論　文

TINA型分散ネットワーキング環境におけるハードウェア・ソフトウェア連携障害管理方式

石井　啓之[†]　西川　博昭[††]　田中　博樹[†]　井上　友二[†††]

Hardware and Software Collaborative Fault Management for TINA Typed Distributed Networking Environment

Hiroshi ISHII[†], Hiroaki NISHIKAWA[††], Hiroki TANAKA[†], and Yuji INOUE[†††]

あらまし　TINA(Telecommunications Information Networking Architecture)は，分散透過性を利用し，再利用性，相互運用性の高いネットワーキングソフトウェアを実現しつつある．今後，更に広範囲な商用展開に備えて，TINA環境を更に確実性の高いものとする必要がある．すなわち，広域に分散したTINAソフトウェアオブジェクト及びそれらを搭載するプロセッサの障害対策が急務となってきている．既に筆者らは，データ駆動原理に基づくハードウェアを中心とした障害管理アーキテクチャについて提案を行っている．本論文は，この考え方とソフトウェアにおけるソフトウェアオブジェクト障害管理との有機的な連携を行うことにより，TINAネットワーキング環境における確実性の高い障害管理アーキテクチャを提案するものである．

キーワード　TINA，分散処理，データ駆動プロセッサ，障害管理

1. まえがき

TINAにおいては，分散オブジェクト指向の利点である，分散透過性と開発工程の短縮効果を最大限利用し，そのうえでテレコムネットワークに必要な網管理やサービス実現のためのアーキテクチャやソフトモジュール仕様を規定し，アプリケーションオブジェクト及び共通的ネットワーキングミドルウェアオブジェクトが定義されつつある[1]．また，その実システムへの応用も盛んになりつつある．

また，TINAソフトウェアを用いない場合であっても，分散処理プラットホームであるCORBA (Common Object Request Broker Architecture) を積極的に応用した通信システムも市場に出現しつつある．

このような分散型テレコムネットワーキング環境においては，中央集権型管理や制御を行わないため，個々のオブジェクトや，それらのオブジェクトを搭載する各TINAノードの管理状態（内部状態ではなく，運用・非運用（障害/スタンバイなどの状況））をクライアントが把握せずにオブジェクト間通信を行い，結果として望みの通信が行えなくなる可能性がある．

現在の分散処理環境におけるソフトウェアオブジェクト管理には，例えばCORBAにおける共通サービス機能としてライフサイクルなどがあるが[2]，障害管理に直接の適用はされていない．また製品レベルではいくつかのORB製品において，内部プロトコルとして，障害オブジェクト管理を実装しているものも見られるが規模の面で十分な機能とはいえない．したがって，TINAのような大規模分散ネットワークにおけるソフトウェアオブジェクトの障害管理機能の実現は必須となる．

また，TINAネットワークのソフトウェアオブジェクトが実際に搭載されるTINAノードやノード間を結ぶネットワークなどのハード面の障害管理は，標準化されたOSI(Open Systems Interconnection)システム管理やTMN(Telecommunication Management Network)において検討されているがオーバヘッドを少なく障害検出を可能とし，かつ系に悪影響を及ぼさずに障害部分を切り離すことが可能なハードアーキテクチャが必要となる．

更に，ソフトウェアとハードウェアの障害管理は，

† NTT情報流通プラットフォーム研究所，武蔵野市
NTT Information Sharing Platform Laboratories, Musashino-shi, 180-8585 Japan
†† 筑波大学電子・情報工学系，つくば市
Institute of Information Science and Electronics, University of Tsukuba, Tsukuba-shi, 305-8573 Japan
††† NTTサービスインテグレーション基盤研究所，武蔵野市
NTT Service Integration Laboratories, Musasino-shi, 180-8585 Japan

