通信情報ネットワーキング環境の
高効率・高信頼実現法に関する研究

石井啓之

システム情報工学研究科

筑波大学

２００１年１月

目次

内容梗概

情報処理においては，メインフレームと端末というセンター・エンド型の関係から，より小型で汎用の計算機が機能及び負荷を分散処理する，分散コンピューティングの様相へ変化して久しい．一方通信の世界においては，元来分散処理を前提とした電話交換ネットワークが１００年の長い歴史を持っている．コンピュータの発明以来，通信と情報処理は比較的独立に発展してきた．ところが近年，インターネットの爆発的発展に見られるように，通信と情報処理の融合したサービスが本格化しつつある．ネットワークやサービスの規模が大きくなり，また地理的に広範囲にわたるこのようなサービスにおいては，機能や負荷の分散が大きな課題となる．このために，分散オブジェクト技術に基づく分散処理環境DPE(Distributed Processing Environment)の利用の機運が高まりつつある．たとえばOMG(Object Management Group)で検討され既に多くの実用化がなされているCORBA(Common Object Request Broker Architecture)がその代表である．さらに，分散処理環境上に，ソフトウェア群を配備して複数のアプリケーションが共同利用し，サービスアプリケーション構築の効率化を図るネットワークミドルウェアの検討もNTTの情報流通プラットフォームの検討に代表されるように活発になされている．また，トランスポート層も含め，分散処理環境，情報流通ネットワークミドルウェア層を利用してネットワーク上の機能・負荷分散型サービスアプリケーションを実現する環境（通信情報ネットワーキング環境）は，TINA-C(TINA Consortium)において，機能分類，インタフェース仕様などが精力的に検討されてきた．

このように，ネットワークノード，サービスノード，ユーザシステムのハードウェアおよび，トランスポート層，分散処理環境，ネットワークミドルウェア，さらにはアプリケーションまで含めたソフトウェアがネットワーク上に分散し，これらの機能群が互いに協調しながらサービスを実現する通信情報ネットワーキング環境は，今後の情報通信サービスの要となりつつある．

本研究は，現在および将来の通信情報ネットワーキング環境に対して，下位ハードウェアの効率的処理の実現法，ハードウェアモジュール・エレメント障害管理，アプリケーションオブジェクト障害管理などを明らかにする．さらに環境総体としての高信頼・高効率な品質を実現する方法を明らかにする．

はじめに，通信情報ネットワーキング環境実現上の留意すべき要件を整理する．

要件１：通信情報ネットワーキング環境における多重処理の効率化

通信情報ネットワーキング環境においては，前述の通りハードウェア・ソフトウェアの多くの機能が互いに協調してサービスを実現するため，機能間での通信が多く発生する．環境の規模の増大に従い，通信すべき相手の数も増え，プロトコル装置における多重処理能力の向上が求められる．また，分散処理環境を運用する上で，分散不可視を可能とするための各種サーバ機能（ネームサーバ，トレーディングサーバ）においても同時に多くの問い合わせに対応するための多重処理能力が必要となる．さらに，高負荷時

にプロセシング能力向上のための効率的マルチプロセッサ構成の実現が望まれる.

要件２：通信情報ネットワーキング環境の高信頼化

２つめの要件は，分散した機能群を管理して，全体としての信頼性を確保する方法である.旧来の環境では多くのネットワーク管理やシステム管理において見られるように，集中管理方式となっている．しかしながら，インターネットでは，集中管理されることなく自律分散型で機能している．従って自律分散ゆえの信頼度の低さが顕在化してきているのも事実である．通信情報ネットワーキング環境では，旧来の環境にはなかったソフトウェアオブジェクト単位の分散までを視野に入れ，また小規模PC(Personal Computer)を用いた分散システム構築を可能とするため,大規模化に耐力の無い集中管理ではなく，またベストエフォートでない信頼性の確保を可能とする新しい管理機能が必要となる．この管理機能の中には，自律的系再構成による高信頼化，アプリケーション障害やノード障害のネットワーク的波及の防止法を含むことが望まれる.

これをふまえて，本要件を以下の３つに分化する.

要件２－１：通信情報ネットワーキング環境を形成するネットワークミドルウェアコンポーネントやそれを用いて実現されるアプリケーションソフトウェアなど，ソフトウェアコンポーネントに対する障害管理法の必要性

要件２－２：本環境を支えるネットワークやノードプロセッサなどのハードウェアの障害管理法の必要性

要件２－３：2-1のソフトウェア障害管理法と2-2のハードウェア障害管理法を連動させたソフトウェア・ハードウェア連携障害管理法の必要性

要件３：その他の要件

そのほかの要件としては，分散処理環境のサーバノードなどトラヒックが集中するノードにおける瞬時的トラヒック増大に対する耐力の実現，マルチメディアコンテンツ情報流通を容易とする高速信号処理等があげられる.

本研究は，以上の要件をふまえて，高効率化，高信頼化による通信情報ネットワーキング環境の構成法を明らかにする.

高効率化の検討

第１の要件である高効率化について本研究では以下のように検討を進めた.

(1)既存ノイマン処理環境の非効率性の検証

まず,既存のノイマン型プロセッサによって実現された通信情報ネットワーキング環境について，実測と数学的解析によってその特性を明らかにし，その問題点を明らかにした.

測定は以下のように行った.典型的なサーバ・クライアント環境において，クライアントがスレッドを生成し，スレッドごとにTCP(Transmission Control Protocol)コネクションを設定し，サーバに並列に処理を要求する環境を作成し，サーバの処理時間，データ転送時間，コネクション設定時間等を実測した.

この結果，スレッド数の増加に従い，スレッドあたりのすべての項目の処理時間が増加することを明らかにした．さらに，キューイングモデルを作成して，数学的に解析することにより，同じ傾向が現れることを検証した．

このことは，逐次処理の拡張としての擬似的多重処理では，オーバヘッドが原理的に免れないことを意味していると思われる．換言すれば，多重処理されるスレッド間での相互干渉が本質的な問題と考えられる．

(2)データ駆動プロセッサの効率性の検討

(1)の問題に対して,西川らが検討してきたストリーム指向データ駆動プロセッサの多重処理性，受動性の活用が有用であることに着目した．

データ駆動方式では，その処理駆動原理からわかるように，一旦起動された処理は互いに独立であることが保証される.そこで,データ駆動プロセッサを用いたマルチスレッド実行に関して各スレッドの実行時間を実測した．その結果,(1)のノイマン型プロセッサでの実行結果とは異なり，スレッドの実行時間（負荷）と数の大小に関わらず，スレッド間の相互干渉によるオーバヘッドは起こらないことを明らかにした．さらに，この現象を上記と同様にキューイングモデルで解析することにより，検証した．

この結果，データ駆動プロセッサの通信情報ネットワーキング環境への適用性が高いことを明らかにした．

(3)高効率な通信情報ネットワーキングアクセス系の要求条件の検討

通信情報ネットワーキング環境の効率化の手始めとして，同ネットワークにアクセスするユーザアクセス系の効率化をデータ駆動プロセッサを応用して実現する方法を検討した．ユーザがネットワークにアクセスする上での要求条件は，低容量アクセス回線の効率的利用，複数コネクションの同時利用が性能に影響を与えないプロトコル処理，高速マルチメディアストリーム情報と低速制御情報が混在し同時使用される場合の効率化，単一端末が複数サービスを受けること，同一宅内に設置された複数のPCにより単一のアクセスを共用できること，PCの持つ既存インタフェースの活用などがあげられる．これらはすべて効率化の要件に大きく関与する．

(4)高効率TCP/IPプロトコル処理のデータ駆動実現の検討

これらのアクセスのための要求条件の中でプロトコル処理に着目した．プロトコルとしては，通信情報ネットワーキング環境において，また昨今のインターネットにおいて最もよく用いられるプロトコルであるTCP/IP(Transmission Control Protocol/Internet Protocol)を取り上げ，そのデータ駆動実現を検討した．

TCP/IPプロトコルの効率化のために，実時間で高精細ビデオ信号処理を実現できるデータ駆動プロセッサシステムをエミュレータとして使用してIPパケット転送部をまずエミュレートしノイマン型とデータ駆動型の多重処理性能を比較した．ノイマン型の模擬においては，繰り返し処理が頻発し，その制御のために多重処理がある値以上の入力では，実現不可能となる．これに対して，パイプライン処理に加えて，繰り返しを防ぐためバッファメモリを用いない並列処理を導入したIP転送処理部を作成し測定すると，不要な繰り返し処理のためのオペレーションが劇的に減少し安定した能力を実現した.このエミュレーション検討より，データ駆動型並列処理がが高効率TCP/IPにふさわしいこ

とを明らかにした.

この結果より, TCP/IP向きデータ駆動プロセッサの完全実装を行った. これを本研究のプロジェクトコード名であるCUE(Coordinating Users' Requirements and Engineering constraints) におけるプロトタイプ第1版としてCUE-pと呼んでいる. 本実装においては, プロセスあたりの処理時間を測定すると一定となり, きわめて安定な効率性を実現することができた.

高信頼化の検討

(1)要件2－1の検討

ソフトウェアコンポーネント (以下オブジェクトと呼ぶ) の障害管理法の必要な状況として以下のことが考えられる. 通信情報ネットワーキング環境においては, 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ノード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上のソフトウェアオブジェクト管理には, たとえばCORBAにおける共通サービス機能として, オブジェクトの生成, 維持, 消去のライフサイクル管理機能などがあるが, 高信頼化のための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言えない. 従って, 大規模分散型ネットワークーキング環境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.

上記要求条件を満たす方策として, 本研究ではSOMSE (Service Operation and Management architecture using Surveillance of application software Elements)を提案する. 通信情報ネットワーキング環境においては, あるアプリケーションは複数のアプリケーション要素であるオブジェクトによって実現され, また個々のオブジェクトは異なるサービスで多重に使用される. そのため, 特定のオブジェクトの異常は複数のアプリケーションに影響を及ぼすことになる. 現在存在する管理方法ではアプリケーションを単位にあるいはそのオブジェクトが搭載されるハードウェア単位の管理しか実現していないため, 上記の異常に有効に対応することができない. SOMSEは, 監視オブジェクトを設けて, それが単数のあるいは複数の被管理オブジェクトの異常を監視し, 異常検出時に当該被管理オブジェクトのクライアントに対して異常を通知するとともに, 総合監視機構に通知し, 必要に応じて, 代替オブジェクトを生成する等の措置を行うことにより, 分散環境に適する自律型の障害処理を実現できる.

本SOMSEにより, 従来実現されなかった, アプリケーションソフトウェア構成単位ごとの監視を可能として, 通信情報ネットワーキング環境に適合した新しい障害管理方式を実現し, 高信頼化の要件をソフトウェア面から充足した.

SOMSEは, システムの大規模化に伴い, アプリケーション数が増加した場合にすべての関連するアプリケーションに同時に障害を通知するのが実効上困難となり, また障害時にすべてのアプリケーションが必ずしもアクティブな状態であるとは限らないため積極的に障害を通知することがオーバヘッドになり得る問題点が存在する. このため, す

べての関連アプリケーションに能動的に異常と代替オブジェクトを通知する代わりに,代替オブジェクトをCORBAの標準化されたオブジェクトサービスの１つであるトレーディングサービスに登録しておく方法を考案した．本方法では，監視オブジェクトを専用に設けず，被管理オブジェクトのクライアント自身が障害検出を行うことにより大規模化によるオブジェクト数増加も軽減できる．この方法は,OMG標準にのっとった共通サービスを用い，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ため，CORBA製品への適用も容易となる．またハードウェア方式とも完全に独立である．本方式により，すでにSOMSEで充足した要件２－１を大規模システムにおいても適用可能な見通しを得ている．

(2)要件２－２の検討

分散型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サーバ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)において検討され多くの実装が存在する．しかしながら障害処理自身が通常の処理を浸食する処理オーバヘッドは無視できない場合がある．また，障害に陥ったハードウェアコンポーネントが，系に積極的な悪影響を及ぼす可能性も否定できない．従って，処理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ウェアアーキテクチャが必要となる．本研究では前述のデータ駆動プロセッサを採用したノードハードウェア障害管理を検討する．

通信情報ネットワーキング環境のハードウェアシステム向きの基本的な障害管理の要件は，経済性の観点から，こわれにくいものよりも，こわれたことを迅速にしかも集中制御無しに検出できるものが望ましいこと，分散処理では複雑にサーバ・クライアント関係が成立しており，故障箇所を利用していたクライアント，故障箇所から利用されていたサーバに，故障を迅速に通知し，故障箇所を迅速に系から除去し，系の再構成が迅速に実行できることが望ましいこと，系から除去されるまでの間，故障箇所が系に悪影響を及ぼさないようなパッシブな仕組みが望ましいことがあげられる．

このため，アプリケーションオブジェクトあるいはオブジェクト群と対応関係を持つデータ駆動プロセッサが負荷分散と相互監視を目的に冗長構成されるアーキテクチャを考案した．

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．この特徴は，監視専門のエンティティを設置する必要がないこと，相互に監視しあ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こと,集中制御でなく，完全なローカル制御となっていること，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こと,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なんら負の効果をもたらさないことにある．

また，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．この特徴は，冗長構成により，基本的に障害の通知が不要であること，迅速に障害箇所を系から切り離せ，切り離すことがそのまま系再構成となること，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なんら影響を与えず，入力を遮断すると機能しないデータ駆動プロセッサの特徴により，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ことにある．

このように，本研究で明らかにしている本ハードウェア高信頼化対策は，データ駆動プロセッサの多重処理性により障害処理の負荷が本来の処理に影響を与えないこと，入力断による障害箇所の切り離しとパッシブ性によって，障害箇所が系全体になんら悪影響を及ぼさない有効性の大きい方式である．

(3)要件２－３の検討

現在の一般的なシステムにおいては，プロトコルの階層間障害連携を除けばハードウェアとソフトウェアが密接に連携している状況になく，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，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．

上で具体化したソフトウェア，ハードウェアそれぞれの障害管理法が独立して機能した場合の問題としてたとえば以下のことが考えられる．ハードウェアの障害によって，ハードウェアの多重化機構が機能して障害エンティティを切り離したとする．このとき，負荷の問題から処理能力が不十分になる場合がある．ハードウェアの情報を持たないソフトウェア障害管理機能も同様に異常を検出（処理レスポンスの遅延など）すると，代替オブジェクトを指定する．しかし，たまたま当該障害発生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ことがある．従って，より効果的な障害管理の実現のためには，ソフトウェア，ハードウェアのそれぞれの障害管理方式に加えて，ソフトおよびハードの障害管理が連携することが必要となる．本研究では，この問題を解決するために以下の方策を考案している．

ノードハードウェアには要件２(2)の解決策であるデータ駆動プロセッサの採用を仮定し，さらに，ソフトウェアオブジェクトの障害管理には要件２(1)の解決策であるトレーダを利用した障害管理を前提とし，さらに，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した．本方式は，ハードウェア障害を直ちにソフトウェア障害管理の要であるトレーダに通知することにより，障害ハードウェアに搭載したオブジェクトをソフトウェア障害管理が指定しないようにすることができ，有効な通信情報ネットワーキング環境のトータルな高信頼化が図れるようになる．

その他の要件の検討

その他の要件である，分散処理環境のサーバノードなどトラヒックが集中するノードにおける瞬時的トラヒック増大に対する耐力の実現，マルチメディアコンテンツ情報流通を容易とする高速信号処理等は，基本的に，本研究で検討したストリーム指向データ駆動プロセッサの性質から容易に解決できることを示した．

以上をまとめる.

本研究では，通信情報ネットワーキング環境の実現法として，ストリーム指向データ駆動プロセッサをハードウェアに採用することにより，高効率なプロトコル処理を実現することを実装や数学モデルによりその有効性を具体的に検証した.

また，高信頼なソフトウェア障害管理アーキテクチャおよびデータ駆動原理を利用したハードウェア障害管理アーキテクチャ，さらにはハードウェア・ソフトウェア連携管理アーキテクチャを明確にした.

関連発表論文

[1] H. Tanaka and H. Ishii, "Service Operation and Management Architecture using Surveillance of Application Software Elements (SOMSE)," Proc. of IEEE Global Telecommunications Conference (GLOBECOM'95), pp.1997-1981, Singapore, Nov. 1995.

[2] H. Ishii, H. Nishikawa, and H. Tanaka, "Reliable TINA-based Telecommunication Networking Environment," Proc. of IASTED International Conference on Parallel and Distributed Systems (Euro-PDS'97), pp.17-22, Barcelona, Spain, Jun. 1997.

[3] 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," 電子情報通信学会論文誌B-I, Vol.J80-B-I, No.6, pp.457-464, Jun. 1997.

[4] H. Ishii, H. Nishikawa, and Y. Inoue, "Data-Driven Fault Management for TINA Applications," IEICE Transactions on Communications, Vol.E80-B, No.6, pp.907-914, Jun. 1997.

[5] H. Ishii, H. Nishikawa, and Y. Inoue, "TINA-based Multimedia Networking Environment Supported by Data-Driven Processor: CUE-p," Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), pp.423-429, Las Vegas, USA, Jul. 1998.

[6] H. Ishii, H. Tanaka, and H. Nishikawa, "Comprehensive Fault Management for Distributed Networking Environment," Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), pp. 454-460, Las Vegas, USA, Jul. 1998.

[7] H. Ishii, H. Nishikawa, and Y. Inoue, "Data-Driven Implementation of TINA-based Multimedia Networking Environment," Proc. of International Conference on Communication Technology (ICCT'98), pp.S14-07-1 - S14-070-5, Beijing, China, Oct. 1998.

[8] H. Ishii, H. Nishikawa, and Y. Inoue, "Data-driven Implementation of Highly Efficient Access to the TINA Network," Proc. of 1998 International IEEE Asia Pacific Conference on Communications/Singapore International Conference on Communication Systems (APCC'98/ICCS'98), pp.367-371, Singapore, Nov. 1998.

[9] 石井啓之, 西川博昭, 田中博樹, 井上友二, "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," 電子情報通信学会論文誌B-I, Vol.J82-B, No.4, pp.522-529, Apr. 1999.

[10] H. Ishii, H. Nishikawa, and Y. Inoue, "Data-Driven Implementation of Highly Efficient TCP/IP Handler to Access TINA Network," IEICE Transactions on Communications, Vol.E83-B No.6, pp.1355-1362, Jun. 2000.

第1章　序論

1.1　研究の背景

昨今，インターネット及びその上のｅビジネスに代表される情報通信サービスは飛躍的に伸びている．またインターネットを前提とした新しいネットワークアプリケーションの開発も盛んとなりつつある．

この背景として，まず，情報処理においては，メインフレームと端末というセンター・エンド型の関係から，より小型の汎用計算機が，機能及び負荷を分散処理する自律分散コンピューティングの様相へ変化したことがあげられる．分散コンピューティングは，機能の拡張性，融通性に富み，また低コストで実現できる特徴を有している[1]．

一方ネットワークの世界においては，その低コスト性，情報処理との親和性を原因として学術的ネットワークであったインターネットが急速に商用サービスとして伸びている[2]．今やインターネットは既存の電話をベースとしたキャリアネットワークを凌駕する勢いであり，IPによる統合サービスの方向が鮮明になってきている．

このような状況において，情報処理と通信が融合した新しいネットワーク型サービスの開発・提供環境すなわち，通信情報ネットワーキング環境と呼べる新しいパラダイムが興ってきている[3][4]．通信情報ネットワーキング環境では，ハードウェア（ネットワークノードハードウェア，サービスノードハードウェア，ユーザハードウェア）及びソフトウェア（伝達層のソフトウェア，分散処理環境，ネットワークミドルウェア，さらにはアプリケーションまで）の機能がネットワーク上に分散している．ネットワークに分散したこれらの機能群が互いに協調しながらサービスが実現される．

通信情報ネットワーキング環境では，ネットワークの伝達機能層とアプリケーション層の両者を有機的に結合し，なおかつ各層が互いにできるだけ独立に定義でき，機能し，運用されるトータルな枠組みが必要となる．すなわち情報通信ネットワーキングアーキテクチャの明確化が重要となっている．

情報通信ネットワーキングアーキテクチャの国際的な活動の代表として，TINA-C(TINA Consortium)[5]があげられる．TINA-Cにおいては，通信情報融合サービスを実現するための機能分類，機能間のインタフェース仕様などが精力的に検討されてきた．基本的なアーキテクチャは，情報伝達を担う伝達ネットワーク，情報伝達ネットワークによって相互に接続され，上位に対して位置や負荷の透過性を確保する分散処理環境ＤＰＥ(Distributed Processing Environment)，最上位のアプリケーション，の構成となっている．この中で，DPEは，OMG(Object Management Group)[1]で検討され既に多くの実用化がなされているCORBA(Common Object Request Broker Architecture)[6]がその代表である．

さらに，複数のアプリケーションが共同で利用でき，サービスアプリケーション構築の効率化を図る，情報流通ミドルウェアプラットフォームと呼ぶ共通アプリケーション層の検討が本格化している．TINA-Cで検討されている多くのアプリケーション機能は，

この情報流通ミドルウェアに含まれている．また，この情報流通ミドルウェアは，NTTにおける情報流通プラットフォーム概念に明確に打ち出されており[7],米国においてはIT業者の中にコンポーネントベンダーと呼ばれる，情報流通ミドルウェアプラットフォームの構成部品を商品としている企業群も現実に現れつつある.また,情報流通ミドルウェアの構築のためのプラットフォームとして，アプリケーション結合機構であるＥＡＩ(Enterprise Application Integration)技術も大きく伸び,多くのアプリケーションサーバ製品が出荷され用いられている．この背景として，インターネットを利用したビジネスが高度化するに従い，SI (System Integration)の負荷が大きくなり，できるだけ機能を共通化すること，しかも機能の組み合わせのための枠組みとしてのプラットフォームが必要となってきたことがあげられる.多くのビジネスパッケージや,アプリケーションサーバーなどは,この代表である.しかしながら,大規模エンタプライズビジネスユーザのように個別に自前のシステムを組める場合とは異なり多くの中小ビジネスユーザにとっては，独自にSI商品を購入することは負担が大きくなる．このために，仲介業者がプラットフォームを提供し,それを複数でシェアする方式が必要となってきている.ERP(Enterprise Resource Planning)などを提供するASP(Application Service Provider)がその代表例である．ここで，SI業者にとっても，ASPにとっても，アプリケーション展開のためのミドルウェアプラットフォームを用意しておくことが，サービス展開上重要となってくる．このプラットフォームの提供は，通信情報ネットワーキング環境の重要な役割である.

通信情報ネットワーキング環境においては，このようなアプリケーション，ミドルウェアの検討によりネットワーク上でのサービス・アプリケーションが高度化し，ネットワークに対するサービスの依存度が高まる．その結果，インターネットのそもそもの発想であるベストエフォート型のネットワークサービスではなく,より高品質,高セキュリティネットワークサービスに対するユーザ要望が増えてきている．たとえば，IP型ネットワークを中心に,現在のベストエフォートに加えていわゆる管理型IPネットワークの検討が行われている．これは，NTTのGMN-CL[8][9]に代表されるように，帯域を保証したり，各種優先度を制御したりして，メッセージからストリームまで，多様なサービス要求に対応できるネットワーク品質を提供でき,またネットワーク側でセキュリティを確保する技術である．また，ネットワーク管理技術，装置管理技術も，ＴＭＮ(Telecommunication Management Network)[10][11]に代表されるように，長いIP以前のネットワーク技術の歴史の上で多くの検討がなされている．これらはすべて，通信情報ネットワーキング環境の検討として位置づけることができる.

しかしながら，通信情報ネットワーキング環境において，アプリケーションやネットワークが一体となって，真に高い品質を実現できるには，上述のネットワークにおける転送機構，制御機能の検討やアプリケーション処理機構・構築機構，ミドルウェア部品の検討では不十分である．たとえば，それらを支える下位のハードウェアテクノロジの高品質化の検討や，アプリケーションソフトウェア，ミドルウェアの正常性を確保する機構の検討など，いわば背景技術の検討が必要である．しかしながら，現在それらが十分に行われているとは言い難い.

1.2 研究の課題と解決法

DPEにはネームサーバやトレーダなどの集約的処理機能が存在し[6]，この機能を物理的に搭載しているノードにはネットワークの負荷が集中する可能性がある．従って，通信情報ネットワーキング環境の接続／安定品質確保のために，集中する負荷への耐力確保法を検討する必要がある．特定サーバへの負荷集中対策については，インターネットのWebサーバについては，サーバのミラーリングやキャッシュサーバの運用によるネットワーク的負荷の分散，さらには，一般サーバの場合，製品レベルのアプリケーションサーバにより，分散処理技術によって負荷や機能分散が図られている．またより装置に近いレベルでは，複数サーバを接続して負荷分散，障害分散を図るクラスタリング技術[12]により，障害耐力増強と負荷分散を実現する方式もかなり一般化してきている．しかしながら，これらはアルゴリズム的にソフトウェア処理，あるいはOS機能の一部として実現されているものであり，個々の物理サーバを構成するハードウェアエレメント，すなわちプロセッサそのものの過負荷耐力の実現法は，いわゆるプロセッサ能力の向上の名の下に隠蔽され，正面から検討されているものは見受けられない．現実には，一般のノイマン型システムでは設計目標を超えた入力が加わると急激にスループットが低下する問題があり，これを避けるために本来プロセッサが有している処理リソースを有効に使用せずに新たなプロセッサの増設やプロセッサ能力の向上で対処している．本研究では，プロセッサアーキテクチャとしてプロセッサ自身が過負荷耐力を有する，エラスティックバッファ機能を具備した西川らのデータ駆動プロセッサ[13]-[17]を適用することによって，瞬時的な過負荷状態におけるあふれを吸収し，この問題を解決できることを示す[18]-[25]．

また，通信情報ネットワーキング環境のネットワークを構成するサーバシステム，ネットワーク管理システムや各種サービスアプリケーションシステムを構成するサーバに対しては，真に処理機構の効率性が追求されているとは言えない．これは，ノイマン型ハードウェア（プロセッサ）アーキテクチャを前提にそれを与えられた条件としてシステムの品質を規定しているためである．すなわち現在のノイマン型アーキテクチャでは多重処理がシーケンシャル処理の繰り返しによってしか実現できない．よって本質的に同時並列処理は「疑似」的なものであり，それらの擬似的な同時処理を切り替えながら処理を行う．従って，切り替えに際して無駄処理が発生して，プロセッサそのもののリソースが効率的に利用されていない．これも前述のように高速のプロセッサを用いることでその問題を顕在化させていないのにすぎない．本研究では，既存の疑似並列処理によるオーバヘッドを有するノイマン型プロセッサでは実現できない効率的処理を可能とする，データ駆動プロセッサに着目して，この問題に正面から取り組む．ネットワークにおける多重処理の典型は複数セッションやコネクションを同時に扱う通信プロトコル処理である．またプロトコル処理は専用処理装置であり，プロトコル装置以外の系と独立に検討が可能であることを勘案し，効率化の検討をプロトコル処理の効率化の観点で検討を行う．既存のプロトコル装置の実現法の検討は，高速化を目的にソフトウェア処理をファーム化，ASIC(Application-Specific Intergrated Circuit)ハード化する観点で検討されているが[26]，コネクションの並列性を生かした並列処理アーキテクチャ

に踏み込んだ検討は見受けられない．具体的には，昨今のインターネットにおいて最も用いられるTCP/IPプロトコル処理をとりあげプロトコル処理の並列性に着目した処理アーキテクチャをデータ駆動型で実現し，並列度と処理能力が線形な関係を示すという高い効率性を明らかにする[27]-[33]．

自律分散システムにおいては，アプリケーションコンポーネントがネットワークワイドに分散しているため，ある特定のコンポーネントやシステムの異常が影響を及ぼす範囲の特定が困難となる可能性がある．このようなランタイムの異常になんらかの対処を施して系の安定性を高めることが重要な課題となる．また現状のシステムでは，アプリケーション構成単位であるコンポーネントやオブジェクトがネットワーク的に分散しているときに局所的な異常を検出しても，それがネットワーク上のほかのどの箇所に影響を与えるのかを見いだすことは不可能である．市場ではCPU処理の状況を監視して，間接的に局所的な問題を発見する機構しかなかったり,分散を前提としないアプリケーション（帳票関係など）の状況監視機構などが一部商品化されている[34][35]．しかし，より粒度の小さいオブジェクトなどのソフトウェア部品（コンポーネント）単位の異常検出，ネットワークレベルでの障害対策はほとんど検討されておらず，ネットワークワイドなアプリケーションへの影響を検出対処できるものは存在しない．従って，大規模な通信情報ネットワーキング環境の高信頼化のためのソフトウェアコンポーネントの障害管理機能を，ORB(Object Request Broker)などの分散処理環境のベンダー製品非依存の形で実現することが望まれる．本研究では，個々のアプリケーションコンポーネントを監視する機構，コンポーネント間の関係を管理する機構をもとにコンポーネントをベースにネットワークレベルでのアプリケーション障害管理を可能とする機構を提案し，実用化し，上記の問題の解決を図る[36]-[39]．

同様にノードシステムやサーバシステムのハードウェア管理面では，装置全体をいわばブラックボックス的に扱う管理手法が中心である．より細部に踏み込んだモジュール管理，エレメント管理の検討は，キャリアにおけるネットワークノードの管理では実行されているが非常に高価な管理システムを前提としており，通信情報ネットワーキング環境には重装備と言える．またネットワークノードシステム管理であっても，プロセッサそのものにネイティブに障害対策を埋め込んだものは無く，外部装置・ソフトウェアによる監視であり，異常対策も冗長システムへの切り替えが主である．通常のサーバに対しては，クラスタリング機構[12]や，ディスクアレイ技術によるディスク二重化技術[40]により障害対策が検討されているが，これも装置をブラックボックスとしてとらえて外部監視による冗長化・切り替え対策である．本研究では，上記データ駆動プロセッサの受動性（入力を絶つと機能しない）と容易なマルチプロセッサ構成に着目する．外部監視では無い個々のプロセッサエレメント自身が相互に監視を行い，障害を検出したときには迅速に障害箇所への入力を断ち,系から切り離す,新しいプロセッサハードウェア障害管理手法を提案する[22][23][41][42]．

さらに障害管理において，ソフトウェアとハードウェアの障害管理は，それぞれが独立に機能しているのが一般的であり，十分な機構としての連携は図れておらず，もっぱら保守者による連携が主となっているのが現状である．これに対して，本研究では，上記のハードウェア及びアプリケーションソフトウェアの障害管理を連携させる方策を考

案し，有効なハードウェアソフトウェア連携障害管理法を提案する[43][44].

1.3 研究の特徴

本研究の特徴は，以下の４点に集約できる.

１点めは，通信情報ネットワーキング環境の実現のために考慮すべき要件を，アプリケーションからハードウェアまで幅広く目を向け明確化していることである.ハードウェアに関する要求条件は，近年の高速プロセッサの発展により，ほとんどが考慮すべき対象となっていないのが現状である．しかしながら，単なる高速化ではなく，いかに与えられた有限な資源を有効に活用するかの観点が抜け落ちていると言える.本研究では,その有効活用の観点で，従来のプロセッサでは実現できない，高多重性，高効率性を研究のひとつの大きな柱に位置づけ，データ駆動プロセッサによってそれを解決できることを明確に示している．またこのハードウェアに関する要求条件とアプリケーションに関するそれを密接に関連づけ，なおかつ双方が階層として独立に機能し得る管理法を明確にしている.

２点めは，データ駆動プロセッサの適用による通信情報ネットワーキング環境の実現である．計算機アーキテクチャの検討では，データ駆動原理の検討が長い歴史を有している．データ駆動プロセッサは，同時併行・パイプライン・多重処理というすべての並列処理性をオーバヘッドなしに実現するできるという特徴を有している．これを生かすことにより上記の通信情報ネットワーキング環境の品質確保をより容易にできることが考えられる．１点めの特徴にも述べたように，今やハードウェアの要求は，高速度と低価格に集中している．しかしながら，ソフトウェアレベルでの分散に対応する形でハードウェアも分散処理を実現できたり，あるいは，通常処理と付加処理（たとえば，付加的な障害探索の業務など)を同時進行させてもなんら通常処理を圧迫することがない,などの要求条件を正面から検討しているものは見受けられない．本研究では，従来高速並列計算アーキテクチャとして検討されてきたデータ駆動プロセッサ技術を高速化ではない，高効率化・高信頼化という新しい観点で検討し，その高い適用性を明確にしている.

３点めは，通信情報ネットワーキング環境へのユーザからのアクセスに注目した高効率なプロトコル処理法を明らかにしている点である.２点めで述べたデータ駆動プロセッサを適用することにより，既存の非効率なノイマン処理を飛躍的に効率化できることを示している.

４点めは，通信情報ネットワーキング環境の分散処理の構造に合致した，自律分散的障害管理のアーキテクチャを明らかにしている点である.従来のTMN(Telecommunication Management Network)に代表される管理アーキテクチャは，情報としても，あるいは機能としても階層化され，最終的な判断は集中化して，処理能力的に，また障害耐力にも限界が生じる．本研究で提案するソフトウェア，ハードウェアにおける障害管理アーキテクチャではできるだけ自律分散構成により集中化をさける新しい障害管理のアーキテクチャを明確にしている.

本論文は６つの章により構成されている.

まず，第１章は本章であり，研究の背景を概観し，本研究の目的，特徴について述べている．

第２章では，通信情報ネットワーキング環境の代表的研究例であるTINAの状況を概観するとともに，通信情報ネットワーキング環境実現上の品質面を中心とした要求条件を抽出する．

第３章では，第２章の要求条件をふまえ，分散処理を前提にした通信情報ネットワーキング環境に対するデータ駆動プロセッサの適用性を検討する．既存のノイマン型処理の非効率性を明確にし，それとの対比でデータ駆動プロセッサの有効性を示している．

第４章では，そのひとつの典型的な適用として，通信情報ネットワーキング環境へのアクセスのためのプロトコル処理をデータ駆動原理により実現し，きわめて高い効率性を実現することを示す．

第５章では，通信情報ネットワーキング環境の高信頼実現のために，ソフトウェア層で自律分散的監視制御を行う新しい障害管理アーキテクチャであるSOMSEを提案し，それをベースに改良を加えた方式を提案する．また，ハードウェア層では，４章で適用したデータ駆動プロセッサをハードウェアの実現法として採用し，上位の分散オブジェクトとの対応をとるデータ駆動ハードウェア障害管理法を提案する．さらに，ソフトウェア層，ハードウェア層の個別の管理を連携させることによりより確実な障害管理を実現するハードウェア・ソフトウェア連携管理法を提案する．以上によって，通信情報ネットワーキング環境の高品質化が容易に実現できることを明確にする．

第６章では，本研究で得られた結論と今後の課題について述べる．

第2章　通信情報ネットワーキング環境実現上の要求条件

2.1　緒言

通信情報ネットワーキング環境は，インターネット上のｅビジネスに代表される，通信とコンピューティングが融合した情報通信サービスを開発･提供するため,ネットワーク上に分散配置される，ハードウェア，ならびに，分散処理環境，ミドルウェア，アプリケーションまで含めたソフトウェアが，互いに協調してサービスを実現しようとするものである.

