マルチメディア通信と分散処理　62－14
（1993. 9. 21）

動的グループ通信プロトコル

鈴木　等, 中村　章人, 滝沢 誠

東京電機大学理工学部経営工学科
埼玉県比企郡鳩山町石坂

あらまし

グループウェア等の分散型応用システムを実現するためには、複数の応用エンティティ間の高信頼なグループ(群)通信が必要となる。群は、通信網の提供する基本通信サービスを用いて、複数のシステムエンティティ間の協調動作により提供される。高信頼な群通信を提供するためには、通信網でのデータ単位の紛失に加えて、システムエンティティの障害に対処できる必要がある。本論文では、提供側のシステムエンティティが停止しても、残りの動作中のエンティティにより、群通信を提供するプロトコルについて論じる。

和文キーワード　プロトコル、グループ通信、分散型システム、放送通信、動的群

Dynamic Group Communication Protocol

Hitoshi Suzuki, Akihito Nakamura, and Makoto Takizawa

Dept. of Information and Systems Engineering
Tokyo Denki University
Ishizaka, Hatoyama, Hiki, Saitama 350-03
tel. 0492-96-2911 ext.2542　fax. 0492-96-6185
e-mail { suzu, naka, taki }@takilab. k. dendai. ac. jp

Abstract　To realize distributed applications like groupware, reliable group (cluster) communication among multiple entities is required. The cluster is provided by the cooperation of underlying multiple system entities using basic communication service which is provided by the communication network. To provide reliable cluster communication, the cope with failure of system entities in addition to the lost of data units on communication network is required. Hence, in this paper we would like to discuss the protocol which provide a reliable cluster communication service by operational entities in the presence of failure of system entities.

英文 key words　protocol, group communication, distributed system, broadcast communication, dynamic cluster

1 はじめに

グループウェア [6] 等の応用では、複数のエンティティ間での協調動作が必要とされる。このためには、OSIプロトコル [18, 26] や TCP/IP[5] 等で提供されている一対のエンティティ間での通信に加えて、複数エンティティから成るグループでの通信が必要となる。本論文では、複数のエンティティから成るグループを群 [20, 21] とする。群通信のためのプロトコルが [1, 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 19, 20, 21, 22, 23, 24, 25] で議論されている。群を利用する応用エンティティが送信するプロトコルデータ単位 (PDU) を、どのような順序で受信するかが問題となる。ここでは、群内の各エンティティで、各々のエンティティから送信された PDU を送信順に受信できる送信順序保存 (OP) サービス [16, 23] を考える。

群通信プロトコル [20, 21] では、群を提供するエンティティが停止すると、群を異常終了する。しかし、分散型応用が冗長要素を含む場合には、ある要素が障害しても、他の冗長要素により処理を継続できる。このような応用では、群を提供しているエンティティが停止した場合にも、正常動作しているエンティティ間に高信頼な群通信サービスを提供し続ける必要がある。

Amoeba[9] では、障害の検出と復旧は、集中的に制御される。エンティティが障害したとき、動作しているエンティティ間で群を再構成する。復旧したエンティティは、集中制御によって群を再構成することで、群に再加入できる。このとき、データ転送を停止する。ISIS[1, 2] では、エンティティの障害は、各サイトで検出される。この後、ある主エンティティの制御の下で、どのエンティティが動作中かについての合意がとられる。エンティティの停止は、そのエンティティを群から除去することを意味し、復旧は、新たに群に加わることと考えられている。これらの合意がとられる間、データ転送が停止される。これらに対して、本論文では、群を、スキーマとインスタンスに分離して考える。スキーマは、群のエンティティ構成を示す。インスタンスは、群が開設されてから終了するまでの状態である。群の状態は、各エンティティの状態の組とし、エンティティの状態は動作中か停止中のいずれかであるとする。スキーマの変化は、エンティティでのプロトコル処理のためのデータ構造を変更する必要があり、実装上の問題がある。本論文では、データ転送を停止せずに、エンティティの停止、復旧によりインスタンスが変化しても、動作中エンティティにより完全分散型の環境下でOPサービスを提供するプロトコルを示す。通信網は、高信頼であり、PDU の紛失はなく、送信順序が保存されるOPサービスを提供するものとする。