プロトコルの階層間障害連携を除けば密接に連携している状況になく，両者が有機的に連携することによるより確実な障害管理の実現が望まれる．

本論文では，TINAネットワーキング環境におけるソフトウェアオブジェクトやノードハードウェアの障害管理に着目し，ソフト・ハード両面の障害管理が連携した障害管理アーキテクチャを提案するものである．まず，ソフトウェア及びハードウェアのそれぞれ独立な障害管理方式を明らかにする．ソフトウェア面では，既に提案したSOMSEアーキテクチャ[3], [4]をベースに，専用の監視オブジェクトを設けず，分散処理環境の共通サービスであるトレーディングを用いる，という拡張を加えた障害管理方式を提案する．ハードウェア面では，筆者らが提案したデータ駆動プロセッサをハードウェアに採用したノードハードウェア障害管理を用いることを前提にしている．データ駆動プロセッサを前提とする理由は，信頼度、多重処理性、マルチメディア情報の効率処理などの面で優れた特徴を有していること[5], [6]，またそれを用いることにより高品質マルチメディアネットワークが構成することができること[7], [8]，分散ネットワーク障害管理法を実現できること[9], [10]，高効率TINA情報転送網が実現可能であること[11]などの結果によるものである．つぎに，これらソフトウェア及びハードウェアそれぞれの障害管理機能をもとに，分散型ネットワークのための確実で包括的な障害管理法をどのようにソフトウェア・ハードウェア障害管理を連携させて実現するかを提案している．

2. TINA環境における障害管理への要求条件

分散処理ネットワークにおけるアプリケーションオブジェクトは，図１に示すように，ネットワーク内に広範囲に分散している．地理的に離れた複数のオブジェクトが連動してアプリケーションを実現する．これらのオブジェクトは物理的にプロセッサ上に搭載された分散処理環境(DPE: Distributed Processing Environment)上に分散する．このような分散ネットワーク環境における障害に対する要求条件としては以下のものが考えられる．

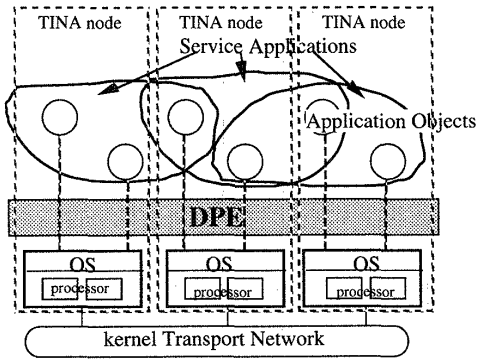


図１　分散処理ネットワーク
Fig. 1　Distributed processing network.

2.1 ソフトウェアオブジェクトの障害

ソフトウェアオブジェクト障害に対する要求条件は以下のように考えられる．

（1） 障害検出：分散環境におけるオブジェクトの障害を早期に検出する．

（2） 障害通知：オブジェクト障害を何らかの手段によりクライアントに通知する．

（3） 代替オブジェクト検出：障害オブジェクトと同等な役割を果たす代替オブジェクトを検出する．

（4） 代替オブジェクト通知：代替オブジェクトをクライアントに通知する．

（5） 障害回復：障害オブジェクトを回復させる．

2.2 プロセッサの障害

分散処理環境及びその上のソフトウェアオブジェクトを物理的に搭載し，その障害がソフトウェアに最も大きな影響を及ぼすノードのプロセッサ障害に対する対策としては，プロセッサの２重化などが考えられる．この場合の要求条件として以下のものが考えられる．

（1） 障害検出：障害検出が迅速に可能となること．しかも，障害検出のための処理が通常処理を圧迫しないこと．

（2） 障害通知：アプリケーション側に必要に応じて障害を通知する．ただし，検知させることなく代替プロセッサが起動できれば陽に通知は不要となる．

（3） 代替プロセッサ起動：プロセッサ障害は２重化などの冗長構成が基本となるが，できる限り迅速な系の切換えが必要である．同時に，障害プロセッサが障害のために系に悪影響を与えないように迅速な系からの切離しが必要である．

（4） 障害回復：障害プロセッサを回復させる．

このようなプロセッサ障害に関しては，筆者らは既にデータ駆動原理基づく障害管理方式を提案している[9]．

3. ソフトウェア及びハードウェアそれぞれの障害管理アーキテクチャ

2.の要求条件をふまえて，本章においては，ソフト

ウェア，ハードウェアそれぞれの観点からの障害対策を考察する．

3.1 トレーダを利用したソフトウェアオブジェクト障害管理方式

2.1の要求条件を満たす方策として，筆者らはSOMSE (Service Operation and Management architecture using Surveillance of application software Elements)を提案した[3], [4]．SOMSEにおいては，ソフトウェアオブジェクトの障害を検出するために各オブジェクトを監視するオブジェクトマネージャを仮定し，クライアントに障害をオブジェクトマネージャが通知するとともに，オブジェクト代替関係データベースを仮定して，そのデータベースを参照することにより，代替オブジェクトを検出することができる．このようにSOMSEは分散環境に適する自律型の障害処理を可能とするソフトウェアオブジェクト障害管理アーキテクチャである．