本章では，通信情報ネットワーキング環境を構築するに当たって，その実現上の要求条件を明らかにする.アプリケーションソフトウェアによるサービスの中身やネットワーク実現法と言った面での検討はこれまでにも精力的に検討され実用化されている．しかし，通信情報ネットワーキング環境を支えるハードウェア，特にプロセッサ，さらにはソフトウェアの正常運用をサポートする機能については十分な検討がなされてこなかった．その観点から，解決すべき課題，要求条件を明確化する.

まず，通信情報ネットワーキング環境のアーキテクチャについて，世界で最も早く着目し，通信業者，コンピュータ業者，ソフトウェアベンダーが協調してアーキテクチャ仕様の標準化を行った，TINA-C(Telecommunications Information Networking Architecture Consortium)[5]とその標準アーキテクチャであるTINA[4][45]について概観する．本研究のベースとして考えているTINAは，マルチキャリア，マルチベンダ通信情報ネットワーキング環境で，ユーザのエンドシステムとネットワークが協調してサービスを定義し運用する枠組みであるアーキテクチャである．特徴は，全てのシステムに共通のソフトウェア構造を適用することにあり,これにより,ソフトウェア部品がネットワークワイドに分散設置されてそれらが協調してサービスを実現する．これにより，サービス導入の迅速化，容易化，ソフトウェアの移植性の向上，移動性が確保され，さらに情報の流通共有が容易になり，本研究の対象としている通信情報ネットワーキング環境上を実現するベースとなる[21].

技術的な背景であるTINAの状況をふまえて,通信情報ネットワーキング環境を実現する上で，従来TINA-Cやそのほかにおいても検討が十分でなかった，環境を支えるハードウェアプラットフォーム，アプリケーション障害管理の観点から，通信情報ネットワーキング環境実現上の問題点の所在を明らかにする[18]-[25].

それらから，通信情報ネットワーキング環境実現のための要求条件を抽出する．第一の要求条件として，通信情報ネットワーキング環境の高効率化の必要性を述べる．現状のシステムにおける多重処理の必要性を例示し，そこにおける非効率性が解決すべき課題として重要であることを示す.

さらに，次の要求条件として，通信情報ネットワーキング環境の高信頼化を述べている．ソフトウェア，ハードウェアそれぞれの観点からの高信頼化，特に障害管理の重要性を明確にする.

加えて，通信情報ネットワーキング環境において特徴的なマルチメディア処理の必要性などその他の要求条件を明らかにする．

2.2 TINA(Telecommunications Information Networking Architecture)

2.2.1 TINAとは

本研究の対象としている通信情報ネットワーキング環境は，1章で述べたように，TINAをその代表として念頭においている．本節ではTINAについて，その概要を述べている．

インターネットに代表されるネットワークを用いたこれからのサービスは，単体のエンドシステムが複数連携することにより実現され，機能的にも負荷的にも分散を前提としている．このときにネットワークには，マルチメディアの各メディアに見合った帯域／品質のパスを提供し，ネットワーク全体の管理機能，ルーチングなどのネットワーク特有機能を提供していくことにより,エンドシステムと連動したアプリケーション／サービスを可能とすることが求められる．

このためには，マルチキャリアネットワーク／マルチベンダ環境下で，エンドシステムとネットワークが協調して，サービスを定義しかつ運用できる枠組み，すなわちネットワーキングアーキテクチャが重要な役割を果たす．

ネットワーキングアーキテクチャへの重要な要求条件には，エンドシステムとネットワークのシームレス化，情報／アクセスのオープン性が挙げられる．シームレス化，オープン化の実現には，従来の物理的なユーザ・網インタフェースを分界点とするエンドシステム側とネットワークの機能分担ではなく，インテリジェンスの分散を前提とした情報の相互流通，アプリケーションの相互運用（インタオペラビリティ），サービス定義／改変の迅速化を可能とする共通基盤（情報流通ミドルウェアと呼ぶ）が必要となる．

このような情報流通ミドルウェア，ミドルウェア相互のインタフェース及びその共通プラットフォームとしての分散処理環境（DPE：Distributed Processing Environment）を定義するために検討されてきたのが,TINA(Telecommunications Information Networking Architecture)である[4][5][21][45]-[47].

TINAは,多様な電気通信サービスを展開できるためにユーザ装置(端末)とネットワークが協調するオープンなアプリケーションソフトウェアアーキテクチャであり，キャリア／コンピュータベンダ／通信機器ベンダが結集したTINA-C (TINA Consortium0)[5]において，インタオペラビリティを持ったミドルウェアの世界的な流通による開発効率の向上も狙いつつ展開が進められた．

2.2.2 TINA仕様の特徴

TINAは個々のサービス／管理機能を実現するアプリケーション層，複数のサービス／管理に共通な機能を提供する情報流通ミドルウェア層（両者を併せてTINAアプリケーションとみなす)，これらを支えるDPE層の階層構造からなっている(図2.2.1)[45]．アーキテクチャにおける各階層で分担すべき機能の詳細と階層間のインタフェースをTINA仕様として規定している．ここで，DPEは別機器上に搭載された上位ソフトをあたかも同一

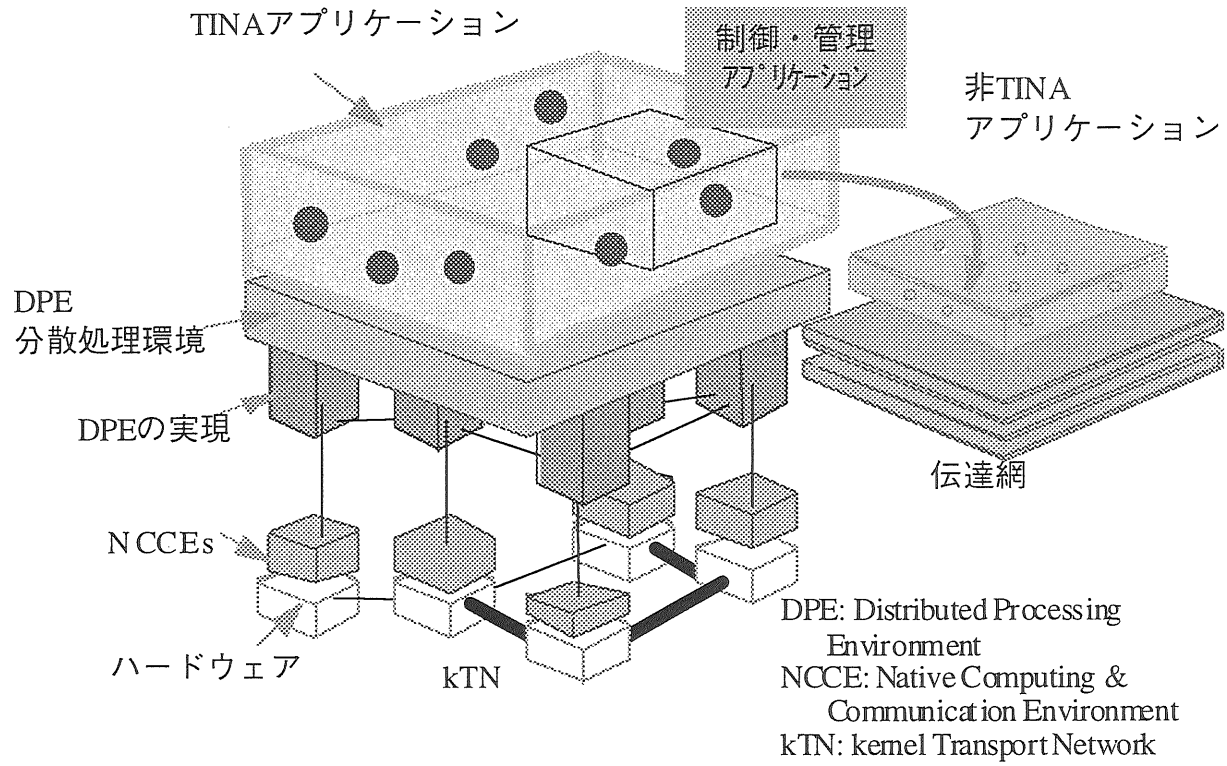


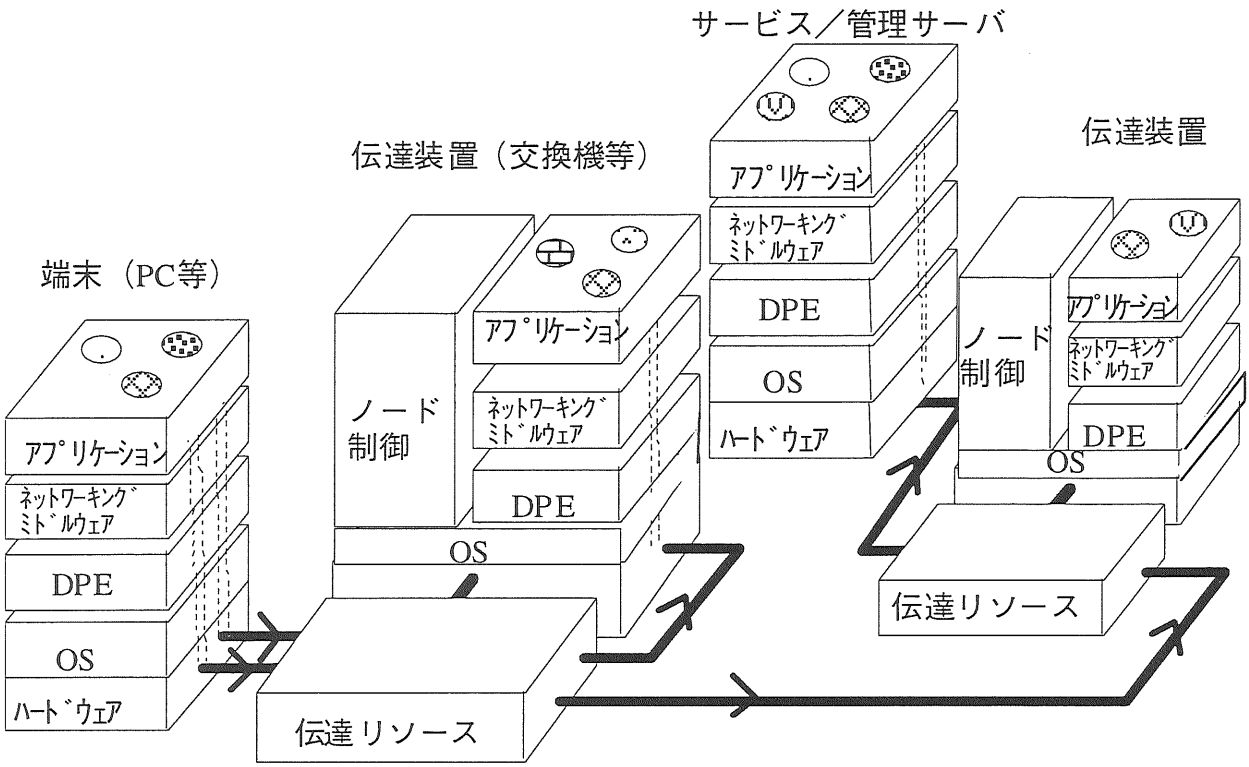
図2.2.1　TINAアーキテクチャ

機器上のソフトウェアであるかのように利用可能とする分散位置透過機能を司っている．DPE仕様の作成においては，CORBA仕様（Common Object Request Broker Architecture：アプリケーションソフトの相互運用性と移植性を保証するためにOMG(Object Management Group)[1]が規定した仕様[6]）をベースに，ユーザ情報と制御・管理情報の分離など，通信特有の要求条件を付加した仕様として規定することとした．

TINAの特徴は，図2.2.2に示すように，パソコン等の端末，交換機，ルータ等の伝達装置，サービス／管理サーバ等，情報通信ネットワークを構成する各機器に，共通のソフトウェア構造を適用することにある[46][47]．TINA仕様の適用により，機能毎に作成されたソフトウェア部品（コンポーネント）が地理的に分散したハードウェア上に自由に配備され，これにより端末上のソフトとネットワーク上のソフトが連携した新しいサービスの提供が可能となるとともにサービス導入の迅速化，効率的開発，移植性の向上，ポータビリティが達成できることになる．加えて，ネットワークサービスアプリケーションが原則として自由にユーザ側から利用できること，同じ情報定義を共有することで，情報の開示／共有が容易になる．

TINA仕様が世界レベルで普及すれば，マルチメディア用ソフトウェア特にネットワーキングミドルウェアの開発費用の削減と開発期間の短縮を実現できる．一方で，各社は同仕様に基づくアプリケーションソフトウェアの実用化・製品化を急ぎ，他社との競争に備える必要がある．このことから，TINA-Cの活動は「競争下の協調活動」と捉えることができる[4][21]．

一方，このようにTINAシステムが現実の世界で実用に供される段階になると，TINAソフトウェアを搭載したシステム（TINAノード）間を接続するネットワーク（kTN: ker-



DPE: Distributed Processing Environment
OS: Operating System

図2.2.2　TINAアーキテクチャを物理装置にマッピングした例

nel Transport Network）及びノードのハードウェアがシステム全体の品質を左右する重要な役割を持つ．しかしながら，TINA-Cにおいては，この観点での検討が十分になされたとは言えない．また，TINAアプリケーションの障害時の対策についても十分に検討されているとは言えない．そこで，本研究では，TINAを代表とする通信情報ネットワーキング環境を実システムとして組み上げていく際に，ここで述べたような十分に検討されていないエリアの問題点を明確化し，それから実現上の要求条件を引き出し，高信頼なkTNの構成法，ノードハードウェアリソースの効率的な運用法，さらにプロセッサレベル及びネットワーク分散したアプリケーションソフトウェアの障害管理の方法を検討していく．以下にTINAに代表される通信情報ネットワーキング環境実現上の問題点の所在と，その要求条件を示す．

なお，付録に，筆者が積極的に運営に関与したTINA-Cの状況及びTINA-C仕様の概要を述べる．

2.3　通信情報ネットワーキング環境実現のための問題点の所在と要求条件

通信情報ネットワーキング環境を実際にインプリメントしていく上ではさまざまな下位テクノロジーへの要求条件やハードウェアソフトウェアの運用管理面での要求条件を考慮に入れる必要がある．

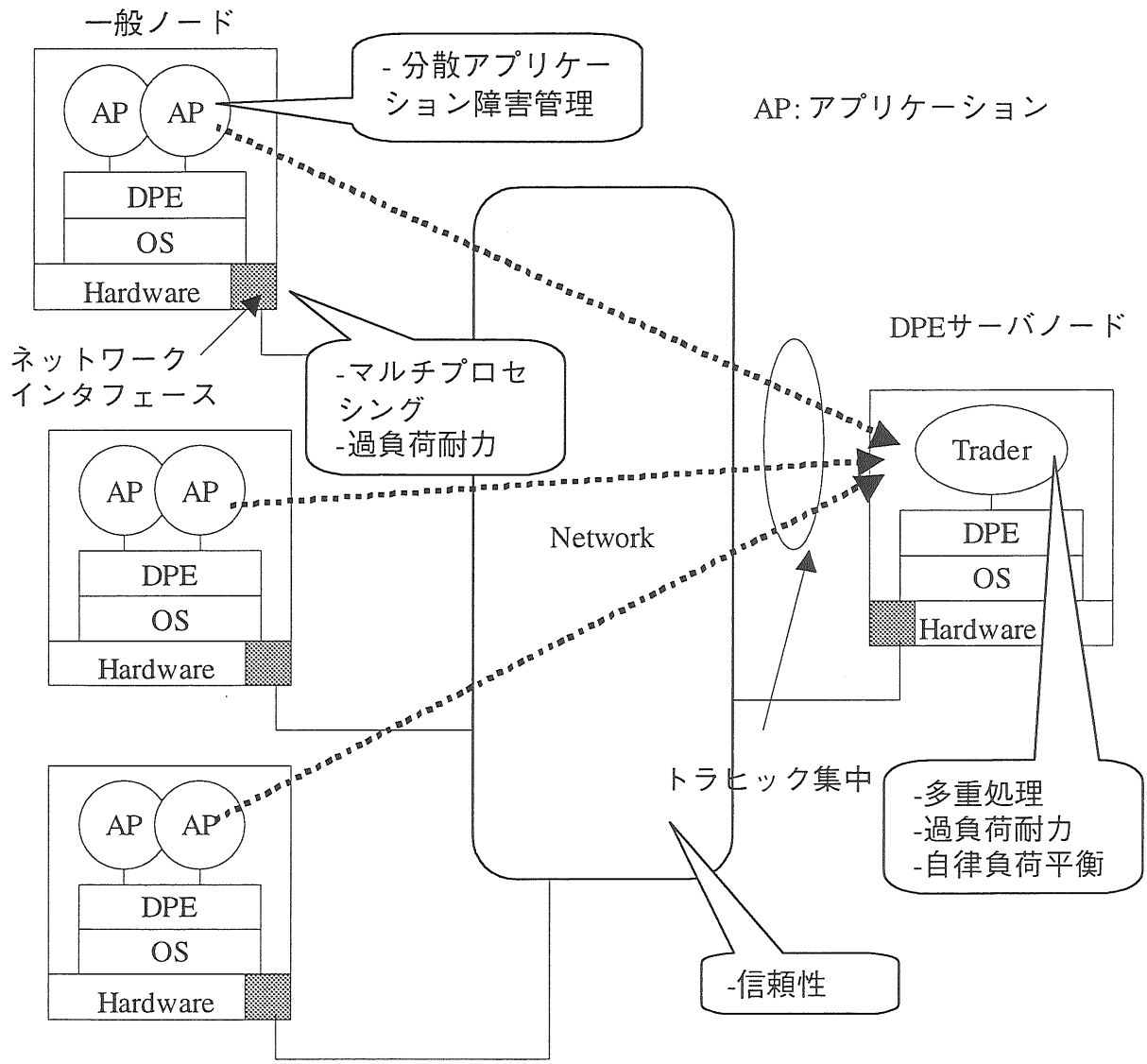


図2.3.1　通信情報ネットワーキング環境において解決すべき課題

図2.3.1に通信情報ネットワーキング環境の物理構成を示す．各ノードにはハードウェアの上に，DPE，情報流通ミドルウェア，アプリケーションを搭載しており，ノードの中にはトレーディング機能やネームサービス機能と言ったDPEの共通サービス機能を有するものもある．この図2.3.1をもとに，通信情報ネットワーキング環境において解決すべき問題の所在を示す[31][32]．

(1)ネットワークインタフェース

各ノードに収容される複数のアプリケーションは複数の他ノードとオブジェクト間メッセージをやりとりしてサービス／管理を実行する．物理的にはこれらのメッセージはネットワークによって転送される．従って，図2.3.1におけるネットワークインタフェース部分においては，転送プロトコルの複数のセッション／コネクションを同時に保持し制御することが求められる．この場合，プロトコル処理は，ハードウェア的にはプロセッサ上の複数のコンテキストがセッションやコネクションに対応する形で処理されると考えられる．現在の逐次処理型ノイマンプロセッサにおいては，頻繁にコンテキ

ストスイッチが発生していると考えられ,そのオーバヘッドは大きく非効率な処理を行っていると考えられる．プロセッサの高速化により，この問題は顕在化していないが，元来有している処理リソースを無駄に消費することの工学的な問題に正面から取り組んだ検討は見られない．今後ますます高速化していく回線速度に応じてコネクションの多重度が高まり，この非効率性が顕在化する可能性に対して何らかの回答を用意する必要がある．

(2)DPEサーバー

通信情報ネットワーキング環境においては，分散処理環境（DPE）によってアプリケーション間メッセージの交換を行う．このDPEには共通的なサービスを提供するDPEサーバが存在する．代表的なものには，ネームサーバ，トレーダがある[6]．各オブジェクトが処理を他のオブジェクト（サーバオブジェクト）に依頼するとき，実アドレスではなく，サーバオブジェクトの名前でサーバを特定することが多い．このとき，サーバオブジェクトの名前を問い合わせると実際のアドレス（レファレンス）に変換してくれる共通サーバがネームサーバである．インターネットにおけるDNSとほぼ同様の働きを行う．一方，サーバオブジェクトを特定しないで，処理の内容で検索する場合，サーバオブジェクトが有すべき属性などを問い合わせるとそれに該当するオブジェクトレファレンスを提供するのがトレーダである．これらのDPEサーバが機能するためには，事前にオブジェクトの情報の登録（名前，属性，レファレンスなど）を行い，運用時には上述のようにクライアントオブジェクトから登録された情報の問い合わせが行われる．場合によっては，これらのDPEサーバに登録／問い合わせのメッセージが集中することがある．このとき，物理的にDPEサーバを搭載しているノードでは扱うトラヒックの総量が大きくなり，かつ多くの登録／問い合わせセッションが同時に存在することになる．従って，上記(1)と同じくオーバヘッドの大きいコンテキストスイッチが問題になると同時に,トラヒックの増大に応じて，負荷に対応できるようにプロセッサの多重化も必要となり，効率的なマルチプロセッサの構成が求められると考えられる．また瞬間的なトラヒックの増大に対応できる過負荷耐力も重要な要件となる．

(3)サーバ（一般・DPEサーバ・ネットワークノードを含む）

ネットワークに分散した大小のサーバノードが障害時にできるだけ上位アプリケーションに影響を与えないで復旧できることが望まれる．大規模分散ネットワークにおいては，管理情報量が膨大となり，できるだけ局所的な障害対策が実行できることが望まれる．また，通信情報ネットワーキング環境においては，マルチメディアコンテンツが流通するため，ストリームを効率よく高速に扱えるノード内処理の実現が大きな課題となる．

(4)ネットワーク

通信情報ネットワーキング環境においては，ユーザ情報も，管理・制御情報も物理的には同じネットワークを転送される．ISDNを含む既存の電話型通信網においては，制御管理情報は共通線信号網という専用のネットワークによって転送されている．共通線信号網は，物理的なネットワーク構成が完全冗長構成である2面構成を採用している．共通線信号網において用いられるプロトコルであるNo．7信号方式のレベル2，3におい

ては，障害の検出機能と，２面構成を利用したリンクの切り替え機能を具備し，極めて信頼性の高い網構成を実現している．そのため信号端局にはコスト的にも機能的にも高いプロトコル処理機能を具備する必要がある．これに対して，インターネット型の通信情報ネットワーキング環境では，ノードには一般のワークステーションや場合によってはPCが採用され，共通線信号端局装置のような高価な専用機能は期待できない．従って，通信情報ネットワーキング環境のノードそのものに大きな影響を与えることなく，自律的な系の再構成など信頼性確保能力をネットワーク及びそのインタフェースにどのように持たせるかが問題となる．

(5)アプリケーション

通信情報ネットワーキング環境では，分散処理環境上にアプリケーションソフトウェアを構成する要素であるコンポーネントがネットワークワイドに分散する．同じコンポーネントが複数のアプリケーションから用いられるなどしてサービス／管理処理が実行される．そのため，あるコンポーネントの障害が複数のサービス／管理処理に影響を与えることは十分考えられる．重要なことは障害の検出とそのコンポーネント障害がどのアプリケーションに影響を与えるかの特定である．すなわち，障害箇所が能動的に系に悪影響を環境に及ぼさないような仕組みが求められる．従って，集中制御ではない分散制御の環境において，いかにアプリケーションの障害の波及を防止するかは重要な課題となる．

以上の問題を要求条件として整理すると以下のようになる．

(R1)効率的多重処理の実現

ネットワークインタフェースにおけるプロトコルのセッションやコネクションの同時処理，DPEサーバにおける問い合わせ，登録処理がシステム全体のボトルネックとならないため，効率的に並列多重処理を行えるハードウェア機構を実現することが必要となる．

(R2)高い信頼性の実現

アプリケーション障害やノード障害が，ネットワーク的に波及しないように，障害の局所化，迅速な障害対策が行え，高い信頼性を確保できる障害管理法の実現が望まれる．

(R3)ネットワークの再構成

広範囲にわたるネットワークが完全な中央制御でネットワーク障害対策を行う方法に加えて，通信情報ネットワーキング環境のような自律分散型のネットワークにおいては，ローカルな情報での自律的ネットワークの再構成の実現が望まれる．

(R4)高速信号処理

通信情報ネットワーキング環境ではマルチメディアストリームの高速・効率的処理が求められる

(R5)過負荷耐力の実現

DPEサーバにおける瞬時的トラヒック集中など，局所的なトラヒックの集中に対する耐力の実現が望まれる．

(R6)容易なマルチプロセッサ構成の実現

高負荷時に処理能力向上のための効率的マルチプロセッサ構成の実現が望まれる．

本研究では,(R1)の効率的並列処理と(R2)の高い信頼性を中心に,通信情報ネットワーキング環境の実現法を明らかにする.

2.4 結言

本章では,今後の情報通信サービスの要となりつつある通信情報ネットワーキング環境の実現に当たっての解決すべき課題,要求条件を明確化した.

まず,通信情報ネットワーキング環境の検討の最も代表的なものとして,世界の主な通信キャリア,コンピュータベンダ,ソフトウェアベンダで構成されたTINAコンソーシアム(TINA-C)及びそこで検討されてきた電気通信情報ネットワーキングアーキテクチャ(TINA) について概観した.

ついで,同環境を実現する上で,主にアプリケーションを支えるハードウェアプラットフォームの観点さらにアプリケーションの障害管理の観点から解決すべき問題点として以下を明らかにした.

(1)ネットワークインタフェースの多重時の非効率性

(2)DPEサーバーなど共通サーバへの負荷集中

(3)サーバノードの信頼性の確保

(4)ネットワークの信頼性の確保

(5)アプリケーション障害波及の防止

これらをもとに,通信情報ネットワーキング環境実現のための要求条件を抽出した.第一の要求条件として,通信情報ネットワーキング環境の高効率化の必要性を述べた.現状のシステムにおける多重処理の必要性を例示し,そこにおける非効率性が解決すべき課題として重要であることを示した.

さらに,次の要求条件として,通信情報ネットワーキング環境の高信頼化を述べた.ソフトウェア,ハードウェアそれぞれの観点からの高信頼化,特に障害管理の重要性を明確にした.

加えて,通信情報ネットワーキング環境において特徴的なマルチメディア処理の必要性,自律的なネットワーク再構成,過負荷耐力の実現などその他の要求条件を明らかにした.

第３章　データ駆動原理の通信情報ﾈｯﾄﾜｰｷﾝｸﾞ環境への適用性

3.1　緒言

本章では，２章で明らかにした，通信情報ネットワーキング環境実現上の要求条件を解決するための手段としてデータ駆動原理をとりあげ，その通信情報ネットワーキング環境への適用性の高さを示す．

まず，通信情報ネットワーキング環境の要求条件実現のために適用できる可能性の高いデータ駆動原理について，西川らが検討してきたストリーム指向データ駆動プロセッサをとりあげ，その原理，アーキテクチャを述べる．その中で，実測を通じ，本プロセッサのエラスティックパイプラインにより，２章であげた要求条件の１つである，過負荷耐力を有すること，各プロセッサが付加的ハードウェアなく相互に接続できることから，もう１つの要求条件である容易なマルチプロセッサ構成を満たしていることを示す．

続いて，通信情報ネットワーキング環境の効率化の観点から検討を行う．まず既存のノイマン型プロトコルの疑似多重処理の非効率性の問題点を明らかにするため，典型的な通信情報ネットワーキング環境の実現形態であるクライアント・サーバシステムを対象に，クライアントとサーバ間処理の多重度を変えることによって，処理多重度と各種処理時間の傾向を測定する．また同じ通信情報ネットワーキング環境を数学モデルによっても解析する．その結果，実測および数学モデルのいずれにおいても，多重度が上がるにつれて，１つのプロセスあたりの処理時間が長くなることを明らかにする．その原因は，ノイマン型疑似多重処理におけるコンテキストスイッチのオーバヘッドにあり，その処理の非効率性を明らかにする．

さらに，この結果を踏まえて，データ駆動プロセッサにより上記のクライアント・サーバシステムをエミュレーションによって実現し，同様の実験を行って，多重度に無関係に処理時間が一定という，きわめて高い処理の効率性を検証する．この結果は数学モデルによっても裏付けている．加えて，２章で示したその他の要求条件についても定性的に検討し，通信情報ネットワーキング環境に対する２章であげた６つの要求条件をデータ駆動プロセッサは満たす可能性が高いことを示す．

3.2 データ駆動原理

データ駆動プロセッサは，Dennisによって提唱されたコンピュータの並列処理アーキテクチャであり[48]，並列多重処理による計算速度向上を主たる目的として検討が進められてきた[49][50]．しかしながら，計算機アーキテクチャそのものによる高速化は，VLSIルールの極めてめざましい発展などによる現実のハードウェアの急速な進化に追いつくことはできず，高速演算そのものへの適用については検討が終息の方向にある．しかしながら，本来データ駆動方式の持つオーバヘッドの無い多重処理能力は，データ駆動プロセッサが対象としてきたジョブのスケジューリングが可能なインハウス型計算機環境ではない，非常に系のふるまいの予測が困難で不確定性が高いシステム（その典型

は通信網であるが）においては有力な処理方式を提供する可能性が高い．データ駆動プロセッサの検討の中で，西川らが検討してきたストリーム指向パイプライン処理[13]-[17]は，きわめて高い過負荷耐力を示しており，これもロバストな通信網環境を実現する上で適用可能性が高いと思われる．さらに，データ駆動原理がもつ，入力条件が充足されて初めて処理が発火するという仕組みは，ある機能の障害時にその処理に入力さえ与えなければ能動的な悪影響を環境に及ぼすことはない.その意味で通信情報ネットワーキング環境における信頼性向上対策に一つの大きなヒントを与えていると思われる．

本節では，このように，通信情報ネットワーキング環境の要求条件実現のために適用できる可能性が高いと思われるストリーム指向データ駆動プロセッサについてその概要を述べる．

データ駆動スキームは，実行に必要な入力トークンがそろった時に初めて処理が実行される並列分散処理原理である．本研究では，既に西川らが検討してきたデータ駆動プロセッサを前提に検討を進めた．西川らが実現したデータ駆動プロセッサはトークンにタグを付加した動的データ駆動原理に基づいている [13][50] ．本プロセッサはすでにVLSI化されており，タグを参照するとともに各処理ステージ間での送受確認のためのハンドシェーキング処理を用いることにより，パケット化されたデータストリーム（タグ付きトークン）の処理を行う[14][15][17].

一般に，並列プロセスにおけるデータ駆動処理の実行には，プロセス間の処理順序などによりデータの処理時間（データ量）に変動が生じる．西川らは，この変動を吸収しスループットを向上させるために,連結した機能間にバッファ機構を具備させている.さらにストリームはパイプライン的に処理されていくが,その分散したパイプラインステージでのボトルネックを回避するためにバッファ機構も設けている．すなわち，系を自律制御型のエラスティックパイプライン構成とし，データ量の変動を吸収し，合流分岐機構を設けてデータの自律ルーチングを可能とする，流れ処理方式を採用している．

図3.2.1に，動的データ駆動原理に基づく流れ処理方式を実現するエラスティックパイプラインで構成されたストリーム指向データ駆動プロセッサのアーキテクチャを示す[14][22].

