Design of Dynamic Group Communication

4P−6 Hitoshi Suzuki, Akihito Nakamura, and Makoto Takizawa
Tokyo Denki University

1 Introduction

In group communication, the configuration of a cluster, i.e. group of entities, may change by some events like entity failure. In some applications, there are cases that the remaining operational entities can continue to communicate with each other even if some entities fail, i.e. stop communication. On the other hand, some entity may join and leave the cluster. [3] discusses a total receipt ordering (*TO*) protocol where every entity receives all the protocol data units (PDUs) broadcast in the cluster in the same order preserving the sending order. In this paper, we would like to discuss a protocol which provides the *TO* service even if the cluster configuration changes. In this paper, we present two kinds of dynamic aspects of group communication. In the first aspect, entities in the cluster may fail and then recover from it. In the other aspect, entities may join and leave the cluster. We present *TO* protocols for such dynamic cluster.

In section 2, we present our system model. In section 3, we discuss properties of dynamic cluster in this paper. In section 4 and 5, we discuss two dynamic aspects. In section 6, we discuss how to manage clusters in the system.

2 System Model

2.1 Cluster

The communication system is composed of multiple layers according to the OSI reference model. An (N) *cluster* C is a set of (N) service access points (SAPs) $S_1,\ldots,S_n$. The (N) service is provided by the cooperation of (N) entities $E_1,\ldots,E_n$ by using the underlying (N−1) service. There are two aspects of the (N) cluster C, i.e. *scheme* and *instance*. A scheme of C shows which entities support C. It is represented as a set of the (N) SAPs. When a cluster C is established, an instance of C is created. When C is closed, the instance disappears. Assume that there exists at most one instance of C for any time.

2.2 System hierarchy

In this paper, the system is considered to be composed of three layers, i.e., *network*, *system*, and *application* layers[Figure 1]. We further assume that the underlying network layer provides the *TO* service for C, i.e. a set of SAPs $S_1,\ldots,S_n$. In the *TO* service, every entity in C can receive all the PDUs sent in C in the same order preserving the sending order. Each system entity E_i at the system layer takes the *TO* service through S_i in C. $E_1,\ldots,E_n$ provide a dynamic *TO* (*DTO*) service for the application entities $A_1,\ldots,A_n$. A_i takes the *DTO* service through a SAP D_i supported by E_i.

We assume that each entity stops by failure.

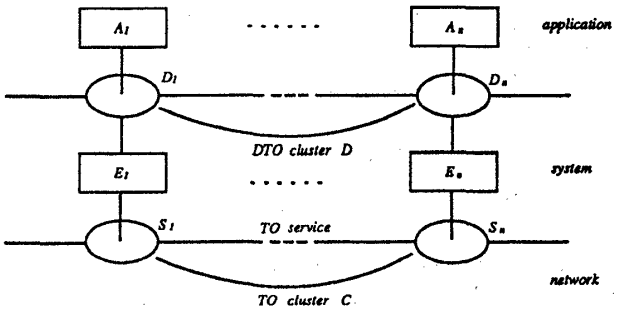

Figure 1: *DTO* cluster

3 Dynamic Cluster

Suppose that a *DTO* cluster D is supported by system entities $E_1,\ldots,E_n$. E_i is *operational* in D if E_i sends and receives PDUs. E_i *fails* if E_i stops sending PDUs. An instance of D is *fully operational* if all the entities in D are operational. An instance of D is *partially operational* if some entity fails. We assume that the failed entity does not recover promptly after it fails. We do not think about the *intermittent* failure of the entities.

D is *dynamic* if each instance of D can be fully or partially operational. D is *static* if all the instances of D have to be fully operational. D is *evolutional* if the scheme can be changed. In the evolutional cluster D, some entity may leave and join D. In the evolution of cluster, the data structure like variables, tables, and PDU formats has to be changed according to the *join* of entities.

4 Dynamic Instance Cluster

First, let us consider a case that some entity E_f fails in the *DTO* cluster D. When E_f fails, i.e. stops sending PDUs, the failure is detected by another operational entities using the time-out mechanism. When E_f fails, the other operational entities have to agree on
(1) which entity fails, i.e. E_f, and
(2) which PDUs from E_f are received by all of them.
In this paper, we would like to present a protocol which provides the agreement on (1) and (2) among all the operational entities.

Each PDU p and E_i have bitmaps $p.BMAP$ and BM, respectively, where each j-th bit denoted by $BMAP_j$ and BM_j represents the states of E_j which E_i knows ($j = 1,\ldots,n$). BM_j is 1 if E_i knows that E_j fails. BM_j

Design of Dynamic Group Communication
Hitoshi Suzuki, Akihito Nakamura, and Makoto Takizawa
Tokyo Denki University

is 0 if E_i knows that E_j is operational.

