Active Replication in Wide-Area Networks *

4G−3

Hiroaki Higaki, Nobumitsu Morishita and Makoto Takizawa †
Tokyo Denki University ‡
e-mail{hig,nobu,taki}@takilab.k.dendai.ac.jp

1 Introduction

According to the advance of computer and network technologies, network applications are widely developed. These applications are realized by the cooperation of multiple *objects*. Here, mission critical applications are also implemented and these applications are required to be executed fault-tolerantly. *Active replication* has been proposed where multiple replicated objects are operational in the network system. In the conventional active replication, all the replicated objects are required to be synchronized. In the network environment, each replicated objects may be placed on different kinds of computers so that the synchronization induces additional time-overhead. The authors have been proposed pseudo-active replication [4]. Here, not all the replicated objects are required to be synchronized. However, the proposed protocol for the pseudo-active replication has been proposed for local-area networks with different kinds of computers. In this paper, we extend the pseudo-active replication to be used in wide-area and large scale network environment and propose a novel protocol. In section 2, we review the pseudo-active replication. The overview of a modified pseudo-active replication for wide-area and large-scale network environment and the design of a protocol are shown in section 3 and 4, respectively.

2 Pseudo Active Replication

In a network system $\mathcal{S}$, an application is realized by cooperation of multiple *objects*. An object is composed of data and methods for manipulating the data. A *client* object o_i^c request a *server* object o_j^s to invoke a method. o_j^s manipulates the data and responds to o_i^c. In this paper, the objects are assumed to communicate in such style called *client-server* style. Each server object o_i^s are replicated for fault-tolerance. o_{ik}^s $(1 \leq k \leq n_i)$ are *replicas* of o_i^s.

There are two main approaches for replicating objects: *passive* and *active* replication. In the passive replication, only one of the replicas is operational. A client object o_i^c sends a request message to one of the server replicas say o_{j1}^s. Only o_{j1}^s invokes the methods requested by o_i^c and sends back a result message to o_i^c. o_{j1}^s sometimes sends the state information to o_{jk}^s $(2 \leq k \leq n_i)$ and o_{ij}^s updates the state information. If o_{j1}^s fails, one of the passive replicas say o_{j2}^s becomes operational. Hence, the recovery procedure takes time because o_{j2}^s has to re-execute the methods which o_{j1}^s has already executed before the failure. In the active replication, all the replicas are operational. A client object o_i^c sends request messages to all of o_{jk}^s. Every o_{jk}^s invokes the method requested by o_i^c and sends back a result message to o_i^c. After receiving all the messages from o_{ik}^s, o_i^c accepts the result and continue to execute the application. Since all the replicas are operational, even if some replica $o_{jk'}^s$ fails, the other replicas o_{jk}^s $(k \neq k')$ can continue to execute the application.

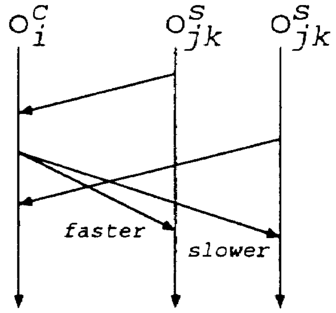

Figure 1: Pseudo-Active replication

In the conventional active replication, every replica o_{jk}^s are assumed to be placed on the same kind processors. That is, all the replicas simultaneously finish the computation for a request from a client object. This assumption is reasonable in local area networks. However, in wide-area networks, e.g. the Internet, each replica may be placed on different kind processors with different speed, reliability and availability. Here, it is difficult for o_i^c to receive all the responses from o_{jk}^s $(1 \leq k \leq n_j)$ simultaneously. That is, the synchronization overhead for receiving the responses is required to be reduced. The authors have been proposed a *pseudo-active replication* where o_i^c only waits for the first response from the replicas. On receiving the first response, o_i^c continues to execute the application. Thus, the synchronization overhead is reduced. However, since o_{jk}^s are placed on processors with different speed and are not synchronized, some replica $o_{jk'}^s$ might finish the computation for all the request from the client objects and another replica $o_{jk''}^s$ might keep many requests not to be computed because $o_{jk''}^s$ is placed on a slow processor. If $o_{jk'}^s$ fails, the recovery procedure takes time because $o_{jk''}^s$ has to execute the methods that $o_{jk'}^s$ has already executed before the failure as in the passive replication. In order to solve this problem, (1) each client object tells the server replicas which server is fast, and (2) if o_{jk}^s finds to be slower, it omits some methods requested by client objects to catch up the faster server replicas [Figure 1].

[Faster/Slower replica] If the response from o_{jk}^s has been received and that from $o_{jk'}^s$ has not yet when o_i^c sends a request to o_j^s, o_i^c informs that o_{jk}^s is a faster replica and $o_{jk'}^s$ is a slower one. □

[Omissible request] If an operation *op* is *identity* or *idempotent*, *op* is defined to be omissible [2]. □

[Omission rule] If the following conditions are satisfied, an operation *op* is omitted in o_{jk}^s:

1. o_{jk}^s is a slower replica.