図3.2.1(a)の発火制御及びプログラムメモリ（FCP)，機能処理ユニット（FP）でデータパケットが処理され次段に送られる．メモリ（Mem）はデータパケットをバッファし，継続的な過負荷状態における瞬時的オーバフローを吸収するエラスティックバッファ機能を実現する．図3.2.1 (b)のキューバッファ（QB）は，負荷の増減に応じて自律的にパケットの通過するステージ数を増減する可変長であり，系全体のエラスティック性を強化している．

このアーキテクチャによって，急激な負荷の変動にも耐えることができる．図3.2.2にその効果を示す[14][18][22]．一般のノイマン型のシステムでは，設計目標を越えた入力が加わると急激にスループットが低下する．しかし，本プロセッサにおいては，過負荷状態においても安定したスループットを実現している．これにより，2.3節の要求条件(R5)の過負荷耐力をデータ駆動プロセッサは実現できることがわかる．

また図3.2.1(c)に示すように，大きな負荷に対応するためにマルチプロセッサ構成も

容易に実現できる．各プロセッサは２入力２出力であり，それらを相互に結合することにより付加的なハードウェアを要することなくマルチプロセッサ構成が可能となる．各プロセッサのＱＢによりパケットフローの変動が吸収され各プロセッサ間のデータフローが平準化される効果を図3.2.3に示す[14][18][22]．これにより，2.3節の要求条件(R6)の容易なマルチプロセッサ構成をデータ駆動プロセッサは実現できることが言える．

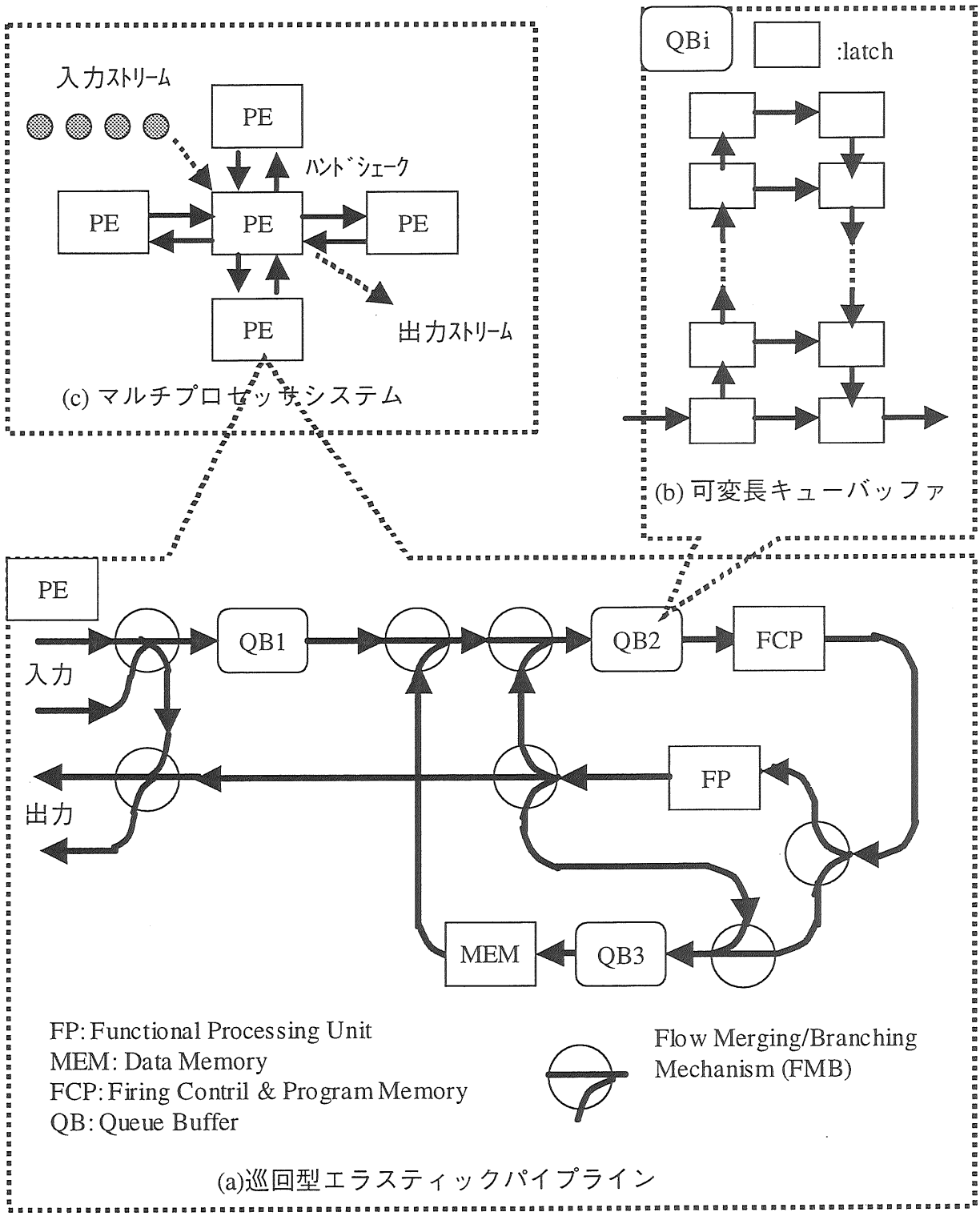


図3.2.1　ストリーム指向データ駆動プロセッサアーキテクチャ

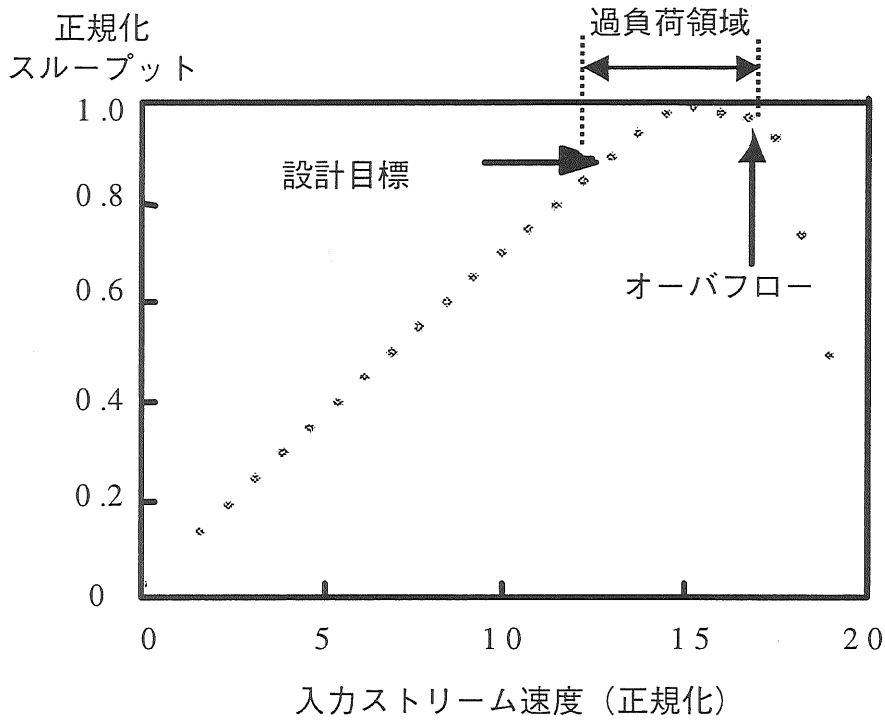


図3.2.2　ストリーム指向データ駆動プロセッサの過負荷耐力

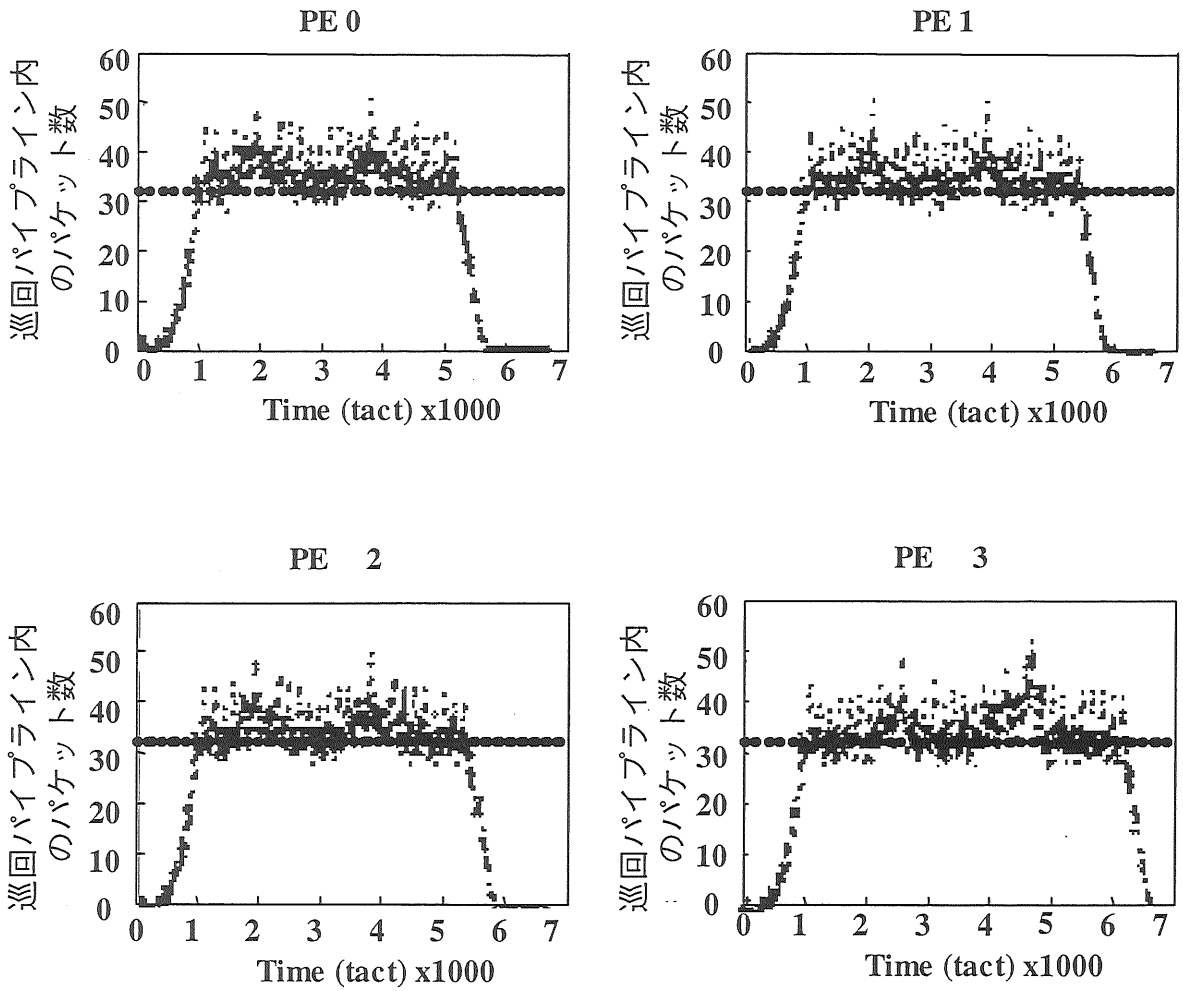


図3.2.3　ストリーム指向データ駆動プロセッサの
マルチプロセッサ構成における負荷平衡

3.3 既存環境の非効率性の検証

ここでは，2.3節の要求条件(R1)について検討する．データ駆動原理の適用性を検討する前に，既存のノイマン型プロセッサによる処理の非効率性を実測と数学モデルで検証する[18][22]．

3.3.1 既存環境の非効率性の実測による検証

2章に述べたように，ネットワークインタフェース及びDPEサーバにおける多重処理の効率化は重要な要求条件である．ここでは，通信情報ネットワーキング環境を構成するクライアント・サーバ構成を取り上げ，その間の処理多重度と各種処理時間の関係を実測する[18][22]．系の構成は，通信情報ネットワーキング環境の最小構成とし，1つのクライアントアプリケーションと1つのサーバアプリケーションを既存のワークステーション（ノイマン型処理）に搭載し，その間をイーサネットワークで接続し，CORBA上で通信（クライアントがサーバに処理を依頼し，その処理結果を受け取る）させて，処理時間，転送処理時間等と処理多重度（スレッド数）の関係を実測した．系構成，測定条件，及び測定項目を以下に述べる．

(1)系構成

2台のワークステーションをイーサネットワーク(100baseT)で接続する．それぞれのワークステーションはOS上にCORBAを搭載し，その上位にサーバアプリケーションとクライアントアプリケーションを載せる（図3.3.1）．

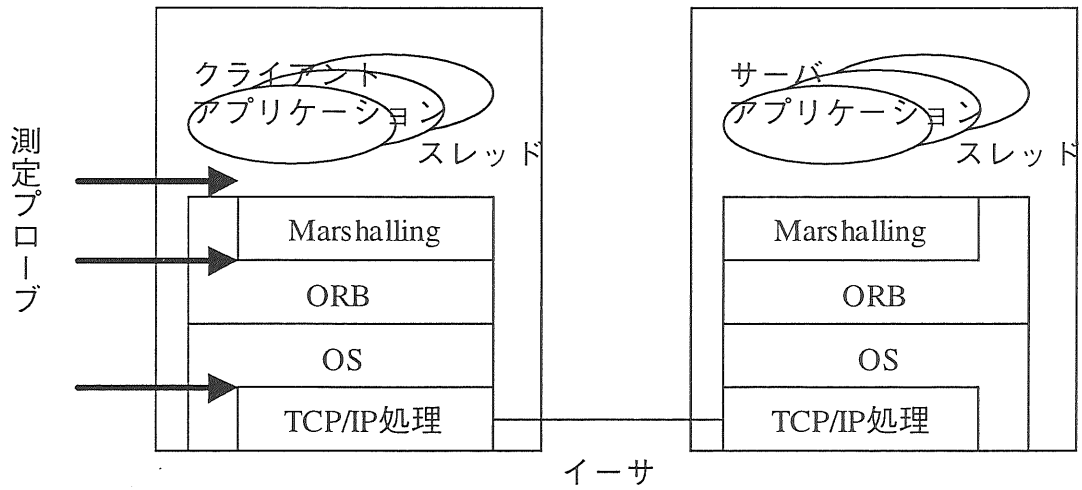


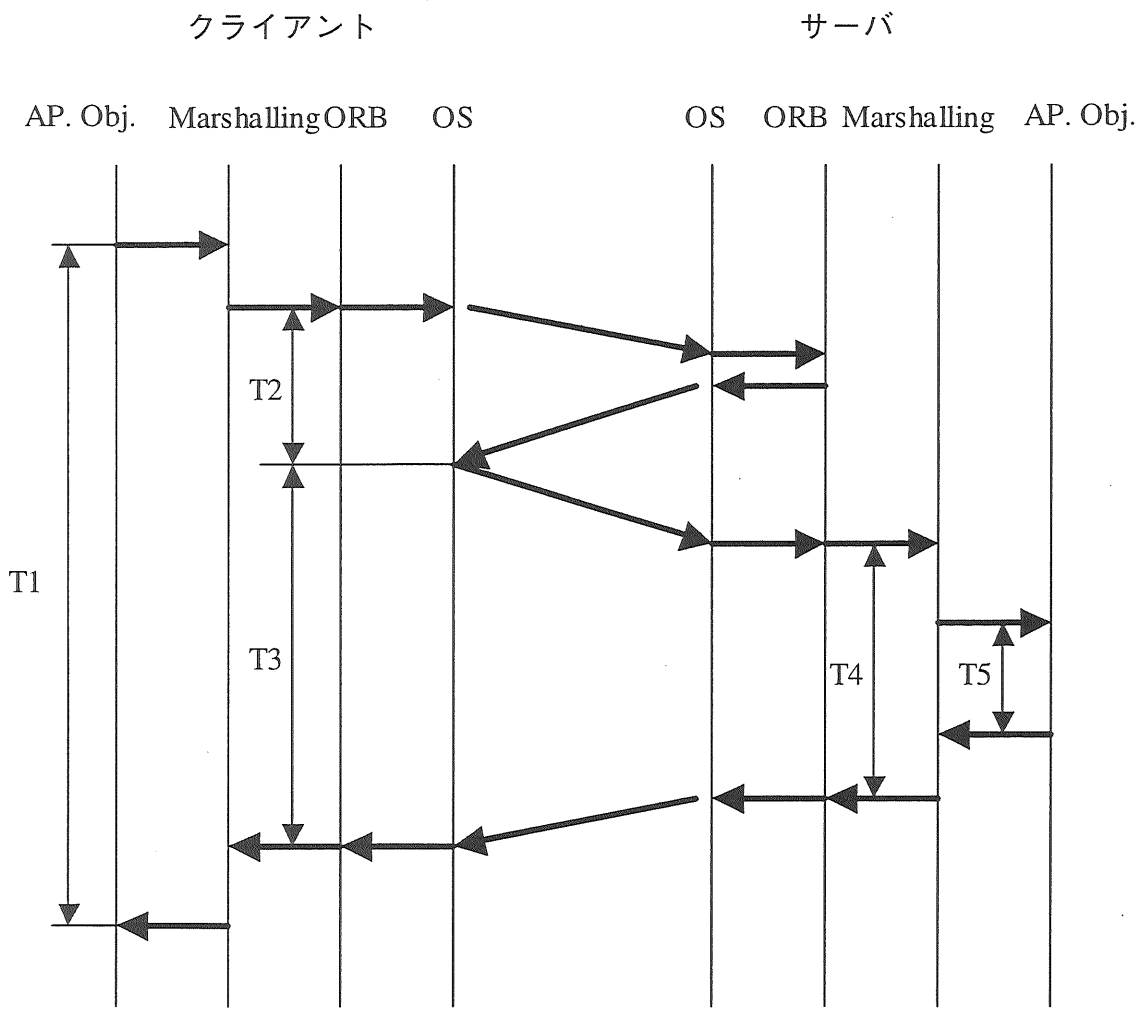
図3.3.1 ノイマンプロセッサの非効率性測定系構成

(2)処理の流れ

・クライアント側アプリケーションで処理の発生に対応づけてスレッドを生成する．
・スレッド生成後クライアントアプリケーションはORBに対して処理要求を渡す．
・クライアントORBは，アプリケーションからの処理要求をIIOP(Internet Inter-ORB Protocol)形式に形式変換する(Marshallingと呼ばれる処理)．Marshallingが終了すると，ORBはOSを介してTCPコネクションを相手ORBとの間に設定する．ORBはTCPコネクションを設定後，TCPパケットにより処理要求メッセージをクライアントOS・通

信路・サーバ側OSを介してサーバORBに送る.
・サーバ側ORBは受け取った処理要求メッセージをサーバ側アプリケーションが理解できるようにMarshallingを行い，サーバアプリケーションに届ける.
・サーバ側アプリケーションは受け取った処理要求メッセージ上のデータをデータベースに書き込み，書き込み終了後，クライアントに処理終了確認メッセージを応答する.
・処理確認メッセージは処理要求メッセージの逆の流れでクライアントアプリケーションに届けられる.

以上の処理を処理要求メッセージ上のデータサイズに応じて，軽微な処理（Lighter: 数バイト引数1msec未満のCPU時間）と重い処理（Heavier: 数百バイト引数10msec程度のCPU時間）の2つの場合に分け，同時に生成されるスレッド数（同時接続数）を1から45まで変化させてスレッドあたりの各種処理時間（(3)に示す）を10回測定しその平均をとった.



AP. Obj.：アプリケーションオブジェクト

図3.3.2　ノイマンプロセッサの非効率性測定項目

(3)測定項目（図3.3.2）

測定プローブは，アプリケーションとORBのMarshalling機能間，ORBのMarshalling機能とORB本体との間，TCPソケットに設け（図3.3.1参照），以下の5種類の時間を測定した．

(i)T1

クライアントアプリケーションが要求メッセージをMarshalling機能に発出後，Marshalling機能から確認メッセージを得るまでの時間，すなわち総応答時間

(ii)T2

クライアント側でMarshalling終了後，ORB本体に対して処理要求を引き継いでから，ORBがTCPコネクション設定要求を発出し，サーバ側ORBからコネクション設定完了を受信し，処理要求TCPメッセージをクライアント側が発出するまでの時間，TCPコネクション設定時間に対応する．

(iii)T3

クライアント側が，要求TCPメッセージ転送を開始してからサーバからの確認メッセージがクライアントMarshalling機能に到着するまでの時間

(iv)T4

サーバ側において，要求メッセージをORB本体からMarshalling機能が受信してから，Marshalling機能がORB本体に確認メッセージを引き継ぐまでの時間

(v)T5

サーバ側でアプリケーションが処理要求をMarshalling機能より受信してから確認メッセージをMarshalling機能に引き継ぐまでの時間，アプリケーション処理時間に相当する．

これらから，次の6種の測定値が得られる．

(a)総応答時間[T1: Total turn around time]
(b)クライアント側でのデータ形式変換時間[T1-(T2+T3): Marshalling(C)]
(c)TCPコネクション設定時間[T2: Connection setup]
(d)イーサ上のTCP/IPデータ転送時間[T3-T4: Data transfer]
(e)サーバ側でのデータ形式変換時間[T4-T5: Marshalling(S)]
(f)サーバ側でのアプリケーション処理時間[T5: Server processing]

軽微な処理と重い処理の場合の測定結果を，図3.3.3の(a)と(b)にそれぞれ示す．

図3.3.3から，サーバ側でのアプリケーション処理時間及びサーバ側データ変換処理時間は，軽微な処理ではスレッド数にほとんど依存しないが，重い処理ではスレッド数の増加に従い増加すること，コネクション設定時間，データ転送時間は，処理の重さによらず，スレッド数の増加に従い増加していることがわかる．以下にその原因を考察する．

(1)サーバでの処理

アプリケーション処理時間およびデータ形式変換時間の傾向を解析する．ノイマン型処理は逐次処理である．従って，多重処理は擬似的に実現されている．すなわち，1つの処理を中断して次の処理に切り替え，という処理を繰り返して擬似的多重処理を実現

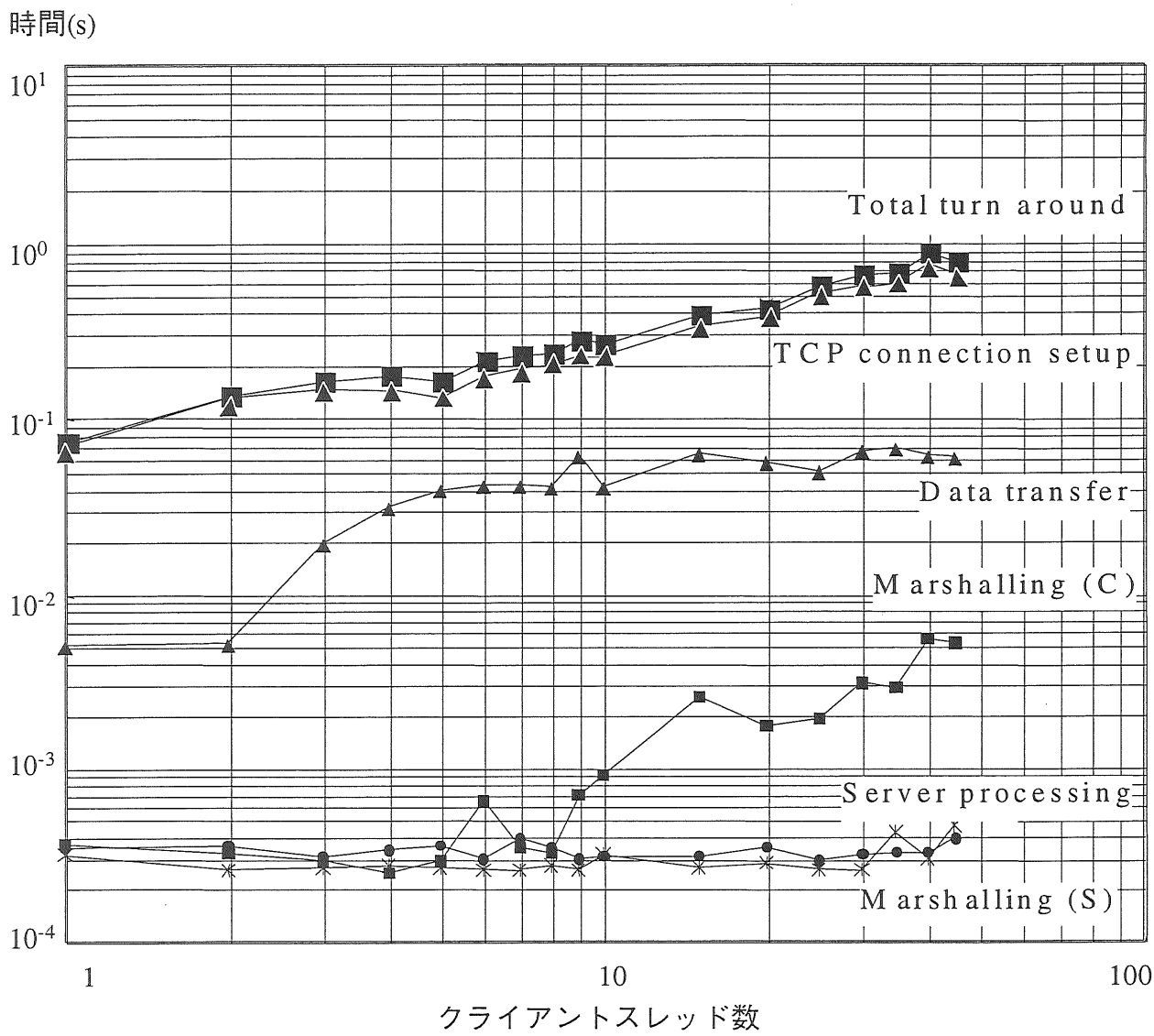


図3.3.3(a)　既存環境のスレッドあたりの処理時間（軽負荷）

している．重い負荷ではスレッドの増加に従い多重処理を実現するため，スレッド実行中に割り込みが生起する割合が高くなり,スレッドの切り替え及び実行待ちのオーバヘッドが増加し処理能力を消費していると考えられる．一方軽微な処理では，処理が短時間であり，スレッド実行中に割り込みが生起する割合が少ないため，頻繁なスレッド切り替えが生起しておらず，スレッド数の増加と処理時間の相関は少ないと考えられる．

(2)ネットワークインタフェースにおけるプロトコル処理

コネクション設定時間，転送時間とも，図3.3.3の(a)と(b)を比較すると，処理の重さ（引数の長さ，即ちデータ長）にかかわらずスレッド数増大に伴い同じ傾向の伸びを示し，かつ同一のスレッド数に対してはほぼ同一の処理時間がかかっている．これらはメッセージがネットワークを転送されるため，時間には，伝搬遅延時間と，クライアント及びサーバノードにおけるプロトコル処理時間が含まれる．

ここで，イーサネットのパケット長（最大1．5kbyte）に比べて，コネクション設定パケット及びデータパケットの長さは十分小さく，クライアントからのパケットはすべ

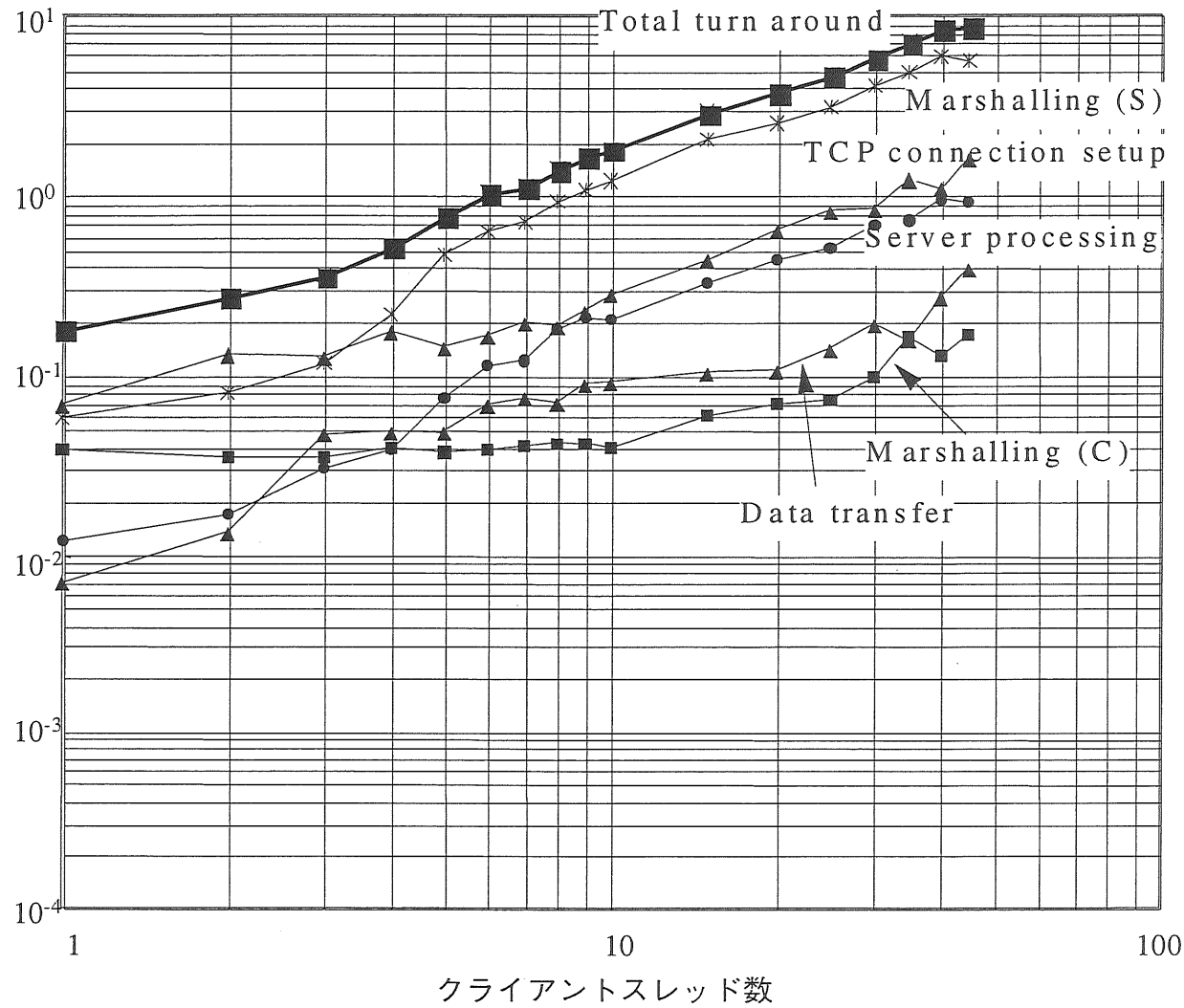


図3.3.3(b)　既存環境のスレッドあたりの処理時間（重い負荷）

て１パケットに収まる．よって，実験に用いたイーサネットの転送能力（実質数Mbit/s）を考慮すると，伝搬遅延は支配項とならないと考えられる．

一方処理時間について，コネクション設定処理はデータを搭載しないパケットにより実行されるため，データ長に依存しない処理時間となる．データ転送処理も，送出元での各レイヤパケットの組立処理・手順処理，イーサの中継ノードでのMACヘッダ解析処理，そして受信側の手順処理・分解処理など，いずれも，パケットヘッダ処理が支配的で，データ長の影響を受けないと考えられる．

従って，同時に発生するデータ転送要求の数（スレッド数）に依存して，TCPコネクションの設定処理，データ転送処理時間が増大する理由は，パケットヘッダの多重処理におけるスレッド切り替えオーバヘッドが増大しているためと考えられる．

以上の測定結果を要約すると，逐次処理の拡張としての擬似的多重処理では，オーバヘッドが原理的に免れない．換言すれば，多重処理されるスレッド間での相互干渉が本質的な問題と考えられる．

3.3.2 既存環境の非効率性の数学モデルによる検証

3.3.1で実測によって，ノイマン型プロセッサのスレッド切替オーバヘッドの問題を指摘した．ここでは待ち行列モデルを用いて，同様にこの問題を検証する[27][28]．

図3.3.4にCORBAを介した通信情報ネットワーキング環境のクライアント・サーバ通信モデルを示す．

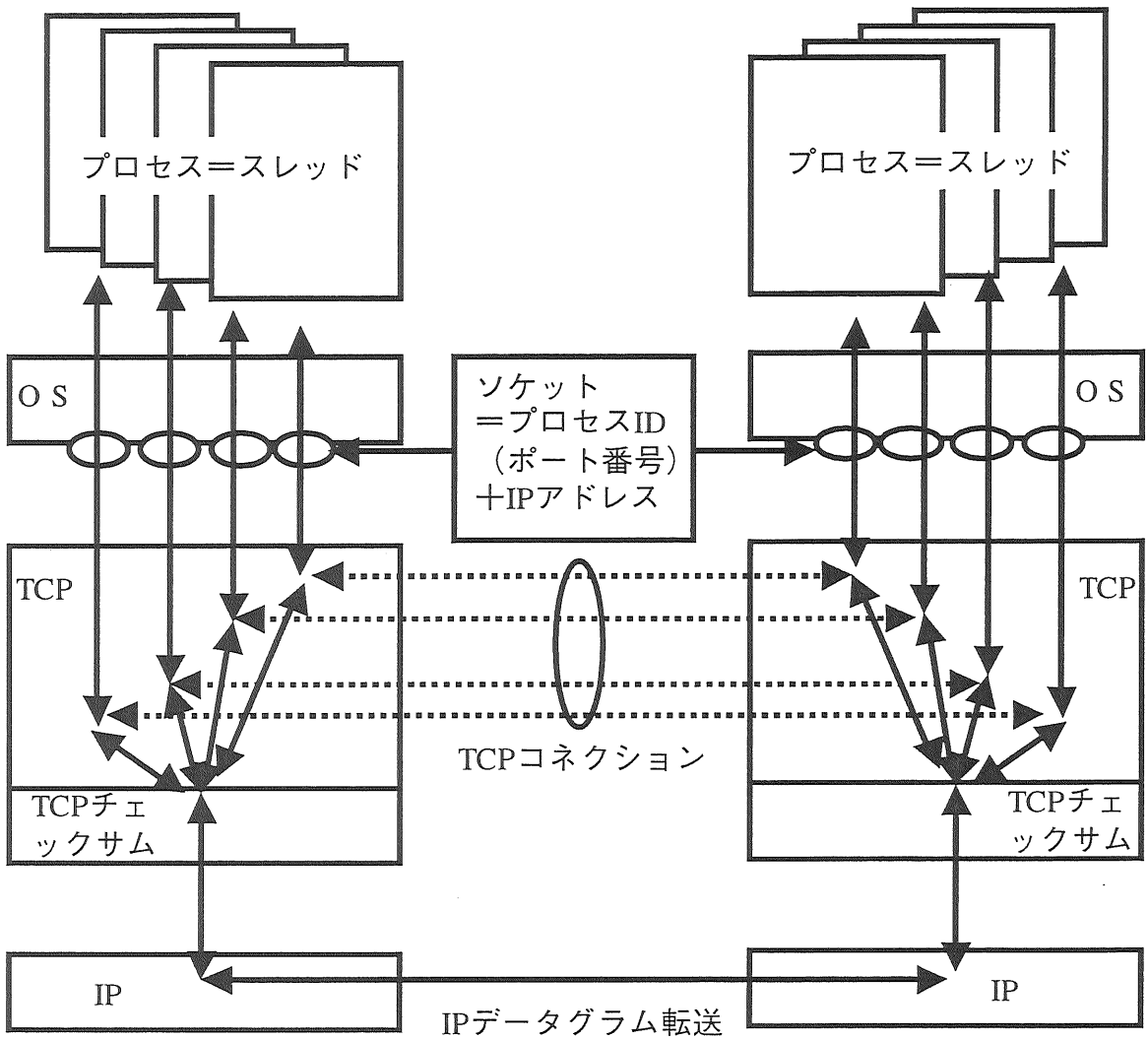


図3.3.4 CORBAを介した通信情報ネットワーキング環境の通信モデル

図3.3.4をキューイングモデルで表現すると，図3.3.5のようになる．TCPとIPのプロトコル処理が直列に実現され，各処理段において，処理中断割込が発生する形になる．

さらに，ひとつのキューを取り出すと図3.3.6のように表現できる．

図3.3.6を説明する．

・到着したパケットは，通常処理部に入る．

・通常処理中に新たな割込が生起したときには，(1)のパスを通り，切り替えオーバ

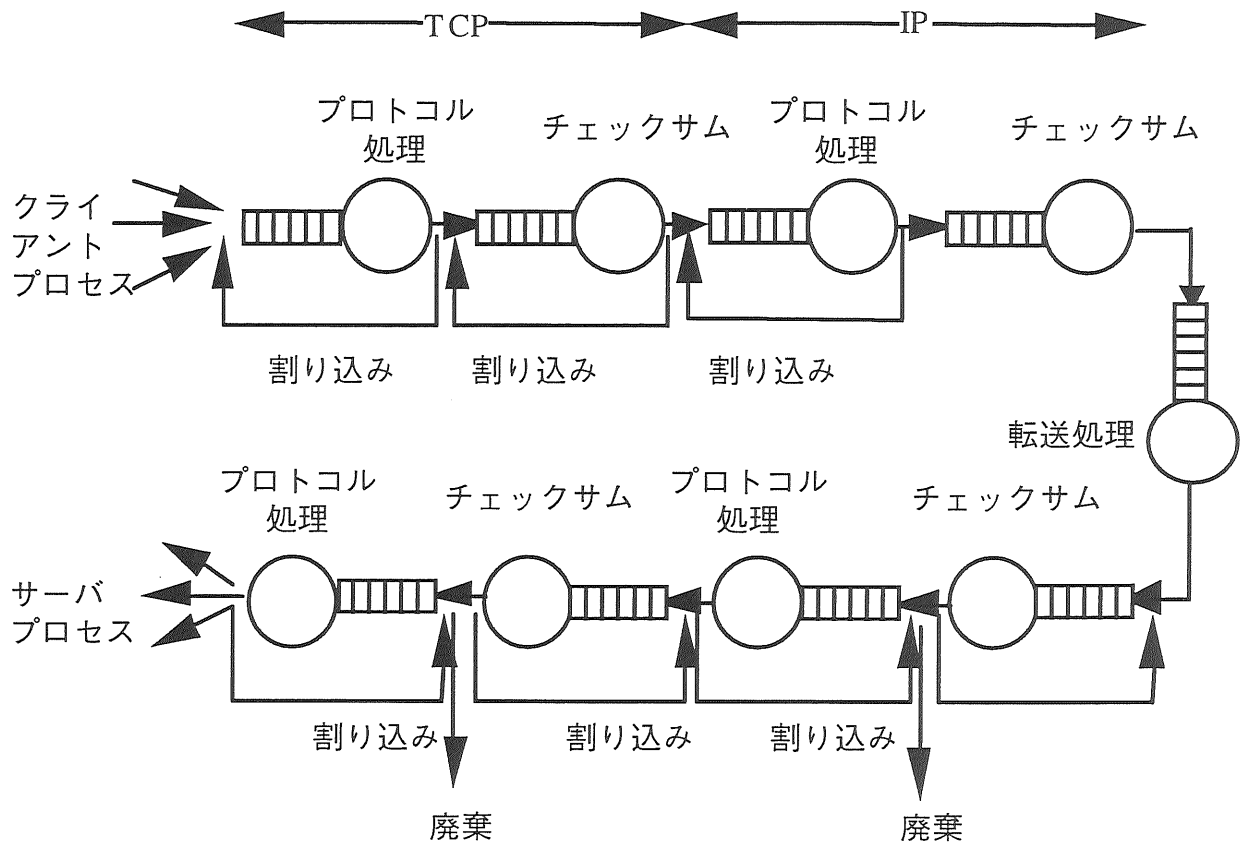


図3.3.5　キューイングモデル

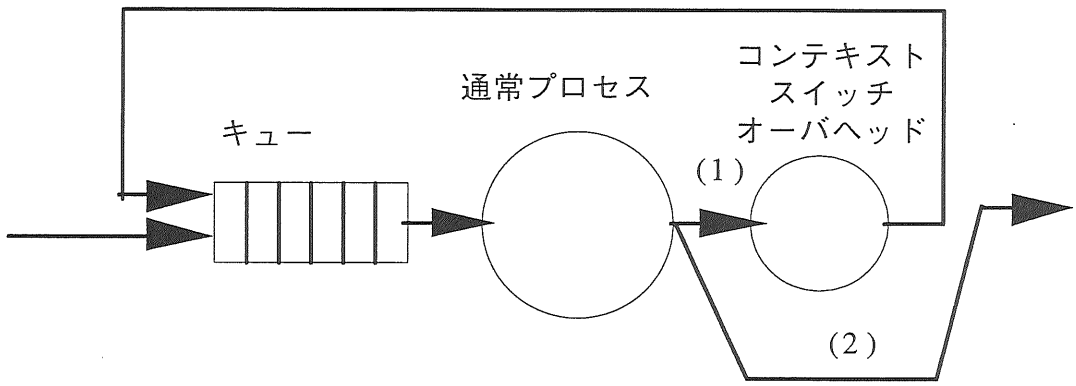


図3.3.6　キューイング詳細モデル

ヘッド時間を消費し，その上で，もう一度残りの処理を待つためにキューに再入力される．

・通常処理中に１個も割り込みが生起しないときは，オーバヘッドなしに出力される．
　図3.3.7に割込の様子を示す．

ここで,

処理能力：C [bit/sec]

パケット長：p [bit]

切替オーバヘッド：s [sec]；一定と仮定

割込生起確率：v＝f (q)；q はキュー長

とすると,

割込発生間隔：1／v [sec]

となり,

実効処理能力＝C×｛(1／v) －s｝／(1／v) ＝C (1－v s)[bit/sec]

1パケット処理時間　T＝p／｛C (1－v s)｝[sec]

となる.

