mechanism is based on the classical control theory and the function of this mechanism is chosen as $P$ (Proportional). The output of the feedback control mechanism is sent as a content of a FIPA message to the leader (there is a centralized agent that controls all the replicas of itself) agent(s) applying the adaptive fault tolerance policy. On the left part of the Figure 1, the internal architecture of an agent applying the adaptive fault tolerance policy is shown. The necessary services to improve fault tolerance such as the membership service, the group communication service in an organization are embedded into the agent internal architecture of SEAGENT’s layered architecture. The replication service is placed over the layer including internal architecture of the agent in SEAGENT platform. The agents pass the message received from the feedback control mechanism to the replication service implemented as a reusable plan structure. Replication Service of the agent replicates new replicas or removes the replicas with the respect to the content of the

![Figure 1: An Abstract Architecture for Adaptive Replication in Fault Tolerant MAS organizations](image)
received FIPA message. Replication Service uses membership service and group communication service implemented in the agency to perform removing and replicating operations [16].

In the next section, we present implementation of services for fault tolerance over SEAGENT framework.

3. SEAGENT ARCHITECTURE

In this section, we introduce briefly SEAGENT’s layered software architecture to explain how adaptive replication scheme is integrated to SEAGENT architecture [1, 16]. Each layer and packages of the layers in SEAGENT platform have been specially designed to provide built-in support for semantic web based multi agent system development. SEAGENT platform architecture is shown in Figure 2. The bottom layer of the platform architecture is responsible of abstracting platform's communication infrastructure implementation. SEAGENT implements FIPA's Agent Communication and Agent Message Transport specifications [2] to handle agent messaging. SEAGENT’s communication layer also supports multicasting. This feature is used for implementing group communication service. Multicasting is a strategy for reliable communication including a mechanism for handling send omission failures. The essential feature of multicast communication is that an agent issues only one “send” operation instead of issuing multiple “send” operation to individual agents. Therefore, it provides stronger delivery guarantees.

The second layer of SEAGENT architecture includes packages, which provide the core functionality of the platform. The first package, called as Agency, handles the internal functionality of an agent. Agency package supports the creation of general purpose and goal directed agents. In order to provide fault tolerance, some services such as group communication service, group membership service, are implemented within the agency package. Multicasting of group communication service is basically supported SEAGENT’s communication infrastructure. However, this basic multicasting mechanism only delivers requests in an arbitrary order. An ordering guarantee in multicasting must be supported for developing fault tolerant systems because consistency and coordination between the replicas is constructed by performing incoming requests in an order. Total ordering scheme is one of the schemes providing ordering guarantee and implemented in the agency package to provide a multicasting mechanism of group communication service.

Group membership service is also implemented within the agency package. A group membership service is responsible for providing an interface for group membership changes by adding or removing an agent to or from a group, monitoring the group members by using a failure detector mechanism. A failure detector mechanism based on unreliable failure detector mechanism is also implemented within the agency package.

The second package of the Core Functionality Layer includes service sub-packages, one for each service of the platform. SEAGENT provides all standard MAS services such as Directory Facilitator (DF) Service and Agent Management Service (AMS) following the previous platform implementations and FIPA standards.

Third layer of the overall architecture includes pre-prepared generic agent plans. We have divided these generic plans into two packages. Generic Behavior package collects domain independent reusable behaviors that may be used by any MAS. On the other hand, Generic Semantic Behaviors package includes only the semantic web related behaviors.

In our approach, the replication service is implemented as reusable plans in the layer consisting generic behaviors plans. The replication service is responsible for creating new replicas and applying different fault tolerance policies such as static and adaptive fault tolerance policies. This service must provide some sub-services such as cloning, leader election, and uses the other services such as the membership service and the group communication service to achieve its purposes. Internal mechanism of the replication service changes depending on the applied policies such as static and adaptive fault tolerance policies. The replication service is implemented as a reusable plan in SEAGENT agent platform. Therefore, this makes our agents flexible in terms of fault tolerance since it is possible to easily modify existing plans, remove some of plans, or include new plans.

Throughout this paper, we assume that the system is in an asynchronous environment and subject to message omissions; agent crashes (fail-silent), and network partitions. We also assume no message corruptions and no malicious faults. We note that SEAGENT platform itself is not fault tolerant against its components failures. It only supports developing fault tolerant multi-agent organizations.