2章では、本論文で用いる基本概念の定義を行う。3章では、DOPプロトコルについて述べる。

2 基本定義

本論文で用いる基本概念の定義を行う。

2.1 送信順序保存 (OP) サービス

各エンティティは、下位層が提供するサービスアクセス点 (SAP) を通じて、他のエンティティと通信を行う。SAP の組 $\langle S_1, \ldots, S_n \rangle$ を群Cとする ($n \geq 2$)。エンティティE_kは、S_kでサービスを提供する ($k = 1, \ldots, n$)。各S_kで送信された PDU は、C内の全 SAP に届けられる。ここで、E_kは Cに**含まれる**とし、$C = \langle E_1, \ldots, E_n \rangle$ と書く。

各E_kが利用する群Cのサービスを、ログの集合としてモデル化する [23, 24]。ログLは、PDU の系列 $<p_1, \ldots, p_m]$ である。ここで、p_1をLの先頭 ($top(L)$)、p_mを最後尾 ($last(L)$) とする。L内で $i < j$ ならば、p_iはp_jに先行するとする。各E_kは、送信ログSL_kと受信ログRL_kを持つ ($k = 1, \ldots, n$)。

OPサービス [16, 23] では、C内の各 SAP で送信された全PDU は、全SAP で送信順に受信される。RL_iの副受信ログRL_{ij}を、E_jから受信した PDU の系列とする。OPサービスでは、$RL_{ij} = SL_j$であるが、任意のiとjについて、$RL_i = RL_j$とは限らない。OPサービスの例を以下に示す。

[例] $C = \langle E_1, E_2, E_3 \rangle$ とする。E_1、E_2、E_3の送受信ログを図1に示す。例えば、SL_1は、E_1が a、b、cの順に PDU を送信し、RL_1は、a、p、x、b、c、yの順に受信したことを示す。図1に示すように、OP サービスを利用するE_1、E_2、 E_3は、各々のエンティティからの PDU を、紛失することなく送信順序を保存して受信する。□

E_1 RL_1: $< a\ p\ x\ b\ c\ y\ ]$ SL_1: $< a\ b\ c\ ]$
E_2 RL_2: $< p\ a\ b\ x\ y\ c\ ]$ SL_2: $< p\ ]$
E_3 RL_3: $< a\ p\ b\ x\ c\ y\ ]$ SL_3: $< x\ y\ ]$

図 1: OP サービスの例

2.2 システムモデル

本システムは、網、システム、応用の3層から構成される [図 2]。網SAP(NSAP) $S_1, \ldots, S_n$の集合を網群Cとする。Cは、システム層のエンティティ$E_1, \ldots, E_n$に、各々$S_1, \ldots, S_n$を通して、OPサービスを提供する。各E_iは、応用エンティティA_iに、システムSAP(SSAP) D_iを通して、ある通信サービスを提供する $(i = 1, \ldots, n)$。

