5V−1

Design and Implementation of Reliable Broadcast Protocol

Akihito Nakamura, Yasuyuki Terauchi, and Makoto Takizawa
Tokyo Denki University

1 Introduction

This paper discusses how to provide reliable broadcast communication for multiple entities in distributed systems by using unreliable broadcast communication services. In distributed applications, each entity rather sends every protocol data unit (PDU) to only the subset than all the entities, and every entity receives only PDUs destined to it from each entity in the same order as they were sent. We name such a broadcast service an SPO service (service for selectively partially ordering PDUs). In this paper, we discuss a design of a protocol, named $\mathcal{SPO}$, which provides the SPO service by using unreliable broadcast networks.

In section 2, we model unreliable and reliable broadcast communication services. In section 3, we present a data transmission procedure of the $\mathcal{SPO}$ protocol. Finally, we discuss the implementation and performance of the $\mathcal{SPO}$ protocol in section 4.

2 Service Model

A cluster C [1] is composed of n (≥ 2) entities $E_1, ..., E_n$. Entities communicate with each other by sending and receiving PDUs. The communication service which every entity uses is modeled as a set of logs [1]. A log is a sequence of PDUs. All the PDUs in a log L are totally ordered by the precedence relation $\rightarrow_L$. For every pair of PDUs p and q in L, $p \rightarrow_L q$ if p precedes q in L. Each entity E_i has two kinds of logs, a sending log SL_i and a receipt log RL_i. SL_i and RL_i are sequences of PDUs which E_i has sent and received, respectively. Also, RL_{ij} is a sublog of RL_i, which is a sequence of PDUs from E_j only ($j = 1, ..., n$).

The Multi-Channel (MC) service [1] is an abstraction of service provided by the systems in which stations are connected by multiple communication channels (multiple Ethernets, for example). Formally, the MC service is defined as following. For every E_i and E_j, and every pair of PDUs p and q in SL_j such that $p \rightarrow_{SL_j} q$, $p \rightarrow_{RL_i} q$ if E_i has received both p and q. That is, PDUs are received in sequence but may be lost.

From the viewpoint of distributed application entities, they require communication service which ensures reliable transfer of data to a number of intended recipients. The SPO service [1] is one which satisfies such a requirement. This service is characterized as follows. For every E_i and E_j, and every pair of PDUs p and q in SL_j such that $p \rightarrow_{SL_j} q$, if both p and q include E_i in their destinations, then $p \rightarrow_{RL_i} q$. Also, RL_i includes all the PDUs intended to E_i. In other words, every entity receives all and only the PDUs destined to it in sequence from each entity.

Design and Implementation of Reliable Broadcast Protocol

Akihito Nakamura, Yasuyuki Terauchi, Makoto Takizawa
Tokyo Denki University

3 $\mathcal{SPO}$ Protocol on the MC Service

We now discuss the $\mathcal{SPO}$ protocol which provides the SPO service to entities in a cluster C by using the MC service.

The data transmission procedure of the $\mathcal{SPO}$ protocol is based on the three-phase receipt [1] to realize the fully distributed control scheme. First, a PDU p is broadcast by E_i and is *accepted* in each entity E_j. Next, E_j knows that every entity has accepted p by collecting the acknowledgment (ACK) from each entity. Here, p is said to be *pre-acknowledged* in E_j. Then, by collecting ACKs for PDUs which carry ACKs for p, E_j knows that every entity has pre-acknowledged p and p is said to be *acknowledged* in E_j.

Every entity has n receipt sublogs $RL_{i1}, ..., RL_{in}$. Each sublog RL_{ij} is divided into three continuous subsequences ARL_{ij}, PRL_{ij}, and RRL_{ij}, each of which contains acknowledged, pre-acknowledged, and accepted PDUs, respectively.

3.1 Variables

A notation p^i is used to denote that a PDU p is sent by E_i. Also, p^i.*field* denotes the *field* in p^i. Every PDU p^i has the following fields ($j = 1, ..., n$).