In SEAGENT, each agent can be replicated many times and with different replication strategies. SEAGENT provides the group membership service, the group communication service and the replication service to the groups. Each replication group has only one leader which coordinates the replica group and communicates with the other agents. When the leader fails, a replica is selected as a new leader in the replica group.

Due to the space limitation of the paper, the integration of the membership service, group communication and replication strategies to the agency package will not be presented here. Next Section describes how the observation service which implements monitoring and the feedback control mechanism are embedded to the SEAGENT’s communication infrastructure.

These modules are explained in the following section in details.
4. MONITORING

Monitoring is necessary for acquiring information to determine the criticality of agents in adaptive fault tolerance policies. The information is acquired from either the system-level information such as communication load, processing time etc. or application level information such as the importance of messages, the roles of the agents etc. [12]. In our approach, the number of requests receiving by an agent, message sizes, fault rates and the role of an agent which captures the importance of the agents are the sources of information. The information related to the communication load of a specific agent, the role of this agent, fault rates of the group that belongs to the agent are used to dynamically compute the agent’s criticality.

The agent’s criticality is a very important in terms of fault tolerance, because the agent’s criticality shows the reliance of other agents on a specific agent and how important an agent is. Therefore, if a critical agent fails, the other agents, which rely on the critical agent, will struggle to achieve their individual goals.

Monitoring is achieved via an observation mechanism. Next section presents the observation mechanism.

4.1 The Observation Mechanism

The observation mechanism is responsible for monitoring the system-level information such as the number of messages, message sizes, and fault rates and the application-level information such as the role concept in our approach. Therefore, we implemented this mechanism in the Agent Communication Channel Module (ACC) of the communication infrastructure. All FIPA messages are received by ACC module and then forwarded to the receiver agents. Since all system-level information and application-level information can be acquired in this module, we prefer to modify this module.

In ACC, we implemented a data structure which stores data about each individual agent who receives requests for a preset period. This data collection period is set during the initialization of an organization. The data consists of the agent’s name and address, the value of the role of the agent, the number of requests and the total size of messages received by the agents, the number of faults in the replica group which belongs to the agent, its normalized information and criticality from the previous period. The information in the data structure is updated for every new request and every new fault report received by ACC, and every message regarding the change of the role of the agent. In addition to this, when a new request is received by ACC module to forward the message to the receiver agent, it increases the total number of requests and the total size of messages that are sent in the multi-agent system.

5. A FEEDBACK CONTROL MECHANISM FOR ADAPTIVE REPLICATION

A typical feedback control system includes a controller, a plant to be controlled, actuators, and sensors. It defines a controlled variable, the quantity of the output which is measured and controlled. The set point is the correct value of the controlled variable. The difference between the current value of the controlled variable and the set point is the error. The manipulated variable is the quantity that is varied by the controller so as to affect the value of the controlled variable.

The adaptation mechanism in our approach utilizes a feedback control technique to achieve satisfactory system performance in replication in spite of unpredictable system dynamics such as communication load, behaviors of agents, and failure rates. We believe that feedback control theory will improve performance of adaptive fault tolerance policy in the presence of failures.

To apply feedback control mechanism in replication, we need to identify the controlled variable, the manipulated variable, the set point, the error, and the control function. The number of replicas in a replica group in an organization is defined as the controlled variable in our approach. The manipulated variable is the fault rate. The set point will be chosen by the programmer since the difference between the manipulated variable and the value of the set point (the error) has an effect on defining the replication degree. In our approach, the set point is defined as the half of the replication degree. The architecture for feedback control adaptive replication is illustrated in Figure 3.

Figure 3: The Architecture for Feedback Control Adaptive Replication

In this architecture, during system initialization, a period is set for the organization and a timer module is implemented in ACC. This period is called the sampling period T and actually defined over a time window ((k-1)T, kT), where k is the sampling instant. Task of the timer is to monitor this period and calculate the criticality of each agent by using the following formulas:

\[
\text{Ratio}(k) = \frac{[a \cdot \text{role}(k) + b \cdot (\text{req_no}(k))]}{(t \cdot \text{req}(k))} \frac{[c \cdot (\text{msize}(k))]}{t \cdot \text{msize}(k)}
\]

(1)

\text{Ratio}(k): This is the degree of an agent’s activity in a MAS organization at the \( kth \) sampling instant. Since we can not observe agents’ internal states, we should observe the agent’s behavior and agent’s interaction events to calculate the degree of the agent’s activity in a MAS organization. The \text{Ratio}(k) is then be used to calculate the agent’s criticality at the \( kth \) sampling instant and it is the disturbance input of the feedback control mechanism.