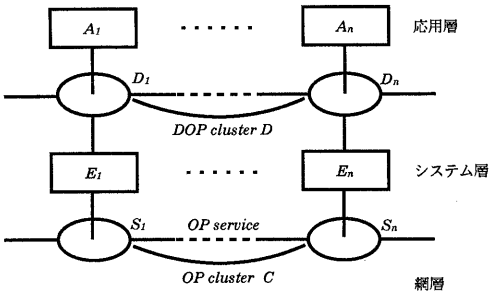


図 2: システム階層

システム群Dは、$D_1, \ldots, D_n$間に存在する。各D_iは、システムエンティティE_iにより提供される。Dには、**スキーマ**と**インスタンス**との2つの側面がある。Dのスキーマは、エンティティの構成を示し、$D = \langle E_1, \ldots, E_n \rangle$と書く。これに対して、$D$のインスタンスは、群が開設された状態である。$E_i$の**停止**とは、$E_i$が一定時間以上、動作しなくなることである。停止中から動作中となることをE_iの**復旧**とする。ここでは、エンティティのビザンチン障害[10]を考えない。このとき、E_iが停止前の状態をどの程度記憶しているかが問題となる。何も記憶していないとき、**完全停止**とする。少なくともDのスキーマについて知っているとき、**部分停止**とする。各E_iの状態を$state(E_i)$と書く。$state(E_i) \in \{$ 動作中(A), 停止中$(S)\}$である。Dのインスタンス状態を$state(D)$とし、動作中のエンティティの組$\langle\langle E_{a1}, \ldots, E_{ah} \rangle\rangle (\subseteq D)$により示す。

従来の放送通信サービス[13, 14, 17, 20, 21, 22]では、あるE_iが停止するとDを終了、即ち、インスタンスを消去していた。冗長な要素を含むシステムでは、ある要素が故障しても残りの冗長要素により、動作を続けられる。このような分散型応用を実現するためには、E_iが停止しても、他の動作中エンティティ間で放送通信サービスを提供し続ける必要がある。このような群を**動的インスタンス群**とする。即ち、動的インスタンス群では、スキーマは変化しないが、インスタンスは変化する。あるエンティティが停止した場合に、群インスタンスが消滅する群[20, 21]を**静的群**とする。一方、新たにエンティティが群に加入、離脱することにより、スキーマが変化する群を**動的スキーマ群**とする。本論文では、OPサービスを提供する動的インスタンス群について論じる。

群Dのインスタンスが存続し得るために、動作中でなければならないエンティティを、Dで**必須**とする。さらに、D内で一定数以上のエンティティが動作中する必要がある場合もある。また、冗長なエンティティの中で一つでも動作しているときは、処理し続けられる場合もある。本論文では、インスタンスが存続できるかどうかは、上位の応用エンティティが決めるものとする。

2.3 動的群$OP(DOP)$ サービス

DOP群$D = \langle E_1, \ldots, E_n \rangle$では、動作中のエンティティにより、$OP$サービスが提供される。ある$E_i$が停止しても、残りの動作中エンティティにより、$OP$サービスが提供される。各$E_i$からみて動作中であるエンティティの組を、$E_i$の$state(D)$の視野とし、$state(D)_i$と書く。$E_i$からみた、$E_j$の状態を$state(E_j)_i$と書く。

動作中のE_iは、D_iでOPサービスを提供する。このとき、DをDOP群とし、提供されるOPサービスをDOPサービスとする。

3 動的群$OP(DOP)$ プロトコル

下位のOPサービスを利用して、DOPサービスを提供するプロトコルについて述べる。

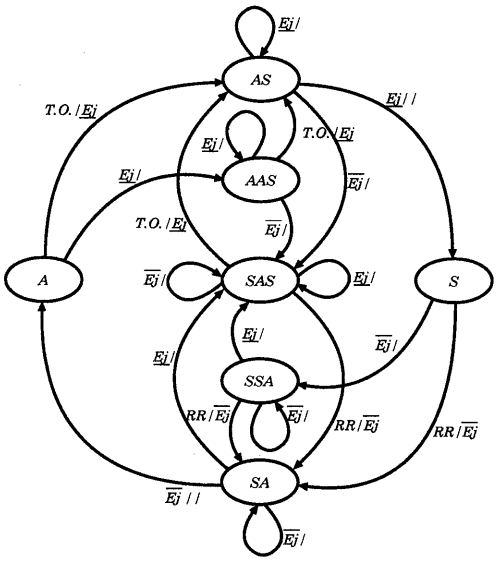
3.1 エンティティ状態

各エンティティE_iからみたE_jの状態$state(E_j)_i$について考える。E_iがE_jから一定時間PDUを受信しないとき、E_iはE_jの**停止を認識する**とする。このとき、$state(E_j)_i$を**停止合意中**(AS)とし、E_jの停止を他のエンティティ

