マルチメディア通信と分散処理　50－4
（1991. 5. 16）

OSI トランザクション処理における回復制御の実装

谷林 陽一、 楠 和浩、 中川路 哲男、 水野 忠則

三菱電機 (株) 情報電子研究所

現在我々は OSI トランザクション処理システムの実装を行なっている。本論文では、OSI-TP における 回復専用の通信路である TP チャネルと、 状態を記録するためのログに関する機能の設計および実装 について述べる。「TP チャネル管理オブジェクト」は、 ノード内の全ての TP チャネルを管理し、 確立および解放を行なう。「ログ管理オブジェクト」は、 ログへのアクセスを管理する。 また、ログ管理オブジェクトは、再起動時にログを読み、 回復制御も行なう。これらの回復制御機能により、 通信障害および システムクラッシュ に対して信頼性の高い 分散トランザクション処理システムを実現できた。

Implementing Recovery Control for OSI Transaction Processing

Yoichi TANIBAYASHI, Kazuhiro KUSUNOKI,
Tetsuo NAKAKAWAJI, Tadanori MIZUNO

Computer & Information Systems Laboratory,
MITSUBISHI ELECTRIC CORPORATION

We are now implementing OSI Distributed Transaction Processing system. In this paper, we describe design and implementation of communication channels for recovery, called *TP channels*, and functions to control all accesses to *Log* records that store states of transactions. *TP channel manager object* manages the TP channels in one node, and establishes and terminates them. *Log manager object* controls all accesses to log records, and, when the system is rebooting after system crash, it reads log records and recovers the transactions. These objects and their recovery control functions realized a reliable distributed transaction processing system against communication failures or system crushes.

1 はじめに

現在、我々の回りには、銀行の CD、新幹線や飛行機の予約などのように、処理するデータを通信回線を経由して離れた場所にあるコンピュータに転送し、処理された結果をできるだけ素早く通信回線経由で処理の依頼元へ返すという トランザクション処理 (以後 TP) が多数存在する。しかしながら、各ベンダが独自のシステムを提供しているため、相互に通信回線を経由して接続することができず、標準化が求められている。

ISO (国際標準化機構) では、ディレクトリサービス、ネットワーク管理など各種の業務に応じた OSI (開放型システム間相互接続) における応用層プロトコルの標準化が進展している。これにともない現在、FTAM (ファイル転送) や MOTIS (電子メール) などが使われ始めており、徐々に OSI の市場も広がりつつある。データ通信の 8 割から 9 割を占めるといわれている TP のための **OSI–TP** を実現することによりこの傾向はますます増すと思われる。OSI–TP の標準化は現在、**DIS (Draft International Standard)** の段階まできており、まもなく国際標準となる見込みである。

これに先立ち、我々は現在 OSI–TP の実装を行なっている。TP を実用的なシステムにするには、高信頼性、すなわち、回復処理が必要不可欠な要素となる。

そこで本論文では、OSI-TP の回復処理の設計および実装について述べる。

本論文では、第 2章で TP について概説し、第 3章では、トランザクションにおける回復処理の解説および 問題点の提示を行なう。第 4章では、前章までの議論を基に、本システムの設計および実装について述べる。そして、第 5章で、結論を述べる。

2 トランザクション処理

本章では、まず一般的なトランザクションについて説明し、次に、OSI でこれがどのように規定されているかについて述べる。

2.1 TP の概要

TP とは、一般に、銀行の CD、新幹線や飛行機の予約などのように、処理するデータを通信回線を経由して離れた場所にあるコンピュータに転送し、処理された結果をできるだけ素早く通信回線経由で処理の依頼元へ返すという処理のことである。TP の特徴は、同時多発のデータを短時間で処理し応答を返すことができ、更に信頼性が高いということである。