\text{role}(k): The value corresponding to the agent’s role at \( kth \) sampling instant. The role is a variable related to the responsibilities of an agent and reasons of its behaviors. The role for an agent is set before the program starts and can be changed during the system operation.

\text{treq}(k): The number of total requests that are sent in a multi-agent system over a time window ((k-1)T, kT);

\text{req_no}(k): The number of requests that are sent to the individual agent over a time window ((k-1)T, kT);

\text{tsize}(k): The total size of messages received by the agents in a multi-agent system over a time window ((k-1)T, kT);
msize \((k)\): The total size of messages received by an agent over a
time window \((kT, kT + T)\);
a, b, c: Coefficients for contributions of the role, and the number of
requests and message sizes to the Ratio.
In calculation of Ratio\((k)\), the role concept and communication
load are considered. The fault concept will be included to the
Ratio\((k)\) after information regarding the fault will be collected and
processed in the controller.
The data related to the fault rates is derived from the failure
reports sent by the failure detectors at every sampling period. If
this data for the replica group of an agent is not larger than the set
point, there is no point worrying about the system’s availability.
In this case, the gain of the controller \((C_p)\) is set to zero. Otherwise,
the data related to the faults is processed in the controller.
The following formulas are used in the controller:
\[
\Delta fault\,(k) = fault\,no\,(k) - replica\,no\,(k)/m \tag{2}
\]
\(fault\,no\,(k)\): The number of faults which is reported by the failure
detector in a replica group in the \(k\)th sampling window;
\(replica\,no\,(k)\): The number of replicas in a replica group in the \(k\)th
sampling window;
m: A coefficient set by the programmer, which rate of the
contribution of faults is allowed.
\(\Delta fault\):(k)\: The difference between \(fault\,no\,(k)\) and \(replica\,no\,(k)\)
which is called error of the feedback control mechanism. If the
value of \(\Delta fault\) is larger than 0, this means that the contribution of
the fault rate to the criticality should be increased. Therefore, the
controller is activated. Proportional gain of the controller is
restricted by the following:
\[
0 < C_p < \frac{(4 \times Tot\,W\,(k-1))/RM}{(3)}
\]
\(C_p\): Proportional gain of the controller. Based on the above
inequality, \(C_p\) should be chosen between the bounds defined by
the inequality. We derive this value from the stability analysis of
the feedback control mechanism. According to control theory, a
system is stable if and only if all the poles of its transfer function
are in the unit circle of \(z\)-plane.
\(RM\): The number of available resources;
\(Tot\,W\,(k-1)\): Total criticality of all agents in a multi-agent system
for the previous sampling period;
\[
fault\,'\,(k) = \Delta falt\,(k) \times C_p \tag{4}
\]
fault\,'\,(k): The error with gain and the output of the controller.
\[
F\,(k) = fault\,'\,(k) / n \tag{5}
\]
n: A coefficient set by the programmer.
\[
\Delta Ratio\,(k) = F\,(k) + Ratio\,(k) - Ratio\,(k-1) \tag{6}
\]
\(\Delta Ratio\,(k)\): The change in the criticality of an agent in the \(k\)th
sampling window.
\(F\,(k)\): The contribution of the fault rate in the change of the
criticality of an agent in the \(k\)th sampling window.
\(Ratio\,(k)\): Ratio determined in the recent sampling period;
\(Ratio\,(k-1)\): Ratio stored in the data structure;
The formula for the actuator of the feedback control mechanism is
given below:
\[
W\,(k) = W\,(k-1) + \Delta Ratio\,(k) \tag{7}
\]
\(W\,(k)\): New criticality value to be applied in the next period;
\(W\,(k-1)\): Old criticality of an agent stored in the data structure and
determined in the \((k-1)\)th sampling period;
Before the \((k+1)\)th sampling period begins, \(t\,req, t\,msize, and
req\,no\) and \(m\,size, fault\,no\) are set to zero for each agent, then
data structure is updated with \(req\,no, m\,size, fault\,no\,Ratio\,(k)\);
\(W\,(k)\). In the next step, we determine the normalized criticality as
follows:
\[
W\,ratio\,(k) = W\,(k)/Tot\,W\,(k) \tag{8}
\]
\(W\,ratio\,(k)\): Normalized criticality of an agent in a MAS
organization in the \(k\)th sampling period. \(W\,ratio\,(k)\) shows how
critical an agent is among other agents in a MAS organization.
\(Tot\,W\,(k)\): Total criticality of all agents in a MAS organization.
\(W\,ratio\,(k)\) is the gain of the actuator. \(W\,ratio\,(k)\) values are sent to
the agents to be used in the adaptive fault tolerance plan which is
implemented as a reusable plan using HTN formalism[3]. In HTN
formalism a complex task is defined as compositions of primitive
and possibly other complex tasks. We call the complex task,
which matches to a goal as a plan.
In the adaptive fault tolerance policy, the leader has the “Adaptive
Fault Tolerance” plan in which adaptive replication mechanism is
performed. We mentioned that the observation mechanism and the
feedback control mechanism are embedded into the
communication infrastructure. The feedback control mechanism
sends the normalized criticality value to each agent in the content
of a FIPA message. When the agent is received this message, it
starts to execute the “Adaptive Fault Tolerance” plan and gets its
criticality value as a provision. The first task of this plan is to
determine the replication degree of the group by using \(W\,ratio\)
values sent by the ACC module. Since the resources are limited,
the replication degree for each adaptive agent is defined as follows:
\[
R\,D = Round\,(W\,ratio \times RM) \tag{9}
\]
\(R\,D\): The replication degree of a replica group that should
be present in current sampling period and \(RM\) is the number of
available resources in an organization,
\(N\,R\): The number of replicas in the current group
\(R\,C\): The number of replicas that must be replicated if \(R\,C\) is a
positive integer or the number of replicas that must be killed if
\(R\,C\) is a negative integer. If \(R\,C\) is a positive integer then the plan,
it replicates new replicas, is executed as explained in[16].
If \(R\,C\) is a negative number, the plan structure, which decreases
replication degree of a group, is executed as explained in[16]. In
order to give an idea about adaptive replication implemented by
using SEAGENT, next we’ll give a case study example.
6. CASE STUDY