に通知する。$state(E_j)_i = A$のときに、E_iが他のE_kより、E_jの停止通知を受ける場合がある。E_iは、E_jの停止通知を受けても、E_jが停止したとは認識しない。E_iがE_jの停止を認識するのは、E_iがE_jから一定時間PDUが届かない場合である。この状態$state(E_j)_i$を**停止認識中**(AAS)とする。以上から、E_iがE_jの停止について合意するための条件は以下である。

(1) E_iがE_jの停止を認識する。

(2) 全動作中のエンティティから、E_jの停止認識通知を受信する。



$\overline{E_j}$: E_jの復旧通知、 $\underline{E_j}$:E_jの停止通知

図 3: $state(E_j)_i$の遷移

次に、部分停止していたE_jが復旧する場合を考える。復旧したE_jは、RCV PDUを放送する。E_jの復旧は、E_jからRCVを受信することによりわかる。E_iが、E_jからRCVを受信したとき、E_i はE_jの**復旧を認識する**とする。E_jの復旧通知を全動作中エンティティに放送する。このとき、$state(E_j)_i$を**復旧合意中**(SA)とする。停止と同様に、他のエンティティから、E_jの復旧通知を受けても、E_iはE_jが復旧したとは認識しない。このとき、$state(E_j)_i$を、**復旧認識中**(SSA)とする。E_iは以下が成り立つとき、E_jの復旧に合意したとする。

(1) E_iは、E_jの復旧を認識する。

(2) E_iは、全動作中エンティティから、E_jの復旧認識通知を受信する。

各E_iからみてた動作中エンティティの集合を$A_i(\in D)$とする。S_i、AS_i、AAS_iを各々、E_iからみた停止中、停止合意中、停止認識中のエンティティ集合とする。SA_i、SSA_iを各々、E_iからみた復旧合意中、復旧認識中のエンティティ集合とする。ここで、$AS_i \cap AAS_i = \phi$、$SA_i \cap SSA_i = \phi$である。$AA_i = A_i \cup AAS_i \cup SA_i$は、$E_i$が動作認識しているエンティティの集合である。一方、$SS_i = S_i \cup AS_i \cup SSA_i$は、$E_i$が停止認識しているエンティティの集合である。

各E_iが送信するPDU pは、以下の項目から構成される。

$p.SRC$ = 送信元のエンティティ (E_i)。

$p.SEQ$ = PDUのシーケンス番号。

$p.ACK_j$ = E_iがE_jから次に受信予定のPDUのシーケンス番号 ($j = 1, \ldots, n$)。

$p.STATE$ = $state(D)_i$を示すビットマップ。

$p.TRANS$ = 状態の変化を示すビットマップ。

ここで、ビットマップBに対して、$B[i]$はi番目のビットを示す。ここで、ビットマップの積、和、差、排他的論理和、否定を各々 &、|、 −、⊕、 ～により示す。各E_iは、$state(D)_i$を示すビットマップ$STATE$と、状態遷移を示すビットマップ$TRANS$、他のE_jから次に受信予定のPDUのシーケンス番号REQ_j ($j = 1, \ldots, n$)を持つ。$STATE$は、E_iが動作中と認識しているエンティティの集合AA_iを示す。即ち、$E_j \in AA_i$のとき、$STATE[j] = 0$で、そうでないとき、$STATE[j] = 1$である。$TRANS$は、E_iが停止又は復旧合意中のエンティティ集合$AS_i \cup SA_i$を示す。即ち、$E_j \in AS_i \cup SA_i$ならば、$TRANS[j] = 1$、そうでなければ、$TRANS[j] = 0$である。E_iは、$p.STATE := STATE$、$p.TARNS := TRANS$としたPDU pを放送する。これにより、E_iは他のエンティティに$state(D)_i$を通知する。

3.2 完全停止

まず、エンティティが完全停止する場合を考える。一度停止したエンティティは、D内に復旧することはない。即ち、エンティティの停止合意だけを考えればよい。このため、図3の状態の中で、A、AS、 AAS、Sのみを考

える。

以下に、停止エンティティを検出し、動作中のエンティティ間で正しい送受信を行なうための手続きを示す。本方式は、動作中のエンティティのデータ転送を停止せずに動作できる。ここで、$broadcast(p)$ はPDU pを、通信網サービスを利用して放送することを示す。