TP には、次に挙げる 4 つの条件がある。

原子性 (atomicity) : 一連の操作が全て実行されるか、何も実行されないかのいずれかしかないこと。

一貫性 (consistency) : 一連の操作で複数のデータを変更する場合、それぞれのデータが互いに矛盾しない状態を維持すること。

独立性 (isolation) : 一連の操作によって導かれた途中結果に対して、他の操作によってこの途中結果が利用されないこと。

耐久性 (robustness) : 一連の操作が終了すると、それによるあらゆる効果は、以後消えることがないこと。

いくつかの地理的に離れたアプリケーションプロセスがネットワークで通信しながら協力し合い、上に挙げた 4 つの条件を満たす仕事をする場合、これを分散トランザクション (distributed transaction) と呼ぶ。

2.2 OSI–TP の概要

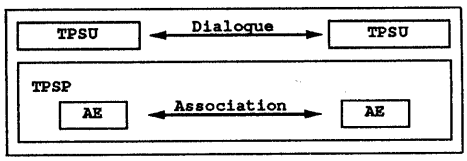


図 1: ダイアログとアソシエーションの関係

OSI が規定している TP では、TP サービスの利用者を **TPSU (Transaction Processing Service User)**、サービスの提供者を **TPSP (Transaction Processing Service Provider)** と呼ぶ。お互いに通信し合う 2 つの TPSU の関係を ダイアログ と呼ぶ。OSI の応用層では、ダイアログの下位の関係として、アソシエーション があり、ダイアログと 1 : 1 に対応しているが、確立、解放は独立に行なわれる (図 1)。

分散した TPSU が協調し合って TP を行なう場合、TPSU の関係は トランザクション処理木 (図 2) と呼ばれる木構造を形成する。

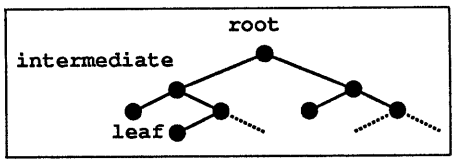


図 2: トランザクション処理木

2 相コミットメント

トランザクションにおける一連の操作を完了し、状態を確定する操作は コミットメント 操作と呼ばれてい

障害の種類	障害の内容	OSI としての現象	回復処理	
プログラム障害	TPSU の障害	なし	資源を矛盾のない状態にし、ダイアログを終了する	×
通信障害	アソシエーション障害	アソシエーションの異常解放	資源を矛盾のない状態にし、ダイアログを終了する	○
応用プロトコル障害	応用プロトコルにおけるエラー	プロトコル違反	資源を矛盾のない状態にし、ダイアログを終了する	×
システムクラッシュ	アソシエーション、TPPM、TPSUI の障害	複数のアソシエーションの異常解放	資源を矛盾のない状態にし、ダイアログを終了する	○
記憶媒体障害	バウンドデータやログの障害	なし	資源をバックアップなどから再現し、資源を矛盾のない状態にする	×
デッドロック	デッドロック	なし	資源をロールバックする	×

本システムでは、TPSP の回復処理について考えているので、「○」のついている項目を対象とし、「×」ついては考慮しない。

表 1: 障害の種類

る。OSI-TP におけるコミットメント プロトコルの基本は、**2 相コミットメントプロトコル** である。

2 相コミットメント プロトコルでは、まず第 1 相で、トランザクションに関係している全プロセスをコミット可能な準備状態にする (**Ready 状態**)。全プロセスが準備状態になると、第 2 相で、実際にコミットを実行し、状態を確定する (**Commit 状態**)。

プリジュームド ロールバック

2 相コミットメント プロトコルを実行している途中で、障害が発生した場合、TPSP は、何らかの回復制御を行ない、トランザクションの「原子性」および「一貫性」という条件の下で、全てのプロセスの状態を確定しなければならない。このために、状態を記録した **ログ** と 回復専用の通信路である **TP チャネル** を利用する。

障害が発生した場合、TPSP は、ログの情報をもとに、全プロセスを一貫した状態に確定しようとする。このとき OSI-TP では、ログが記録されていなかった場合は、トランザクションのロールバックを行なう。これを **プリジュームド ロールバック プロトコル** と呼ぶ。