簡単のために，スレッド数をキュー長とみなし，割り込みはキュー長に単純比例して起こると仮定し,

f (q)＝k (q－1) とすると；k は定数

q＝n のとき,

T＝p／｛C (1－k (n－1) s)｝ ただし，n＜1／s ・・・式(1)

となる.

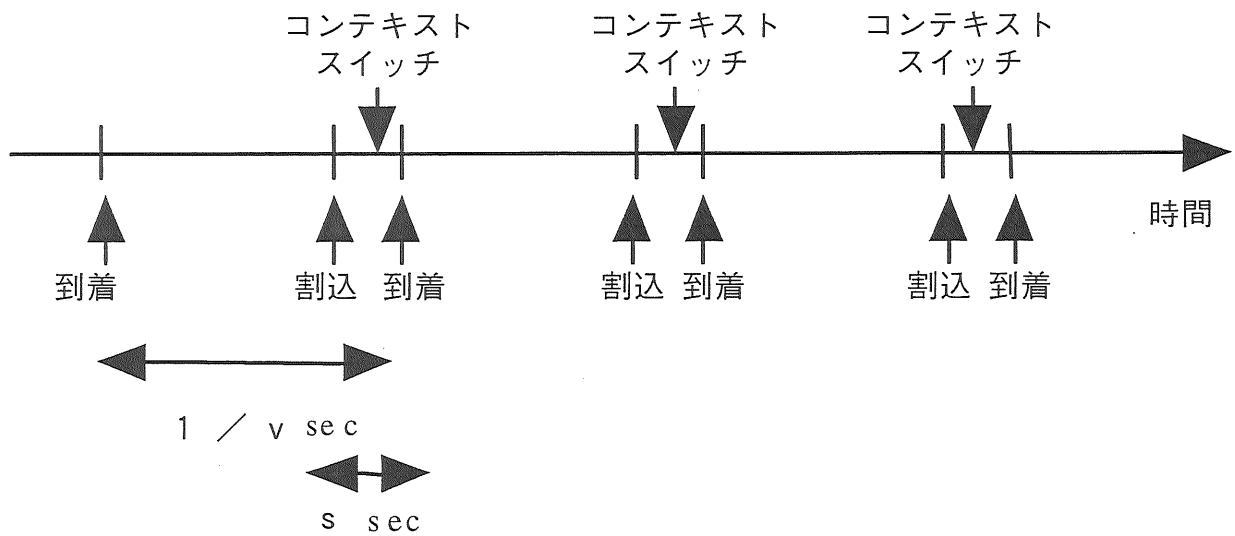


図3.3.7　割り込みのようす

図3.3.8に1パケット処理時間とキュー（スレッド数に対応）の関係を示す.

明らかに，オーバヘッドによって，単一パケット処理時間の上昇傾向が見え，3.3.1の実測結果を裏付けている.

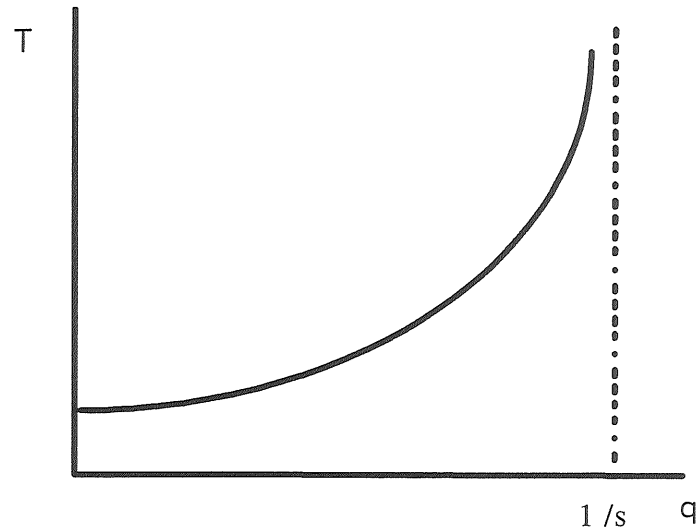


図3.3.8　キュー長と処理時間

3.4　データ駆動原理の適用性

3.4.1　多重処理／過負荷耐力実現への適用性

データ駆動方式では，その処理駆動原理からわかるように，一旦起動された処理は互いに独立であることが保証される．図3.4.1に，3.3節に対応させて，データ駆動プロ

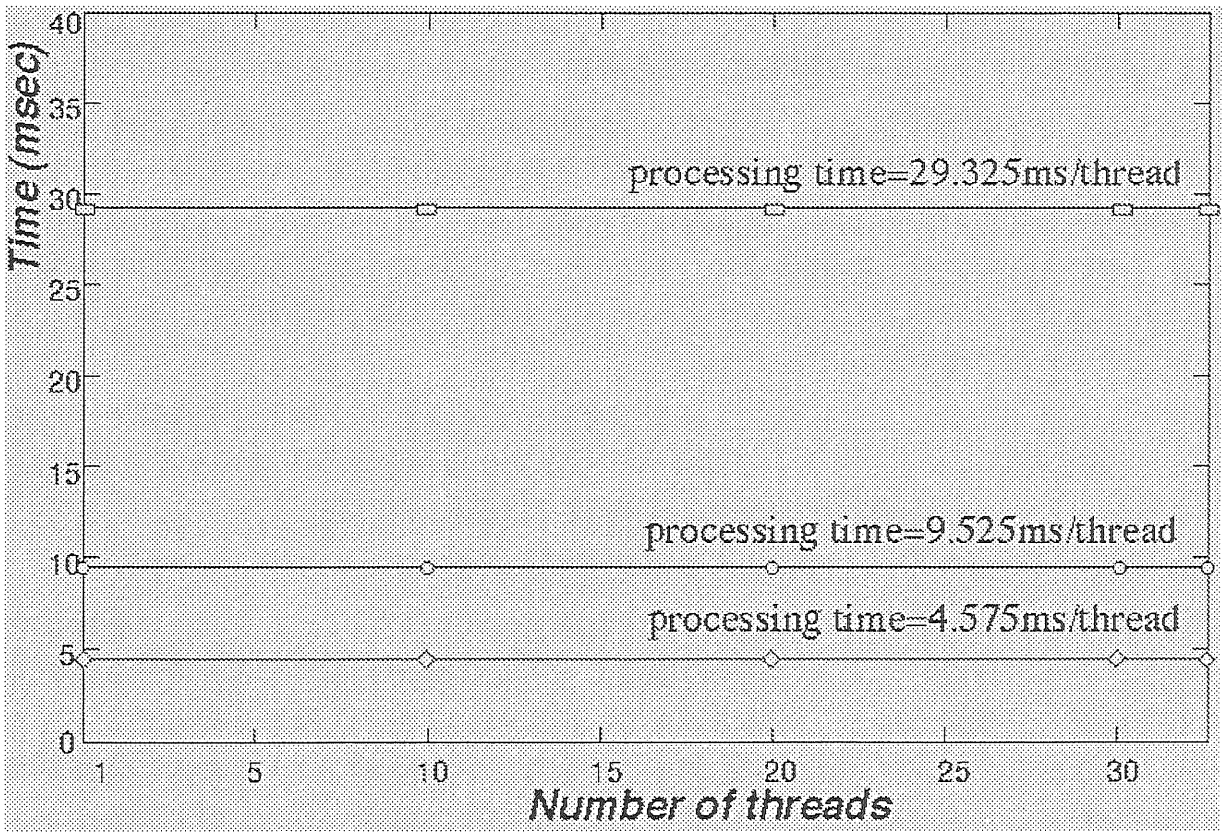


図3.4.1　データ駆動プロセッサの場合のスレッド当たり処理時間

セッサを用いたマルチスレッド実行に関して各スレッドの実行時間を図に示す3種類に設定した場合の実行結果を示す．この結果からわかるように，ノイマン型プロセッサでの実行結果とは異なり，スレッドの処理負荷の大きさと同時処理スレッド数の大小に関わらず，処理時間は一定となっている．このことは，スレッド間の相互干渉によるオーバヘッドは起こっていないことを意味すると考えられる．

数学モデル的には，既に述べたように，過負荷状態に陥らない限り，コンテキスト切り替えオーバヘッドが存在せず，3.3.2の(式1)において

$S=0$

となり，パケットあたり処理時間は，

$T = p / C\ [sec]$

と一定となり，実測を裏付けている．

3.3節で検証した既存環境の非効率性は，プロセッサの処理能力の向上により，現象的には大きな問題とはなっていない．しかし，今後のDPE上の処理量の増加，特定サーバへの処理集中を考慮すると，上記のオーバヘッドを原理的には全く有さないデータ駆動原理の適用を検討する価値は高いと考えられる．すなわち，2.3節の要求条件(R1)の効率的多重処理を満たしていることがわかる．さらに，今後の実環境においては，瞬時的な負荷の増大による過負荷状態も十分想定され，また，負荷の増大に対するプロセッサのスケーラビリティも要求されることは明らかである．このことは3.2で示した，過負荷耐力の大きさ（2.3節の要求条件(R5)）と負荷均衡化の容易さ（2.3節の要求条件(R6)容易なマルチプロセッサ構成）にすぐれるデータ駆動プロセッサの適用性の高さを示している．

3.4.2 信頼性・ネットワーク再構成への適用性

ネットワークの信頼度は，eビジネスへの展開を考慮すると，インターネット上の通信情報ネットワーキング環境においてひとつの重要な課題である．ネットワークそのものの信頼度向上のための方策は，通信路の多重化，ネットワーク管理システムの適用など，多く検討されている．本研究では，インターネットのような信頼度が十分に実現できないネットワークを利用した通信情報ネットワーキング環境の構築時に，エンドシステムとしてできる方策を考察する．

既存のシステムやネットワークに大きな影響を与えることなく信頼性を確保できる方策として,通信情報ネットワーキング環境のノードにつながるネットワークを2重化し，データ駆動プロセッサを用いた自動的な系再構成を検討した[24]．

図3.4.2に，自律系再構成方式案の基本系構成を示す．

本方式においては，ネットワークを介して対向するエンドシステムをデータ駆動プロセッサシステムで構成し，メッセージの転送ルートを二重化する．そのシステム間で通常ルートに対して試験パケットを循環させる．遅延や不達などの異常を検出すると，そのルートすなわちネットワークに問題が生じたとみなし，自律的にデータ転送経路を予備側に切り替える方式である．これにより，ネットワーク障害への対応の可能性が生まれる．現実にはこれらの二重化ルートが，ネットワーク内で物理的に別経路を通過する

ような対策は別途必要であるが，構内系等それらのネットワーク対策が比較的容易な系においてはその適用性は高いと考えられる．これにより，２章の要求条件(R3)のネットワークの再構成を満たしうることがわかる．

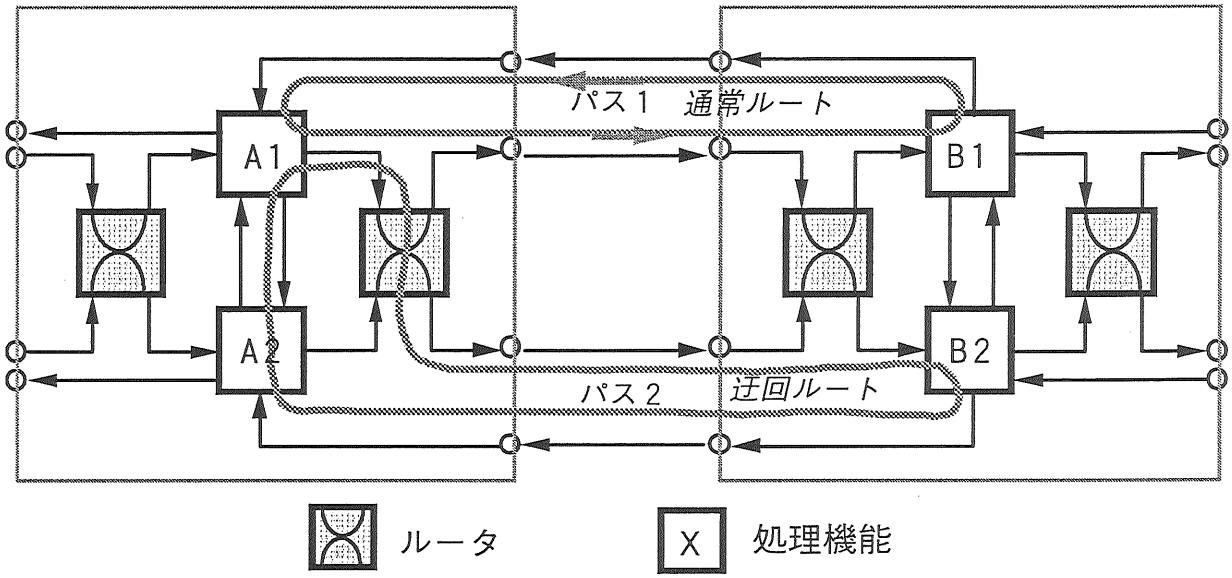


図3.4.2　自律系再構成方式案

3.4.3　信頼性（障害波及防止方式）への適用性

通信情報ネットワーキング環境の障害において問題となるのは，障害箇所の特定，障害部分の他への悪影響の防止，そして迅速な回復である．

通信情報ネットワーキング環境のDPE上のアプリケーションによるサービスは一般人間社会を模擬するものとも考えられる．アプリケーションソフトウェアのオブジェクトを個々の人間と考えると，人間が自律的に分散して社会を形成するように，オブジェクトが分散処理環境を形成している．従って，そのアナロジで考えると，オブジェクトの故障対策にも人間システムと同様な仕組みが必要と考えてよい[27][28]．

表3.4.1に，人間社会と通信情報ネットワーキング環境を対比して示す．パッシブ障害（病気）は障害発生時にそれ自身が機能を停止することが多いのに対して，ポジティブ障害では障害発生（犯罪発生）から障害箇所が特定され，それが系から切り離される(逮捕)までの間に障害を系にポジティブにまきちらす可能性が問題になる．ポジティブ障害発生を防ぐ方法として，データ駆動原理の適用性が高い．データ駆動原理の入力条件が充足されて初めて処理が発火するという仕組みは，ある機能の障害時にその処理に入力さえ与えなければ能動的な悪影響を環境に及ぼすことはないと考えられる．これによりポジティブ障害は発生せず，パッシブ障害のみとなり，上記の要求条件に合致する．適用箇所としては，サーバー系の多くの処理が比較的集中しやすい箇所が考えられる．

また，障害回復は人間社会では，社会全体を一カ所で集中して管理する機構は存在せず，社会ルールにのっとり，障害の生起した周辺地域でローカルに処置されるものである．このアナロジーで，データ駆動プロセッサをローカルに相互監視する仕組みを設け

れば同様に障害対策の局所化が可能となる．これにより，2.3節の要求条件(R2)の高い信頼性，のうちハードウェア部分について満たしていることがわかる．

表3.4.1　人間社会と通信情報ネットワーキング環境の対比

	人間社会	通信情報ネットワーキング環境
共通	人間	オブジェクト
	生産活動、社会活動	処理
事象（パッシブ）	病気、ケガ	障害
	自分自身，身の回りの知人	障害監視機構
	病院	障害の診断、回復
事象（ポジティブ）	犯罪	障害
	警察	障害監視機構
	裁判所	障害の診断
	刑務所	回復
共通	会社、組織	代替オブジェクト割り当て

3.4.4　高速信号処理

既に西川らの開発したストリーム指向データ駆動プロセッサは，実用化され，製品化されている[51][29]．主な適用先は高速信号処理であり，HDTV信号処理などに活用されている．これから，２章であげた要求条件(R4)高速信号処理はすでに満たされていることがわかる．

3.5　結言

本章では，データ駆動原理その通信情報ネットワーキング環境への適用性の高さを示した．これにより，２章で明らかにした，通信情報ネットワーキング環境実現上の要求条件を解決できる可能性を示した．

まず，西川らが検討してきたストリーム指向データ駆動プロセッサの原理，アーキテクチャを述べた．実測により，２章であげた要求条件の１つである，過負荷耐力を有すること，もう１つの要求条件である容易なマルチプロセッサ構成を満たしていることを示した．

続いて，典型的な通信情報ネットワーキング環境の実現形態であるクライアント・サーバ間の処理の多重度と各種処理時間の関係を，実測および数学モデルによって解析した．その結果，多重度が上がるにつれて，１つのプロセスあたりの処理時間が長くなることから，多重処理におけるコンテキストスイッチのオーバヘッドが大きくなっていることを示した．これにより，既存のノイマン型プロセッサの多重処理の非効率性を明

らかにした.

さらに，データ駆動プロセッサをエミュレーションによって実現し，多重度に無関係に処理時間が一定という，きわめて高い処理の効率性を検証した．また，数学モデルによっても効率性を示した．ネットワーク再構成，障害対策，高速信号処理などその他の要求条件についても定性的に検討した．その結果，通信情報ネットワーキング環境に対する2章であげた6つの要求条件をデータ駆動プロセッサは満たしていることを示した．これらから，通信情報ネットワーキング環境へのデータ駆動プロセッサの適用性の高さを示した.

第4章　TCP/IPプロトコル処理のデータ駆動実現

4.1　緒言

本章では，通信情報ネットワーキング環境の要求条件の１つである効率的多重処理に着目し，具体的な適用先として，通信網における多重処理が典型的になされるプロトコル処理を取り上げた．プロトコル処理は，それ自身で閉じた機能ブロックを構成し，既存ノイマン型ではないデータ駆動プロセッサを適用する敷居が低いこともあり，３章の結果を踏まえ，プロトコル処理へのデータ駆動プロセッサの適用を検討する．

多くの通信プロトコルの中から，通信情報ネットワーキング環境において，また昨今のインターネットにおいて最もよく用いられるプロトコルであるであるTCP/IPを例に取り上げ，そのデータ駆動実現を検討する．

まず，IP処理部のみを対象に，ノイマン型処理とデータ駆動型（並列）処理をデータ駆動プロセッサによりエミュレーションし,処理多重度と処理時間の関係を比較する.その結果，複数の同時処理がお互いに影響しない後者の高い効率性を示し，データ駆動型並列処理のTCP/IP処理への適用性の高さを明確にする．

その結果にもとづき，TCP/IPプロトコル処理装置をデータ駆動プロセッサの並列処理を生かして実装し，性能の検証を行い，データ駆動プロセッサを用いたプロトコル処理の高い効率性と通信情報ネットワーキング環境への適用性を示す．

続いて，このデータ駆動TCP/IPプロトコル処理装置の適用先として，通信情報ネットワーキング環境へのアクセス回線を選び，得られた効率性を実際の環境に適用する検討を行う．

通信情報ネットワーキング環境では，比較的軽量で機能がそれほど高くないユーザ装置が環境の一環に組み込まれ，サービス実現する．そのために，アクセス回線の効率的利用が重要な課題となり，本研究の成果であるデータ駆動型TCP/IPプロトコル処理装置の適用先として適切と考えられる．同時に，端末による通信情報ネットワーキング環境へのアクセスのためのその他の要求条件を検討し，その要求条件にもとづいた通信情報ネットワーキング環境のアクセス系の実現法を検討する．

4.2　TCP/IPプロトコルのデータ駆動実現

4.2.1　エミュレーション

３章の結果を踏まえて，通信情報ネットワーキング環境が利用するインターネットプロトコル（TCP/IP）を多重処理性にすぐれるデータ駆動型で構成することにより，高効率性を実現する検討を行った．

まず，データ駆動型並列処理の高効率性を実証するために，疑似並列処理のノイマン型プロトコル処理と並列処理の両者をエミュレータによって性能を比較することにした．傾向を見るために，IP受信処理部のみをエミュレートした[27][28][30]-[33]．

既に開発されている実時間で高精細ビデオ信号処理を実現できる１チップ型データ駆

動プロセッサシステムをエミュレータとして使用した[29]．図4.2.1にエミュレータ構成図を示す．

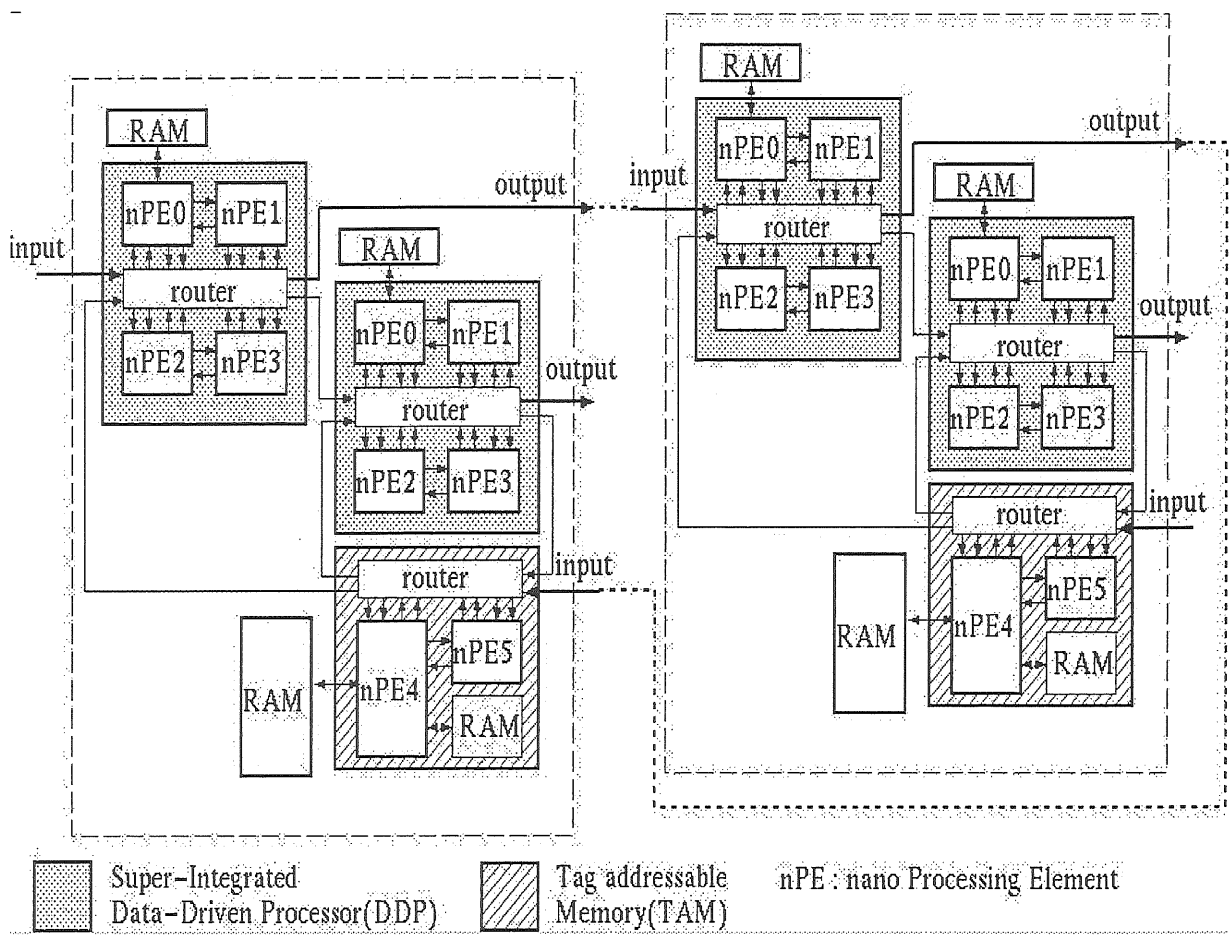


図4.2.1　TCP/IPエミュレータブロック図

ここで，IPプロトコル処理装置エミュレータは次の範囲をエミュレートしている．

(1)IPパケットヘッダの受信部

(2)IPパケットヘッダの検査

(3)検査結果に基づいて，正常であれば上位レイヤにパケットを出力し，異常であれば廃棄する

このIP処理部のエミュレータにより，ノイマン型(Sequential version)とデータ駆動型(Concurrent version)の多重処理性能を比較した．比較結果を図4.2.2に示す．

縦軸はプロセス当たりの処理時間を示している．ここで，プロセスとは「マルチプロセシング」におけるプロセスに対応するもので，並列して同時に処理されるIPプロトコル処理における各処理単位を意味している．上記エミュレーション範囲に示したように，ここでの処理時間はIPパケットを受信してから上位レイヤに出力するまでの時間となる．横軸は入力ストリーム速度を示す．ノイマンプロセッサにおけるスレッドはデータ駆動プロセッサにおけるトークンに対応する．トークンの長さはメッセージ長に依存する．こ

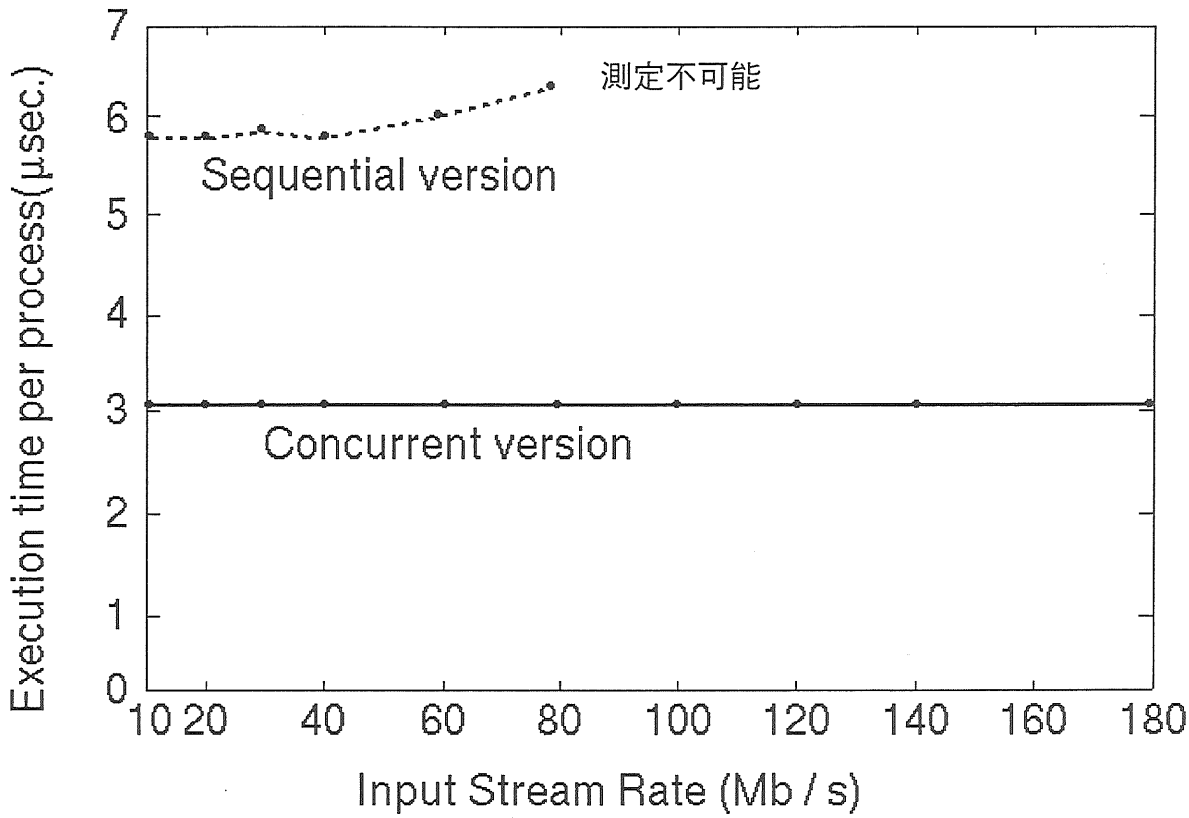


図4.2.2　データ駆動エミュレータによるTCP/IP処理のノイマン型実現とデータ駆動型実現の性能比較

こでメッセージ長を50バイトとすると，20Mbit/sの入力ストリーム速度は450，000 スレッドに対応する．

ノイマン型の模擬は，パケットのパイプライン処理を，バッファメモリとメモリへの直列アクセスのみを用いて実現した．この処理においては，繰り返し処理が頻発し，その制御のために結果として3.3の数学モデル(図3.3.8)と同様に，ある値以上(約80Mbit/s)の入力では，多重処理が実現不可能となった．

これに対して，データ駆動型IP転送処理部は，パイプライン処理に加えて，繰り返しを防ぐためバッファメモリを用いない並列処理を導入した．結果として不要な繰り返し処理のためのオペレーションが劇的に減少し安定した能力を実現した．

これらのエミュレーション検討により，データ駆動プロセッサは効率的なプロトコル処理に適用性が高いことを明らかにした．

4.2.2　TCP/IP実装

4.2.1のエミュレーション検討より，データ駆動型並列処理は，高効率プロトコル処理にふさわしいことがわかった．これを踏まえて，データ駆動プロセッサを用いた並列処理型TCP/IPプロトコル処理装置の実装を行った[27][30]-[33]．ここで作成したデータ駆動プロセッサは，本研究のプロジェクトコード名であるCUE(Coordinating Users' Requirements and Engineering constraints) におけるプロトタイプ第1版としてCUE-pと呼んでいる．図4.2.3にCUE-pボードとそのブロックダイアグラムを示す．

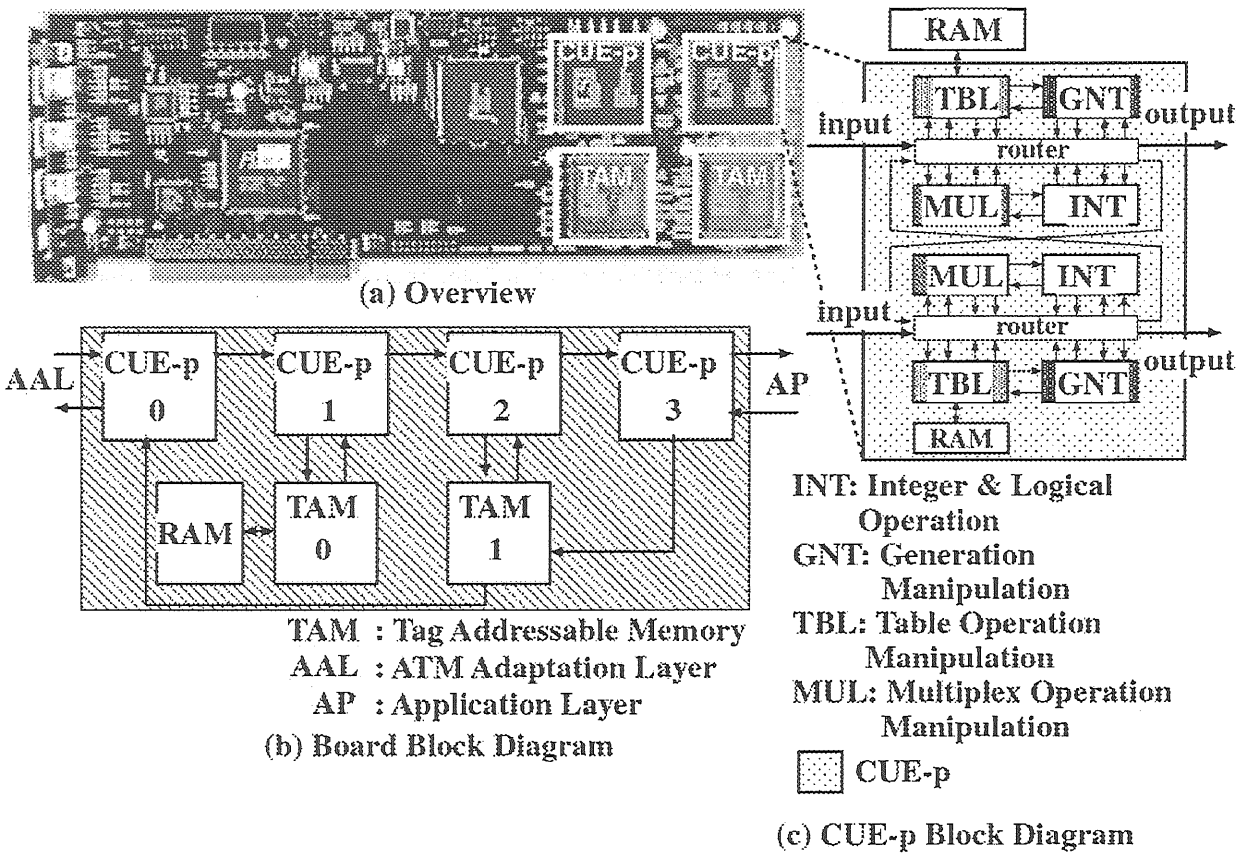


図4.2.3　TCP/IPプロトコル処理データ駆動プロセッサ(CUE-p)ボード

図4.2.4にTCP/IPのデータ駆動実装のマクロレベルのデータフローグラフを示す.