*広域ネットワークにおける能動的多重化
†桧垣 博章　森下 展光　滝沢 誠
‡東京電機大学

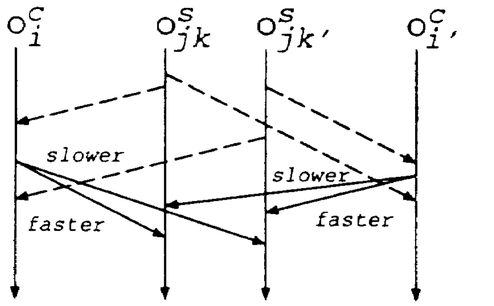

Figure 2: Pseudo-Active in a wide-area network

2. op is an omissible operation.
3. Some $o^s_{jk'}$ has already executed op. □

In [2] and [4], by using *vector clocks* [3], rule 1 and 3 are checked in o^s_{jk}. In addition, every request is assumed to be delivered to all the server replicas in the same order, i.e. *totally ordered delivery* is assumed.

3 Pseudo-Active in a Wide-Area Network

First, we remove Omission rule 3 in order for slower servers to catch up faster servers. By removing the confirmation whether op has been executed before omitting op, the difference of the number of the operations not yet executed among the replicas can be reduced. This is because the procedure omitting the operations can be invoked more frequently.

In a wide-area network, processors on which the replicas o^s_{jk} of server objects may be connected to different sub-networks, e.g. one is in Japan and another is in Europe, for executing applications more fault-tolerantly. In this case, all the replicated server objects may be informed to be slower [Figure 2]. Here, some operation op may be omitted by all the replicated servers o^s_{jk} and the client object o^c_i which requests op cannot receive any response from the servers.

In a wide-area network, the difference among the times when a client object o^c_i receives a response from each o^s_{jk} is caused by both the processing speed of the processors on which o^s_{ik} are placed and the message transmission delay in the channel between o^s_{ij} and o^c_i. Furthermore, in a large-scale network, i.e. the system $\mathcal{S}$ includes large number of client objects, since multiple clients may send requests simultaneously and the processing speed information piggy backed to the request is relative, all the replicas may be informed to be slower. Finally, the judgment of the processing speed is based on the receiving order of the previous responses in a client object. This information does not reflect *current* processing speed. In order to solve these problems, we design another protocol where the processing speed information is piggied back to the messages used for a total ordering protocol.

4 Protocol

In this section, we propose another protocol for pseudo active replication based on the total ordering protocol [1]. Each replicated server object o^s_{jk} manipulates the following variables:

- Logical clock cl_{jk} for totally ordering the requests from client objects.
- Last executed operation index loi_{jk} for the measurement of processing speed of server objects.

In the following total ordering protocol, the information which operations have been executed in every o^s_{jk} is exchanged among the replicated servers [Figure 3]:

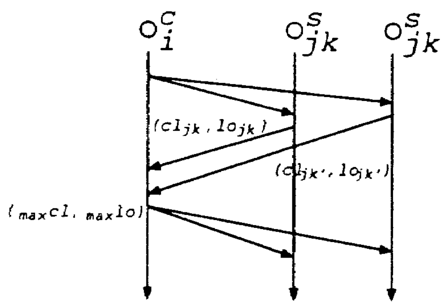

Figure 3: Total ordering protocol for pseudo active

[Total ordering protocol]

1. A client object o^c_i sends request messages $req(op)$ with an operation op to all the replicated server objects o^s_{jk} $(1 \leq k \leq n_j)$.
2. On receipt of $req(op)$, o^s_{jk} stores op in the buffer with cl_{jk}. o^s_{jk} sends back an ordering message $ord(cl_{jk}, loijk)$ piggy backing cl_{jk} and $loijk$. cl_{jk} is incremented by one.
3. After receiving all the ordering messages from o^s_{jk} $(1 \leq k \leq n_j)$, o^c_i sends final messages $fin(\max cl, \max loi)$ where $\max cl = \max_k cl_{jk}$ and $\max loi = \max_k cl_{jk}$.
4. On receipt of $fin(\max cl, \max loi)$, op is restored from the buffer and enqueued to APQ ordered by $oi(op) = \max cl$. □

APQ is an FIFO message queue and the application dequeues messages from APQ. If the application finishes an operation op with $oi(op)$, loi_{jk} is updated to $oi(op)$. Hence, loi_{jk} is always incremented. $\max loi$ piggy backed to fin message means that the fastest server object has finished to execute an operation with $\max loi$. Hence, the procedure for omitting operations is invoked as follows:

[Omitting operations]

- If $\max loi - loi_{jk} > threshold$, omissible operations in APQ is removed. □

5 Concluding Remarks

In order to apply pseudo active replication in wide-area and large-scale network systems, we proposed a novel protocol which is designed based on the total ordering protocol. We are now implementing a prototype system and evaluating our protocol comparing our previous pseudo active replication protocol.

References

[1] Birman, K.P. and Joseph, T.A., "Reliable Communication in the Presence of Failures," ACM Trans. on Computer Systems, Vol. 5, No. 1, pp. 47–76 (1987).

[2] Ishida, T., Higaki, H. and Takizawa, M., "Pseudo-Active Replication of Objects in Heterogeneous Processors," IPSJ Technical Report, vol. 98, No. 15, pp. 67–72 (1998).

[3] Mattern, F., "Virtual Time and Global States of Distributed Systems," Parallel and Distributed Algorithms, North-Holland, pp. 215–226 (1989).

[4] Shima, K., Higaki, H. and Takizawa, M., "Pseudo-Active Replication in Heterogeneous Clusters," IPSJ Trans., Vol. 39, No. 2, pp. 379–387 (1998).