プリジュームド ロールバック プロトコルでは、スーペリア が サブオーディネイト を ログに記録するのは、サブオーディネイトが COMMIT することに同意したときのみである。この記録は、ルートでは **COMMIT レコード**、中間ノードでは **READY レコード** になされる。

3 TP における回復制御

本章では、前章で述べた TP において回復制御を実現するために必要な機能および特に実装において検討すべき問題点について述べる。

本研究では、TPSP の回復制御を対象とし、ここで考えなければならない障害として「システムクラッシュ」と「通信障害」を挙げ、これらについて詳しく検討する。具体的な解決方法については、4章で述べる。

3.1 障害の種類

TP 実行中に起こり得る障害としては、一般に表 1に示す 6 つが考えられる。ただし、本研究では、通信プロトコルを制御する TPSP および通信路 の回復制御を対象としている。したがって、「プログラム障害」、「記憶媒体障害」、「デッドロック」 については、通信路の状態としては現象が現れないため、考慮しない。「応用プロトコル障害」は、各応用プロトコル毎に設計するので、本研究では、TPSP として考えなければならない障害として、通信路の切断という最も深刻な状況を伴う「通信障害」 と「システムクラッシュ」 の 2 つを対象とする。

3.2 通信障害

通信障害とは、あるアソシエーションに障害が起き、そのアソシエーションで通信を続行できなくなる障害である。この場合、OSI-TP では **TP チャネル** と呼ぶ回復用の通信路を用いる。TP チャネルは、TPSP にとっては、ダイアログとほとんど同じだが、以下のような特徴がある。

- TPSU には、見えない。

- リカバリ専用の通信路で、 *C-RECOVER* メッセージのみが交換される。
- 回復処理が終了しても直ちに解放する必要はなく、 他のトランザクションの回復処理に 使用しても良い。
- 複数の回復処理を同時に実行できない。

TP チャネルを実現するためには、 次の 3つの機能が必要となる。

1. その時点での回復制御の必要に応じて、 TP チャネルを確立、解放する機能
2. TP チャネル上で回復制御情報を交換する機能
3. トランザクションを一貫した状態に確定する (コミットまたはロールバック) 機能

この場合、以下のことが問題となる。

- TP チャネルの確保および解放
- TP チャネルの管理
- TP チャネル上のイベントの処理

3.3 システムクラッシュ

システムクラッシュ、 すなわちプロセスまたはマシンに障害が生じた場合、 そのノードを再起動し処理を続行しなければならない。しかし、この場合、 メモリ中に保持されているトランザクションの状態や通信路に関する情報は障害とともに消滅してしまうため、 これら必要な情報を stable なログに保存し、 回復に備えなえなければならない。したがって、 システムクラッシュに対する回復処理を実現するには、 次の 4つの機能が必要となる。

1. ログに状態を書き込む機能
2. ログから回復処理の必要なトランザクションを識別する機能
3. ログの情報をもとに、 トランザクションの状態を再現する機能
4. 3.2節で述べたように 通信路の回復を行なう機能

これらのうち、 4以外は、全てログに関することで、 次のような問題を解決しなければならない。

- ログの内容
- ログに書き出すタイミング
- 回復時に、ログから必要な情報を取り出す方法
- ログの情報をもとに、トランザクションを再現する方法

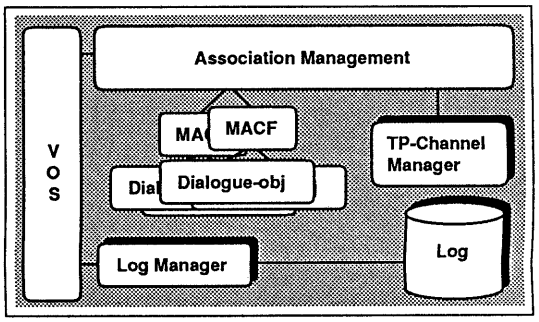
4 設計および実装

一般のプロトコル処理は、 ほとんどが受動的なもので、規格に定められた状態遷移に従った処理を 行なえば良いが、 回復制御については、 能動的な要素が多く、 規格で定められていない部分もかなりある。