並列性を生かすためにIP処理とTCP処理を並行して行うとともに，TCPヘッダ処理とチェックサムも並行して行う．以下に処理の流れを示す.

(1)入力IPパケットは，TCP/IPプロトコル処理装置に入る前にIP headerとIP data (TCP segment)に分割される.

(2)IPヘッダは，"Check IP header & Generate pseudo header"（IPヘッダ検査及び疑似ヘッダ生成）部に入力される．ここで，IPのチェックサムが検査され，誤りヘッダは廃棄される．誤りのなかったヘッダは疑似ヘッダに変換され，正常結果を示す信号が"output transaction"（出力トランザクション）部に送られ，疑似ヘッダは"Check TCP header"（TCPヘッダ検査）部及び"Generate Ack segment"（Ackセグメント生成）部に送られる.

(3)IPデータはTCP segmentであり，"Check TCP header"部に入力される．ここでIPデータは疑似ヘッダとともにTCPチェックサムが検査され，TCPデータとともに検査結果は"Output transaction"部に送られる．TCPヘッダそのものは"Generate Ack segment"部に送られる.

(4)TCP確認信号(Ack)は"Generate Ack segment"部で生成される.

(5) "Output transaction" 部では，IP header check signalとTCP header check

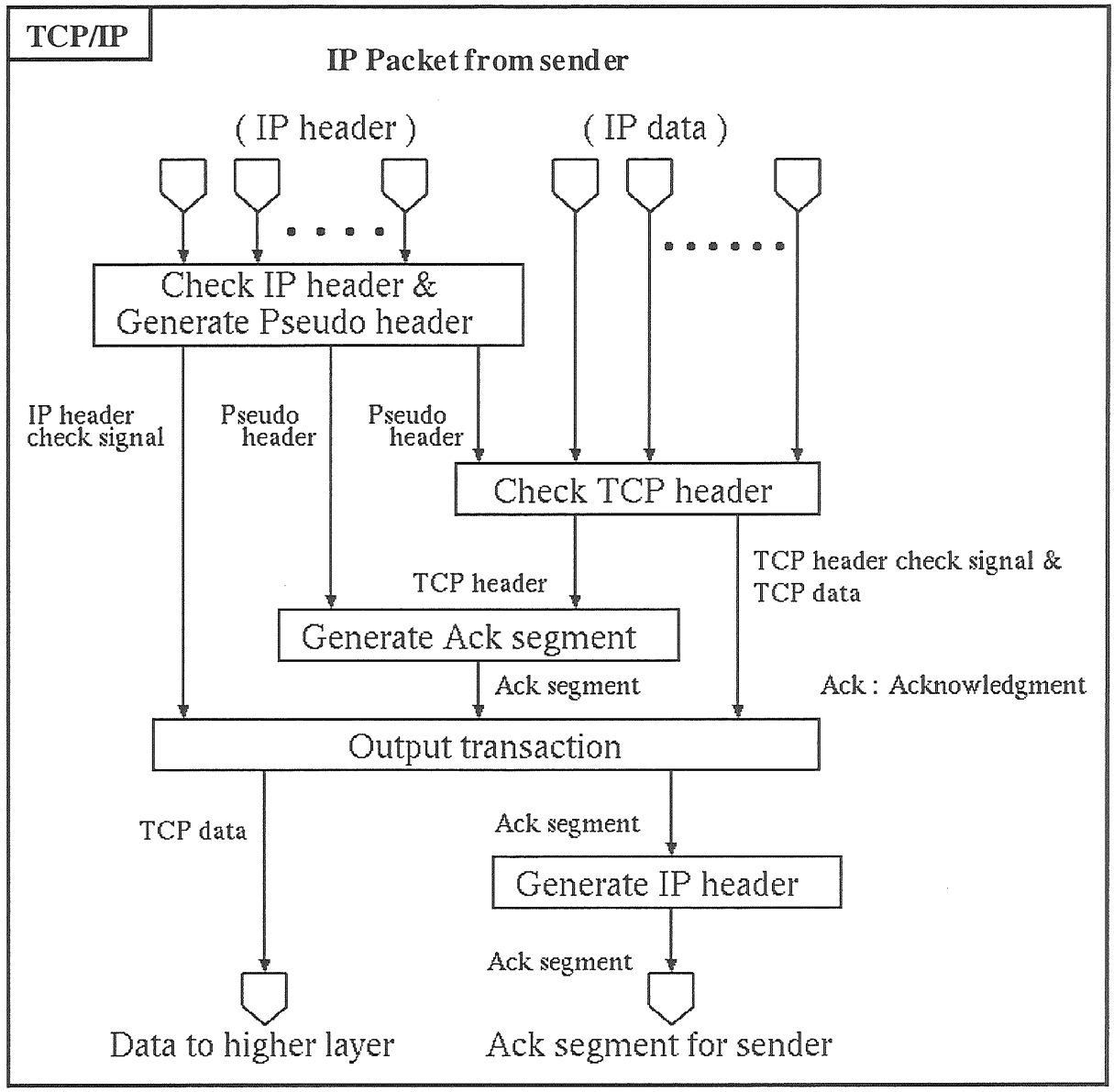


図4.2.4　データ駆動型TCP/IPデータフローグラフ

signalの結果を参照し，正常な TCP データは上位レイヤに送られ，Ackセグメントが"Generate IP header"部に送られる．header check結果が異常であったデータはここで廃棄される．

(6)"Generate IP header" 部はTCPレイヤの確認のためのTCPヘッダを含むIPパケットを生成し，IPメッセージを下位レイヤに送る．

このように，各IPメッセージ対応に処理を行うプロセスが複数同時に走っている．このCUE-p実装において，プロセスあたりの処理時間を測定すると図4.2.5のようになった． 図4.2.2と同様に，縦軸はプロセス当たりの処理時間を示し，横軸はノイマンプロセッサにおけるスレッドに対応する入力ストリーム速度を示す．エミュレーションにおいて予想されたように，図4.2.5は，図4.2.2の並列型と同じ傾向を示している．明らかに処理オーバヘッドは存在せず処理時間は一定となった．図4.2.2と図4.2.5の処理時間の差異は実装されたステージ数によるものである．前者のエミュレーションはIP

ヘッダ生成部のみをエミュレートしたが，後者はIPとTCPの全体の処理ステージを実現している．この結果は，データ駆動型TCP/IPプロトコル処理の効率性とすでに述べた要求条件（2.3節のR1）を満足する高い可能性を有していることを証明している．

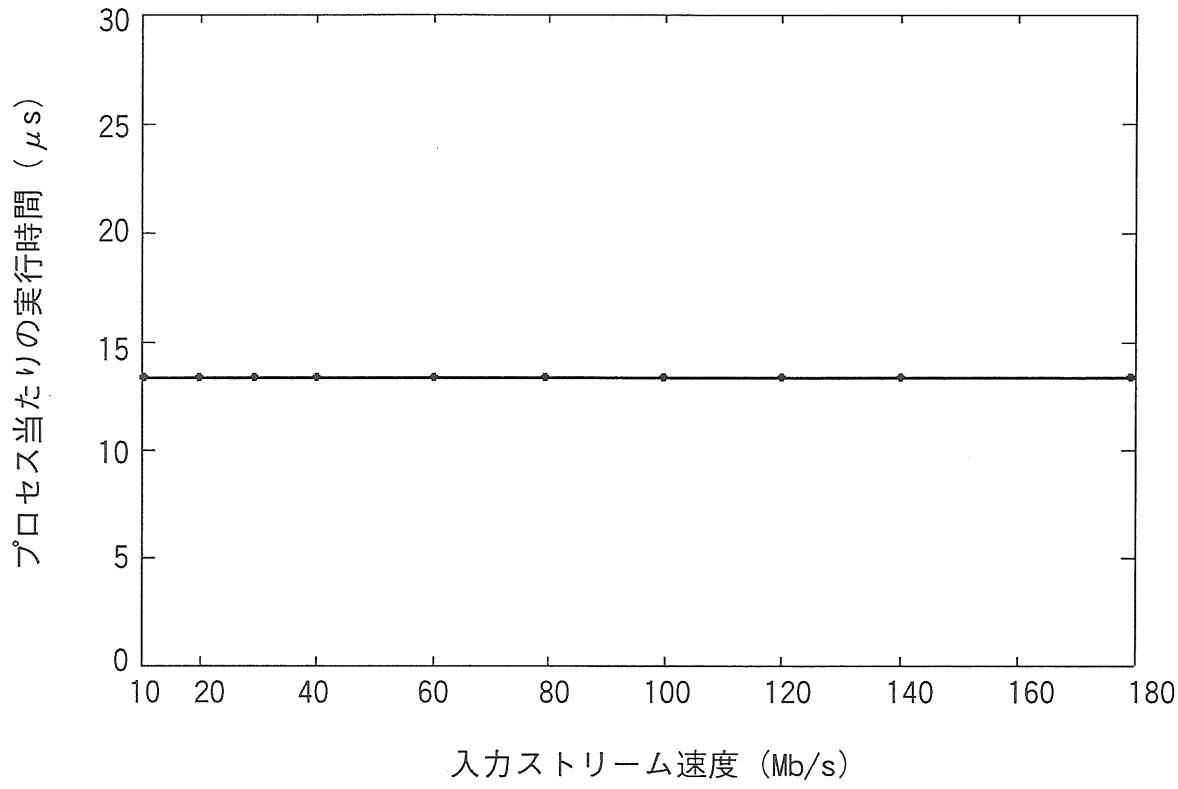


図4.2.5　TCP/IPプロトコル処理のデータ駆動実現における実行時間

4.3　通信情報ネットワーキング環境アクセス系の要求条件

通信情報ネットワーキング環境におけるユーザとネットワークの協調を実現するためには，軽量型ユーザ装置すなわちPCなどを念頭にネットワーク内高度機能との接続を考慮しなければならない．昨今，webとJAVAを組み合わせ，CORBAのクライアント機能が使える環境（CORBAスタブ）をPCにダウンロードする技術が製品化されつつある．現在の標準CORBA間通信プロトコルIIOP（Internet Inter-ORB Protocol）は下位転送機構としてインターネットすなわちTCP/IPを前提としている．これにより，なんらCORBA機能を持たないPCでも，インタネットアクセス機能を有し，JAVAエンジンを搭載したブラウザを具備し，CORBAスタブをダウンロードすればTINA/CORBA端末をベースにした通信情報ネットワーキング環境にアクセスすることができる．図4.3.1に，web/JAVA/CORBA[52]を用いてNTTがTINA-Cの検討の一環としてTTT(The TINA Trial)[53]で実現したユーザネットワーク協調型サービス実現システムの概要を示す[54]．

TCP/IP以下には，ISDNやATMなどが用いられる．このとき，ユーザが通信情報ネットワーキング環境にアクセスする上での要求条件は以下のように考えられる．

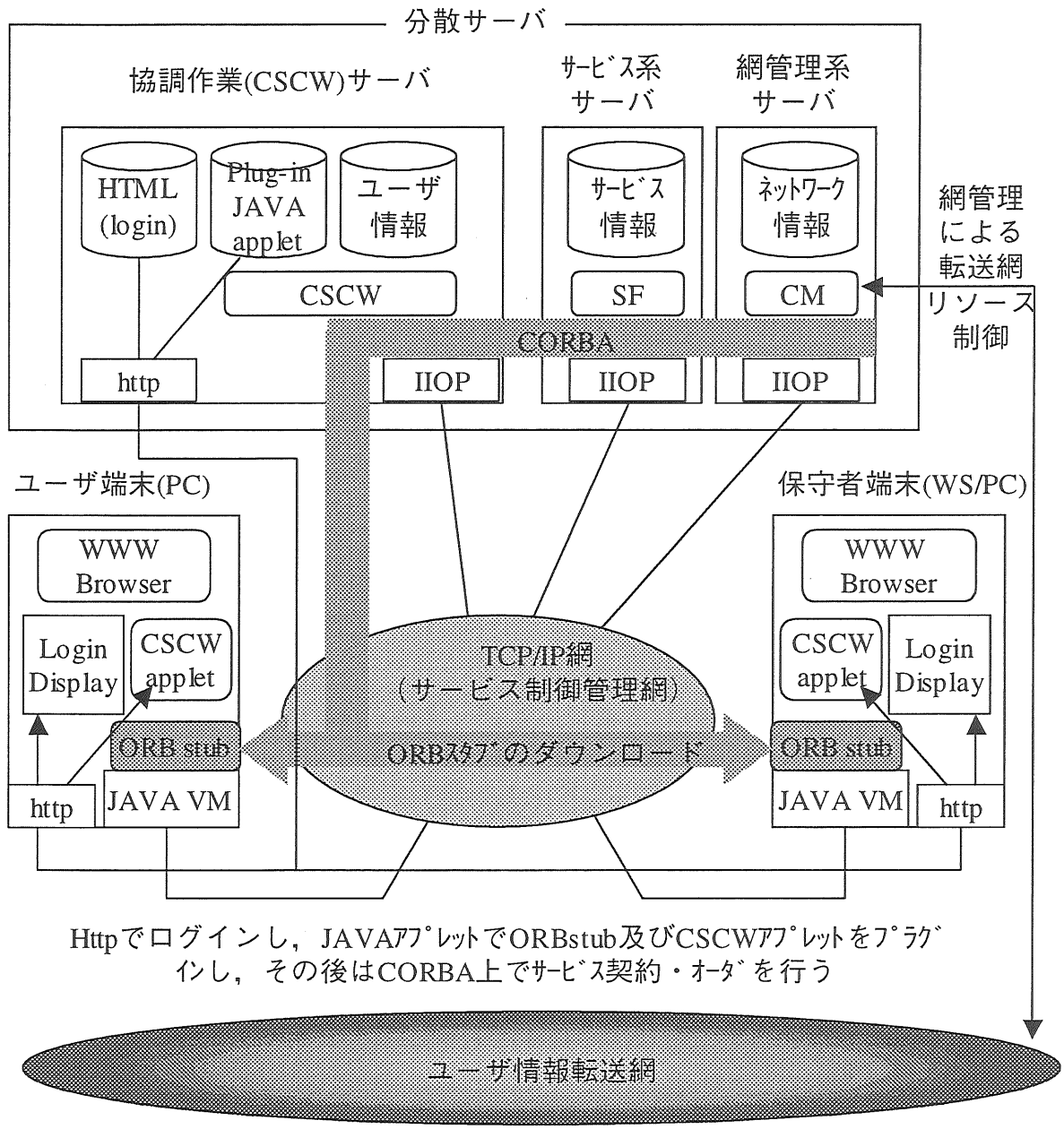


図4.3.1　ユーザ・ネットワーク協調型サービス実現システム

(1)アクセス回線の効率的利用

一般に大きな容量を確保できないアクセスにおいては，できるだけ無駄の少ない効率的な回線の利用が望まれる．通信情報ネットワーキング環境では複数の網内サーバ上の分散オブジェクトとの同時通信が考えられアクセスの効率化のために，複数コネクションの同時利用が性能に影響を与えないプロトコル処理が望まれる．

(2)ストリームとメッセージの共用

制御や管理メッセージとそれによって制御管理されるストリーム情報が論理的に分離されていても，物理的には同一のアクセスを共用することも十分にあり得る．特に，ATMを利用した数Mから数十Mbit/s程度のマルチメディア情報と数キロ程度の制御情報が混在し同時使用される場合の効率化も求められる．

(3)複数端末構成

単一端末が複数サービスを受けるのみならず,同一宅内に設置された複数のPCにより単一のアクセスを共用できることも要求される.

(4)簡易なインタフェース

簡易なPCに高度な機能を付加することは望ましくなく,PCの持つ既存インタフェースを活用することが要求される.

以上の要求条件のうち,これからのユーザ装置(PCなど)への分散技術の展開上,特に考慮すべきアクセスの効率化の要求条件については,4.2節のデータ駆動TCP/IP処理により充足できる可能性が高い.

4.4 通信情報ネットワーキング環境アクセス系の実現

既に行った検討により,データ駆動TCP/IPプログラム処理を最大限利用することにより,4.3節に述べた4つの通信情報ネットワーキング環境へのアクセスのための要求条件((1)アクセス回線の効率的利用,(2) ストリームとメッセージの共用,(3)複数端末構成,(4)簡易なインタフェース)を満足する望ましいアクセス系構成が実現できることが言える.

要求条件の(1)は,4.2で検討したように,データ駆動TCP/IPプロトコル処理の高効率性により満足できる.

(2)から(4)の要求条件を満足するため,複数のPCをIEEE1394[55]を介して収容するTCP/IPハブの構築をめざした.IEEE1394インタフェースは通常のPCによって作られたホームバス環境に導入されつつある高速インタフェースである(要求条件(4)).

この構成では,複数PCがプラグアンドプレイで簡易に実現できる(要求条件(3)).

IEEE1394は,内部バス速度(数百Mbit/s)と同等な速度を実現でき,同期・非同期いずれの転送モードも実現できる.この特徴はストリームと管理メッセージの双方を提供する通信情報ネットワーキング環境に適合している(要求条件(2)).

図4.4.1に系構成の概要を示す.現在のところ, TCP/IP over IEEE1394に関する標準が存在しないため,以下を仮定した.

(a)データ駆動TCP/IPハブには1つのIPアドレスが割り振られ,その上に複数のTCPコネクションが多重される.

(b)PCからのあるいはPC宛の情報はハブで組み立て・分解されたTCPパケットで転送される.

(c)ハブの中では,ORBスタブがインストールされている(たとえば,JAVAアプレットのダウンロードなどで実現).各PCは,IIOP (Internet Inter-ORB Protocol)を介して高速のORB通信を実現でき,それらは単一のアクセス回線にTCP/IPハブによって多重され,結果として効率的なアクセス回線利用が可能となる.

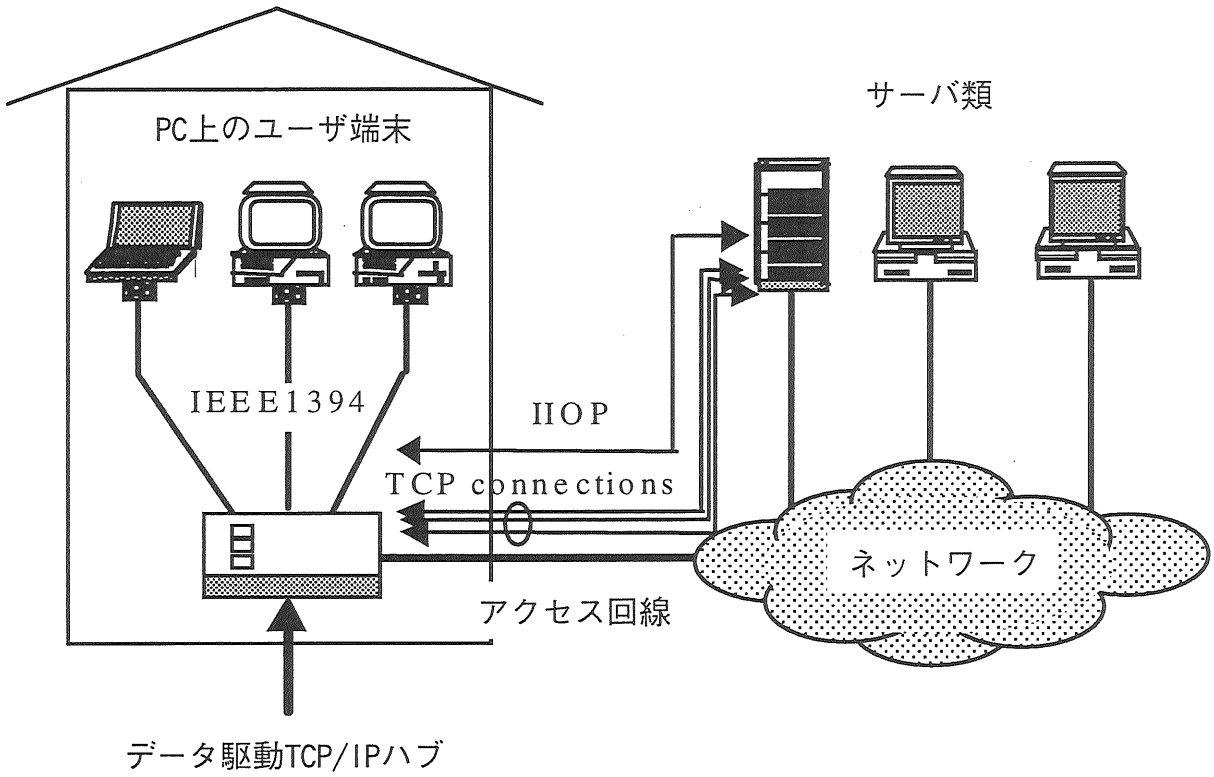


図4.4.1　通信情報ネットワーキング環境アクセス系構成例

4.5　結言

本章では，通信情報ネットワーキング環境実現のための要求条件の１つである効率的多重処理に着目し，プロトコル処理へのデータ駆動プロセッサの適用を検討した．

通信情報ネットワーキング環境やインターネットにおいて最もよく用いられるプロトコルであるTCP/IPを取り上げ，３章で明らかにした多重処理の高い効率性を有するデータ駆動プロセッサにより，TCP/IP処理の実現を検討した．

まず，IP処理部のみをとりあげ，ノイマン型処理と並列型処理をデータ駆動プロセッサでエミュレーションし，プロセス多重度と処理速度の関係を比較した．複数の同時処理がお互いに影響しない後者の並列型が高い効率性を示し，データ駆動プロセッサの並列性のTCP/IP処理への高い適用性を明確にした．

これをふまえ，TCP/IP処理のデータ駆動実装を行い，性能の検証を行った．その結果，データ駆動プロセッサを用いたプロトコル処理の高い効率性と通信情報ネットワーキング環境への適用性を実証した．

さらに，ユーザとネットワークの協調を実現することが特徴である通信情報ネットワーキング環境では，軽量で機能がそれほど高くないユーザ装置が通信情報ネットワーキング環境の一環に組み込まれ，サービスを実現するための，ユーザのネットワークへのアクセスに対する要求条件として，

(1)アクセス回線の効率的使用，(2)ストリームとメッセージでのアクセスの共用，(3)複数端末の収容，(4) 簡易なインタフェースを上げた．

最後にアクセスの要求条件にもとづいた通信情報ネットワーキング環境のアクセス系の実現法を検討した.

今後の課題として以下が考えられる.

(1)実装したデータ駆動型TCP/IPプロトコル処理を，上位インタフェース（ソケット）を明確にした上で，実際のノイマン型装置の中に組み込み，総体としての性能を明確にすること

(2)効果が明確になったTCP/IP実装をふまえて，そのほかのプロトコルへの適用を検討すること

(3)アクセス系の実験を実装し，性能を評価すること.

第5章　通信情報ネットワーキング環境の高信頼化

5.1　緒言

通信情報ネットワーキング環境においてはすでに述べたように，複数のノードに分散配置されたソフトウェア部品（コンポーネント）を協調動作させてアプリケーションを実現している．クライアント・サーバ関係で処理が実行されていくが，クライアントであるコンポーネントが，サーバコンポーネントやサーバコンポーネントを搭載するノードの状況（障害か正常かなど）を把握せずにコンポーネント間通信を試み，結果として障害や異常状況に遭遇して望みの通信を行えない場合がある．

ソフトウェア工学的に，ソフトウェアの設計，構築時における信頼性向上策を行うのとは異なり，一般の分散オブジェクト指向型ソフトウェアで構成されたシステムにおいて各オブジェクト・コンポーネントに対して後付けで障害検出機構，性能監視機構を設け，オブジェクトというブラックボックス単位での故障発見，置換を行い，しかもネットワークワイドに分散したコンポーネント間の関係をもとに影響を受けるアプリケーションを特定すると言うソフトウェア障害管理の側面からの検討はほとんどなされていない．

現在の分散処理環境におけるソフトウェアコンポーネント管理には，たとえばCORBAにおける共通サービス機能としてオブジェクトライフサイクルなどがあるが[56]，これはオブジェクトを生成し削除する機能であり，障害管理に直接の適用はされていない．市場ではCPU処理の状況を監視して，間接的に局所的な問題を発見する機構しかなかったり，分散を前提としないアプリケーション（帳票関係など）の状況監視機構などが一部商品化されているにすぎず[34][35]，ネットワークワイドなアプリケーションへの影響を検出対処できるものは存在しない．従って，大規模な通信情報ネットワーキング環境の高信頼化のためのソフトウェアコンポーネントの障害管理機能を，ORBなどの分散処理環境のベンダー製品非依存の形で実現することが望まれる．

また，通信情報ネットワーキング環境のソフトウェアコンポーネントが実際に搭載されるサーバノードやサーバ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[8][9][57]．これらは，キャリアの交換ノードなどを除き，サーバノードそのものをブラックボックス化して管理するのが通常であり，ディスクアレイ[40]やクラスタリング技術[12]も，プロセッサレベルの粒度の小さいモジュール，エレメント単位での障害対策ではない．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．

さらに，ソフトウェアとハードウェアの障害管理は，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．

本章では，このような通信情報ネットワーキング環境の高信頼化を目的に（2章要求

条件R2),同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，障害管理方式を提案する．

まずソフトウェア面では，個々のアプリケーションオブジェクトを監視する機構，オブジェクト間の関係を管理する機構をもとにコンポーネントをベースとしたネットワークワイドなアプリケーション障害管理機構により上記の問題の解決を図っているSOMSE (Service Operation and Management architecture using Surveillance of application software Elements)アーキテクチャを提案する[36]-[39]．さら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[38][43][44]．

ハードウェア面では，すでに通信情報ネットワーキング環境への適用性の高さを示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提案する[22][41][42][43][44]．

次に，これらソフトウェアおよびハードウェアそれぞれの障害管理機能をもとに，通信情報ネットワーキング環境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する[22][43][44]．

5.2 通信情報ネットワーキング環境における障害管理への要求条件

通信情報ネットワーキング環境におけるアプリケーションを形成するソフトウェアコンポーネントは，図5.2.1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コンポーネントが連動してアプリケーションを実現する．これらのコンポーネントは物理的には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．このような分散型処理を前提とした通信情報ネットワーキング環境における障害（バグによるもの，組み合わせの問題，処理能力不足によるもの等正常でない状況を指す）に対する対処の要求条件としては以下のものが考えられる[41][43]．対象としているオブジェクトはCORBAオブジェクト及びEJBコンポーネントなど，ソフトウェアコンポーネント構成時の信頼性を確保する手法はとられていない通常の分散コンポーネントである．

5.2.1 ソフトウェアの障害

通信情報ネットワーキング環境では，アプリケーションは図5.2.1に示すように，分散配備された複数のコンポーネントが連動して実現される．またコンポーネントが持つ個々の機能は，複数のアプリケーションによって多重に利用される．

このとき，アプリケーションの実行中にコンポーネントの障害が発生したときの対処への要求条件は以下のように考えられる．

(1)障害検出：コンポーネントの障害を迅速かつ確実に検出する．

(2)障害波及範囲の把握：コンポーネント障害により，その影響が波及する可能性のあるすべてのコンポーネントあるいはアプリケーションを迅速に把握する．コンポーネントはネットワークワイドに分散しており，それを利用するアプリケーションも複数の可能性がある．よって，あるコンポーネントの障害によりほかのどのコンポーネントが影響を受け，結果としてどのアプリケーションに影響を与えるかを迅速に把握で

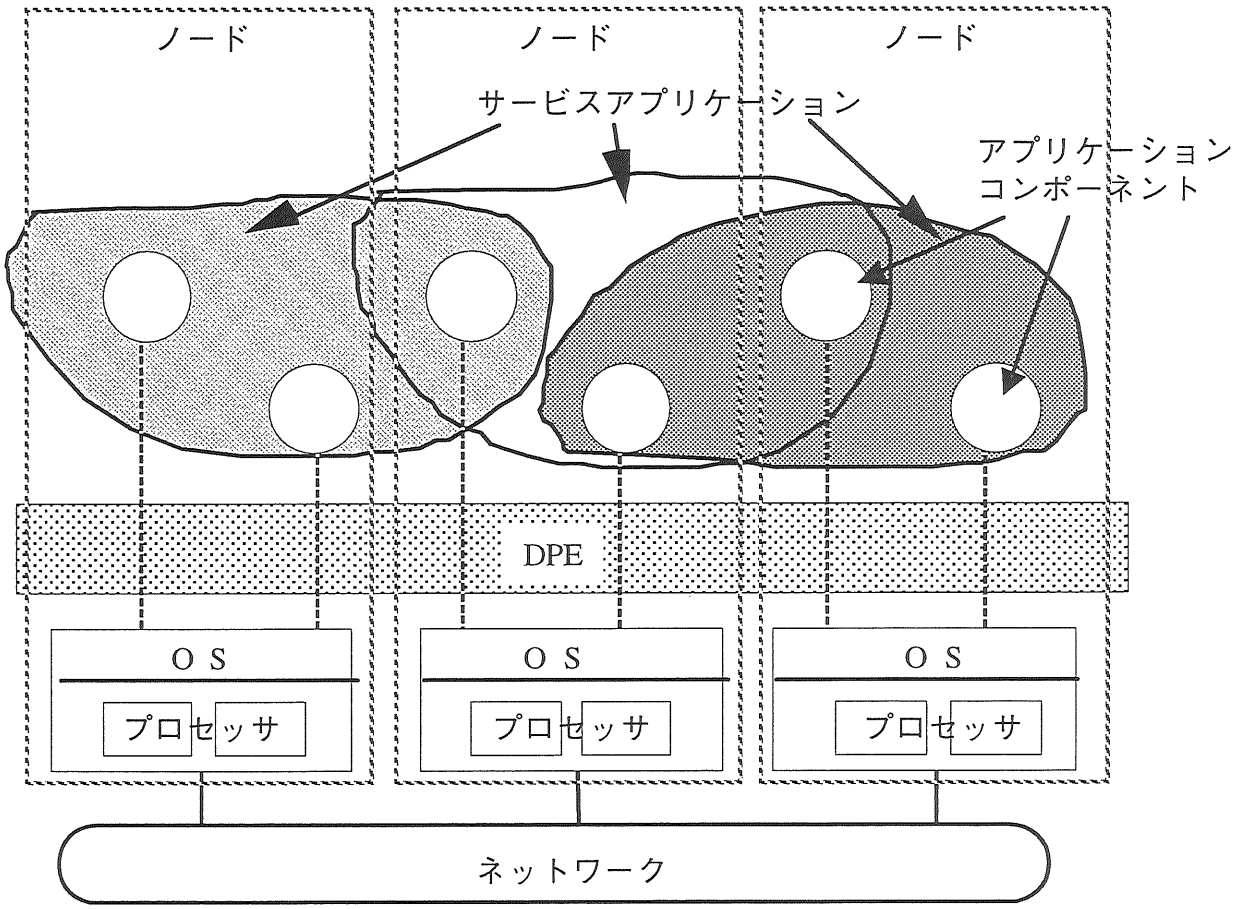


図5.2.1　分散処理を前提とした通信情報ネットワーキング環境

きねばならない.

(3)障害影響の除去：障害箇所以外に影響を与えることなく，迅速に障害に対処し，影響の及ぶ可能性のあるコンポーネントやアプリケーションへの影響を除去する．言い換えれば，できるだけ早くそのコンポーネントに代替できるコンポーネントを見いだし，それを関係するコンポーネントに通知する仕組みである.

(1)を実現するには,個々のコンポーネントが実行する処理をきめ細かに監視できる仕組みが必要となる（要求条件１）．また個々のコンポーネントは自らが保持する機能の実行時に必要に応じて他のコンポーネントに処理を依頼するため,対処(2),(3)を実現するには，動的に変化するコンポーネント間の関係（処理依頼関係及び代用関係）を正確に把握し，その情報にもとづき的確に障害の対処を実現する仕組みが必要となる（要求条件２）.