しかしながら，障害検出機構を被管理オブジェクト以外に設けること，全体的な知識を利用した代替オブジェクトの管理法など，実用に供するうえでの課題は残されていた．

そこで，SOMSEの自律性を生かしつつ，上記の問題を解決するために，専用の障害検出機構を設けずサーバ・クライアント関係を利用し，代替オブジェクト管理にCORBAの標準化されたオブジェクトサービスの一つであるトレーディングサービスを利用してSOMSEの考え方をより実用的なものとする方法を検討した．トレーディングサービスを実行するトレーダは，サーバオブジェクトのサービス属性，タイプなどのサービスオファとそのオブジェクトのレファレンスを登録しておき（サービスオブジェクトからのエクスポートと呼ぶ），クライアントオブジェクトからトレーダに対する問合せ（タイプ情報などをかぎとする）により，対応するサーバオブジェクトリファレンスをクライアントに通知する（クライアントがトレーダからインポートする）機能を有する．

トレーダを利用した障害管理法を，**2.**の要求条件に応じて以下のように検討した．

（1） 障害検出

あるサーバオブジェクトに障害が発生した場合，検出の方法として以下のものが考えられる．

（i） クライアントによる検出：サーバオブジェクトが障害に陥ると，そのサービスを受けているクライアントオブジェクトは，望みのサービスを完了できなくなり，結果として障害の可能性を検出する．

（ii） 障害監視機構による検出：SOMSEにおける検出法であり，各オブジェクトに障害監視機構を仮定し，それが，当該オブジェクトの処理の実行状況を監視することにより検出する．

（iii） オブジェクト自身による検出：障害オブジェクトが障害に陥ったことを自身で検出する．

実用的観点からは，各オブジェクトの監視機構を設ける方法は監視機構の信頼性や監視機構も含めたオブジェクト数の増加などの問題がある．また，ソフトウェアの構成単位であるオブジェクト自身に障害検出機構を具備する方法は，オブジェクト自身の障害時に機能しない可能性が高く明らかに問題がある．ここではクライアントによる検出方法を基本とする．最も一般的な検出方法は，一定時間内に応答を受けない場合や，異常な応答を受ける場合などがある．

（2） 障害の通知

障害を検出したクライアントは当然障害を既に知っているが，同じ障害サーバオブジェクトを使用する他のクライアントにはまだ障害が通知されていない．そのため，他のクライアントが独立に障害サーバオブジェクトにアクセスすると，障害に遭遇する．分散処理環境において，障害サーバにアクセスする可能性のあるすべてのクライアントに障害を通知することは，オブジェクトインスタンス数が大きい場合には実行上難しい．また，どのクライアントがどのサーバにアクセスするかをSOMSEのように管理する方法も考えられるが，これも大規模分散環境においては実際的ではない．したがって，ここでは積極的にクライアントに障害を通知することは行わない．

（3） 代替オブジェクト検出

(2)において障害を検出したクライアントはその旨をトレーダに通知することにする．これにより，トレーダは障害サーバを認識する．トレーダには，種々のサービスオファが事前にエクスポートされているため，障害サーバのサービスオファに相当する，あるいはそれに同等なサービスオファを有するオブジェクトを検出することができる．

（4） 代替オブジェクト通知

(3)により，ほかのクライアントがトレーダに当該サーバの提供するサービスをかぎに問い合わせてきたときに，障害オブジェクトのリファレンスを通知することなく，障害サーバのサービスオファに相当する，あるいはそれに同等なサービスオファを有するオブ

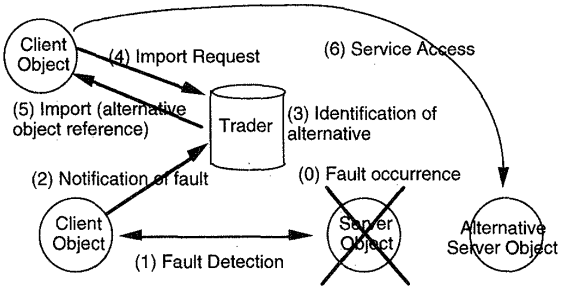


図2　オブジェクト障害管理
Fig. 2　Object fault management.