3章では、 TP における回復制御を実現する上で TP チャネルおよびログ に関する処理が必要であることを述べ、 その問題点を検討した。本章では、 まず、本システムの全体像を示し、 次に、 回復制御を実現する上での具体的な問題点と、 その解決案としての本システムの設計および実装について述べる。

4.1 基本設計

本システム主要部の構成を図 3に示す。 この図におい



影のついた部分が今回の実装で作成された部分を表している。
(VOS: Virtual Operating System)

図 3: システム主要部の構成図

て、 影のついた部分は今回新たに作成した部分である。それ以外は、 [楠 89] で述べられているものとほぼ同じである。

以下、図 3の各モジュールについて概説する。

TP-Channel Manager (TP チャネル管理オブジェクト)[新規]

システムに常駐し、 回復処理用の TP チャネルの確立、解放、状態遷移をテーブルをもとに管理する。

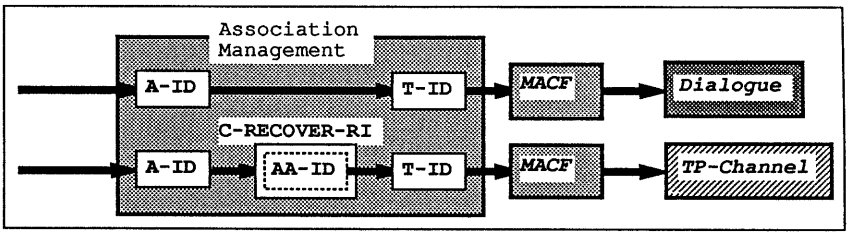
Log(ログ)[新規]

2 次記憶上のファイル。 トランザクションや通信路の状態が記録されている。

Log Manager (ログ管理オブジェクト)[新規]

ログの読み書き、 検索 など、 ログへの実際のアクセスを受け持つ。

Association Management



A-ID =アソシエーション識別子
AA-ID=アトミックアクション識別子
D-ID =ダイアログ識別子
T-ID =トランザクション識別子

図 4: TP チャネルへの入力データの流れ

アソシエーションプールの情報をテーブルとして持ち、アソシエーションとダイアログ/TP チャネルのマッピングなどを管理している。

また、各トランザクションを MACF オブジェクトとして管理し、ローカルでトランザクションを区別するために、各トランザクションにトランザクション識別子を割り当てている。

Commit MACF

コミットメントの調停など、複数のダイアログにまたがるトランザクションに関わる状態遷移を行なう。

トランザクション毎に 1 つ存在する。

VOS(Virtual Operating System)

OSI-TP プロトコルモジュール内のモジュール間でやりとりされるデータの経路制御を行なう。

Dialogue-obj

ダイアログ毎に 1 つ存在し、ダイアログの状態遷移、下位サービスとやりとり などを行なう。

以降の節では、今回新たに拡張した機能であるログと TP チャネルについて、詳しく述べる。

4.2 TP チャネル

TP チャネルは、ダイアログと同様にアソシエーションの上位の関係として見ることができるが、通常のダイアログ上でのプロトコル処理とは異なり、次のような実装上の問題点がある。

1. TP チャネルは、あるトランザクションの回復が終了した後、他のトランザクションの回復のために使用することができる。すなわち、アソシエーションとトランザクションの関係が動的に変化する。このため、TP チャネル用のアソシエーションでメッセージを受け取ると、動的に適切な MACF を判断する機能が必要となる。