[停止合意手続き] (1) E_iが、E_hの停止を検出したとき、$ST\text{-}ATE[h] := TRANS[h] := 1$、$state(E_h)_i := AS$とする。また、$s.ACK_j := REQ_j (j = 1, \ldots, n)$、$s.STATE := STATE$、$s.TRANS := TRANS$なる$STOP$ PDU sをつくる。$broadcast(s)$;

(2) sを受信した各E_jは、$(s.STATE \ \& \ s.TRANS)[h] = 1$なる$E_h$に対して、$STATE = 0$、$TRANS = 0 (state(E_h)_i = A)$ならば、$state(E_h)_j := AAS$。また、タイムアウトにより$E_j$が$E_h$の停止を認識したとき、$STATE[h] := TRANS[h] := 1 (state(E_h)_j := AS)$。$sa.STATE := STATE$、$sa.ACK_j := REQ_j (j = 1, \ldots, n)$なる$STOP_ACK$ saをつくる。$broadcast(sa)$;

(3) 各E_iは、動作中の全$E_j (\in A_i)$から$STOP_ACK$ saを受信したとき、saの$STATE$、$TRANS$、REQが、E_iと同一か調べる。異っていれば、(2) を行う。同一であれば、動作中の全エンティティがE_hの停止を認識していることがわかる。このとき、$state(D)$について合意がとられたことになり、$state(E_h)_i := S$とする。また、停止中の各E_hから受信したPDUのうち、$p.SEQ \geq REQ_h$なるPDU pをログから消去する。

(4) 各E_iは、$STATE[h] = 1$なるE_hについての送受信処理を行わない。□

[停止合意手続きの例] 図4に、$D = \langle E_1, E_2, E_3, E_4 \rangle$の停止合意手続きの例を示す。

(1) E_3が停止したとする。E_3からのPDUを一定時間受信しないことで、E_1は、E_3の停止を検出する。E_1は、$STATE[3] := TRANS[3] := 1 (state(E_3)_1 := AS)$とし、$s.STATE := STATE$、$s.TRANS := TRANS$なる$STOP$ PDU sを放送する。

(2) E_1からのsを受信した動作中の各$E_k (k = 1, 2, 4)$は、$(s.STATE \ \& \ s.TRANS)[3] = 1$である$E_3$に対して、$state(E_3)_k = AAS$とし、$E_3$についてのタイムアウトを待つ。各$E_k$は、各々のタイムアウトにより$E_3$の停止を認識した後、$state(E_3)_k = AS$、即ち各$E_k$で、$STATE := 0100$、$TRANS := 0100$とする。$sa.STATE := STATE$、$sa.TRANS := TRANS$なる$STOP_ACK$ saを放送する。

(3) E_kは、A_k内の全エンティティからsaを受信したとき、全動作中エンティティが同一の$STATE$を持つことがわかる。このとき、$state(D)$についての合意がとられたことになり、各E_kは、$state(E_3)_k = S$とする。$TRANS[3] := 0$。ここで、各E_kは、E_3から受信したPDU pのうち、$p.DSEQ \geq REQ_3$なる全てのpを受信ログ内から廃棄する。□

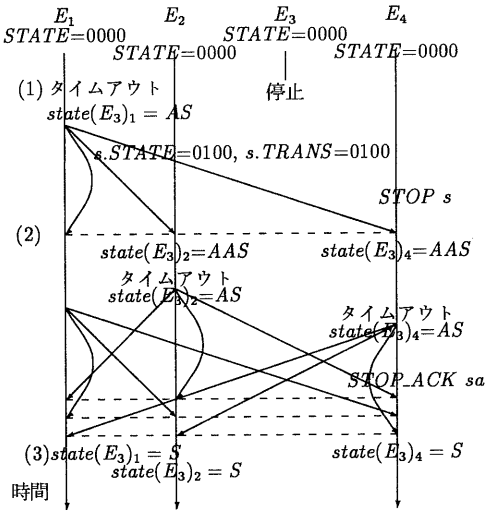


図 4: 停止合意手続きの例

3.3 部分停止

本節では、エンティティが部分停止する場合を考える。即ち、エンティティが停止し、後に復旧したとき、群Dの構成についての記憶があり、再びD内で通信を行う場合がある。従って、エンティティの停止合意中に、他のエンティティが復旧する場合と、復旧合意中に、動作中のエンティティが停止する場合とを考えねばならない。以下に、D内のあるエンティティが停止し、後に復旧したとき、A_i内の全エンティティ間でインスタンス変化に対して合意するための手続きを示す。

