7T-2

Totally Ordering (TO) and Partially Ordering (PO) Broadcast Protocol

Akihito NAKAMURA and Makoto TAKIZAWA
Tokyo Denki University

1. INTRODUCTION

The cooperation of a collection of multipl entities is required to realize concurrency control, commitment control, and distributed query processing in distributed database systems. In this paper, we discuss how to design reliable broadcast communication systems on unreliable broadcast networks like Ethernet and radio networks. We try to provide on the unreliable broadcast service totally ordering broadcast (TO) service where every entity receives all messages in the same order and partially ordering broadcast (PO) service where they receive all messages which broadcast by same entity in the same order.

2. COMMUNICATION MODEL

We adopt a distributed control where each entity decides correct receipt of a PDU by itself.

[Def.] [1,2] E_k is said to receive correctly an PDU "p" iff (1) E_k receives "p", (2) E_k knows that every entity in p.DST has already received "p", and (3) E_k knows that every entity in p.DST has already known that every entity in p.DST had received "p". □

At (1), (2), and (3), "p" is said to be received, pre-acknowledged, and acknowledged in E_k, respectively.

[Assumption] PDUs which an entity broadcasts are received by itself. □

[Def.] An (N-1) reliable broadcast service is said to be partially ordering (PO) iff every (N)PDUs "p" and "q" which broadcast by same entity are received at every common destination SAP of them in the same order. It is said to be totally ordering (TO) iff all PDUs which broadcast by every entity are received in the same order. □

A cluster[1,2] is an extention of the conventional connection concept among two SAPs to one among n (>2) SAPs.

[Def.] A one-channel (1C) service is an unreliable broadcast service where PDUs may be lost and every entity receives PDUs in the same sequence if they are received. It is said to be multi-channel (MC) one iff PDUs broadcast by the same entity are received by every entity in the same sequence but some of them may be lost. □

3. ONE-CHANNEL (1C) SERVICE AND MULTI-CHANNEL (MC) SERVICE

We define logs as the sequence of PDUs. For every entity E_k, let a sending log $SL_k = (SP_k, <_{sk})$ and receipt log $RL_k = (RP_k, <_{RK})$ be sequences of PDUs which E_k broadcast and received, respectively. Here, SP_k and RP_k are sets of PDUs broadcast and received by E_k, respectively. $p <_{RK} q$ iff E_k received "p" before "q". $p <_{sk} q$ iff E_k broadcast "p" before "q". Let $RL_k[j]$ ($SL_k[j]$) be the j-th element in RL_k (SL_k) and j be the index of the element. Let RL_k^j be a prefix of RL_k whose last element is $RL_k[j]$. For "p" and "q" in RL_k (SL_k), $p <<_{RK} q$ ($p <<_{sk} q$) iff for some j, $p = RL_k[j]$ ($SL_k[j]$) and $q = RL_k[j+1]$ ($SL_k[j+1]$). For every RL_k, let RL_{kj} be a subsequence of RL_k which includes all PDUs broadcast by E_j.

[Def.] A receipt log RL_k is said to be legal iff for all "p" and "q" in RL_k, if $p <<_{sj} q$ in some SL_j, then $p <_{RK} q$ in RL_k. □

[Def.] For receipt logs $RL_1, \ldots, RL_n$, the index f is said to be a failure point (FP) iff (1) for all E_j and E_k, $RL_j^{f-1} = RL_k^{f-1}$ and they are legal, and (2) for some E_j, RL_j^f is not legal. □

Let APL_k, PPL_k, and RPL_k be sublogs of RL_k which are the subsequences composed of acknowledged, pre-acknowledged, and received PDUs, respectively. Here, RL_k is $APL_k \mid PPL_k \mid RPL_k$.

4. TOTALLY ORDERING (TO) PROTOCOL ON THE 1C SERVICE

We already discussed the data transmission procedure in the totally ordering broadcast (TO) protocol in [3,4]. The TO protocol provides on unreliable broadcast service TO service where every entity receives same PDUs in the same order. Here, we discuss the performance of the TO protocol.

PDU complexity is measured by the number of PDUs broadcast at SAPs to acknowledge a PDU "p". A round is a maximum delay time from an SAP to a destination. Time complexity is the number of rounds. In the best case, every entity does not broadcast PDUs until a PDU "g" which finally pre-acknowledges "p" is received, every entity broadcasts a PDU which pre-acknowledges "g". Here, $1+(n-1)+n=2n$ PDUs are broadcast. In the worst case, each time when a PDU "g" which pre-acknowledges "p" is received, every entity broadcasts a PDU which pre-acknowledges "g". Here, $1+(n-1)+(n-1)*(n-1)=n^2-n+1$ PDUs are broadcast.

In the best case, PDUs which pre-acknowledge "p" are broadcast in parallel.

Totally Ordering (TO) and Partially Ordering (PO) broadcast protocol
Akihito NAKAMURA and Makoto TAKIZAWA
Tokyo Denki University

Here, the number of rounds is 3. On the other hand, if PDUs cannot be broadcast in parallel like the Ethernet MAC service, $1+(n-1)+n=2n$ rounds at the best and $1+(n-1)+(n-1)*(n-1)=n^2-n+1$ rounds at the worst are required.

5. PARTIALLY ORDERING (PO) PROTOCOL ON THE MC SERVICE

Next, we design a protocol which provides the partially ordering broadcast (PO) service by using the MC service. In the PO protocol, every entity can receive the same PDUs broadcast by an entity in the same sequence. In order to realize the PO protocol, a mechanism similar to the TO protocol[3,4] is used.

