5S-05

Ordered Delivery of Message in Object-Based Systems *

Takao Komiya, Youhei Timura, Katsuya Tanaka, and Makoto Takizawa †
Tokyo Denki University ‡
Email : {komi,timura, katsu, taki}@takilab.k.dendai.ac.jp

1 Introduction

In a distributed application like a teleconference, multiple processes are cooperating to achieve some objectives. A message is required to be delivered to all the destinations of the message. The group protocol using the vector clock [1,2] totally implies $O(n^2)$ computation and communication overheads for the number n of the processes in the group. The overheads can be reduced if only messages required to be ordered by the applications have to be causally and atomically delivered. An application is realized by a collection of processes each of which manipulates data like files in the computers and exchanges messages with other processes. Processes manipulating the data, are encapsulated in an object and the computation is performed by a message-passing mechanism.

An object in a computer sends a message to one destination, i.e. *unicast* or multiple destinations, i.e. *multicast*. In addition, some object can invoke multiple methods in parallel, i.e. parallel invocation. Here, different messages are simultaneously sent to different destinations. Objects wait for multiple responses after multiple methods are invoked in parallel. There are conjunctive and disjunctive ways to receive multiple messages. In the conjunctive receipt, the computer waits for all the messages. Hence, even if the computer sends a message while receiving these messages, there is no causally precedent relation between the message sent and the messages received. In the disjunctive receipt, the computer waits for only a message which arrives at the computer earlier than the others. The computer is not required to receive all the other messages. In this paper, we discuss a new type of causally precedent relation among messages in a system where messages are unicast, multicast, and parallel-cast, and received by single-message and multi-message receipt.

2 Object-based Systems

The objects are distributed in computers interconnected with reliable networks. A *computer* supports a collection of objects and does not necessarily mean a physical *computer*.

A group G is composed of computers supporting objects o_1, ..., o_n and transactions. A transaction is initiated in one of the computers and invokes methods on an object in the group G. The objects are cooperating with each other in the group G. This means every method on an object invokes only methods on objects in the group G.

Methods are invoked in a nested manner in object-based systems. There are *synchronous*, *asynchronous*, and *one-way* invocations of a method with respect to how an invoker, e.g. transaction waits for the response of the method. In the synchronous invocation, the invoker waits for a response of the method. In the asynchronous one, the invoker is being performed without blocking while eventually receiving the response of the method. In the one-way invocation, the invoker does not wait for the response.

There are *serial* and *parallel* invocations of multiple methods. In the serial invocation, at most one method is invoked at a time. On the other hand, multiple methods can be simultaneously invoked in the parallel invocation.

3 Delivery of Messages in Objects

3.1 Transmission

An object sends messages by invoking the transmission methods **ucast**, **mcast**, and **pcast** of the *com* object. The messages sent by the object are transmitted in a network by the computer. For example, if an object o in a computer p_s multicasts a message m by **mcast**(m,$\langle p_t, p_u \rangle$), the computer p_s sends a pair of message instances m_1 and m_2 of m to two computers p_t and p_u, respectively. We discuss how these message instances transmitted in the network to be related. Suppose an object in a computer p_s sends a pair of messages m_1 and m_2. The messages m_1 and m_2 transmitted in the network are related according to the following relations depending on through which transmission method the object sends the messages m_1 and m_2:

1. **Multicast**: m_1 and m_2 are multicast ($m_1 \approx m_2$) iff $m_1 = m_2$ and m_1 is sent to multiple destinations, i.e. m_1 and m_2 are instances of a message m and m is sent by **mcast**.
2. **Parallel-cast**: m_1 and m_2 are parallel-cast ($m_1 \equiv m_1$) iff m_1 and m_2 are to be sent at a same time, i.e. m_1 and m_2 are sent by **pcast**.
3. **Serial transmission**: m_1 is serially sent before m_2 ($m_1 \prec\!\!\prec m_2$) iff m_1 is to be sent before m_2 by **ucast**.

3.2 Receipt

Each object receives a response after sending a request to an object. In addition, an object parallely invoked methods. Here, the object waits for response messages from multiple objects. There are multiple ways to receive messages; *single-message* and *multi-message* receipts where an object waits for only one message and multiple messages, respectively. The *com* object supports three types of primitive methods for receiving messages: 1.**srec**(p_1). 2.**crec**(p_1, ..., p_k) ($k \geq 1$). 3.**drec**(p_1, ..., p_k) ($k \geq 1$). By invoking the method **srec**(p_1) on the *com* object, one message is received from an object in a computer p_1, i.e. single-message receipt. By invoking **crec** and **drec** methods, an object receives messages from multiple objects. There are two further ways to receive multiple messages from multiple computers; *conjunc-*