E_jが停止したとする。E_iは以下の手順によりE_jの停止

に合意する。

[停止合意手続き]

(1) $state(E_j)_i = A$のとき、

(a) タイムアウトにより、E_jの停止を認識:

$STATE[j] := TRANS[j] := 1;$
$state(E_j)_i := AS;$
$A_i := A_i - \{E_j\};\ AS_i := AS_i \cup \{E_j\};$
$s.STATE := STATE;$
$s.TRANS := TRANS;$

(b) E_jの停止通知の受信:

$state(E_j)_i := AAS;$
$A_i := A_i - \{E_i\};$
$AAS_i := AAS_i \cup \{E_j\};$

(2) $state(E_j)_i = AAS$で、E_jについてのタイムアウト:

$STATE[j] := TRANS[j] := 1;$
$state(E_j)_i := AS;$
$AS_i := AS_i \cup \{E_j\};$
$AAS_i := AAS_i - \{E_j\};$
$sa.STATE := STATE;$
$sa.TRANS := TRANS;$

(3) $state(E_j)_i = AS$のとき: A_i内の全E_jについて、$state(E_j)_i = state(E_j)_i$のとき、$E_j$の停止についての合意する:

$TRANS[j] := 0;\ state(E_j)_i := S;$
$S_i := S_i \cup \{E_j\};\ AS_i := AS_i - \{E_j\};$□

次にE_jが停止から復旧したとする。E_iは以下の手順によりE_jの復旧に合意する。

[復旧合意手続き]

(1) $state(E_j)_i = S$のとき、

(a) E_iが、E_jからRCVを受信(E_jの復旧認識):

$STATE[j] := 0;\ TRANS[j] := 1;$
$state(E_j)_i := SA;$
$S_i := S_i - \{E_j\};\ SA_i := SA_i \cup \{E_j\};$
$ra.STATE := STATE;$
$ra.TRANS := TRANS;$

(b) E_iが、E_jの復旧通知の受信:

$state(E_j)_i := SSA;$
$S_i := S_i - \{E_j\};$
$SSA_i := SSA_i \cup \{E_j\};$

(2) $state(E_j)_i = SSA$で、E_jからRCVの受信:

$STATE[j] := 0;\ TRANS[j] := 1;$
$state(E_j)_i := SA;$
$SA_i := SA_i \cup \{E_j\};\ SSA_i := SSA_i - \{E_j\};$

(3) $state(E_j)_i = SA$で、A_i内の全E_kについて、$state(E_j)_k = state(E_j)_i$ならば、$E_j$の復旧について合意する:

$TRANS[j] := 0;\ state(E_j)_i := A;$
$A_i := A_i \cup \{E_j\};\ SA_i := SA_i - \{E_j\};$□

[例] 図5に、$D = \langle E_1, E_2, E_3, E_4 \rangle$の部分停止合意手続きの例を示す。はじめに、$E_3$が停止しており、$D$内で合意されているとする。即ち$state(E_3)_k = S$である。$E_3$が復旧するとする。$E_1$は、$E_2$の停止合意中に$E_3$の復旧通知を、$E_4$は、$E_3$の復旧合意中に$E_2$の停止通知を受信する。このとき、$E_1$と$E_4$ が、E_2とE_3の状態変化について合意するまでの手続きの例を以下に示す。