Each entity E_k has n logs RL_{kj} for E_j $(j=1,\ldots,n)$. RL_{kj} is $APL_{kj} \mid PPL_{kj} \mid RPL_{kj}$.

Every PDU "p" (from E_k) includes the following information.

p.SRC = E_k.
p.SEQ = a sequence number of "p".
$p.A_j$ = a sequence number of a PDU which E_k expects to receive next from E_j $(j=1,\ldots,n)$.
p.BUF = the number of buffers available in E_k.

Each E_k has variables SEQ, REQ_j, and AL_{jl} $(j,l=1,\ldots,n)$ in order to check the sequence number of PDUs.

SEQ = the sequence number of a PDU which E_k expects to broadcast next.
REQ_j = the sequence number of a PDU which E_k expects to receive next.
AL_{hj} = the sequence number of a PDU which E_k knows E_j expects to receive next from E_h (for $j,h=1,\ldots,n$).

Let $minAL_j$ denote the minimum of $AL_{j1},\ldots,AL_{jn}$. Let ISS_k be an initial sequence number of E_k. Initially, SEQ = ISS_k and $REQ_j = AL_{jh} = ISS_j$ (for $j,h=1,\ldots,n$)[1,2]. Each entity E_k has n variables $F_1,\ldots,F_n$, where F_j denotes the number of buffers in E_j which E_k knows$(j=1,\ldots,n)$. Let minF denote the minimum in $F_1,\ldots,F_n$.

If the flow condition holds, E_k broadcasts a PDU "p" by the transmission action. Here, W and C (>1) are constants.

[Flow Condition] $minAL_k \leq SEQ < minAL_k + \min(W, minF/(C*n^2))$. □

[Transmission Action] (1) $p.A_j := REQ_j$ $(j=1,\ldots,n)$. (2) p.BUF := the current number of buffer available. (3) p.SEQ := SEQ and SEQ := SEQ + 1. (4) E_k broadcasts "p" and append to the tail of SL_k. □

On receipt of "p" from E_j, if "p" satisfies the receipt condition, "p" is received like the TO protocol and queued into RL_{kj}. PDUs in RL_{kj} are pre-acknowledged if they satisfy the pre-acknowledgment (PACK) condition. Also, if "p" satisfies the acknowledgment (ACK) condition, "p" is acknowledged.

[Receipt Condition] (1) p.SEQ = REQ_j (where $p.SRC = E_j$), and (2) $AL_{hj} \leq p.A_h \leq REQ_h$ (for $h=1,\ldots,n$). □

[Pre-acknowledgment (PACK) Condition] p.SEQ < $minAL_j$ (where $p.SRC = E_j$). □

[Def.] For each E_k and PDUs "p"$(p.SRC = E_j)$ and "q"$(p.SRC = E_h)$ in RL_k, "q" is said to pre-acknowledge "p" (on E_h) iff $p <_{RK} q$ in RL_k and $p.SEQ < q.A_j$. □

[Def.] For each E_k and PDUs "p" and "q" in RL_k, "q" is said to first pre-acknowledge "p" on E_h (= q.SRC) iff (1) "q" pre-acknowledges "p" on E_h, and (2) there is no PDU "g" such that $g <_{RK} q$ and "g" pre-acknowledges "p" on E_h. "q" is said to finally pre-acknowledge "p" iff (1) "q" first pre-acknowledges "p" on E_h (= q.SRC), and (2) there is no PDU "g" such that $q <_{RK} g$ and "g" first pre-acknowledges "p". □

[Acknowledgment (ACK) Condition] For every E_h, a PDU which first pre-acknowledges "p" on E_h is pre-acknowledged in E_k. □

[Receipt Action] (1) If "p" satisfies the receipt condition, "p" is appended to the tail of RPL_{kj}. Also, SEQ, REQ, and AL are updated in the same manner as the TO protocol does. (2) For each RPL_{kl}, while the top "q" in RPL_{kl} satisfies the PACK condition, "q" is dequeued from RPL_{kl} and enqueued into PPL_{kl}, and for each PPL_{kh}, while the top "g" in PPL_{kh} satisfies the ACK condition, "g" is dequeued from PPL_{kh} and enqueued into APL_{kh}. □

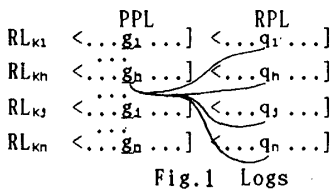

Fig.1 Logs

When the MC service is used, PDUs may be lost. Lost PDUs are detected by the same manner as in the TO protocol.

6. CONCLUDING REMARKS

In this paper, we have discussed a design and implementation of data transmission procedures which provide a totally ordering broadcast (TO) and partially ordering broadcast (PO) service by using unreliable broadcast services like the Ethernet. The protocols are based on the distributed control and the cluster concept.

REFERENCE

[1] M. Takizawa, "Cluster Control Protocol for Highly Reliable Broadcast Communication," Proc. of the IFIP Conf. on Distributed Processing, Amsterdam, 1987.
[2] M. Takizawa, "Design of Highly Reliable Broadcast Communication Protocol," Proc. of IEEE COMPSAC87, 1987, pp.731-740.
[3] A. Nakamura, and M. Takizawa, "Totally Ordering Broadcast Protocol on Multi-channel System", IPSJ, DPS, 39-1, 1988, pp.1-8.
[4] A. Nakamura, and M, Takizawa, "Totally Ordering (TO) and Partially Ordering (PO) Broadcast Protocol," JWCC89, 1989.