*分散オブジェクト環境におけるグループ通信プロトコル
†小宮 貴雄, 千村 洋平, 田中 勝也, 滝沢 誠
‡東京電機大学

tive and *disjunctive* ones. The conjunctive receipt method **crec**(p_1, ..., p_k) means that messages are received from all the computers p_1, ..., p_k. That is, a method instance invoking the method **crec** waits for messages from all the computers p_1, ..., p_k. The disjunctive receipt method **drec**(p_1, ..., p_k) means that one message is received from at least one computer out of the computers p_1, ..., p_k. A method inst ance invoking **drec** blocks until a message is received from at least one of the computers. Suppose an object in a computer p_s receives multiple messages m_1, ... , m_k by invoking one multi-message receipt **crec** or **drec** [Figure 1]. Here, let $M(m_i)$ be a collection of messages $\{m_1, \ldots, m_k\}$ to be received with a message m_i at a multi-message receipt. For every message m_j in $M(m_i)$, $M(m_j) = M(m_i)$. In the conjunctive receipt **crec**, suppose the computer p_t finishes receiving the messages in $M(m_i)$ on when p_t receives a message m_k after receiving all the other messages in $M(m_i)$. Here, m_k is referred to as *most significant* for the messages m_1, ..., m_k in $M(m_i)$ for **crec**. Suppose the computer p_t receives a message m_1 before all the other messages in $M(m_1)$. The message m_1 is referred to as the *first message* in $M(m_1)$. On the other hand, in the disjunctive receipt **drec**, p_t finishes receiving the messages m_1, ..., m_k only if p_t receives the first message m_1 before all the other messages. The message m_1 is *most significant* for the messages in $M(m_1)$ for **drec**. Here, the other messages m_2, ..., m_k are not so significant that the computer p_t is required to receive the messages. Suppose that a computer p_t receives a

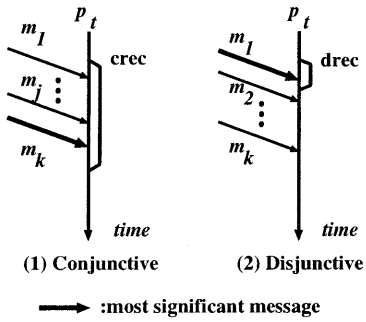

Figure 1: Multi-message receipt.

pair of messages m_1 and m_2 in a network. Let "$m_1 \prec m_2$" show that p_t receives m_1 before m_2.

3.3 Receipt and transmission

We discuss a causally precedent relation ($\rightarrow$) among messages sent and/or received in a computer p_t.

[Definition] A message m_1 sent *precedes* another message m_2 sent in a computer p_s ($m_1 \rightarrow m_2$) if $m_1 \prec m_2$ and one of the following conditions holds:

1. m_1 and m_2 are sent through different transmissions methods by a same method of an object in the computer p_s.
2. m_1 and m_2 are sent by different methods t_1 and t_2 of an object in the computer p_s, respectively, and t_1 and t_2 conflict.
3. $m_1 \rightarrow m_3$ and $m_3 \rightarrow m_2$ for some m_3. □

[Definition] m_1 received precedes m_2 received in p_t ($m_1 \rightarrow m_2$) if one of the following conditions is satisfied:

1. m_1 received by **srec**:
   a. m_2 is received by **srec**.
   b. m_2 is received by **crec** and $m_1 \prec m_3$ fo r the most significant message m_3 in $M(m_2)$.
   c. m_2 is received by **drec**, $m_1 \prec m_2$, and m_2 is the most significant in $M(m_2)$.
2. m_1 is received by **crec**:
   a. m_2 is received by **srec** and $m_3 \prec m_2$ fo r the most significant message m_3 in $M(m_1)$.
   b. m_2 is received by **crec** and $m_3 \prec m_4$ fo r the most significant messages m_3 and m_4 in $M(m_1)$ and $M(m_2)$, respectively.
3. m_1 is received by **drec**:
   a. m_2 is received by **srec**, $m_1 \prec m_2$, and m_1 is the most significant message in $M(m_1)$
   b. m_2 is received by **crec**, m_1 is the most significant message in $M(m_1)$, and $m_1 \prec m_3$ for the most significant message m_3 in $M(m_2)$.
4. $m_1 \rightarrow m_3$ and $m_3 \rightarrow m_2$ for some m_3. □

[Definition] m_1 received *precedes* m_2 sent in p_s ($m_1 \rightarrow m_2$) if one $m_1 \prec m_2$ and one of the following conditions is satisfied;

1. m_2 is unicast by **ucast**:
   a. m_1 is received by **srec**.
   b. m_1 is received by **crec** and $m_3 \prec m_2$ for the most significant message m_3 in $M(m_1)$.
   c. m_1 is received by **drec** and m_1 is the most significant in $M(m_1)$.
2. m_2 is multicast by **mcast** or parallel-cast by **pcast**:
   a. m_1 is received by **srec** and $m_1 \prec m_3$ for some message m_3 such that $m_2 \approx m_3$ or $m_2 \equiv m_3$.
   b. m_1 is received by **crec** and $m_3 \prec m_4$ for the most significant message m_3 in $M(m_1)$ and some message m_4 such that $m_2 \approx m_4$ or $m_2 \equiv m_4$.
   c. m_1 is received by **drec** and $m_1 \prec m_4$ for the most-significant message m_1 in $M(m_1)$ and some message m_4 such that $m_2 \approx m_4$ of $m_2 \equiv m_4$.
3. $m_1 \rightarrow m_3$ and $m_3 \rightarrow m_2$ for some m_3. □

4 Concluding Remarks

In the object-based system, methods are not only serially but also in parallel invoked and responses are received in various ways. An object multicast one message to multiple destinations, i.e. *multicast* and different messages are *parallel-cast* to multiple destinations. Objects receive multiple messages in conjunctive and disjunctive receipt ways. We defined new types of causally precedent relations among messages transmitted by multicast **mcast** and parallel-cast **pcast** and received by conjunctive **crec** and disjunctive receipts **drec** in addition to *unicast* and single-message receipt **srec**.

References

[1] Birman, K., Schiper, A., and Stephenson, P., "Lightweight Causal and Atomic Group Multicast," *ACM Trans. on Computer Systems*, Vol.9, No.3, 1991, pp.272-314.

[2] Mattern, F., "Virtual Time and Global States of Distributed Systems," *Parallel and Distributed Algorithms* (Cosnard, M. and , P. eds.), *North-Holland*, 1989, pp.215–226.