5.2.2　ハードウェアの障害

分散処理環境およびその上のソフトウェアコンポーネン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サーバノードのハードウェアの障害対策は，市販ワークステーションにおいては十分になされていない．高価なサーバシステムにおいては，RAID(Redundant Arrays of Independent Disks)によるディスクの二重化[40]，クラスタリングによるシステムＮ重化[12]，プロセッサ障害に対する対策としては，プ

ロセッサの二重化などが行われている．ここでは，ハードウェア障害対策として，市販品で対処されているサーバノード全体としての障害対策ではなく，さらに粒度の小さいプロセッサ障害対策に着目する．単なるプロセッサ二重化より効果的な障害対策のために，プロセッサ自身に障害対策機構を埋め込み，しかもそれが通常処理を圧迫しない方式を検討する必要がある．プロセッサハードウェア障害対策に対する要求条件として以下のものが考えられる．

(1)障害監視・検出：プロセッサの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)障害箇所特定，障害箇所の切り離し：プロセッサ障害対策として２重化などの冗長構成を基本とする．(1)の検出後，どのプロセッサが障害であるかを特定し，障害プロセッサが系に悪影響を及ぼす前に，障害プロセッサを系から切り離すことが必要である．

(3)局所障害管理機構：系全体の知識をできるだけ持たずに，局所的に対処できる障害管理が必要である．

(4)効率的管理処理：障害管理のための処理が通常処理を圧迫しないことが必要である．

(5)過負荷耐力：障害時の過負荷に耐えうるハードウェア構成が必要である．

5.3 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(SOMSE)

5.3.1 SOMSE アーキテクチャ[36]-[39]

5.3.1.1 提案方式

5.2.1をふまえて，図5.3.1にソフトウェアコンポーネント障害監視・復旧の機能アーキテクチャSOMSE (Service Operation and Management architecture using Surveillance of application software Elements)を示す．

図中の矢印は，機能間の関係を示す．

このアーキテクチャは，以下の２つの特徴により，5.2.1であげた障害監視・復旧への２つの要求条件を満たすことをねらいとしている．

(1)信頼性の高いコンポーネント監視を実現するため，コンポーネントが実行する処理の正常動作を監視するACM(Application Component Manager)を個々のコンポーネントに対応させてもうける．

(2)コンポーネント障害に的確に対処するため，コンポーネント間の相互関係の情報を保持するACD(Application Component Database)と，その情報を利用してグローバルなレベルでコンポーネント障害に対処し，コンポーネントの復旧を行うGCM(Global application Component Manager)と，コンポーネントの動的生成・消去を実行するACIR(Application Component Installer and Remover)を設ける．

5.3.1.2 SOMSE コンポーネント

ここではSOMSEを構成する部品について述べる．

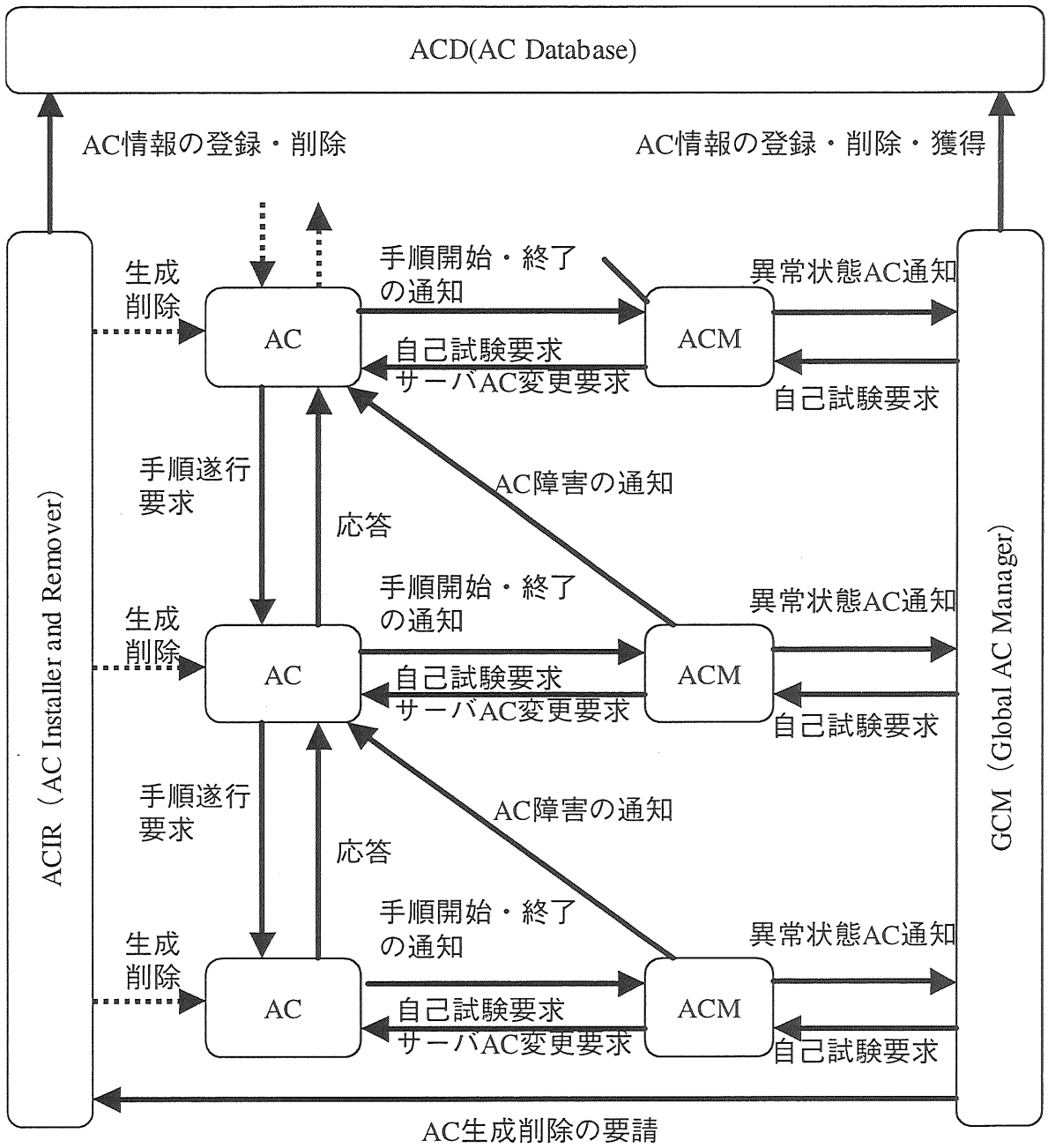


図5.3.1　SOMSEアーキテクチャ

Ⅰ．AC

アプリケーションの機能要素（コンポーネント）．CORBAオブジェクトやEJBのBeansに対応する．個々のACはクライアントACから処理の依頼を受けると，処理開始をACMに通知し，サーバACからの処理依頼応答を受けると処理終了を同様にACMに通知する．個々のACは自らの処理の依頼先（サーバAC）をACごとにアクセス情報としてその名前あるいはアドレスを保持する．処理性能を考慮して，本アクセス情報は処理依頼先のACに異常が発生しない限り有効とする．

Ⅱ．ACM

ACの正常動作を監視するAC監視機能．以下の５つの機能を持つ．

(1)ACが実行する処理の正常動作の監視

監視下のACがそのクライアントACから依頼された処理のうち，所定の処理期限時間内に実行を完了できなかった処理を異常処理として検出する．またACとクライアントACの障害を切り分け，その結果をGCMに通知する．

(2)ACが実行した処理のログ情報の保持

監視下のACが実行し終えた処理に関するログ情報（AC名，クライアントAC名，処理終了時刻，終了状態）をある規定された時間だけ保持する．

(3)ACに対する自己診断実行の要求

異常検出の確実性向上のため，処理実行時以外にも，ACの負荷を考慮しながら，適宜ACの動作正常性を確認する．

(4)処理依頼先切り替え要求

監視下のACが障害を起こしたサーバACに処理を依頼しないように，そのACが処理を依頼する先のサーバACを変更するように要求する．

(5)処理異常終了通知

監視下のACの障害発生時に，そのACに代わり，クライアントACに対してACが実行していた処理の異常終了を通知する．

ACを監視するための情報は，実行処理管理テーブルと，依頼処理管理テーブルで保持される（図5.3.2参照）．おのおののテーブル上の情報は，ACが処理の依頼を受けてから実行終了（あるいは障害により終了）するまで一時的に作成される．

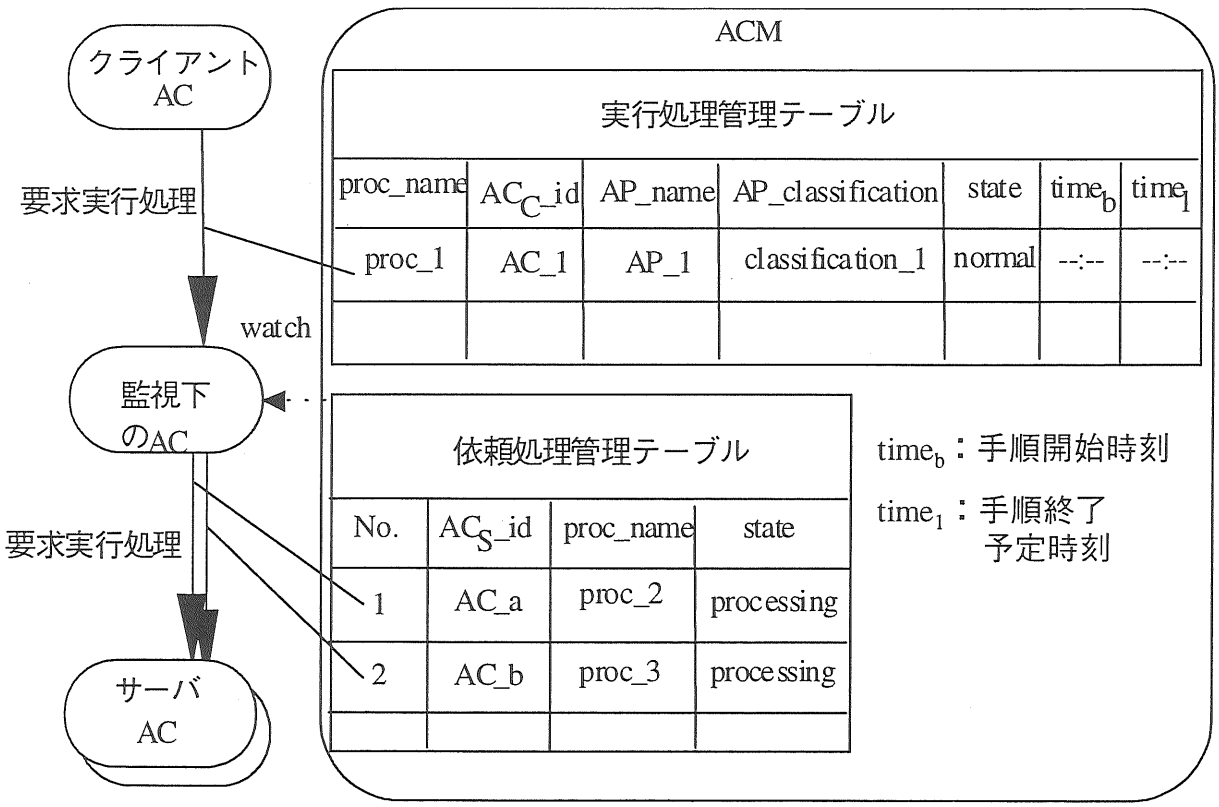


図5.3.2　各テーブルが保有する情報

各テーブルの役割及び保持する情報を以下に示す.

(1)実行処理管理テーブル

本テーブルはクライアントACから監視下のACに対して依頼された処理ごとにその処理に関する以下の情報を保持する.

①クライアントAC名

ACの障害時に,ACの代わりにACMがその障害の旨をクライアントACに通知するために用いられる.

②アプリケーション名およびアプリケーション種別

ACに依頼された処理に関わるアプリケーションの名前と,その処理に課せられる制約を表すアプリケーションの種別.アプリケーション名はAC障害を引き起こしたアプリケーション名,あるいはACの異常の影響を受けうる実行中のアプリケーション名をGCMに通知するために参照される.アプリケーション種別は,処理期限時刻と処理の優先度を決めるために参照される.

③処理の実行状態

ACが実行している処理が,現時点で設定時間内(normal)/設定時間経過(over)のいずれの状態にあるかを表す.ACMが自己診断を行うとき,処理の障害発生率((状態overの処理の数/(実行している全処理数))を算出するために用いられる.

④処理開始時刻

ACMがACから処理開始通知を受けた時刻.ある時点で複数の処理で障害が発生していた場合,障害が発生した原因となった処理を(ACMあるいはGCMが)特定するための参考情報として用いる.

⑤処理終了予定時刻

ACMがACから処理終了通知を受けるべき期限時刻.ACMが,ACあるいはサーバACの障害を検出するために用いる.

(2)依頼処理管理テーブル

本テーブルは,クライアントACからACに依頼された処理を実行するために,ACがおのおののサーバACに依頼した処理に関する以下の情報を保持する.

①依頼処理番号

ACが各サーバACに依頼した処理の順番を表した番号.ループ処理の障害などにより,正常に処理が終了した場合にはあり得ない規定値以上の数の依頼処理が生じたことを検出するために用いる.

②依頼処理名

ACが処理を依頼したサーバAC名およびその依頼処理名.ACのACMが障害の疑いのあるサーバAC名をGCMに申告するために用いる.

③終了チェック

ACがサーバACに依頼した処理が終了しているか,実行中であるかを表す(終了した処理の情報は,一定時間経過後に消去される).ACMが,監視下のACが実行する処理に障害が発生したと判定したときに,それがACの障害によるものかサーバACの障害によるも

のかを切り分けるために参照する.

Ⅲ.GCM

ACMからAC障害の通知を受け，障害箇所の特定及びアプリケーション障害復旧のためのAC切り替えを行う．また，通知内容や通知受信時にACDが保持する他の（障害が起きたACの代用となりうる）ACの運用状況をもとに，AC間の処理依頼関係の切り替え（代用ACへの処理依頼先変更）やACIRへの新規AC生成依頼などの処理を実行する．また，ACMが有する障害監視機構の信頼性を確保するため，定期的におのおののACMに対して自己の診断を要求し，障害が発見された場合に，その切り替え処理を実行する．

Ⅳ.ACD

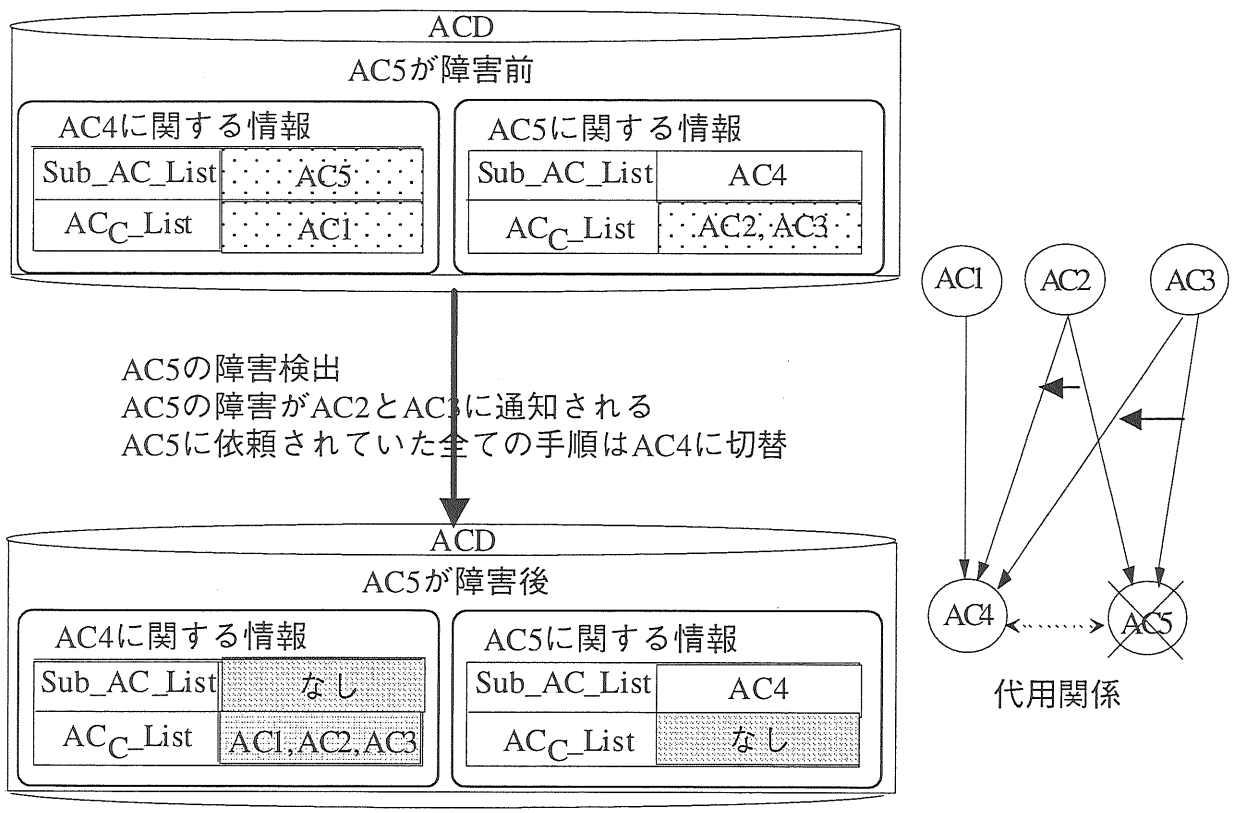
ACを運用するために必要である以下の情報を保持する．

(1)AC名

(2)ACが実行する処理名

(3)代用AC名リスト

注目しているACの代用となりうる（運用中の）AC名のリスト．ACに障害が発生したときに，それ以後代わりに用いるAC（代用AC）を決定するときに参照される（図5.3.3参照）



Sub_AC_List： 代用ACリスト

AC_C_List：クライアントACリスト

図5.3.3　ACDの管理する情報

(4)クライアントACリスト

注目しているACを用いるクライアントAC名のリスト．ACに障害が発生したときに，その障害ACに処理を依頼する可能性のある全てのクライアントACに対して，以後その障害を起こしたACを用いないことの指示を行い，新規の処理依頼先のAC名あるいはアドレスを通知する時に参照される

(5)使用可・不可

注目しているACに処理の依頼ができるかできないかを表す．AC障害の発生時は，そのACに関するこの項目を「不可」とする．これにより，それ以降そのACに処理要求が受け渡されることがなくなり，その障害となったACに起因するサービス障害を防止する．

(6)負荷レベル

ある時点で，注目しているACが他のACから依頼を受けて実行している処理の数．一定時間ごとに最新の値が反映される.AC間の処理の負荷のバランスを取るために用いる．

Ⅴ．ACIR

GCMからの要求により，ACの生成および消去を実行する．ACIRがACを配備（ノード内のプロセスとして生成）した場合，ACIRはそのAC名，代用AC名リスト，クライアントACリストをACDに登録する．

5.3.2 AC障害監視／アプリケーション障害復旧手順

5.3.2.1 AC障害検出手順

ACがクライアントACから処理γの依頼を受けると，ACはACMに対して，処理γの依頼を受け付けた旨の通知を発行する．ACMは，通知の引数としてクライアントAC名，アプリケーション名，アプリケーション種別を受けると，それらの情報と処理時間開始時刻，終了時刻，実行処理状態(normal)を実行処理管理テーブルに記録する．

処理γの実行中にACがサーバACに処理δを依頼するとACはACMに対して，処理δの依頼した旨の通知を行う．ACMは，通知の引数として依頼処理データ番号，終了チェック(processing) を依頼処理管理テーブルに記録する．

ACはクライアントACから依頼された処理γを実行するために必要な全ての処理(ACが自ら行う処理，および各サーバACに依頼した処理) が終了したとき，その旨をACMに通知する．ACMは，処理期限時間内にその通知を受けたときは，処理γに関連する実行処理管理テーブルおよび依頼処理管理テーブル上の情報を消去し，ログに移し替える．

ACMは一定時間ごとに，監視下のACが実行している全ての処理に対して，処理期限時刻を経過していないかどうかをチェックする．処理期限時刻を経過した処理を検出したとき，その処理に関連するAC，サーバAC，AC間通信機構のいずれかの動作の正常性が損なわれたものとみなす．

このとき，ACMはまず，そのACに自己診断の実行を要求する．この結果と実行処理管理テーブル上の処理γのアプリケーション種別をもとに，GCMに以下のa)，b)いずれかの内容の通知を行うか，あるいは処理期限時間を延ばす処置をとる．なお，ACの自己診断では，ACが保持する変数データ値のチェックを行う．

a)ACMの監視下のACの障害

b)監視下のACのサーバAC，またはAC－サーバAC間での通信障害の疑い

ACの自己診断の結果として，ACからAC障害の通知がACMに送られるか，あるいは応答期限時間のうちに応答が返らなければ，そのACMは，監視下のACの障害をGCMに通知し，現在ACが実行している同一内容の処理（実行処理管理テーブルから検索）を全て強制終了させ，おのおのの処理のクライアントACに対してACでの処理が異常終了したことを通知する．さらに，処理γのクライアントACに対し，以後代用のACへの依頼先切り替え処理（後述）が行われるまで，ACへの処理依頼を止めるように通知する．

また，ACの自己診断の結果として，ACから正常の通知がACMに送られれば，サーバACあるいはAC間通信機構に障害が発生している疑いがあるものとして，その旨をGCMに通知する．

以上の手続きにより，おのおののACMは監視下のACの障害を検出する．

5.3.2.2　アプリケーション障害復旧のためのAC切り替え手順

ACMが監視下のACの障害を検出したとき，およびACMが監視下のACのサーバACの障害の可能性を検出したときそれぞれの場合でのアプリケーション障害復旧のためのAC切り替え手順の例を示す．なお，ACMが監視下のACのサーバAC障害の可能性を検出することは，サーバACとそのサーバACを監視するACMが同時に障害を起こしたことを迅速に検出するために役立つ．

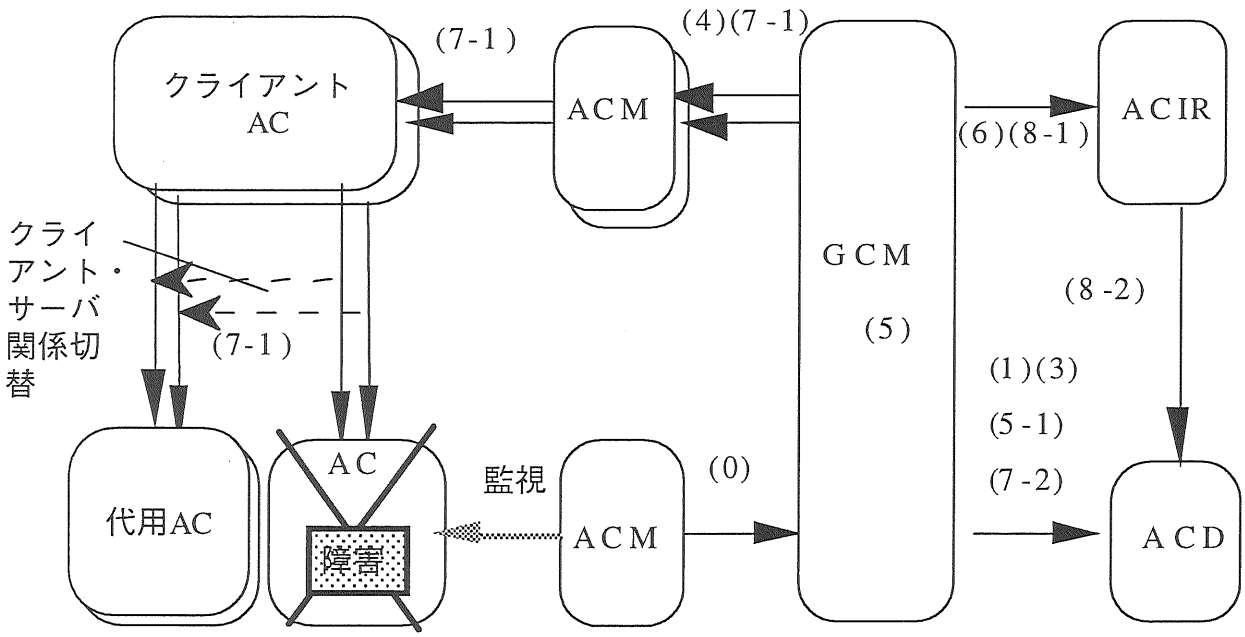


図5.3.4　AC障害復旧手順（ACの障害）

＜AC切り替え手順Ⅰ：ACMが監視下のACの障害を検出したとき（図5.3.4参照）＞

GCMは，ACMからその監視下のAC障害の通知を受けると（図5.3.4(0)），以下の手順に

よりACの切り替え手続きを実行する．

(1)ACDが保持する情報のうち，障害と申告されたAC（障害AC）に関する使用可／不可の項目を「不可」に変更するように要求する（図5.3.4(1)）．これにより，障害ACが他のACの代用として割り当てられることがなくなる．

(2)通知された障害の内容（エラー内容，サービス種別）により，ACの切り替え，サービス管理者への通知などを行う（図5.3.4(2)）．

(3)ACDに対し，障害ACに処理を依頼しうる全てのクライアントAC名のリストを要求する（図5.3.4(3)）

(4)(3)のクライアントリストの全てのクライアントACに対し，GCMから代用AC名あるいはそのアドレスを知らされるまで障害ACに対する処理依頼を全て一時停止するように要求する（図5.3.4(4)）．

(5)ACDに対し，障害ACの代用として用いることのできるAC名のリストを要求し（図5. 3. 4(5-1)），得られたリストから，代用として用いるAC（代用AC）を決定する（図5.3.4(5-2)）．代用ACを一つとするとAC間の負荷のバランスが偏る場合は，代用ACを複数設定する．

(6)適用な代用ACが存在しない場合，あるいはACの切り替えにより代用ACの負荷（代用ACに処理の実行を要求するクライアントAC数，及びAC－クライアントAC間メッセージ交信頻度）が大きくなる場合には，新規に代用ACを生成するようにACIRに要求する（図5.3.4(6)）

(7)(3)のクライアントリストのすべてのACに対して(6)までのステップで決定した代用ACに処理を依頼するように要求する（図5.3.4(7-1)）．このとき，クライアントリストの各クライアントACに対し，おのおののクライアントACが保持している処理依頼先名を障害と申告されたACから代用ACに変更するよう要求する．これ以後，リスト中のクライアントACは，全て障害ACの代わりに代用ACに処理を依頼するようになる．

処理依頼先変更の要求先のクライアントACから了承の返答を受けると，GCMはさらに，代用ACが障害となった際にAC切り替えが実行できるように，ACD内の代用ACのクライアントリストの項目に，新たに代用ACのクライアントとなったAC名を（追加）登録するようにACDに対して要求する（図5.3.4(7-2)）．

(8)障害ACの消去の実行をACIRに要求する（図5.3.4(8-1)）．ACIRは，そのACを消去した後にACD内のそのACに関する情報を消去するように要求する（図5.3.4(8-2)）．

＜AC切り替え手順Ⅱ：ACのACMがサーバACの障害の可能性を検出したとき（図5.3.5参照）＞

GCMは，ACMからその監視下のACのサーバACに障害が発生した可能性がある旨の通知を受けると（図5.3.5(0)），以下の手続きで，障害箇所の特定およびその障害の対処を行う．

(1)ACDが保持する情報のうち，障害と申告されたサーバAC（障害サーバAC）に関する使用可／不可の項目を「不可」に変更するように要求する（図5.3.5(1)）．

(2)ACDに対し，障害サーバACを用いる全てのAC名のリストを要求する（図5.3.5(2)）．

(3)ACリストの全てのACに対して，GCMから代用サーバAC名あるいはそのアドレスを知らされるまで，障害サーバACに対する処理依頼を全て一時停止するように要求する(図5.3.5(3))．
(4)この時点では，通知された障害が，障害サーバACの障害によるものか，あるいは処理の要求時／応答時の通信機能の障害によるものかが判別できない．そこでGCMは，障害サーバACを監視するACMに対して，当該サーバACの試験を要求する（図5.3.5(4)）．その応答内容により以下のような手続きが実行される．
(4-1)応答の内容が「正常」のとき
障害サーバACのACMに，ACが障害サーバACに依頼した処理を実行中であるかを問い合わせる（図5.3.5(4-1-1)）
a)応答が「実行中」のとき
GCMは，障害サーバACがさらに他のAC(ACss)に処理を依頼しているかを障害サーバACのACMに問い合わせる（図5.3.5(4-1-2)）．その応答内容対応に，以下の手順を実行する．
－障害サーバACがさらに他のACに処理を依頼しているとき
ACssのACMに，ACssの自己診断を行わせるように依頼する（図5.3.5(4-1-3)）．ACssが障害であれば，サーバACのACMおよびACssが障害であるとみなし，サーバACのACMとACssの双方の切り替え処理を実行する．ACssが正常であれば，以降繰り返しACssが依頼している処理について調べ，その結果，最終的に障害を起こしたACが判明した時点で，その障害を起こしたACとその処理依頼元のACを監視するACMの切り替え処理を実行する．
－障害サーバACがさらに他のACに処理を依頼していないとき
試験では検出できない障害がサーバACに発生しているとみなし，そのサーバACの切り替え処理を実行する（図5.3.5(4-1-4)）．
b)応答が「非実行中」のとき
サーバACのACMに，障害を申告したACが障害と申告されたサーバACに依頼した処理のログ情報が登録されているかをGCMが問い合わせる（図5.3.5(4-1-5)）．サーバACのACMのログ情報にサーバACが実行した処理名が登録されていれば，応答時にサーバAC－AC間の通信障害が起きたものとみなし，通信処理管理部にネットワークの障害情報を問い合わせる（図5.3.5(4-1-6)）．また，サーバACのACMにサーバACが実行した処理名が登録されていなければ，処理要求時にAC－サーバAC間の通信障害が起きたものとみなし，通信処理管理部にネットワークの障害情報を問い合わせる（図5.3.5(4-1-7)）．
なお，サービス障害の原因が通信処理障害と判定されたとき，GCMは，それまで仮の代用サーバACを用いていたリスト中のACに対し，それ以後，当初用いていたサーバACを用いるようにACの切り戻し処理を実行する．
(4-2)応答の内容が「障害」あるいは応答がないとき
障害と申告されたサーバACの切り替え処理を実行する．

以上の手続きで，ACMとGCMが協調動作することにより，ACの障害の検出及び切り替えが行われる．なお，AC切り替え処理に伴って，そのACのACMについても同様の切り替

え処理がGCMにより実行される.

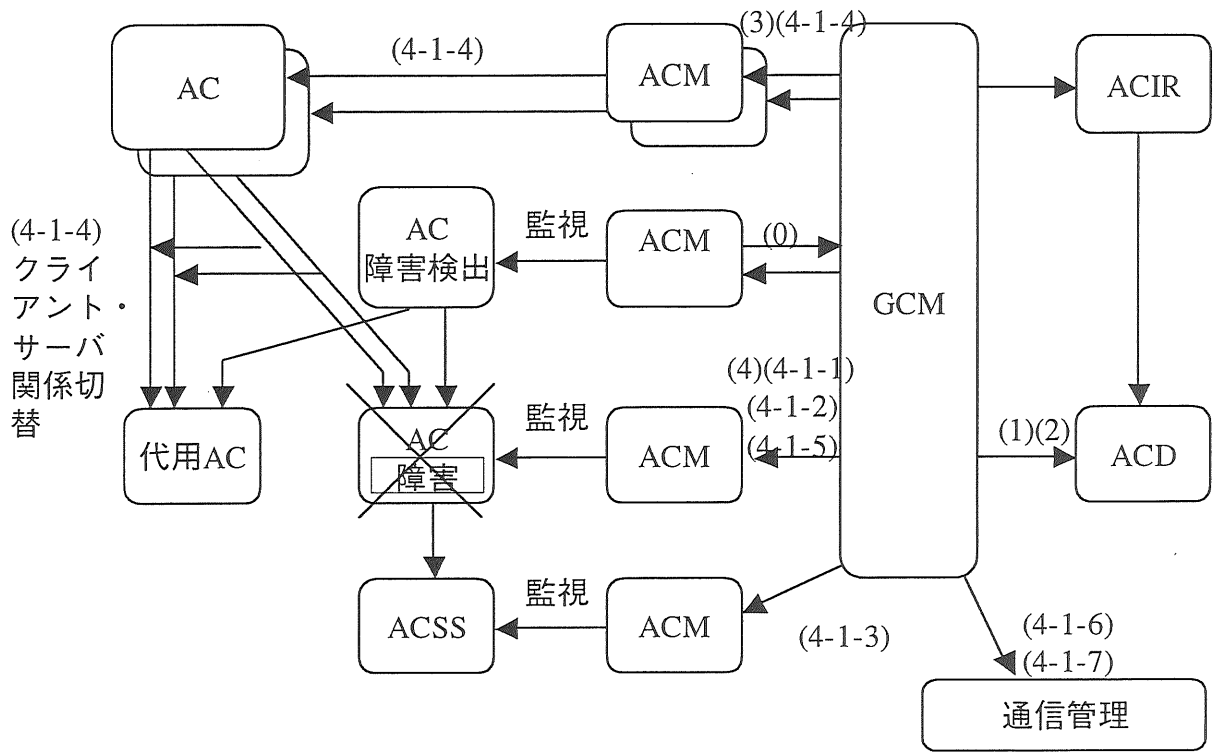


図5.3.5　AC障害復旧手順（サーバACの障害）

5.4　トレーディングサービスに基づくソフトウェアコンポーネント障害管理法

5.2節の要求条件を満たす方策として5.3節に，SOMSEを提案した．SOMSEにおいては，アプリケーションソフトウェアコンポーネント(AC)の障害を検出するために各コンポーネントを監視するアプリケーションコンポーネントマネージャ(ACM)を仮定する．クライアントに障害をACMが通知するとともに，コンポーネント代替関係データベース(ACD)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，障害検出機構であるACMを被管理コンポーネントの外部に設けることはシステム全体のコンポーネント数を増加せしめ,ACM自身の障害管理が別途必要になるという課題がある．また，システム全体のオブジェクト間関係をコンポーネント代替関係データベースを別に設けて管理することは，システム規模の増加に伴う負荷の増大が課題となる．

そこで,SOMSEの自律性を生かしつつ,上記の問題を解決することを検討した[41][43][44].

まず検討の前提として,対象とするコンポーネントは,CORBAオブジェクトとした．こ

れは現状で最も標準化が進んでいること,広範囲に利用されつつあることから選択した.ほかに考えられるコンポーネントとしては, EJB(Enterprise JAVA Beans), Microsoft COM などがあるが, これらは今後の検討課題としたい.

以下に, 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１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する.

CORBA プラットフォームにおいて, 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く (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い合わせ(タイプ情報などを鍵とする)により, 対応するサーバオブジェクトレ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) (図5.4.1).

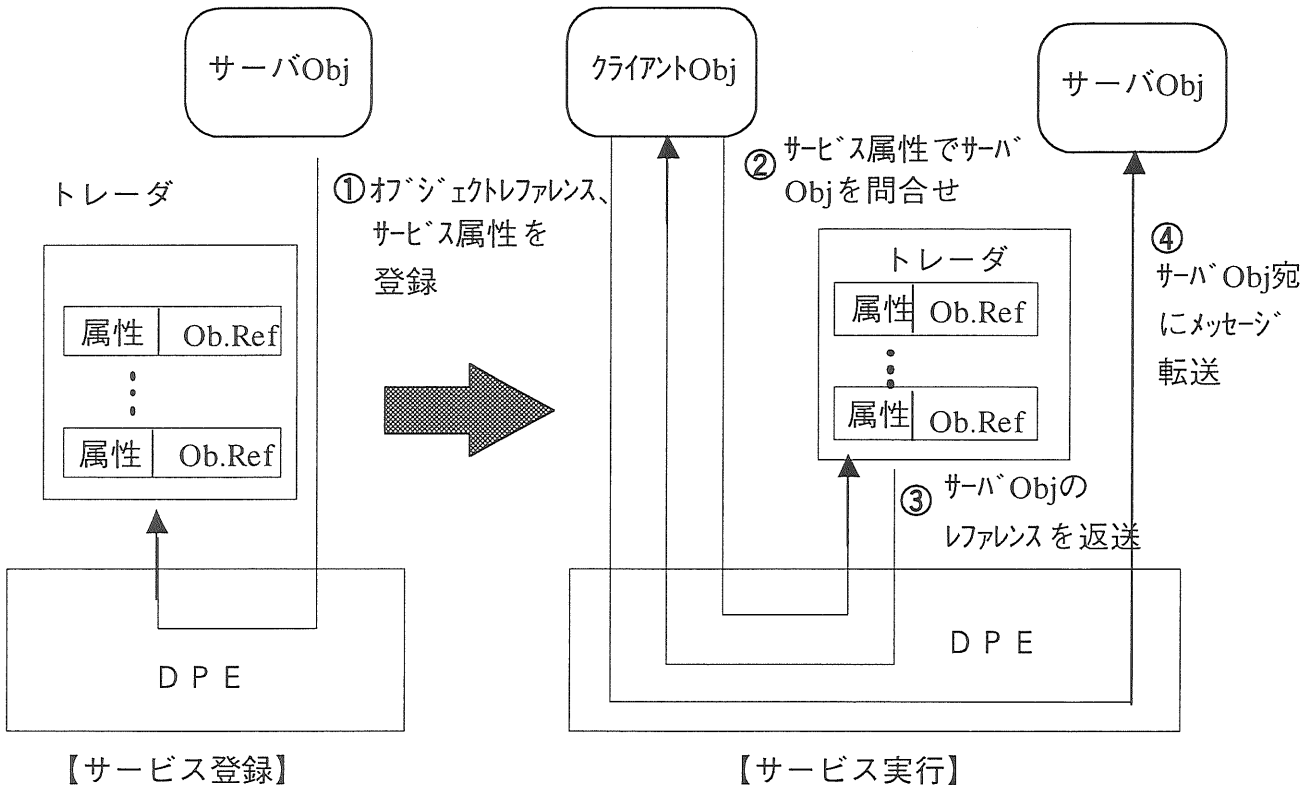


図5.4.1　CORBAにおけるトレーダの役割

本障害管理法を, 5.2.1 の要求条件に対応させて順に手順を述べる.