Our fault tolerance approach presented in this paper has been implemented within SEAGENT’s internal architecture. By using our approach, we try to show the effectiveness of our approach.

For the evaluation of our approach, we designed an agent system which includes some specific agents which are called library assistant agents and some other agents that are specially designed for querying library assistant agents. A library assistant agent holds the library ontology for the books which exist in the library of our department. Instances of this ontology hold the properties of books including name, JSSBN, authors’ names and keywords of the books. The library assistant agent is queried by the agents to find out the situation of a specific book. In this case study, the library assistant agent has only one plan that matches the request to the book ontology instance(s) and returns the matched books descriptions within a FIPA message. Some agents have also simple plans that directly query the library assistant agents and present the results returned by the library assistant agent to the user interface. In this case study, the other agents depend on the library assistant agent. Therefore, the library assistant agent is a single of point of failure. Since it is a critical agent for the system operation, it must be initialized as a fault tolerant agent. Although, our agent plan is very simple, in this case study its general characteristic is very realistic in terms of fault tolerance. Therefore it must be implemented as a fault tolerant agent to make the system more robust.

The agent system is implemented in SEAGENT agent platform and Java Version 1.5.0. The tests are performed on nine computers with Intel Celeron CPU running at 1.2 GHz and 256MB of RAM, running Windows 2000. SEAGENT agent platform including ACC module runs on one of the computers. We distribute up to 100 agents, which execute the same plan, to nine computers. Particularly, leader agents and their replicas run on different computers.

The evaluation consists of two tests:

1. In this test, we have evaluated the cost resulting from including observation and feedback control mechanisms to the system as the number of replicas increases in replica groups. Therefore, we implemented a test bed consisting of two library assistant leaders, their replicas in the number range from 10 to 40, and two agents that query the library assistant agents. One of the library assistant agent leaders applies a static fault tolerance policy (which means no monitoring is applied), while the other one applies the adaptive fault tolerance policy simply including the observation mechanism and the feedback control mechanism. In order to see the monitoring cost, querying agents send requests to the leaders and the response times of the queries are measured. The response time is the time that takes a querying agent to receive the reply from the leader agent after sending its request to the leader. In this test, we try to observe the effect of the monitoring and the feedback control cost to the overall system performance by including the observation mechanism and the feedback control mechanism.