- $p^i.SRC = E_i$, i.e. the entity which sends p^i.
- $p^i.DST$ = set of destination entities of p^i.
- $p^i.TSEQ$ = total sequence number of p^i.
- $p^i.PSEQ_j$ = partial sequence number for E_j.
- $p^i.ACK_j$ = total sequence number of a PDU which E_i expects to receive next from E_j.
- $p^i.BUF$ = number of buffers available in E_i.
- $p^i.DATA$ = data to be broadcast.

When E_j receives p^i, if $E_j \in p^i.DST$, then E_j has to accept p^i. Otherwise, E_j can discard p^i. Each p^i has a unique sequence number $p^i.TSEQ$ which denotes the position in the total sequence of PDUs broadcast by E_i. Also, p^i has a unique sequence number $p^i.PSEQ_j$ for each E_j which denotes the position of the sequence of PDUs broadcast by E_i and destined to E_j . $p^i.ACK_j$ informs every entity in C that E_i has received every PDU q^j where $q^j.TSEQ < p^i.ACK_j$.

Each E_i has the following variables ($j, k = 1, ..., n$).

- $TSEQ$ = total sequence number of a PDU which E_i expects to send next.
- $PSEQ_j$ = partial sequence number of a PDU which E_i expects to send to E_j next.
- $TREQ_j$ = total sequence number of a PDU which E_i expects to receive next from E_j.
- $PREQ_j$ = partial sequence number of a PDU which E_i expects to receive next from E_j.
- AL_{jk} = total sequence number of a PDU which E_i knows E_k expects to receive next from E_j.
- PAL_{jk} = total sequence number of a PDU which E_i knows that E_k expects to pre-acknowledge from E_j.
- BUF_j = number of buffers in E_j which E_i knows of.

Let $minAL_j$ denote the minimum among $AL_{j1}, ..., AL_{jn}$. This means that all the entities have already received every PDU

g^j where $g^j.TSEQ < minAL_j$. Let $minBUF$ denote the minimum among $BUF_1, ..., BUF_n$. The initial values of the variables are initiated when C is established [1].

3.2 Transmission and Acceptance

E_i broadcasts and accepts a PDU by the following procedures. Here, W and H are constants.

[Transmission Procedure for p^i in E_i]

```
if (minAL_i ≤ TSEQ < minAL_i +min(W, minBUF/(H × n²)) {
    p^i.TSEQ := TSEQ;    TSEQ := TSEQ + 1;
    for (j = 1, ..., n) {
        p^i.ACK_j := TREQ_j;    p^i.PSEQ_j := PSEQ_j;
        if (E_j ∈ p^i.DST) {
            PSEQ_j := PSEQ_j + 1;
            p^i.DST := p^i.DST ∪ {E_j};
        }
    }
    enqueue(SL_i, p^i);    broadcast(p^i); } □
```

[Accept Procedure for p^j in E_i]

```
if ( (p^j.TSEQ = TREQ_j or p^j.PSEQ_i = PREQ_j) and
  for (k = 1, ..., n) p^j.ACK_k ≤ TREQ_k) {
    TREQ_j := p^j.TSEQ;
    for (k = 1, ..., n) AL_kj := p^j.ACK_k;
    if (E_i ∈ p^j.DST) {
        PREQ_j := p^j.PSEQ_i + 1;    enqueue(RL_ij, p^j);
    } } □
```

3.3 Pre-Acknowledgment and Acknowledgment

The problem is how each entity E_i decides the correct receipt of p^j based on received PDUs in the distributed control scheme. Here, the following notations are introduced.

- $AL_j(p^j) = \{AL_{jk} \mid E_k \in p^j.DST\}$.
- $minAL_j(p^j)$ = minimum number in $AL_j(p^j)$.

That is, every entity in $p^j.DST$ has received PDUs q^j such that $q^j.TSEQ < minAL_j(p^j)$. The PDUs accepted in every entity are pre-acknowledged by the following procedure.

[Pre-acknowledgment Procedure in E_i]

```
for (j = 1, ..., n) {
    while (p^j = top(RPL_ij), p^j.TSEQ < minAL_j(p^j)) {
        p^j := dequeue(RRL_ij);    enqueue(PRL_ij, p^j);
        for (k = 1, ..., n) PAL_kj := p^j.ACK_k;
    } } □
```

Next, we consider a procedure for the acknowledgment of PDUs. Here, the following notations are introduced.