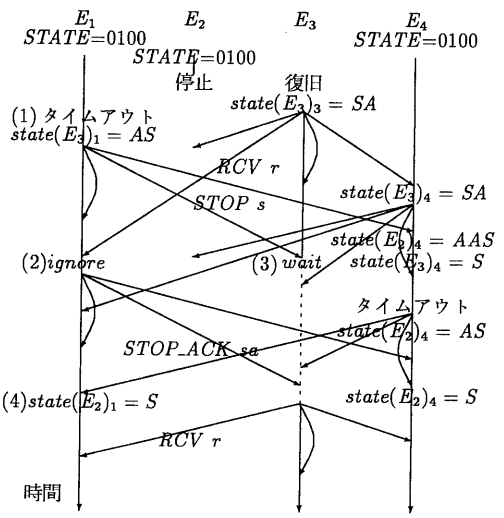


図 5: 部分停止合意手続きの例

(1) E_2が停止し、E_3が復旧したとする。このとき、E_3は、$STATE := 1011$、$TRANS := 0100$、即ち$state(E_3)_3$

:= SAとし、$RCV\ r$を放送する。E_4は、RCVを受信し、$STATE$:= 0000、$TRANS$:= 0100、即ち、$state(E_3)_4$:= SAとした後、$RCV_ACK\ ra$を放送する。また、E_1は、タイムアウトによりE_2の停止を検出し、$STATE$:= 0110、$TRANS$:= 0010、即ち$state(E_2)_1$:= ASとして、$STOP\ s$を放送する。

(2) E_3は、E_1からの$STOP$を受信したならば、一定時間待機($wait$)した後、RCVを再送する。E_1は、停止合意中にE_3とE_4から受信した、RCVとRCV_ACKのそれぞれを無視する。E_4は、E_3の復旧合意中にE_1から受信した$STOP$により、RCVを廃棄し、$state(E_2)_4$:= AAS、$STATE$:= 0100、$TRANS$:= 0000、即ち$state(E_3)_4$:= Sとする。

(3) E_4は、E_2のタイムアウトを待った後、$state(E_2)_4$:= ASとし、$STOP_ACK$を放送する。E_3は、E_1 からの$STOP$を受信していなければ、E_1とE_2からのRCV_ACKを待つことでタイムアウトする。このとき、E_3は、(2) と同様に一定時間待機した後、RCVを再送する。

(4) E_1とE_4は、各々の$STOP_ACK$を受信することで、E_2の停止について互いに合意していることがわかり、それぞれ$STATE$:= 0110、$TRANS$:= 0000、即ち$(state(E_3)_1 = state(E_3)_4) := S$とする。

(5) この後、E_3がRCVを再放送し、E_3の復旧についての合意がとられる。ここでE_3は、E_1とE_4から受信したPDU pのうち、復旧合意がなされる以前に受信したpを受信ログ内から廃棄する。□

4 まとめ

本論文では、群を、群の構成を示すスキーマと、状態を示すインスタンスとに分離した。群内のエンティティが停止し復旧する場合がある動的インスタンス群に対して、動作中の全エンティティ間でOPサービスを提供するDOPプロトコルを示した。エンティティの障害として、スキーマの情報も消失してしまう完全停止と、スキーマを記憶している部分停止とを考えた。部分停止したエンティティが復旧した場合には、群D内で通信を再開することがある。本論文では、完全停止障害に対する停止合意手続きと、部分停止障害に対する停止合意手続きと復旧の合意手続きを示した。これらは、動作中のエンティティのデータ転送を停止せずに実行できる分散型の手続きである。

参考文献

[1] Birman, K. and Joseph, J., "Reliable Communication in the Presence of Failure," *ACM Trans. on Computer Systems*, Vol.9, No.1, 1987, pp.47-76.

[2] Birman, K., Schiper, A., and Stephenson, P., "Lightweight Causal and Atomic Group Multicast," *ACM Trans. on Computer Systems*, Vol.5, No.3, 1991, pp.272-314.

[3] Chang, J. M. and Maxemchuk, N. F., "Reliable Broadcast Protocols," *ACM Trans. on Computer Systems*, Vol.2, No.3, 1984, pp.251-273.