2. In this test, we evaluated the effectiveness of our approach as the number of faults in a replica group increases up to 120. Therefore, we implemented a test bed consisting of five library assistant agent leaders and five querying agents, and a failure simulator. In the first case, we employ the feedback control mechanism in which Cp is set to zero. Therefore, the closed loop of the feedback control mechanism is canceled. This means we only consider the communication load and the role of agents for calculating the criticality of agents. Thus, we try to observe the behavior of a control mechanism similar to the explained in [12, 13]. In the second case, we employ the feedback control mechanism in which Cp is set to the value given by formula (3). In both cases, the failure simulator sends “kill” message to the agents in a certain period to simulate the presence of faults. The agents receiving the “kill” messages will stop their threads. Most of the kill messages are sent to the more critical agents while less of the kill messages are sent to the less critical agents. In this test, the number of maximum available resources is given as 100. Therefore, the number of replicas in the organization must be 100. In this test, we try to observe the effect of the feedback control mechanism by comparing two cases, as the number of faults sent to the organization increases.

6.1 Discussions

The results of Test1 are illustrated in Figure 4. The results show the monitoring cost, which is the difference between the response times of the library assistant agents that apply static and dynamic fault tolerance policies. While applying the static fault tolerance policy in the library system, the system runs without monitoring and feedback control mechanisms. However, applying the adaptive fault tolerance policy in the library system, monitoring and feedback control mechanisms are included for the system operation. Therefore, the results (called monitoring cost in ms) given in Figure 5 are yielded by subtracting the response time of the system applying the static fault tolerance policy from the response time of the system applying the adaptive fault tolerance policy. As seen from the figure the monitoring cost is almost constant as the number of replicas increases. This result was expected, since there is only one leader library assistant agent that applies adaptive fault tolerance policy and is monitored by the observation mechanism and controlled by a feedback control mechanism in ACC module. Even in this case, monitoring cost does not change in considerable level, if the number of leader agents that apply adaptive fault tolerance increases the monitoring cost will also increase.

The results of Test 2 are illustrated in Figure 5. As seen from the figure, as the number of the faults increases over 100, the success rate, which means the percentage of the replica groups that can complete their tasks, in the organization eventually decreases since only 100 replicas can be present in the organization (Rm, the number of available resources, is set to 100). This result was expected since as the number of faults linearly increases the feedback control mechanism in ACC module has to consider the number of faults to determine the criticality values of agents and send these values to the agents. By setting the value of Cp to zero, the number of faults will not be considered in determination of the criticality of agents. Therefore, when the fault rate increases in the organization, the feedback control mechanism shows the better performance comparing to the other control mechanisms. This result proves that the integration of control theory to adaptive replication improves fault tolerance.
The Monitoring Cost

<table>
<thead>
<tr>
<th>replica no.</th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>time(ms)</td>
<td>0</td>
<td>1000</td>
<td>2000</td>
<td>3000</td>
<td>4000</td>
</tr>
</tbody>
</table>

Figure 4: The Monitoring Cost of the Feedback Control Mechanism

The Number of Replicas

Success Rate

<table>
<thead>
<tr>
<th>% success</th>
<th>0</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>120</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Number of Faults</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
<td>110</td>
<td>120</td>
</tr>
</tbody>
</table>

Figure 5: The Effectiveness of the Feedback Control Mechanism

7. CONCLUSION

In this paper, a new approach has been proposed for adaptive replication based on the classical control theory. This approach is used to evaluate dynamically the criticality of agents. The quantities that define the criticality of agents are sent in a FIPA message content to the agents to activate replication services implemented as reusable plan structure. According to contents of the messages, the agents start either cloning themselves or killing their replicas used in multi-agent organizations to improve fault tolerance.

Although, we know that the adaptive fault tolerance policy is a useful technique in terms of fault tolerance, we observe that applying an adaptive fault tolerance policy to the groups increases the cost due to the observation and feedback control mechanism. We again observe that our new approach to adaptive replication shows better performance comparing the other replication mechanisms which considers only communication load and behaviors of the agents in determination of the criticality of agents. As a conclusion, first results are encouraging and indicating that efficient replication is sustainable using this model.

8. REFERENCES