5.4.1　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

してサーバオブジェクトにおける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設け，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障害を検出する.
(iii)オブジェクト自身による検出：オブジェクトが障害に陥ったことをオブジェクト自身で検出する.

(ii)の各オブジェクトの監視機構を設ける方法は，監視機構の信頼性確保方法の検討が必要なことや監視機構も含めたオブジェクト数が増加することによりシステム全体としての管理対象が増加することなどの問題がある．また,(iii)の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.

従ってここでは(i)のクライアントによる検出方法を採用する.ただし,当該サーバオブジェクト（子サーバと呼ぶ）がさらに別のサーバオブジェクト（孫サーバと呼ぶ）にクライアントとして当該処理を依頼している場合，当該子サーバの障害ではなく孫サーバ障害の可能性もあることに留意する.

[障害可能性検出方法]

あるクライアントオブジェクトがサーバオブジェクトに処理を依頼して一定時間内にサーバオブジェクトから応答を受けない場合や，異常な応答を受ける場合に，当該サーバオブジェクトを障害の可能性ありとみなし，障害可能性のあるサーバオブジェクトリファレンスと自オブジェクトレファレンスをトレーダに通知する.(孫サーバ異常の場合,子サーバは孫サーバ障害の可能性を検出し,同様にトレーダに通知していることになる)

[障害の特定]

(1)の障害可能性検出結果(自オブジェクトレファレンス＝α,障害可能性オブジェクトレファレンス＝β）を受けたトレーダは，前後一定時間内に，βを自オブジェクトレファレンスとしている別の障害可能性検出結果を受けなければ，βを障害とみなす.

βを自オブジェクトレファレンスとしている別の障害可能性検出結果（障害可能性オブジェクトレファレンス＝γ）を受けた場合，同様に前後一定時間内にγを自オブジェクトレファレンスとしている別の障害可能性検出結果を受けなければ，γを障害とみなす.

5.4.2 障害の通知

障害を検出したクライアントは当然障害をすでに知っているが，障害発生以前にトレーディングにより，障害サーバオブジェクトレファレンスを取得しているクライアントオブジェクトで，まだそのサーバにアクセスしていないオブジェクトにはまだ障害が通知されていない．再びトレーダに問い合わせることはしないため，サービスを受けようとした時点で障害に遭遇することになり，それを未然に防ぐことができないことがあげられる.(孫サーバ障害のときであって,他のクライアントの処理が障害孫サーバに処理を依頼しない場合は，障害に遭遇しない).

この方式では代替関係データベースを利用せず,トレーダにおいて問い合わせ履歴を残すことにより，障害サーバのレファレンスを既に取得しているクライアントオブジェ

クトのみに積極的に障害を通知することで防ぐことができる.

[障害の通知]

トレーダへのインポート履歴に従って以前に障害の特定がなされたオブジェクトレファレンスをインポートしたクライアントオブジェクトのみに，障害を通知する．通知を受けたクライアントオブジェクトは，障害オブジェクトに依頼する内容の処理が発生したときにトレーダに問い合わせ，以下に述べるように代替オブジェクトレファレンスを得る.

5.4.3 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オブジェクトがその旨をトレーダに通知することにより，トレーダは障害サーバオブジェクトを認識できる．トレーダは，障害オブジェクトをインポート対象から除外する．このときに，障害オブジェクト回復のための手順として適切な動作を取る（(5)に示す）．同時に，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しておく.

[代替オブジェクト検出]

障害を特定したトレーダは当該オブジェクトをインポート対象から除き，障害オブジェクトと同様のサービスオファを持つ代替オブジェクトを検出する.

5.4.4 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障害サーバオブジェクトの提供するサービスを鍵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ジェクトを代替オブジェクトとして通知でき，未然に障害遭遇を予防できる.

[代替オブジェクト通知]

オブジェクトレファレンスインポート要求に，通常のトレーディングと全く同じ処理を行う（代替オブジェクトが検出されているため）

5.4.5 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.

[障害回復]

保守者に通知,あるいはライフサイクル管理を起動して障害オブジェクトを削除する.

図5.4.2，図5.4.3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.

以上の方法は，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障害通知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.

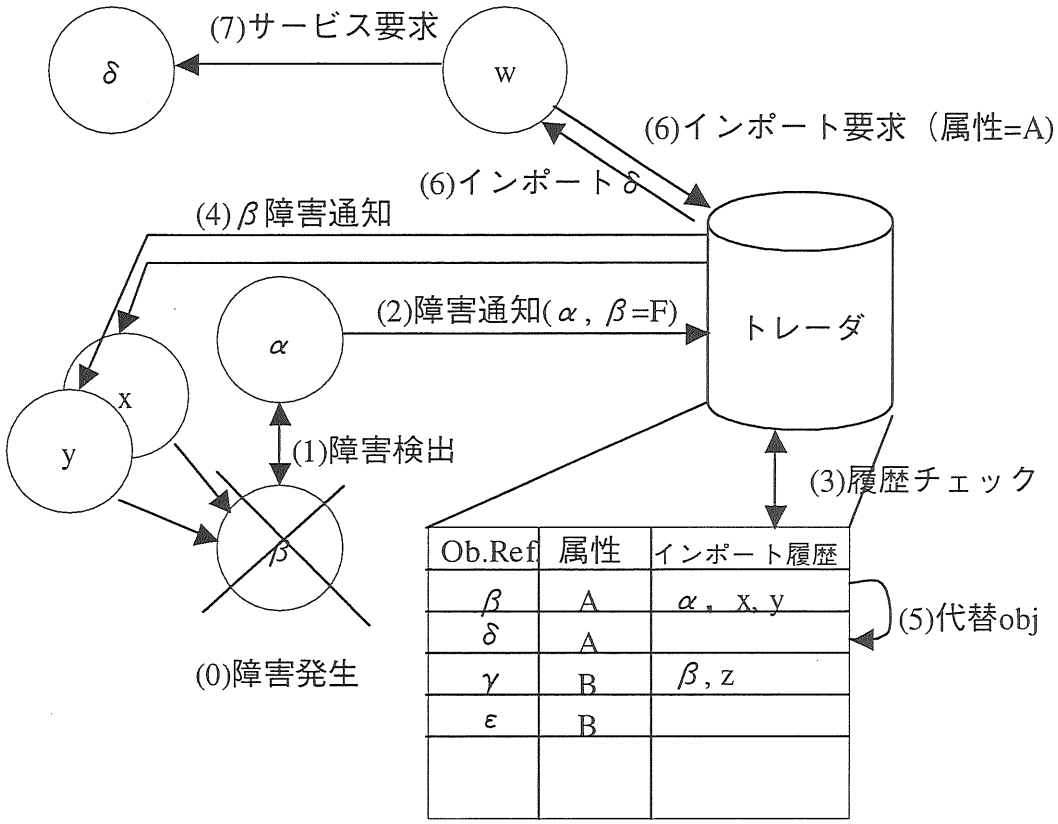


図5.4.2　トレーダ利用オブジェクト障害管理

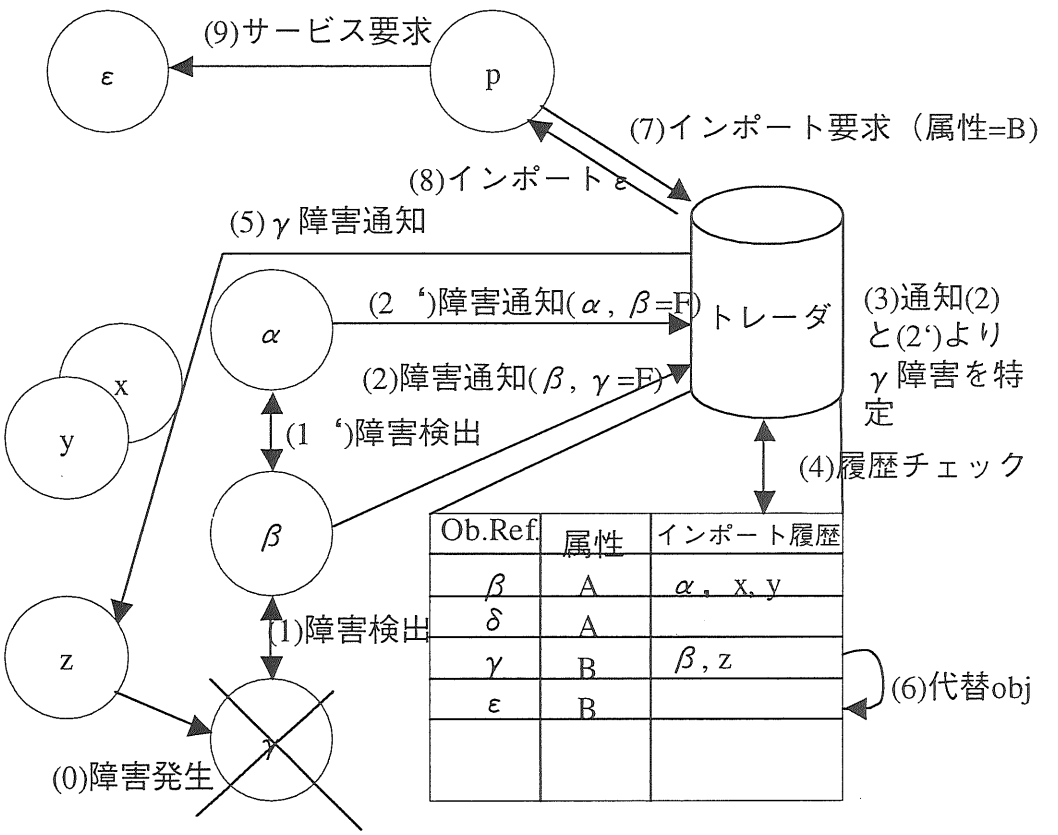


図5.4.3　トレーダ利用オブジェクト障害管理
（孫サーバ障害）

5.5 ハードウェア障害管理方式

本節では，ストリーム型データ駆動プロセッサを通信情報ネットワーキング環境に適用し，5.2.2のハードウェア障害への要求条件を満たすハードウェア障害管理方式を検討する[25][37][41][43][44]．クラスタリング技術[12]は，サーバをひとつのブラックボックスとして，それらの多重化・切り替えをOSやソフトウェアアルゴリズムによって実現している．それに対して，ここでは，さらに粒度の小さなプロセッサレベルに着目し，プロセッサ自身が障害監視と切り替えを可能とする自律分散プロセッサ障害管理方式を提案するものである．

5.5.1 前提

通信情報ネットワーキング環境のサーバノードハードウェア（プロセッサ）へのデータ駆動プロセッサの適用において，以下の前提を設けた．

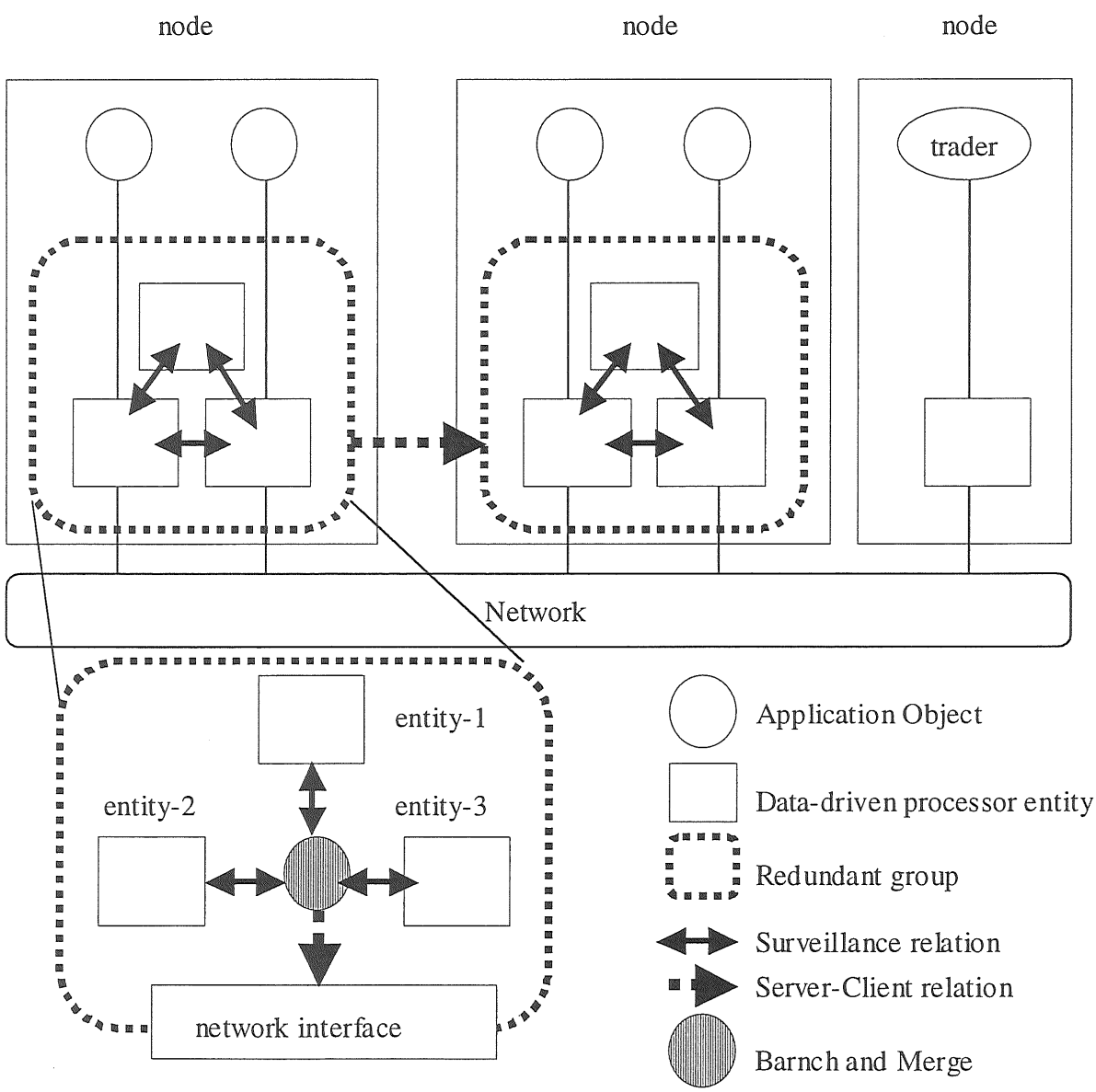


図5.5.1 ハードウェア障害管理構成

(1)障害管理を有効とするためにシステムに冗長性を持たせる（マルチプロセッサ構成とし，処理能力に余剰能力を持たせる構成とする）
(2)上記冗長性は同時に負荷平衡のためにも利用する
(3)「冗長群」と呼ぶ冗長性を共有するプロセッサグループを定義し，その冗長群に属するプロセッサ個々をエンティティと呼ぶ．各エンティティは同一冗長群に属する他のエンティティに冗長性を与え他のエンティティと負荷を共有する．
(4)同一冗長群内のエンティティ間の関係は同じ冗長群に属する他のエンティティによって知られている．

図5.5.1にこの提案にもとづくシステム構成を示す．冗長群は物理的な入出力ポートを有し，その入出力ポートは内部インタフェースあるいはネットワークインタフェースに接続されている．ひとつの冗長群中のエンティティ数は負荷の大きさと障害耐力の程度によって決定される．以下に5.2節の要求条件をいかに解決するかを示す．

5.5.2 要求条件1：障害監視・検出

冗長群の中では，全エンティティの処理段を一巡するする監視パス上に試験メッセージを送り，システム内に循環させることにより，他のエンティティを監視する．図5.5.2に，冗長群に2つのエンティティのみがある場合を例として監視機構を示す．

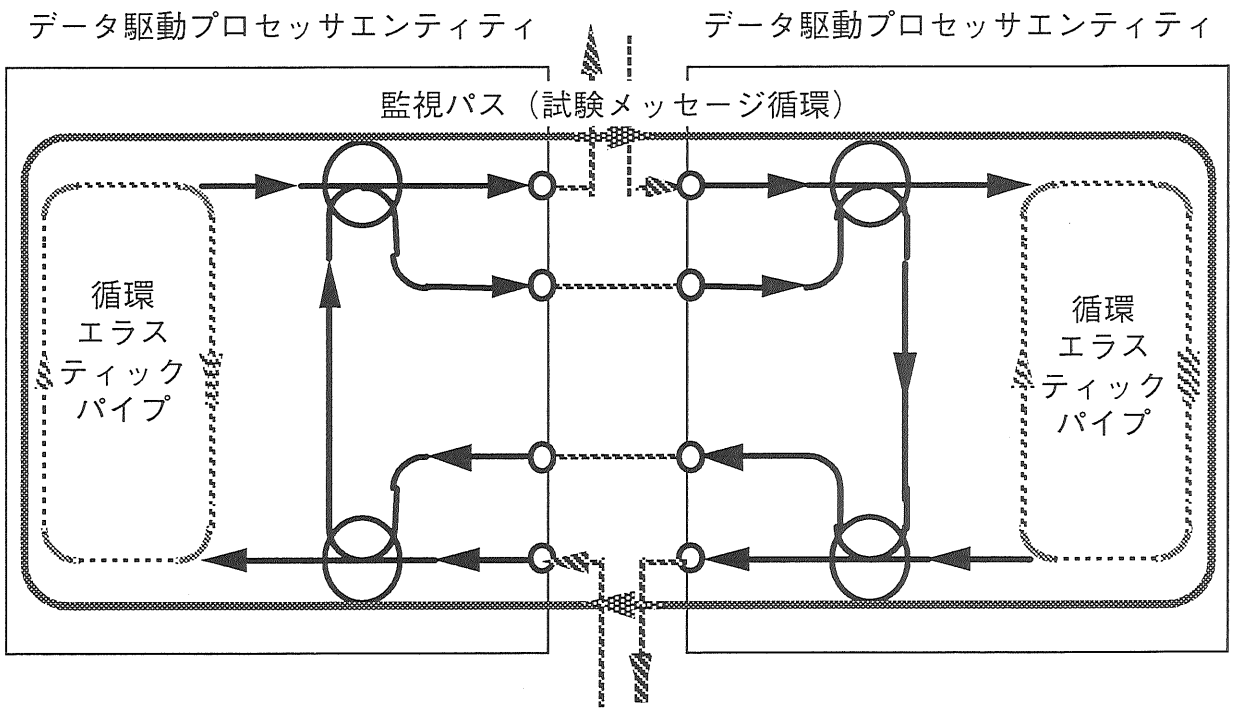


図5.5.2 障害監視機構

3.2節で述べたストリーム型データ駆動プロセッサの特性により以下が言える．
・循環パイプラインの中で消費されるテストメッセージのターンアラウンドタイムはステージ数により決定されるため，安定して変動は少ない．

・並列処理の特性を生かし，テストメッセージに優先度を与えることにより過負荷状態でもテストメッセージの送受が可能である．

図5.5.3に通常処理に優先度をつけた試験メッセージを付加したときのターンアラウンドタイムを示す．若干の変動はあるが，ほぼ系を通過するに必要な時間で一定の値となっている．

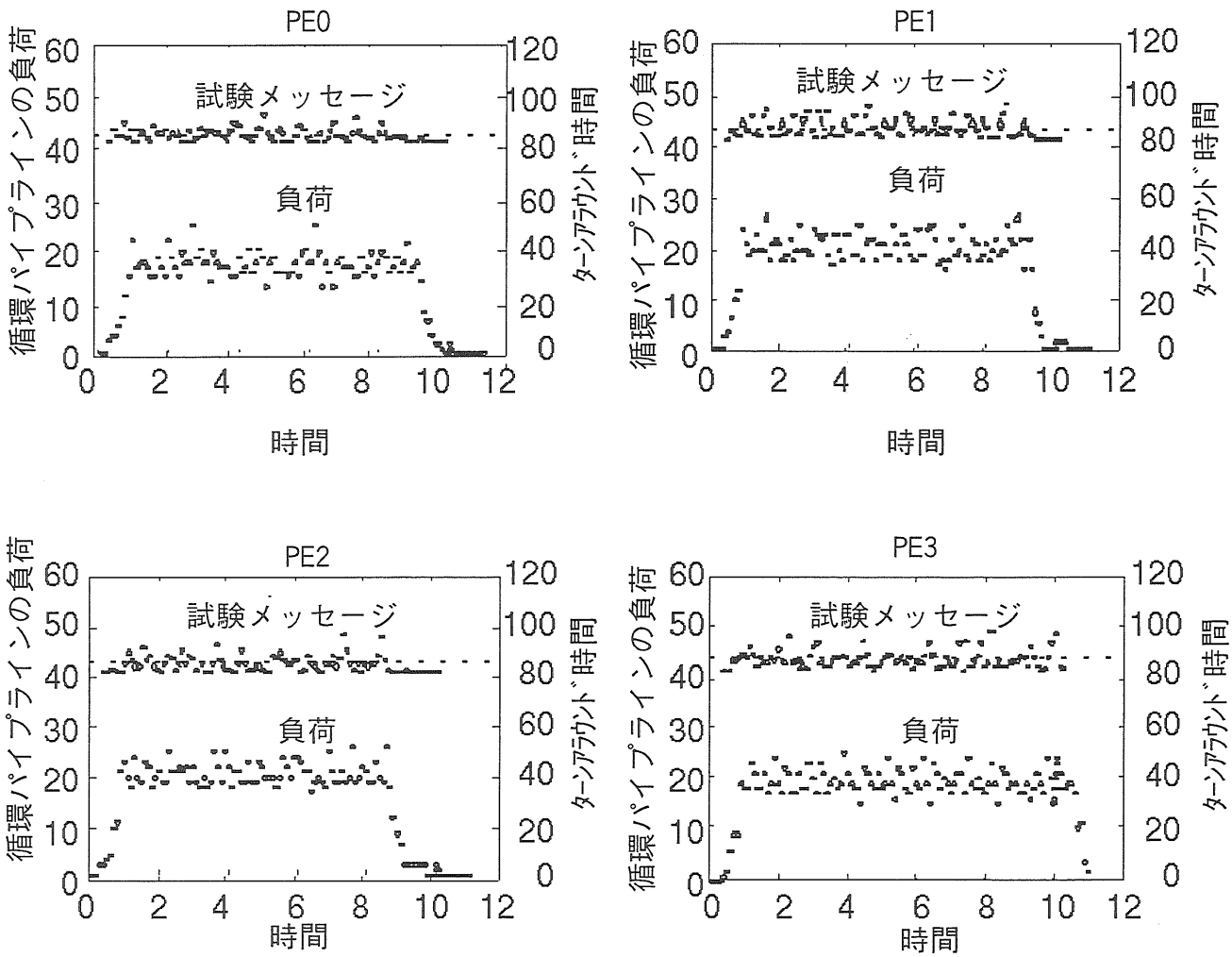


図5.5.3　試験メッセージの実行時間

一般に，障害監視を行うためには，付加的なハードウェアによる専用装置が必要となってくる．しかしながら，このアーキテクチャにおいては負荷分散のために同じ機能を複数のエンティティに割り当てており，各エンティティはお互いに監視を行うため外付けの付加的な装置は必要としない．同時にこれらのエンティティは高い信頼性のあるシステムを実現するのに不可欠な冗長性を与えている．ノイマン型プロセッサの場合，監視のためのテストメッセージはシステムのスループットを下げる．しかし，データ駆動プロセッサは，オーバヘッドなくマルチプロセス可能な利点により，テストメッセージが実行されても通常のスループットを維持できる（図5.5.3）．

従って，上記提案監視機構はオーバヘッドを与えることなく容易に障害監視を以下の

ように実現できる．

(1)障害検出

もしテストメッセージが一定時間内に戻らなかった場合，なんらかの障害状況が被試験エンティティにおいて発生している可能性がある．

(2)過負荷検出

テストメッセージが一定時間内に戻ったとしてもターンアラウンドタイムが長くなっている場合，過負荷状況が被試験エンティティにおいて生じている可能性がある．

5.5.3 要求条件2：迅速な障害箇所特定，障害箇所の切り離し

5.5.2において述べたように障害や過負荷が検出されると，以下の機構が問題を処理する．

(1)障害が検出されたとき

図5.5.4に示すように，障害エンティティの入力を当該エンティティに入れることなく直接出力ポートにバイパスすることにより障害エンティティを系から切り離すことができる．ここで特筆すべきことは，データ駆動プロセッサは入力が遮断されると全く機能しなくなり，結果として障害エンティティが系全体に悪影響を与えることはなくなることである．系に積極的に影響するアクティブ障害のときには非常に有効である．ノイマンプロセッサではアクティブな影響を避けることが難しいことがある．このように容易に系の再構成が可能となる．

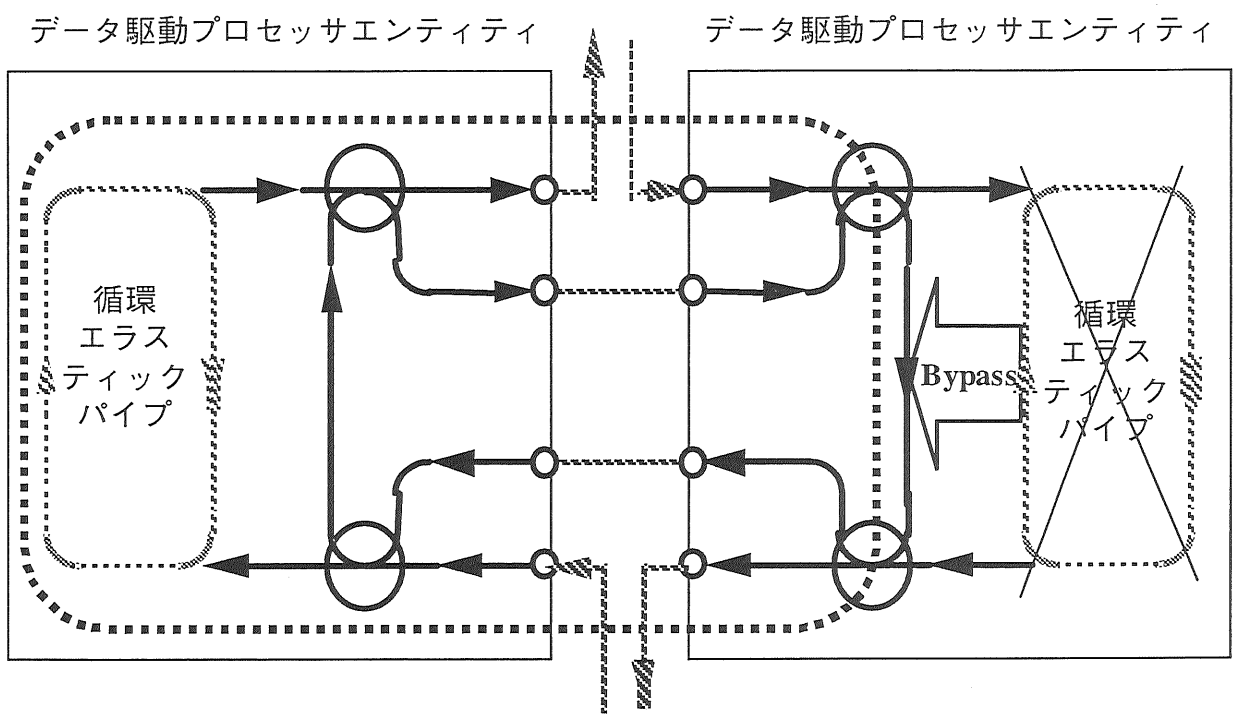


図5.5.4 障害エンティティの切り離し

(2)過負荷が検出されたとき

データ駆動マルチプロセッサ構成においては，すでに示した図3.2.3のように負荷平衡が自動的に行われる．従って，ひとつのエンティティのみが過負荷になることはなく，過負荷条件が検出されたときは，冗長群全体が過負荷条件になっていることになる．この状況は,この冗長群をサーバとして利用している他のクライアント冗長群によって,同一冗長群内で行うとまったく同様な試験メッセージの循環で検出できる．

5.5.4 要求条件3：局所障害管理機構

各エンティティは自分が属する冗長群の情報を有している．系全体的な知識は必要としない．必要な情報は同一冗長群の中の構成情報のみである．障害は5.5.3に述べたように主として同一冗長群内で解決され，その冗長群のサーバやクライアントに影響を与えることは原則としてない．

ただし，冗長群全体が障害や過負荷条件になったとき，その冗長群のクライアントは3.2.3の機構によって状況を知ることができる．この場合には，DPEにおけるトレーディング機構のようなものを設けることによって，障害冗長群のクライアントが代替の冗長群を見つけることを助ける．

このように，ここで提案した障害管理機構は，全体的な知識を必要としない局所管理を可能としており，例外的に冗長群そのものが障害や過負荷に陥った場合にのみ，共通サービスとしてのトレーディングを必要とする．

この例外的な対策は，後述のソフトウェア・ハードウェア連携管理において詳細に述べる．

5.5.5 要求条件4：効率的管理処理

本アーキテクチャはいかなるプロセスも並列処理される．明らかにデータ駆動プロセッサは多重処理にオーバヘッドを与えないため，管理処理と通常処理が互いに影響しないで非常に効率的な処理が行われる．

5.5.6 要求条件4：過負荷耐力

過負荷に対する耐力はすでに3.2節で示している（図3.2.2）．過負荷条件においてもエラスティックパイプラインによって試験メッセージを処理することができる．

5.6 ハードウェア・ソフトウェア連携障害管理方式の提案[43][44]

5.6.1 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）障害管理のソフトおよ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,5.5節に述べたハードウェア障害管理方式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隠蔽されることが多い．しかしながら，5.5節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，迅速な切替が難しい．また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．

従って，5.5節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時，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持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持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，処理能力の低下をさらに悪化させる．

(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毎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．

従って，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コンポーネント，プロセッサハードウェアのそれぞれの障害管理方式に加えて，ソフトおよびハードの障害管理が連携することが必要となる．

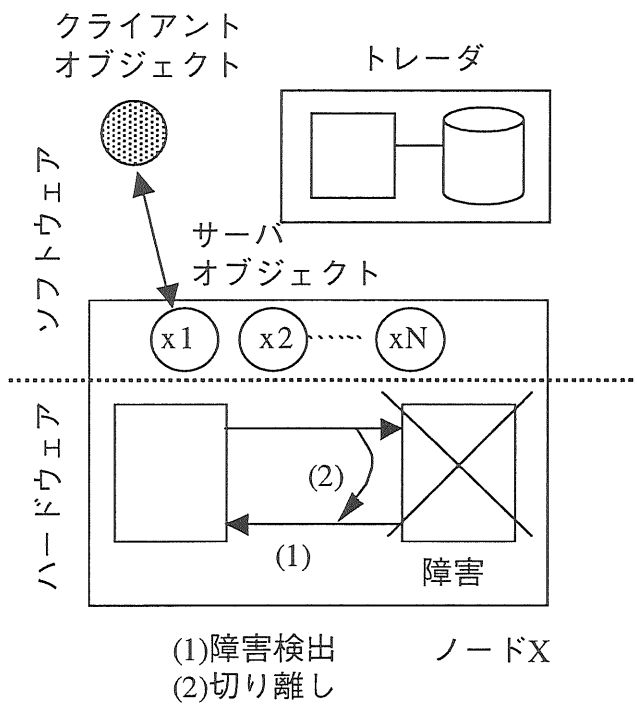
5.6.2 ハードウェア・ソフトウェア連携障害管理方式

5.6.1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と5.5節のハードウェア障害管理機能の採用を仮定する．ソフトウェアオブジェクトの障害管理には5.4節で述べたトレーダを利用したソフトウェアコンポーネント障害管理を前提とする．さらに，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．

5.6.2.1 5.6.1(a),(b)を解決する方策

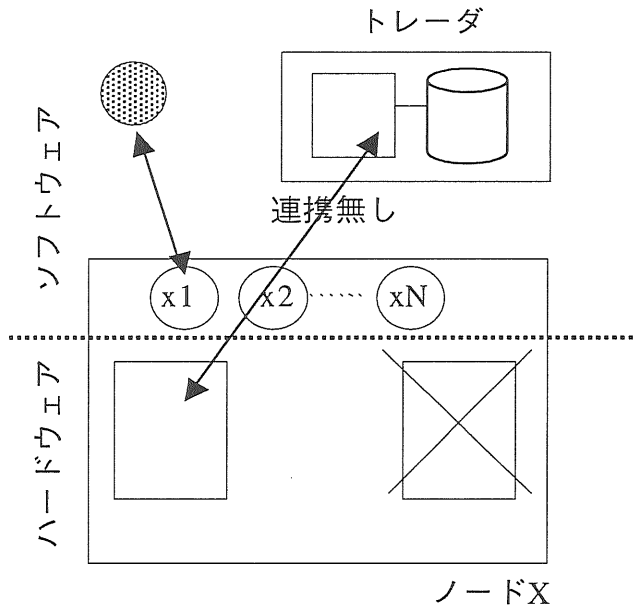
5.6.1の問題(a)(b)を解決するための処理の流れを以下に提案する．

(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（図5.6.1）．



プロセッサ障害発生
（ハードウェア障害管理により障害エンティティ切り離し）

図5.6.1　ソフトウェア・ハードウェア連携障害管理の流れ(1)