ジェクトを代替オブジェクトとして通知できる．

（5）障害回復

(4)までの手順により，障害オブジェクトの代替が可能となった時点で回復措置をとることができる．

図2に，ここで提案したトレーダを利用するソフトウェアオブジェクト障害管理方法の一連の流れを示す．この方法は，OMG標準（TINAの標準でもある）にのっとったサービスを用い，クライアント側からのメッセージ種別を追加するだけで実現できるものである．またハードウェア方式とも完全に独立である．

3.2　ノードプロセッサを対象としたハードウェア障害管理

ハードウェア障害に関しては，筆者らは既にデータ駆動原理基づく障害管理方式を提案している[9]．詳しくは引用文献に譲るが，基本的な方式は以下のとおりである．

（1）障害検出

本アーキテクチャでは，データ駆動プロセッサ（エンティティと呼ぶ）が相互にテストメッセージを巡回させ，その巡回時間と相手のパイプライン長との対応により，障害を検出する．ここでの特徴は以下のとおりである．

（i）監視専門のエンティティを設置する必要がない

（ii）相互に監視し合う各エンティティは，ネットワークを介した地理的な分散も可能である．

(iii) 中央制御でなく，完全なローカル制御となっている．

(iv) 障害のみならず，相手エンティティにおける過負荷状態も監視メッセージの巡回時間の増長により検出できる．

（v）データ駆動プロセッサの多重処理性により，監視メッセージの追加が，各エンティティの処理に何ら負の効果をもたらさない．

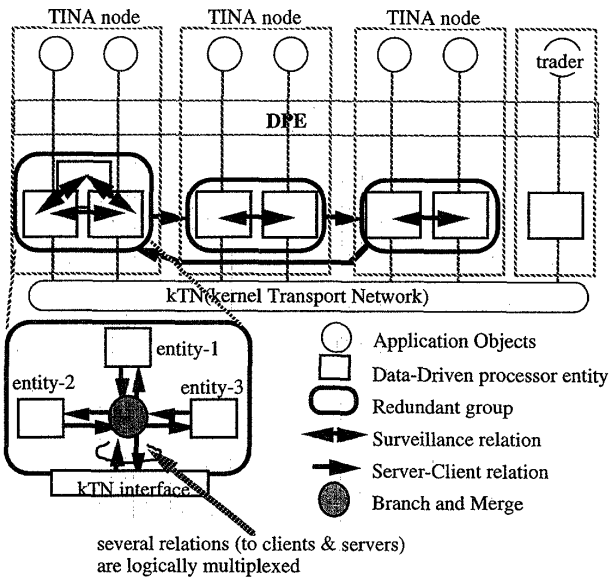


図3　分散型データ駆動障害管理アーキテクチャ
Fig. 3　Distributed data-driven fault management architecture.

（2）自律へいそくを伴う自律再構成（障害通知，代替プロセッサ起動，障害回復）

本アーキテクチャでは，相互監視群内で相互に障害/過負荷を検出すると，エンティティの入力と出力を直接接続し，障害エンティティ内にデータが流入しないようにバイパスする．これにより，以下の特徴が得られる．