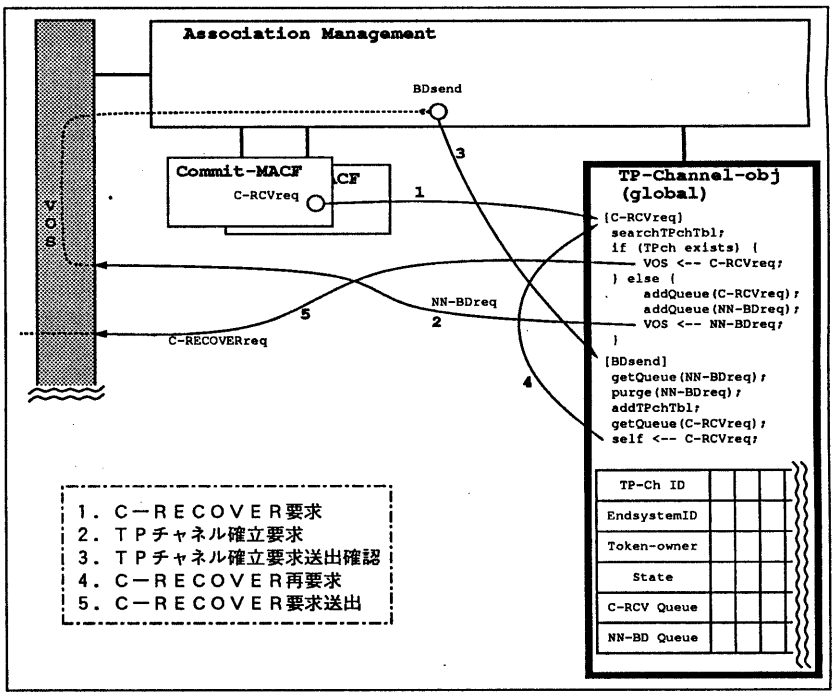
2. 通常のダイアログの 確立および 解放は、TPSU またはリモートからの指示に従って行なえばよい。しかし、TP チャネルは、TPSP が自発的に判断し、必要に応じてこれら処理を行なわなければならない。

まず、1については、次のように解決した。本システムでは、通常はアソシエーションにデータが送られて来ると、アソシエーション識別子からトランザクション識別子に変換し、目的の MACF に送る。しかし、TP チャネルは、ダイアログと異なり、複数のトランザクションの回復のために使用される可能性があるため、アソシエーション識別子からトランザクションを一意に決定することはできない。そこで、TP チャネルの場合には、送られてきた *C-RECOVER* メッセージの中にパラメータとして含まれている **アトミックアクション識別子** (各トランザクションをネットワーク内で一意に区別するための識別子) を読取り、これをトランザクション識別子に変換する処理を行なうようにする (図 4)。

一方、2については、**TP チャネル管理オブジェクト** が自動的に TP チャネルの確立・解放を行なうことで対応している。TP チャネル管理オブジェクトは、システム内の全 TP チャネルを 1 つのテーブルで管理しており、TP チャネルのプールとしての役割も持っている。TP チャネル管理オブジェクトは、表 2に示すようなオペレーションを提供している。Commit-MACF は *C-RECOVER-req* を送信する際には、まず TP チャネル管理オブジェクトに、その要求を送る。TP チャネル管理オブジェクトは、TP チャネル管理テーブルから目的のエンドシステムに送るための TP チャネルを検索する。ここで、適切な TP チャネルが見つかれば、そこに向けて *C-RECOVERreq* を送信する。しかし、適切な TP チャネルが見つからなかった場合は、TP チャネルを新たに確立する必要がある。この場合、TP チャネル管理オブジェク