[Failure Procedure] Suppose that E_i detects that E_f fails. E_i invokes the following procedure. Initially, all the bits in BM are 0.

if (E_i detects E_f fails) {
  $BM_f := 1$; s.BMAP := BM;
  for ($j = 1,\ldots,n$)
    $s.ACK_j := REQ_j$;
}
broadcast($STOP\ s^j$); □

On receipt of a $STOP$ PDU s from E_j, E_i invokes following procedure:

if (BM != s.BMAP){
  $BM := s$.BMAP $\vee\ BM$;
  for ($k = 1,\ldots,n$)
    $AL_k := s.ACK_k$;
indicate(DTO_scheme_view) to A_j;
} □

[Recovery Procedure] Suppose that E_f recovers from the failure. E_f broadcasts an RCV PDU to inform all the operational entity of the recovery. E_f invokes following procedure:

for ($i = 1,\ldots,n$)($i \neq j$) $BM_i := 1$;
$BM_f := 0$; rc.BMAP := BM;
broadcast($RCV\ rc$); □

On receipt of the RCV or RCV_PACK rcp, each E_i invokes following procedure:

if (rcp.BMAP != BM) {
  if (rcp.SRC == E_f)
    $BM := rcp$.BMAP $\wedge\ BM$;
  else $BM := rcp$.BMAP $\vee\ BM$;
}
rcp.BMAP := BM; broadcast(rcp); □

5 Evolutional Cluster

In the evolutional cluster, the number of entities in the cluster may change according to the joining and leaving of entities.

[Join (*grow*) procedure]
(1) *Recruiting* : When a new entity E_i would like to join the existential cluster D after E_i gets the cluster information from the *Cluster Manager*(CM), E_i invokes the same procedure which is invoked when entity recovers.
(2) *Nomination* : Because the CM provides an *escorting* service, any cluster can use it when E_j is required to join the cluster. The following procedure provides the *escorting* service:

CM sends a "*join-us*" PDU to E_j.
If E_j receives the "*join-us*" PDU,
  E_j informs the application entity A_j of it.
If E_j receives an "OK" from A_j,
  E_j invokes the *recovery procedure*. □

[Leave (*shrink*) procedure]
Suppose that E_l decides to leave D. Then, E_l invokes the following procedure:
(1)E_l broadcasts a *leave request* to all of entities in D.
(2)On receipt of the *leave request*, each E_i in D broadcast $ACK.leave$.
(3)After E_l has left, each operational entity E_j invokes a procedure the same as the failure procedure. □

6 Cluster Management

We describe how to manage clusters and how to provide the information of clusters. Figure 2 shows a cluster management system(CM). *MIB*(*Management Information Base*) stores various information on the clusters in the system. *MIB Manager* maintains integrity of *MIB* and authenticates system users, access method, and so on. *MIB Server* updates *MIB* according to the change of the cluster schemes. System entities can get any cluster information because the CM provides a *Notice Board* service where the permitted entities can query the *MIB*.

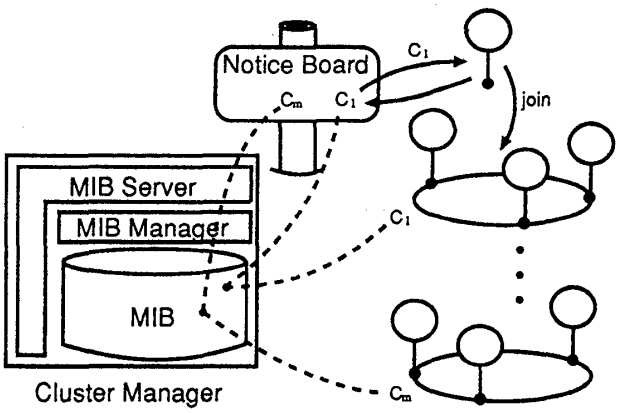

Figure 2: Cluster management environment

7 Concluding Remarks

In this paper, we have discussed the broadcast communication protocols which provide the TO property among all the operational entities in the presence of the cluster configuration change. We have shown two protocols, one for failure detection, and the other for failure recover. They can be executed without terminating the data transmission of the operational entities. Further, we have shown how to manage multi-cluster environment when entities leave and join the cluster.

References

[1] Nakamura, A. and Takizawa, M., "Reliable Broadcast Protocol for Selectively Ordering PDUs," *Proc. of the 11th IEEE International Conf. on Distributed Computing Systems (ICDCS-11)*, 1991, pp.239-246.

[2] Nakamura, A. and Takizawa, M., "Priority-Based Totally Ordering and Semi-Totally Ordering Protocols," *Proc. of the IEEE Conf. on Distributed Computer Systems ICDCS-12*, 1992, pp.178-185.

[3] Takizawa, M. and Nakamura, A., "Totally Ordering Broadcast (TO) Protocol on the Ethernet," *Proc. of the IEEE Pacific RIM Conf. on Communications, Computers and Signal Processing*, 1989, pp.16-21.