（i）冗長構成により，基本的に障害の通知が不要である．

（ii）迅速に障害箇所を系から切り離せる．

(iii) 切り離すことがそのまま系再構成となる．

(iv) 障害エンティティを含む相互監視群のクライアントあるいはサーバに何ら影響を与えない．

（v）データ駆動プロセッサにおいては，入力を遮断すると機能しないため，入出力間バイパスにより機能が停止し，暴走などにより系全体に負の影響を与えることはない．

図3に，ノードプロセッサを対象としたハードウェア障害管理のための分散型データ駆動障害管理アーキテクチャを示す．

4.　ハードウェア・ソフトウェア連携障害管理方式の提案

4.1　基本的考え方と連携の必要性

ここでは，分散処理ネットワークにおけるソフトウェアオブジェクト障害管理とノードハードウェア（プロセッサ）障害管理のソフト及びハード両面の障

害管理が連携する必要性を述べる．

一般に，**3.2**に述べた既提案のハードウェア障害管理方式[9]を採用しない場合でも，ハードウェアの多重化によって，ハード障害はソフトウェア側にいんぺいされることが多い．しかしながら，**3.2**に述べたようなローカル制御性，障害検出専用ハードウェアの有無，監視メッセージのオーバヘッド性，プロセッサの受動性，などの面で従来の方策は，検出オーバヘッドが大きく，迅速な切換えが難しく，障害箇所が暴走して能動的に系に悪影響を与える可能性がある，などの問題を有する．

したがって，**3.2**のデータ駆動型障害管理法をハードウェア障害管理に適用すると，より迅速確実な対策が可能となり，ソフトウェアへの影響を最小限にとどめることができる．

しかしながら，ソフトウェアとハードウェアの障害管理がそれぞれに機能していても，以下の場合，両障害管理が独立して機能することによる問題が顕在化することが考えられる．

(a) ハードウェアの冗長構成によりハードウェア障害管理が機能して，継続したハードの機能を提供したとき，負荷の問題から処理能力が不十分になる場合がある．このとき，このノードのソフトウェアオブジェクトをサーバとして利用するクライアントオブジェクトには，例えば処理時間タイムアウトなどにより，オブジェクト障害が検出され，ソフトウェアオブジェクト障害管理機能（トレーダ）に障害が通知される場合が考えられる．このとき，ハードウェアの情報をもたないトレーダが代替オブジェクトとして同じノードに搭載されているオブジェクトを指定すると，この障害対策は有効とはならない．

(b) (a)のように処理能力が低下しているときに，新たなサーバオブジェクトレファレンスのインポート要求があったとき，ハードウェアの情報をもたないトレーダが当該ノード上オブジェクトを通知するとき，処理能力の低下を更に悪化させる．

(c) ソフトウェアオブジェクト障害管理は基本的にクライアントが発見することにより起動される．(a)や(b)の場合のようにノードの処理能力が低下する場合，多くのサーバオブジェクトが同時に障害状態（それ自身は障害ではないが，ハードウェアに起因するレスポンス時間の増大などによって障害に見える）に陥る可能性がある．この場合，個々のオブジェクトごとにオブジェクト障害管理が起動され多くの回復処置をトレーダで実行する必要が生じるという問題がある．

(d) ソフトウェアオブジェクト障害管理において検出された障害がサーバオブジェクト障害ではなく，通信路やクライアントのノードの通信装置障害の場合，障害に陥っていないサーバオブジェクトが排除されてしまう．

したがって，より効果的な障害管理の実現のためには，本論文で提案したソフトウェア，既提案のハードウェアのそれぞれの障害管理方式に加えて，ソフト及びハードの障害管理が連携することが必要となる．

4.2 ハードウェア・ソフトウェア連携障害管理法方式

4.1の問題を解決するために以下の方策を提案する．ノードハードウェアにはデータ駆動プロセッサの採用を仮定し，更に，ソフトウェアオブジェクトの障害管理には**3.1**で述べたトレーダを利用した障害管理を前提とし，更に，各プロセッサは，ソフトウェアオブジェクト障害管理を行うトレーダとの通信リンクを有していることを前提とする．