[4] Chanson, S., Neufeld, G., and Liang, L., "A Bibliography on Multicast and Group Communications", *ACM SIGOPS Operating Systems Review*, Vol.23, No.4, Oct. 1989.

[5] Defense Communications Agency, "DDN Protocol Handbook," *NIC* 50004-50005, Vol.1 - 3, 1985.

[6] Ellis, C. A., Gibbs, S. J., and Rein, G. L., "Groupware," *Comm. of ACM*, No.1, 1991, pp.38-58.

[7] Garcia-Molina, H. and Kogan, B., "An Implementation of Reliable Broadcast Using an Unreliable Multicast Facility," *Proc. of the IEEE Symp. on Reliable Distributed Systems*, 1988, pp.428-437.

[8] Garcia-Molina, H. and Spauster, A., "Message Ordering in a Multicast Environment," *Proc. of the IEEE ICDCS-9*, 1989, pp.354-361.

[9] Kaashoek, M. F., Tanenbaum, A. S., Hummel, S. F., and Bal, H. E., "An Efficient Reliable Broadcast Protocol," *ACM Operating Systems Review*, Vol.23, No.4, 1989, pp.5-19.

[10] Lamport, R., Shostak, R., and Pease, M., "The Byzantine Generals Problem," *ACM Trans. on Programming Language and Systems*, Vol.4, No.3, 1982, pp.382-401.

[11] Luan, S. W. and Gligor, V. D., "A Fault-Tolerant Protocol for Atomic Broadcast," *IEEE Trans.*

on Parallel and Distributed Systems, Vol.1, No.3, 1990, pp.271-285.

[12] Melliar-Smith, P. M., Moser, L. E., and Agrawala, V., "Broadcast Protocols for Distributed Systems," *IEEE Trans. on Parallel and Distributed Systems*, Vol.1, No.1, 1990, pp.17-25.

[13] Nakamura, A. and Takizawa, M., "Reliable Broadcast Protocol for Selectively Ordering PDUs," *Proc. of the IEEE ICDCS-11*, 1991, pp.239-246.

[14] Nakamura, A. and Takizawa, M., "Design of Reliable Broadcast Communication Protocol for Selectively Partially Ordered PDUs," *Proc. of the IEEE COMPSAC91*, 1991, pp.673-679.

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

[16] 中村 章人, 滝沢 誠: 多チャネルシステム上の送信順序保存放送通信プロトコル, 情報処理学会論文誌, Vol.34, No.1, pp.135-143 (1993).

[17] 中村 章人, 滝沢 誠: 多チャネルシステム上の選択的放送通信プロトコルの データ転送手続き, 情報処理学会論文誌, Vol.33, No.2, pp.223-233 (1993).

[18] ISO "Data Processing – Open Systems Interconnection – Basic Reference Model," *ISO 7498*, 1987.

[19] Schneider, F. B., Gries, D., and Schlichting, R. D., "Fault-Tolerant Broadcasts," *Science of Computer Programming*, Vol.4, pp.1-15, 1984.

[20] Takizawa, M., "Cluster Control Protocol for Highly Reliable Broadcast Communication," *Proc. of the IFIP Conf. on Distributed Processing*, 1987, pp.431-445.

[21] Takizawa, M., "Design of Highly Reliable Broadcast Communication Protocol," *Proc. of the IEEE COMPSAC87*, 1987, pp.731-740.

[22] 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.

[23] Takizawa, M. and Nakamura, A., "Partially Ordering Broadcast (PO) Protocol," *Proc. of the IEEE INFOCOM90*, 1990, pp.357-364.

[24] Takizawa, M. and Nakamura, A., "Reliable Broadcast Communication," *Proc. of IPSJ Int'l Conf. on Information Technology (InfoJapan)*, 1990, pp.325-332.

[25] 滝沢 誠, 中村 章人: 1チャネルシステム上の全順序放送通信プロトコルにおける データ転送手続き, 情報処理学会論文誌, Vol.31, No.4, pp.609-616 (1990).

[26] Tanenbaum, A., "Computer Networks," *Prentice-Hall Int'l, Inc.*, 1987.