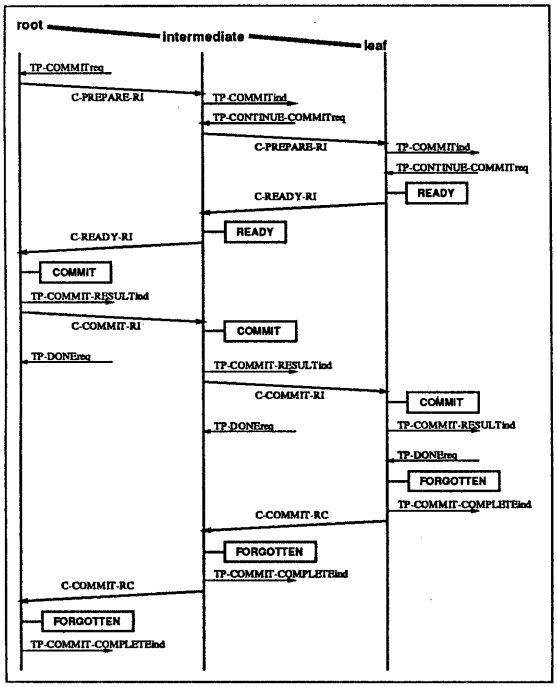
メッセージおよびイベント	概要
C-RECOVERreq	Commit-MACF が送信しようとしている *C-RECOVERreq* を相手のエンドシステムに向けて送信する。 ただし、使用できる TP チャネルがないときは、 新たに確立しようとする。
C-RECOVERind	Commit-MACF が受信した *C-RECOVERind* に応じて TP チャネルの状態遷移を行なう。
DIALOGUE-RI-SEND イベント	TP チャネルを新たに確立するために送信した *NN-BEGIN-DIALOGUEreq* の送信が完了し、 保留していた *C-RECOVERreq* を送信する。
NN-REJECTind	TP チャネル確立要求が拒否されたため、 タイマを設定し、 *NN-BEGIN-DIALOGUEreq* の再送に備える。
TIME-OUT イベント	以前に拒否された *NN-BEGIN-DIALOGUEreq* を再送する。
NN-BEGIN-DIALOGUEind	リモートからの TP チャネル確立要求に対する処理。
NN-END-DIALOGUEind	リモートからの TP チャネル解放要求に対する処理。

表 2: TP チャネル管理オブジェクトの主なオペレーション



(TP チャネルが確立されていない場合)

図 5: *C-RECOVERreq* の送信



READY および COMMIT : ログレコードの書き込み。
FORGOTTEN : ログレコードの消去。

図 6: ログを書き込むタイミング

トは、 *NN-BEGIN-DIALOGUEreq* を発行し、 相手のエンドシステムとの間に TP チャネルを確立しようとする。TP チャネルが確立されるまでは、 *C-RECOVERreq* はキューに蓄えられる。そして、 TP チャネルが確立されてから、 改めて *C-RECOVERreq* を送信する。また、本システムでは、 *NN-BEGIN-DIALOGUEreq* が拒否され *NN-REJECTind* が返された場合を想定して、 一定時間後に *NN-BEGIN-DIALOGUEreq* を再送する機能も有している。 TP チャネル管理オブジェクトの動作の 1 例を図 5に示す。

4.3 ログ

ログに関する処理は、 通常のプロトコル処理にはない回復制御の特徴的な点である。ログを書き込むタイミングを図 6に示す。 図 6では、 正常に終了した場合を示しているが、 この一連の流れの途中でシステムクラッシュが発生し、 再起動された場合、 その時点でのログの内容を読み、回復処理を行なわなければならない。これを実現するためには、 次のこと問題点となる。

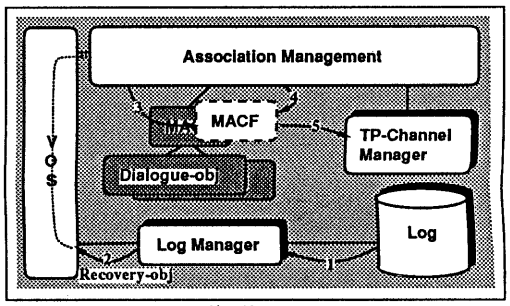
1. ログに記録する項目

有効 / 無効 フラグ	トランザクションの状態 (Ready / Commit)	アトミックアクション識別子	ブランチ情報 (配列)

図 7: ログレコードの構造

2. ログの情報をもとにした回復動作

まず、 1については、 次の通りである。ログレコードには、 コミットを実行する準備ができている状態を示す **READY ログレコード** と、 コミットが確定したことを示す **COMMIT ログレコード** がある。ログレコードがない場合は、 ロールバックを意味する (プリジュームドロールバック)。各ログレコードには、 トランザクションを識別するために必要な アトミックアクション識別子、 回復動作を決定するために必要な トランザクションの状態、 通信路を回復するために必要な ブランチ情報 が含まれている (図 7)。