- $PAL_j(p^j) = \{PAL_{jk} \mid E_k \in p^j.DST\}$.
- $minPAL_j(p^j)$ = minimum number in $PAL_j(p^j)$.

[Acknowledgment Procedure in E_i]

```
for (j = 1, ..., n) {
    while (p^j = top(PRL_ij), p^j.TSEQ < minPAL_j(p^j)) {
        p^j := dequeue(PRL_ij);    enqueue(ARL_ij, p^j);
    } } □
```

3.4 Failures

When the MC service is used, PDUs may be lost. Lost PDUs can be detected by the following lost conditions (LCs).

LC1. On receipt of p^j, if $PREQ_j < p^j.PSEQ_i$, then E_i has not received g^j such that $PREQ_j \leq g^j.PSEQ_i < p^j.PSEQ_i$ $(j = 1, ..., n)$.

LC2. On receipt of q^k, for some j $(\neq k)$, if $TREQ_j < q^k.ACK_j$, then E_i has not received g^j such that $TREQ_j \leq g^j.TSEQ < q^k.ACK_j$ $(k = 1, ..., n)$.

If LC1 holds, E_i has failed to receive some PDU and has to receive every lost PDU g^j. On the other hand, if LC2 holds, E_i detects some lost PDU g^j but does not know whether g^j is destined to E_i or not. E_i has to receive only g^j such that $E_i \in g^j.DST$. If E_i requests the retransmission as soon as LC2 holds, it may be meaningless for E_j to rebroadcast g^j, because $g^j.DST$ may not include E_i. Hence, E_i waits on some PDU from E_j for a while. Suppose that E_i receives a PDU r^j. If $r^j.PSEQ_i = PREQ_j$, E_i does not need to receive g^j. If $PREQ_j < r^j.PSEQ_i$, E_i should have received g^j. We adopt the selective retransmission scheme to recover from the PDU loss, where only the lost PDUs are retransmitted. When E_i requests the retransmission, the request PDU includes the source entity of the lost PDUs, and the minimum and maximum $PSEQ$s of them.

4 Implementation

The $\mathcal{SPO}$ protocol is implemented on SunOS 4.1 (SunOS is trademark of Sun Microsystems, Inc.). The stations are connected by 10Mbps Ethernet. The size of the program is about 5K steps in C language, and the size of the executable object-code is about 50K bytes. Figure 1 and 2 illustrate the performance aspects of the $\mathcal{SPO}$ protocol.

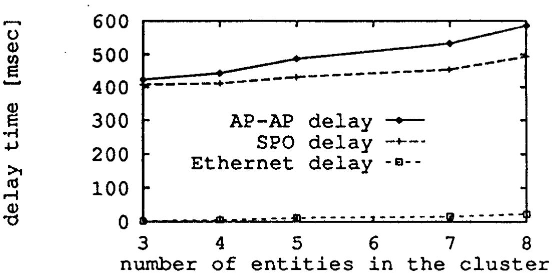

Figure 1: Delay Time vs. the Number of Entities

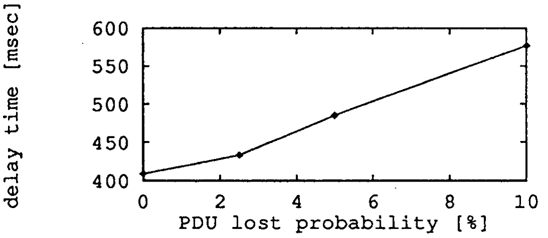

Figure 2: Delay Time vs. Probability of PDU Loss

5 Concluding Remarks

In this paper, we have discussed a design of reliable broadcast protocol to provide reliable data transfer service among multiple entities. We show the $\mathcal{SPO}$ protocol that offers the SPO service by using the underlying MC service in which PDU may be lost.

References

[1] Nakamura, A. and Takizawa, M., "Data Transmission Procedure of Selective Broadcast Protocol on Multi-Channel," *Trans. of IPSJ*, Vol.33, No.2, 1992, pp.223-233.