4.2.1 4.1(a), (b)を解決する方策

4.1の問題(a), (b)を解決するための処理の流れを以下に提案する．

(1) ハードウェア側の障害としてプロセッサが障害に陥る．

(2) **3.2**に提案したデータ駆動型障害管理方式により，障害プロセッサを相互監視する同一ノード内のプロセッサが，監視メッセージにより障害を検出する．

(3) 直ちに障害プロセッサは入力を断たれ，切り離され，そのままハードウェアは継続運転される．

(4) 障害検出したプロセッサは，以下のようにソフトウェアオブジェクト障害管理との連携を図る．

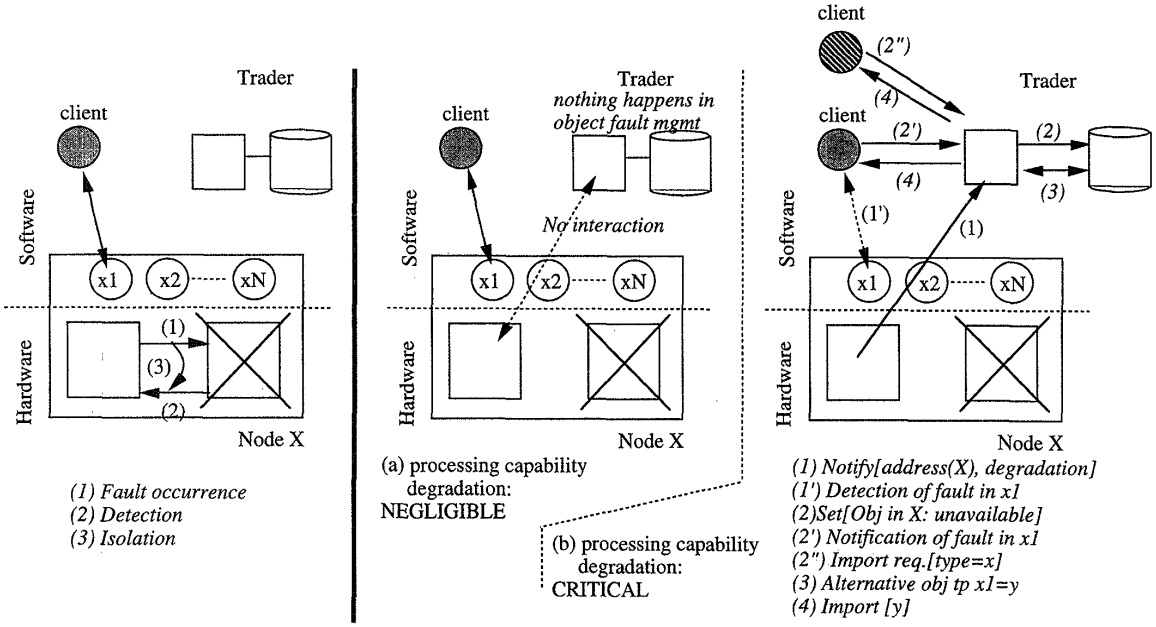
(4-1) 十分な冗長構成により処理能力の低減が大きな問題とならない場合

何も通知しない．積極的な連動がなくとも問題なく系は機能している．オブジェクト障害管理は起動されない．

(4-2) 処理能力の低下が心配される場合

プロセッサはトレーダに対して，当該ノードの処理能力上高負荷に耐えられない可能性を通知する．ここで，通知パラメータは物理アドレスと負荷状態である．一般にオブジェクトレファレンスには物理アドレスが埋め込まれているために，物理アドレスを通知すればオブジェクトレファレンスが引ける．

オブジェクト障害管理は，処理能力低下に起因する



(a) 1. Fault detection phase　　(b) 2. Hardware/software collaboration phase

図4　ソフトウェア・ハードウェア連携障害管理の流れ
Fig. 4　Flow of software/hardware collaborative fault management.

クライアントオブジェクトからの障害通知に対して，当該ノード以外のノードプロセッサ上のオブジェクトを通知することができ，**4.1**(a)の問題は避けられる．

また，新たなインポート要求に対しても同様に，当該ノード以外に搭載されるオブジェクトのレファレンスを通知し，当該ノード向けの処理トラヒックの増大を防ぐ．これにより**4.1**(b)の問題は解決できる．

図4 (a), (b)にここまでの流れを示す．

4.2.2　**4.1**(c)を解決する方策

4.2.1でハードウェアから処理能力低下を通知する対策によっても，ソフトウェアオブジェクト障害管理では，問合せがあって初めて代替オブジェクトを通知する仕掛となっているために，(c)の問題は解決できない．

障害に遭遇したしたクライアントのうち，障害ハードウェアに搭載されるサーバオブジェクトレファレンスを通知したものに対して，代替オブジェクトレファレンス，あるいは障害発生の事態のみを能動的に通知する．これにより不定期なクライアント側からの問合せに対応するためのオブジェクト障害管理の負荷を軽減することができる．