1. ログ読み込み
2. リカバリメッセージ送出
3. MACF の生成および回復
4. リカバリメソッド呼び出し
5. C-RECOVER 要求送出

図 8: システムクラッシュからの復帰

2については、 **ログ管理オブジェクト** が行なう。本システムは、 システムクラッシュ後、 再起動されると、 初期設定の最後にログ管理オブジェクトに制御を移し、 ログを検索する。 ログ管理オブジェクトは、 ログの内容を読み込み、 回復が必要なトランザクションを見つけると、 読み込んだログに保持されているトランザクションの状態と通信路に関する情報をもとにトランザクションの回復動作を開始する。システムクラッシュから回復する動作を図 8に示す。この図では、 1 つのトランザクションの回復を表しており、 実際には、ログレコードがなくなるまで繰り返される。また、図では矢印 5 で、 *C-RECOVERreq* を TP チャネル管理オブジェクトに送信する時点までしか描いてないが、 その後の処理は、 図 5で同様に通信路の回復が行なわれる。

以上の機能を実現するために、本システムでは、次の様な拡張を行なっている。

データユニットクラス

モジュール間でやりとりされるデータには、「TP データユニットクラス」、「タイムアウトクラス」などのデータクラスがあるが、本システムでは、これらに「リカバリデータユニットクラス」を追加した。図8では、"*Recovery-obj*" がリカバリデータユニットクラスのインスタンスである。リカバリ管理オブジェクトは、ログレコード毎に *Recovery-obj* を生成し、アソシエーション管理オブジェクトに向けて送り出す。

Commit-MACF オブジェクト

"*recover*" メソッドを追加した。ここでは、ログレコードの内容を受け取り、トランザクションの状態に応じて、*C-RECOVERreq* のパラメータを設定し、TP チャネル管理オブジェクトに送る。

アソシエーション管理オブジェクト

リカバリデータユニットクラスの追加に伴い、リカバリ用のオペレーションを追加した。ここで行なう動作は、以下の通りである。

1. 必要な Commit-MACF を生成し、MACF 管理テーブルに追加する。
2. Commit-MACF の *recover* メソッドを呼び出す。

5 結論

本論文では、OSI-TP にとって必要不可欠な回復制御の機能を実現するために、問題点を検討し、設計および実装について述べた。

本システムでは、通信障害およびシステムクラッシュからの回復を可能とするために、回復専用の通信回線である TP チャネルとログに関する機能を追加した。

TP チャネルに関しては、1つの TP チャネル管理オブジェクトが、システム内の全ての TP チャネルを管理するようにした。ログに関しては、ログ管理オブジェクトが、ログへの全てのアクセスを管理するようにし、再起動時の、回復処理の制御も行なう。このように、TP チャネルとログの管理をオブジェクトとしてカプセル化することにより、モジュール性を高め、今までのプログラムの変更点を最小限に抑えることができた。

以上の回復制御機能により、通信障害およびシステムクラッシュに対して信頼性の高い分散 TP システムを実現し、OSI-TP をより実用的なものにすることがきる。

参考文献

[ISO89a] ISO: Information Processing Systems — Open Systems Interconnection — Distributed Transaction Processing — Part 1: Model. ISO/IEC DIS 10026-1, 1989.

[ISO89b] ISO: Information Processing Systems — Open Systems Interconnection — Distributed Transaction Processing — Part 2: Service Definition. ISO/IEC DIS 10026-2, 1989.

[ISO90] ISO: Information Technology — Open Systems Interconnection — Distributed Transaction Processing — Part 3: Protocol Specification. ISO-/IEC DIS 10026-3, 1990.

[谷林 91] 谷林 陽一 ほか: OSI トランザクション処理における回復制御の設計. 電子通信学会春期全国大会, 1991.

[楠 89] 楠 和浩 ほか: OSI トランザクション処理システムの試作. マルチメディア通信と分散処理研究会, 1989.