図5に，本方策を示す．

4.2.3　**4.1**(d)の対策

4.1(d)の場合，データ駆動型プロセッサ障害管理から

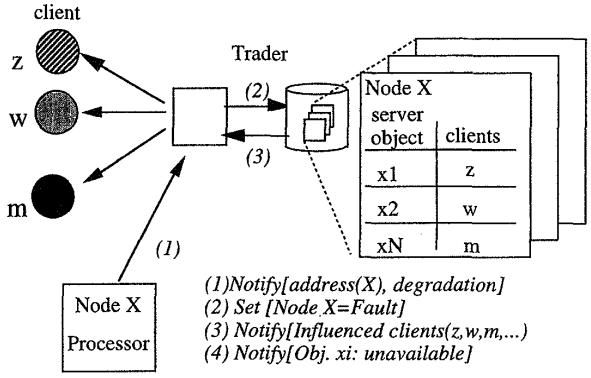


図5　障害ノード上のオブジェクトのクライアントに対する通知
Fig. 5　Notification to clients of objects on faulty node.

オブジェクト障害管理への通知はなされず，オブジェクト障害管理のみが起動され，他のオブジェクトが通知される．しかし，これは本質的な解決にはならない．

まず通信路障害の場合は，本論文の検討対象外であるが，一般に通信路障害は多くの対策が既にとられており，ノードハードウェアやソフトウェアとは独立に解決されることが期待できる．したがって，オブジェクト障害管理が起動されて代替オブジェクトに切り換えられるが，実効上の問題とはならない．

クライアントのハードウェアの通信装置障害の場合は，当該方路へのトラヒックは障害が解決されるまで流すことができなくなる．この場合も本論文の検討対象外であるが，文献[11]で提案するようなデータ駆動型

のプロトコル処理装置を適用すると，通信装置障害への耐力が増加することが考えられる．この件については別稿に譲る．

4.2.4 障害回復後の措置

障害プロセッサが修理などにより機能回復すると，直ちに系に組み込まれる．この場合もデータ駆動型ハードウェアアーキテクチャでは，断っていた入力を再び当該プロセッサに再開すれば直ちに機能する．

処理能力が回復すると，処理能力低下をトレーダに通知したプロセッサは再び処理能力回復を通知する．これにより，ソフトウェアオブジェクト障害管理をつかさどるトレーダは，再び当該ノード上のオブジェクトレファレンスを通知可能となる．このとき，当該オブジェクトを利用していたクライアントに積極的に回復を通知することもオプションとして可能である．

5. むすび

本論文では，現在主流になりつつあるTINA/CORBAを利用した通信ネットワークシステムの広域性と高信頼度への要求に十分にこたえるための障害管理方式を明らかにした．まず，ハードウェアとソフトウェアのそれぞれにおいて必要となる障害管理方式を検討した．ハードウェアに関しては，既に筆者らが基本方式を提案したデータ駆動型障害管理方式を前提とした．ソフトウェア障害管理については，既に提案したSOMSEアーキテクチャをもとに，専用監視オブジェクトを設けず，CORBA標準サービスであるトレーダを用いたるという改良を加えたソフトウェアオブジェクト障害管理方式を新たに提案し，その有効性を示した．更に，ハードウェア障害管理からソフトウェア障害管理に対してプロセッサの負荷状態を通知することにより，ハードウェアとソフトウェアが連携したむだのない確実な障害管理が実現できることを示した．

今後は，通信プロトコル処理装置，ネットワークなどの障害管理も含めた，包括的な障害管理アーキテクチャを明確化していく．また，データ駆動プロセッサとノイマン型プロセッサが混在する系におけるハードウェアアーキテクチャの場合の連携管理法についても検討を進める．更に長期的課題として，実時間性を要求されるネットワークシステムにおけるTINAの適用性とその場合の障害管理方式についても検討を進める．

謝辞 本検討に際して，有益な議論と激励をいただいたNTT情報流通プラットフォーム研究所メガメディアプロジェクト各位に感謝します．また，データ駆動プロセッサアーキテクチャの具体化に関して常に有益な議論をいただくシャープ株式会社宮田宗一氏に感謝します．本研究の一部は文部省科学研究費用基盤研究(B)(2)08458060の援助を受けて行ったものである．

文　献

[1] G.Nilsson, F.Dupuy, and M.Chapman, "An Overview of the Telecommunications Information Networking Architecture," TINA95, Melbourne, Feb. 1995.

[2] Object Management Group, "The Common Request Broker: Architecture and Specification," rev.2.1, Aug. 1997.

[3] H.Tanaka and H.Ishii, "Service Operation and Management Architecture using Surveillance of Application Software Elements (SOMSE)," GLOBECOM'95, pp.1997-1981, Nov. 1995.

[4] 田中博樹，石井啓之，"アプリケーションソフトウェア構成要素の個別監視を用いたサービス運用管理方式の一検討，" 信学技報，IN94-109/CS94-137，pp.7-14，Nov. 1994.

[5] 石井啓之，小林秀承，田中博樹，西川博昭，井上友二，"TINA環境へのデータ駆動プロセッサ適用の検討，" 信学NAWS，Dec. 1996.

[6] H.Nishikawa, H.Ishii, and Y.Inoue, "A Stream-Orinented Data-Driven Processor Realizing Hyper-Distributed Systems,"IASTED PDCS 96, pp.47-51, Oct. 1996.

[7] 石井啓之，西川博昭，小林秀承，井上友二，"TINA型高品質マルチメディアネットワークの実現法の検討，" 信学論(B-I)，vol.J80-B-I，no.6，pp.457-464，June 1997.

[8] H.Nishikawa, S.Miyata, S.Yoshida, T.Muramatsu, H.Ishii, H.Kobayashi, and Y.Inoue, "A Data-Driven Implementation of Telecommunication Network Systems," ISADS97, April 1997.

[9] H.Ishii, H.Nishikawa, and Y.Inoue, "Data-Driven Fault Management for TINA Applications," IEICE Trans. Commun., vol.E80-B, no.6, pp.907-914, 1997.

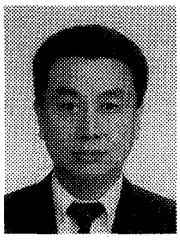
[10] H.Ishii, H.Tanaka, and H.Nishikawa, "Reliable TINA-based Telecommunication Networking Environment," IASTED Euro-PDS'97, 1997.

[11] H.Nishikawa, S.Miyata, S.Yoshida, T.Muramatsu, H.Ishii, Y.Inoue, and K.Kitami, "Data-Driven Implementation of TINA kernel Transport Network," TINA'97, Nov.1997.

（平成10年3月24日受付，10月29日再受付）

石井　啓之　（正員）

昭52阪大・工・通信卒．昭54同大大学院前期課程了．同年日本電信電話公社電気通信研究所入所．以来，データ通信網構成法，ISDNユーザ・網インタフェースプロトコル，通信情報ネットワーキングアーキテクチャ(TINA)の研究開発に従事．現在，NTT情報流通プラットフォーム研究所主幹研究員，グループリーダ．情報処理学会，IEEE各会員．



西川　博昭　（正員）

昭51阪大・工・電子卒．昭59同大大学院博士課程了．日本学術振興会奨励研究員，同大助手，講師，筑波大・電子・情報工学系助教授を経て，現在，同教授．工博．平６・７月～７・８月マサチューセッツ工科大招聘研究員．データ駆動型超分散システムとその仕様記述環境などの研究に従事．昭61年度高柳賞受賞．情報処理学会，IEEE各会員．

田中　博樹　（正員）

平２東工大・工・電子物理卒．平４同大大学院修士課程了．同年日本電信電話(株)通信網研究所入所．以来，分散型ネットワークサービス定義法，分散実時間プラットホーム，分散アプリケーションの管理法の研究開発に従事．平９学術奨励賞受賞．現在，NTT情報流通プラットフォーム研究所研究主任．

井上　友二　（正員）

昭46九大・工・電子卒．昭48同大大学院修士課程了．同年日本電信電話公社電気通信研究所入所．以来，網同期方式，ディジタルネットワーク，ISDN，B-ISDN，SDH，ネットワークアーキテクチャ，マルチメディアネットワークサービスなどの研究開発に従事．現在，NTTサービスインテグレーション基盤研究所長．TINA-C技術フォーラム議長．工博．共著書「ISDN」，「ネットワークアーキテクチャ」など．IEEEシニア会員．