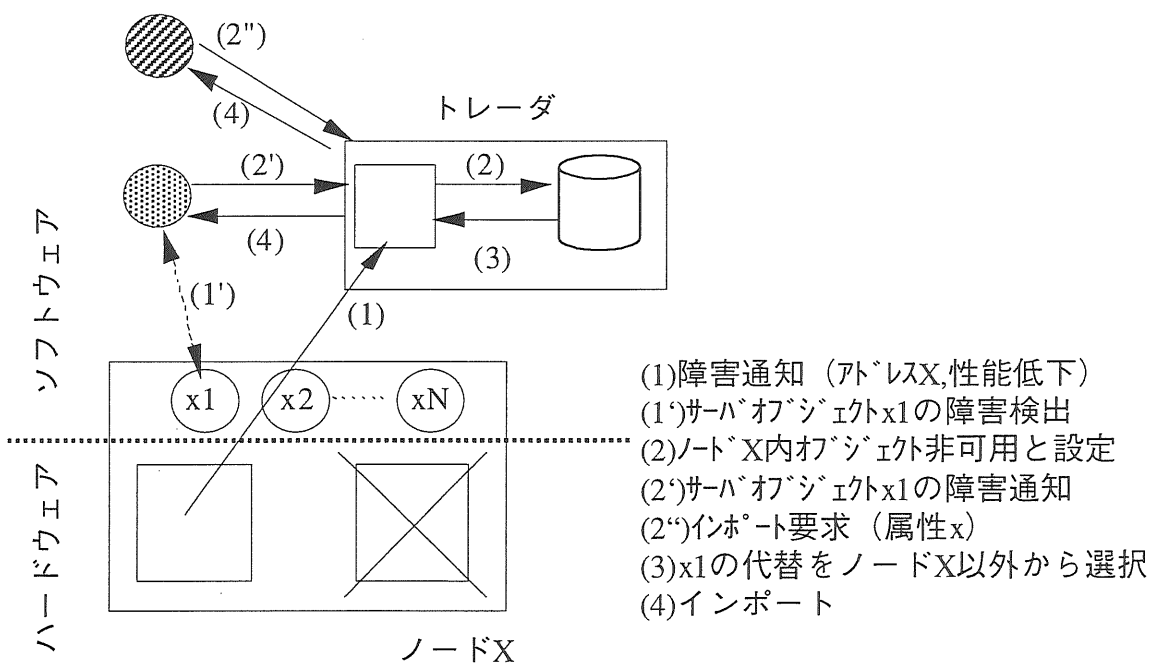
処理能力低下が影響無い場合
（ソフトウェアへの影響無し）

図5.6.2　ソフトウェア・ハードウェア連携障害管理の流れ(2)

(2)5.5節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（図5.6.2）
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（図5.6.3）
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，5.6.1(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5.6.1の(b)の問題は解決できる．



処理能力低下が大きい場合
（連携管理が起動する）

図5.6.3　ソフトウェア・ハードウェア連携障害管理の流れ(3)

5.6.2.2　5.6.1(c)を解決する方策

5.6.2.1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，問い合わせがあって初めて代替オブジェクトを通知するしかけ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い合わ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5.6.4に，本方策を示す．

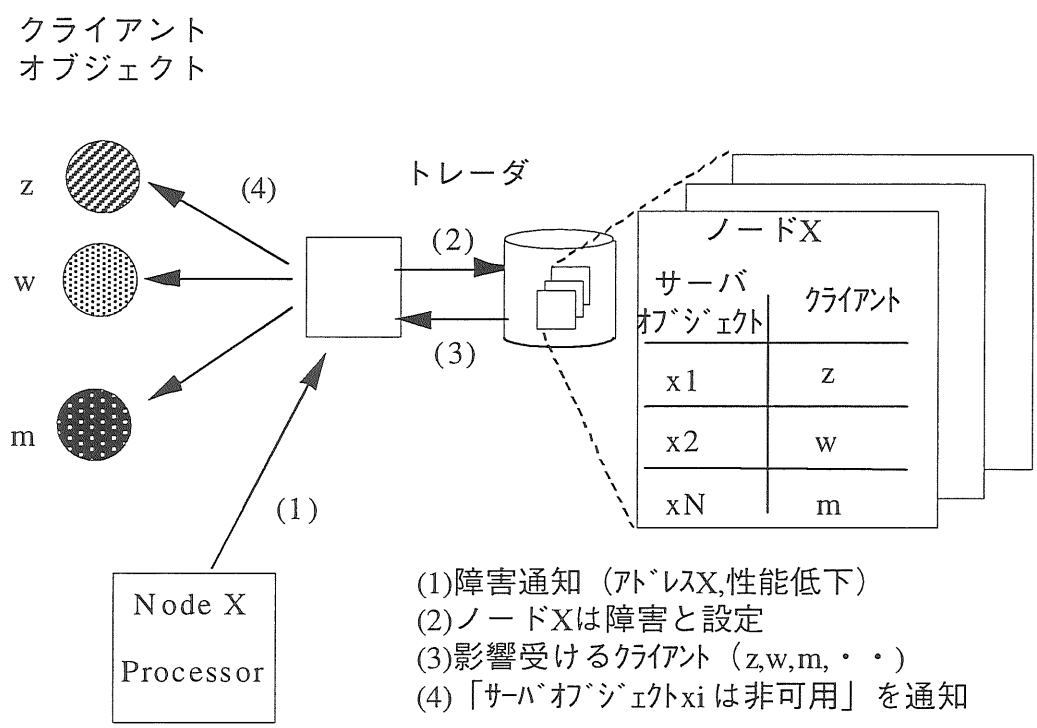


図5.6.4　障害ノード上のオブジェクトのクライアントに対する通知

5.6.2.3　5.6.1(d)の対策

5.6.1(d)の場合，データ駆動型プロセッサ障害管理から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，本研究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取られており，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．従って，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替えられるが，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，4章で提案したようなデー

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．

5.6.2.4 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，ただ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司るトレーダは，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．

5.7 結言

本章では，通信情報ネットワーキング環境の高信頼化のため，ソフトウェアオブジェクトやプロセッサの障害管理に着目した．

まずソフトウェア障害管理に対してSOMSE (Service Operation and Management architecture using Surveillance of application software Elements)アーキテクチャを提案した．SOMSEでは，通信情報ネットワーキング環境において，物理的地理的に分散したアプリケーションソフトウェアコンポーネント個々の状況を監視する監視コンポーネントを設置し，状況に応じて，影響を及ぼすアプリケーションを明確にし，障害が波及しないように，代替コンポーネントを設定したり，障害コンポーネントを系から排除するなどの対策を可能とする．さら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に若干の追加機能を設けることにより，障害の波及を防止する，という拡張を加えた障害管理方式を提案した．

ハードウェア面では，すでに通信情報ネットワーキング環境への適用性の高さを示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提案した．ここでは，上位のアプリケーションコンポーネントに対応した形でのハードウェアプロセッサの運用を前提に，専用に監視装置を設けることなく，冗長構成の中で相互監視を行う仕組みを提案した．

次に，これらソフトウェアおよびハードウェアそれぞれの障害管理機能をもとに，通信情報ネットワーキング環境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た．ハードウェア，ソフトウェア個々の障害管理が連動しないで起こる問題を明確にし，それらを解決する手段として，ハードウェア障害をソフトウェア障害管理に通知する手段を新たに設けることにより，効率的で有効な総合障害管理方式を明らかにした．

今後の課題としては以下のものがあげられる．

(1)既に，実際のアプリケーションサーバへの適用を念頭に，筆者らはNTTにおいてCORBAオブジェクト管理機構の実装を行い，事業導入を行いつつある．このサービス面

での有効性の検証が緊急課題である，
(2)ハードウェア障害管理機構は，たとえばトラヒックの集中する共通サーバ群など，データ駆動プロセッサの適用領域を明確にし，その範囲においてまず実装を開始し，その実際的有効性を検証すること
(3)その上で，ハードウェア・ソフトウェア連携管理を実際のサービスに供すること

第6章　結論

インターネット利用型サービスに代表される通信と情報処理の融合が急速に進んできている．これに合わせて，ネットワークノード，サービスノード，ユーザシステムなどのハードウェア及び伝達層，分散処理環境，ネットワークミドルウェア，さらにはアプリケーションまでのソフトウェアがネットワーク上に分散し，これらの機能群が互いに協調しながらサービスを実現するようになってきた．このような新しいネットワーク型サービスの開発・提供環境すなわち，通信情報ネットワーキング環境と呼べる新しいパラダイムが興ってきている．

本研究の目的は，高信頼・高効率な通信情報ネットワーキング環境の実現法の明確化である．ハードウェア面，ソフトウェア面，それぞれ個別に着目するとともに，両者が連携した観点でも検討を行うことにより，従来十分に検討されていない，環境総体として品質を実現する方法を明らかにする．

本研究成果は，以下の４点に集約できる．

１点めは，通信情報ネットワーキング環境の実現のために考慮すべき要件を，アプリケーションからハードウェアまで幅広く目を向け明確化したことである．ハードウェアに関する要求条件は，近年の高速プロセッサの発展により，ほとんどが考慮すべき対象となっていない状況である．しかしながら，単なる高速化ではなく，いかに与えられた有限な資源を有効に活用するかについての検討は抜け落ちていると言える.本研究では，資源有効活用の観点，即ち，高多重性，高効率性の実現を研究のひとつの大きな柱に位置づけ，その実現法を明確に示した．またこのハードウェアに関する要求条件とアプリケーションに関するそれを密接に関連づけ，なおかつ双方が階層として独立に機能し得る障害管理法を明確にした．

２点めは，データ駆動プロセッサの適用による通信情報ネットワーキング環境の実現である．１点めの特徴にも述べたように，今やハードウェアの要求は，高速度と低価格に集中している.しかしながら,ソフトウェアレベルでの分散に対応する形でハードウェアも分散処理を実現できたり，あるいは，通常処理と付加処理（たとえば，付加的な障害探索の業務など）を同時進行させてもなんら通常処理を圧迫することがない，などの要求条件を正面から検討しているものは見受けられない．本研究では，従来高速並列計算機アーキテクチャとして検討されてきたデータ駆動プロセッサ技術を，高速化ではない，高信頼化，高効率化という新しい観点で検討し，その高い適用性を明確にした．

３点めは，通信情報ネットワーキング環境へのユーザからのアクセスに注目した高効率なプロトコル処理法を明らかにしている点である.２点めで述べたデータ駆動プロセッサを適用することにより，既存の非効率なノイマン処理を飛躍的に効率化できることを示した．

４点めは，通信情報ネットワーキング環境の分散処理の構造に合致した，自律分散的障害管理のアーキテクチャを明らかにしている点である．従来のTMNに代表される管理アーキテクチャは，情報としても，あるいは機能としても階層化され，最終的な判断は

集中化して，処理能力的に，また障害耐力にも限界が生じる．本研究で提案したソフトウェア，ハードウェアにおける障害管理アーキテクチャではできるだけ自律分散構成により集中化をさける新しい障害管理のアーキテクチャを明確にした.ソフトウェア,ハードウェアそれぞれの階層で独立に機能する障害管理機構に加えて，両者が有機的に連動する総合的障害管理機構を明らかにした.

具体的には，以下の順で研究成果をまとめた.

２章では，通信情報ネットワーキング環境の実現に当たっての解決すべき課題，要求条件を明確化した.

まず，通信情報ネットワーキング環境の検討の最も代表的なものとして，電気通信情報ネットワーキングアーキテクチャ（TINA）について概観した.

ついで，同環境を実現する上で，主にアプリケーションを支えるハードウェアプラットフォームの観点さらにアプリケーションの障害管理の観点から解決すべき問題点として，(1)ネットワークインタフェースの多重時の非効率性,(2)DPEサーバーなど共通サーバへの負荷集中,(3)サーバノードの信頼性の確保,(4)ネットワークの信頼性の確保,(5)アプリケーション障害波及の防止，を明らかにした.

それらをもとに，通信情報ネットワーキング環境実現のための要求条件を抽出した.第一の要求条件として,通信情報ネットワーキング環境の高効率化の必要性を述べた.現状のシステムにおける多重処理の必要性を例示し，そこにおける非効率性が解決すべき課題として重要であることを示した.

さらに，次の要求条件として，通信情報ネットワーキング環境の高信頼化を述べた.すでに多くの研究がこの分野でなされてきたが，伝達ネットワークと上位のアプリケーションなどが連携した形で信頼性を確保するという試みは十分になされていない．ソフトウェア，ハードウェアそれぞれの観点からの高信頼化，特に障害管理の重要性を明確にした.

加えて，通信情報ネットワーキング環境において特徴的なマルチメディア処理の必要性などその他の要求条件を明らかにした.

３章では，２章で明らかにした，通信情報ネットワーキング環境実現上の要求条件を具体化するための研究結果としてデータ駆動原理の通信情報ネットワーキング環境への適用性の高さを明確にした.

まず，西川らが検討してきたストリーム指向データ駆動プロセッサをとりあげ，その原理，アーキテクチャを述べた．その中で，実測を通じ，本プロセッサのエラスティックパイプラインにより，２章であげた要求条件の１つである，過負荷耐力を有すること，各プロセッサが付加的ハードウェアなく相互に接続でき，もう１つの要求条件である容易なマルチプロセッサ構成を満たしていることを示した.

続いて，既存のノイマン型プロトコルの多重処理の非効率性を検証するために，典型的な通信情報ネットワーキング環境の実現形態であるクライアント・サーバ間の処理の多重度を変えることによる各種処理時間の傾向を，実測および数学モデルによって解析

した．その結果，多重度が上がるにつれて，１つのプロセスあたりの処理時間が長くなることから，多重処理におけるコンテキストスイッチのオーバヘッドが大きくなっていることを示し，その処理の非効率性を明らかにした．

さらに，この結果を踏まえて，データ駆動プロセッサをエミュレーションによって実現し，多重度に無関係に処理時間が一定という，きわめて高い処理の効率性を検証するとともに数学モデルによっても効率性を示した．加えて，ネットワーク再構成，障害対策，高速信号処理などその他の要求条件についても定性的に検討し，通信情報ネットワーキング環境に対する２章であげた６つの要求条件をデータ駆動プロセッサは満たし得ることから，通信情報ネットワーキング環境へのデータ駆動プロセッサの適用性の高さを示した．

４章では，通信情報ネットワーキング環境実現のための要求条件の１つである処理の効率性に着目し，プロトコル処理へのデータ駆動プロセッサの適用を検討した．

通信情報ネットワーキング環境において，また昨今のインターネットにおいて最もよく用いられるプロトコルであるTCP/IPを取り上げ，３章で明らかにした多重処理の高い効率性を有するデータ駆動プロセッサにより，TCP/IP処理の実現を検討した．

まず，IP処理部のみをとりあげ，ノイマン型処理と並列型処理をデータ駆動プロセッサでエミュレーションし，プロセス多重度と処理速度の関係を比較した．複数の同時処理がお互いに影響しない後者の並列型が高い効率性を示し，データ駆動プロセッサの並列性のTCP/IP処理への高い適用性を明確にした．

その結果にもとづき，TCP/IPのデータ駆動実装を行い，性能の検証を行い，データ駆動プロセッサを用いたプロトコル処理の高い効率性と通信情報ネットワーキング環境への適用性を実証した．

さらに，ユーザとネットワークの協調を実現することが特徴である通信情報ネットワーキング環境では，軽量で機能がそれほど高くないユーザ装置が通信情報ネットワーキング環境の一環に組み込まれ，サービスを実現するための，ユーザのネットワークへのアクセスに対する要求条件として，

(1)アクセス回線の効率的使用，(2)ストリームとメッセージでのアクセスの共用，(3)複数端末の収容，(4) 簡易なインタフェースを上げた．

最後にアクセスの要求条件にもとづいた通信情報ネットワーキング環境のアクセス系の実現法を検討した．

５章では，通信情報ネットワーキング環境の高信頼実現法の検討として，現在主流になりつつあるTINA/CORBAを利用した通信情報ネットワーキング環境の広域性と高信頼度への要求に十分に応えるための障害管理方式を明らかにした．まず，ソフトウェア，ハードウェアのそれぞれについて障害管理方式を検討した．ソフトウェア面では，ソフトウェアコンポーネント障害管理であるSOMSE (Service Operation and Management architecture using Surveillance of application software Elements)アーキテクチャを提案した．SOMSEは，通信情報ネットワーキング環境において，物理的地理的に分散配備されたアプリケーションソフトウェアコンポーネント個々の状況を監視する監視コンポーネントを設置し，状況に応じて，影響を及ぼすアプリケーションを明確にし，障害が波

及しないように，代替コンポーネントを設定したり，障害コンポーネントを系から排除するなどの対策を可能とする．さら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に追加機能を設けることにより，障害の波及を防止する，という拡張を加えた障害管理方式を提案した．

ハードウェア面では，すでに通信情報ネットワーキング環境への適用性の高さを示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特にプロセッサ障害管理を提案した．ここでは，上位のアプリケーションコンポーネントに対応した形でのハードウェアプロセッサの運用を前提に，専用に監視装置を設けることなく，冗長構成の中で相互監視を行う仕組みを提案した．

次に，これらソフトウェアおよびハードウェアそれぞれの障害管理機能をもとに，通信情報ネットワーキング環境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た．ハードウェア，ソフトウェア個々の障害管理が連動しないで起こる問題を明確にし，それらを解決する手段として，ハードウェア障害をソフトウェア障害管理に通知する手段を新たに設けることにより，効率的で有効な総合障害管理方式を明らかにした．

この研究の今後の検討課題と，それをふまえた今後の研究の発展方向を以下に示す．

(1)データ駆動プロセッサの適用領域の実環境における明確化

まず，実際のシステムにおけるデータ駆動プロセッサの適用領域を明確にしていく必要がある．本研究においては，プロトコル処理，障害管理を中心に具体的な適用領域を明確にした．しかしながら，すでに現状ほとんど全てのシステムがノイマン型処理装置で構成されており，その置き換えを議論することは現実的ではない．従って，既存システムとの明示的でオープンなインタフェースを定義し，もっとも有効な領域に効果的に適用する方法をさらに明らかにせねばならない．たとえば，本研究で明らかにしたように，トラヒックの集中する部分，多重処理がボトルネックとなる部分等に，部分的に適用していき，その上で，本研究で明らかにしたプロトコル装置や障害管理機構を実装していき，実装の中で，効果を検証することが重要な課題と考えられる．

(2)障害管理の適用

本研究で提案したソフトウェア障害管理機構は，さらに適用対象をCORBAからEJB(Enterprise Java Beans)にまで拡張することを進めている．本方式のように上位アプリケーションソフトウェアのコンポーネントレベルまでの管理機構は，世界的に見ていまだ実用化されておらず，きわめて斬新な技術として，NTTのデータセンタ事業に採用がほぼ決定し，ASP(Application Service Provider)の保守アウトソーシングサービスの強化によるサービス差異化に役立つものと思われる．今後は，ここで述べたハードウェアとの連携管理や，既存のネットワーク管理との有機的な連携を図る必要があり，現在精力的に検討を進めており，eビジネスのトータルなサービス管理を明確にし，実用化していきたい．

(3)プロトコル装置の適用

プロトコル装置については，上位レイヤとのインタフェースを明確にし，トータルシステムの中に組み込んでいける方法を明確にする．具体的には，UNIXソケットの明示的

な定義が考えられる．また適用領域を広げて，UDPや上位プロトコルへの適用を図り，効率性の特徴を生かす方向をさぐっていく．

謝　辞

本研究の動機付けから，遂行，まとめ，に至る全過程を通じて，終始懇切な御指導，御鞭撻を賜った筑波大学電子・情報工学系教授　西川博昭先生に心より感謝の意を表します．

また，本研究の審査にあたり有益な御助言をいただきました，筑波大学電子・情報工学系教授　斉藤恒雄先生，北脇信彦先生，井田哲雄先生，田中二郎先生に厚く感謝いたします．

本研究は，筆者が日本電信電話株式会社(NTT)研究所における研究業務の一環として行ったものであり，上司の御理解の賜物です．この間，研究の機会をいただき御指導を賜りました，NTTデータ株式会社取締役・ビジネス企画開発本部ITサービスパートナー本部長　井上友二博士，NTT情報流通基盤総合研究所主席研究員　北見憲一博士，NTTネットワークサービスシステム研究所プロジェクトマネージャ　鈴木孝氏,株式会社NTTドコモ研究開発本部ネットワーク研究所長　今井和雄氏,NTT情報流通プラットフォーム研究所プロジェクトマネージャ　今瀬真博士に深く感謝いたします．特に，本論文をまとめる強い動機付けをいただき終始叱咤激励をいただいた井上友二博士に重ねて感謝の意を表します．

本研究に関し，重要なる情報の提供，論文執筆上の大いなる助力と御討論をいただき，さらにシステム開発，事業導入の面で大変なご努力をいただいたNTT情報流通プラットフォーム研究所研究主任　田中博樹氏に深く御礼申し上げます．

本研究に関し，主に実装面で多大なるご協力と有益な御討論をいただきました，シャープ株式会社IC開発本部ネットワークシステムLSI開発センター副所長宮田宗一博士，筑波大学大学院博士課程工学研究科電子・情報工学専攻　青木一浩氏，我孫子泰祐氏を始めとする西川研究室の皆様に厚く感謝します．

本研究の実験データ収集に関してご協力をいただいたNTT情報流通プラットフォーム研究所主任研究員　小林秀承氏に深く感謝いたします．

本検討に際して，激励をいただいた，TINA-C関係者，特にTINA-C創設者のひとりであり初代副会長であるNTTデータ株式会社代表取締役社長　青木利晴博士，最後のTINA-C会長であるNTTエグゼキュティブアドバイザ・エジンバラ大学客員教授早稲田大学教授加納貞彦博士，元TINA-C副会長であるNTT取締役第三部門長　鈴木滋彦博士，元TINA-C CEO Deb Guha氏,元TINA-C CTO TINA International Scientific Committee Hendrik Berndt博士に深く感謝します．

本研究の過程において，適切な御助言をいただいた，NTTサービスインテグレーション基盤研究所主幹研究員　宮岸修氏，OMG代表　Richard M. Soley博士に厚く感謝します．

TINA関連共同検討などで，有益な討論をいただいた，CSELT Luigi Grossi博士，ナレッジファーム社長　更科賢一氏，テレコムイタリア社　Armando Limongielo氏，ACTS REFORMプロジェクト関連者，特にAlgosystems社　Panos Gorgatsos氏，信学会ネット

ワーキングアーキテクチャワークショップ関連者の方々に感謝します．

本検討のすべての過程においてご協力をいただいた，NTT研究所における諸先輩ならびに同僚のみなさま，特に元マルチメディアネットワーク研究所ネットワーキング研究部ネットワークアーキテクチャ研究グループのみなさま，情報流通プラットフォーム研究所IPネットワーキングプロジェクトのみなさまに感謝いたします．

最後に，忍耐強く論文作成を見守ってくれた家族，朋友，所沢マンドリンクラブのみなさまに感謝の意を表します．

参考文献

[1] Object Management Group, “http://www.omg.org/”.

[2] 通信白書,” http://www.mpt.go.jp/policyreports/japanese/papers/h12/index.html”.

[3] T. Boyd, B. Barr, and Y. Inoue, “The TINA Initiative,” IEEE Communication Magazine., 31, No.3, pp. 70-76, Mar. 1993.

[4] F. Dupuy, G. Nilsson, and Y. Inoue, “The TINA Consortium: Towards Networking Telecommunications Information Services,” Proc. of XV International Switching Symposium (ISS'95), B5.2, pp.207-211, Berlin, Germany, Apr. 1995.

[5] TINA-Consortium, “http://www.tinac.com/” および “http://www.tinac.org/”.

[6] CORBA, “http://www.corba.org/”.

[7] 斉藤孝文, 山本修一郎, 畠中優行, 川崎隆二, “情報流通プラットフォーム構築の取組み,” NTT技術ジャーナル, Vol.11 No.5：通巻122号, pp.8-12, May 1999.

[8] 杉岡寛, 小柳恵一, 安達徹, 千葉常之, “次世代コンピュータネットワークに向けた展開構想,” NTTR&D Vol.47, No.4, pp.411-414, May 1999.

[9] 川原崎雅敏, 斉藤孝文, 小谷野浩, 重田信夫, “次世代コンピュータネットワークのサービス構成法,” NTTR&D Vol.47, No.4, pp.414-417, Apr. 1998.

[10] ITU-T Draft Recommendation M.3400, “TMN management function”.

[11] ITU-T Recommendation M.3010, “Principle for a Telecommunications Management Network”.

[12] “PCサーバーのクラスタリング,” 日経オープンシステム, pp.124-131, Aug. 2000.

[13] H. Nishikawa, K. Asada, and H. Terada: “A decentralized controlled multi-processor system based on the data-driven scheme,” Proc. of 3rd International Conference on Distributed Computing System, pp.639-64, Miami/Ft. Lauderdale, USA, Oct.1982.

[14] H. Nishikawa, H. Terada, K. Komatsu, S. Yoshida, T. Okamoto, Y. Tsuji,S. Takakura, T. Tokura, Y. Nishikawa, S. Hara, and M. Meichi, “Architecture of a one-chip data-driven processor: Q-p,” Proc. of the 16th International Conference on Parallel Processing, pp.319-326, St. Charles, USA, Aug. 1987

[15] H. Nishikawa, H. Terada, S. Komori, K. Shima, T. Okamoto, and S. Miyata, "Architecture of a VLSI-oriented Data-Driven processor: The Q-v1: Advanced topics in data-flow computing," Chapter 9, pp.247-264, Printice-Hall Inc., 1991.

[16] 金倉広志, 宮田宗一, "動的データ駆動型プロセッサによる並列処理方式の検討," 情報処理学会マイクロコンピュータ・アーキテクチャシンポジウム予稿集, pp.9-18, Nov. 1991.

[17] H. Terada, H. Nishikawa, K. Asada, S. Matsumoto, S. Miyata, S. Komori, and K. Shima., "VLSI design of a one-chip data-driven processor: Q-v1," Proc. of 1987 Fall Joint Computer Conference, Dallas, USA, Oct. 1987.

[18] 石井啓之, 西川博昭, 小林秀承, 田中博樹, 井上友二, "TINA環境へのデータ駆動プロセッサ適用の検討," 電子情報通信学会ネットワーキングアーキテクチャワークショップ予稿集 pp. 一般3-3-1 - 一般3-3-8, Dec. 1996.

[19] H. Nishikawa, H. Ishii, and Y. Inoue, "A Stream-Orinented Data-Driven Processor Realizing Hyper-Distributed Systems," Proc. of IASTED International Conference on Parallel and Distributed Computing Systems (PDCS'96), pp.47-51, Chicago, USA, Oct. 1996.

[20] H. Ishii, H. Nishikawa, and Y. Inoue, "TINA-based Multimedia Networking Environment Supported by Data-Driven Processor: CUE-p," Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), pp.423-429, Las Vegas, USA, Jul. 1998.

[21] H. Ishii and H. Nishikawa, "TINA as the basis for supporting multimedia Networking," Anritsu News Vol.17, No. 86, pp.16-21, Oct. 1997.

[22] 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", 電子情報通信学会論文誌B-I, Vol.J80-B-I, No.6, pp.457-464, Jun. 1997.

[23] H. Nishikawa, S. Miyata, S. Yoshida, T. Muramatsu, H. Ishii, H. Kobayashi, and Y. Inoue, "A Data-Driven Implementation of Telecommunication Network Systems," Proc. of 3rd International Symposium on Autonomous Decentralized Systems (ISADS'97), pp.51-58, Berlin, Germany, Apr. 1997.

[24] H. Nishikawa, S. Miyata, S. Yoshida, T. Muramatsu, H. Ishii, Y. Inoue, and K. Kitami, "Data-Driven Implementation of TINA kernel Transport Network," Proc. of TINA'97, pp.184-192, Santiago, Chile, Nov. 1997.

[25] H. Ishii, H. Nishikawa, and Y. Inoue, "Data-Driven Implementation of TINA-based Multimedia Networking Environment," Proc. of International Conference on Communication Technology (ICCT'98), pp.S14-07-1 - S14-070-5, Beijing, China, Oct. 1998.

[26] 長田孝彦, 東海林敏夫, 山下博之, 塩川鎮夫, "マルチメディアコンテンツ転送向け高性能TCP/IP通信ボードの構成と評価," 情報処理学会論文誌, Vol.39, No.2, pp.347-355, Feb. 1998.

[27] H. Ishii, H. Nishikawa, and Y. Inoue, "Data-Driven Implementation of Highly Efficient TCP/IP Handler to Access TINA Network," IEICE Transactions on Communications, Vol.E83-B No.6, pp.1355-1362, Jun. 2000.

[28] 石井啓之, 西川博昭, 井上友二, "分散型ネットワークへの高効率アクセスのデータ駆動型実現法," 電子情報通信学会ネットワーキングアーキテクチャワークショップ予稿集, pp. 一般A-4-1 - 一般A-4-7, Jan. 1998.

[29] 芳田眞一, 紫竹リカルド毅史, 松浦康弘, 村松剛司, 岡本俊弥, 宮田宗一," 映像信号処理向きデータ駆動プロセッサ," 信学会集積回路研究会予稿集, ICD95-158, pp.39-46, Oct. 1995.

[30] H. Nishikawa and K. Aoki, "Data-Driven Implementation of Protocol Handling," Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), pp.430-437, Las Vegas, USA, Jul. 1998.

[31] H. Ishii, H. Nishikawa, and Y. Inoue, "Data-driven Implementation of Highly Efficient Access to the TINA Network," Proc. of 1998 International IEEE Asia Pacific Conference on Communications/Singapore International Conference on Communication Systems (APCC'98/ICCS'98), pp.367-371, Singapore, Nov. 1998.

[32] H. Nishikawa, H. Ishii, R. Kurebayashi, K. Aoki, and N. Komatsu, "Data-driven TCP/IP multi-processor Implementation with Optimized Program Allocation," Proc. of 1998 International IEEE Asia Pacific Conference on Communications/Singapore International Conference on Communication Systems (APCC'98/ICCS'98), pp.786-790, Singapore, Nov. 1998.

[33] K. Aoki, H. Ishii, S. Miyata, and H. Nishikawa, "Super-Integrated Data-Driven Processor for TINA-kTN Protocol Handling," Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), pp. 454-460, Las Vegas, USA, Jul. 1999.

[34] Unicenter TNG, "http://www.caj.co.jp/".

[35] JP1, “http://www.hitachi.co.jp/Prod/comp/soft1/jp1/”.

[36] 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.

[37] H. Tanaka and H. Ishii, ”Service Operation and Management Architecture using Surveillance of application Software Elements (SOMSE),” Proc. of Global Telecommunications Conference (GLOBECOM'95), pp.1997-1981, Singapore, Nov. 1995.

[38] H. Tanaka and H. Ishii, “An Environment for Distribuited Application Management,” Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), pp. 998-1004, Las Vegas, USA, Jul. 1999.

[39] H. Tanaka, H. Kobayashi, and H. Ishii, “Integrated Distributed Application Management under Multi-ORB-vendor Environment,” Proc. of 4th International Symposium on Autonomous Decentralized Systems (ISADS'99), pp.190-197, Tokyo, Japan, Mar. 1999.

[40] 宇野俊夫, “ディスクアレイテクノロジ RAID,” エーアイ出版, 東京, 2000.

[41] H. Ishii, H. Nishikawa, and Y. Inoue, “Data-Driven Fault Management for TINA Applications,” IEICE Transactions on Communications, Vol.E80-B, No.6, pp.907-914, Jun. 1997.

[42] H. Ishii, H. Nishikawa, and H. Tanaka, “Reliable TINA-based Telecommunication Networking Environment,” Proc. of IASTED International Conference on Parallel and Distributed Systems (Euro-PDS'97), pp.17-22, Barcelona, Spain, Jun. 1997.

[43] 石井啓之, 西川博昭, 田中博樹, 井上友二, “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,” 電子情報通信学会論文誌B-I, Vol.J82-B, No.4, pp.522-529, Apr. 1999.

[44] H. Ishii, H. Tanaka, and H. Nishikawa, “Comprehensive Fault Management for Distributed Networking Environment,” Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98), pp. 454-460, Las Vegas, USA, Jul. 1998.

[45] G. Nilsson, F. Dupuy, and M. Chapman, “An Overview of the Telecommunications Information Networking Architecture,” Proc. of TINA'95, pp.1-12, Melbourne, Australia, Feb. 1995.

[46] 北見, “マルチメディアネットワーキングサービス実現へ向けて,” NTT R&D, vol. 44, No.11 ,pp. 991-998, Nov. 1995.

[47] 宮岸, "TINAを適用したマルチメディアネットワーキング技術の展開," NTT R&D, Vol.44, No.11 ,pp.1007-1014, Nov. 1995.

[48] J.B. Dennis, "Dataflow Schemes," Project MAC, pp.187-216, MIT Jul. 1972.

[49] J.R.Gurd, C.Kirkham, and I.Watson, "The Manchester Prototype Dataflow Computer," Commun. ACM, Vol.28, No.1, pp.34-52 Jan. 1985.

[50] Arvind and R.S.Nikhil, "Executing a Program on the MIT Tagged-Token Dataflow Architecture," IEEE, Trans. on Computers, Vol.39, No.3, pp.300-318, Mar. 1990.

[51] Y. Ninomiya, Y. Ohtsuka, Y. Izumi, S. Gohshi, and Y. Iwadate, "An HDTV Broadcasting System Utilizing a Bandwidth Compression Technique-MUSE," IEEE Trans. on Broadcasting, Vol. BC-33, No. 4, Dec. 1987.

[52] Tsuchiya, S. Chikara, F. Sato, and H. Ishii, "An Implementation of TINA Connection Management system for ATM Networks," Proc. of TINA'99, pp.239-248, Oahu, USA, Apr. 1999.

[53] Y. Tsuchiya and S. Chikara, "Operator-Oriented Approach for the Interwork of Service and Network Management," Proc. of TINA'97, pp.144-150, Santiago, Chile, Nov. 1997.

[54] Y. Tsuchiya, S. Chikara, F. Sato, and H. Ishii, "An Implementation of TINA Connection Management system for ATM Networks," IEICE Transactions on Communications, Vol.E82-B, No.11, pp.1780-1792, Nov. 1999.

[55] IEEE1394, "http://www.1394ta.org/".

[56] CORBA Life Cycle Service, version 1.1, "http://www.omg.org/technology/documents/formal/life_cycle_service.htm".

[57] H. Suzuki, H. Kawamura, T. Akiyama, and N. Takahashi, "A method for automatic evaluation of fault effects in the advanced Intelligent Network," Proc. of IEEE Global Telecommunications Conference (GLOBECOM'93), pp. 234-238, Houston, USA, Nov. 1993.

[58] International Telecommunications Union, "http://www.itu.int/ITU-T/index.html".

[59] 石井啓之, 星元雄, "TINA-C仕様の概要とこれまでの成果," NTT R&D, Vol.47, No.7, pp.761-764, Jul. 1998.

[60] ODL Manual Version: 2.3, "http://www.tinac.com/specifications/specifications.htm".

[61] Business Model and Reference Points Version: 4.0, "http://www.tinac.com/specifications/specifications.htm".

[62] P. Farley and R. Westerga, "The Ret reference point definition and prototype implementation," Proc. of TINA'97, pp.193-204, Santiago, Chile, Nov. 1997.

[63] E. Kelly, N. Mercouroff, and P. Graubmann, "TINA-C DPE Architecture and Tools," Proc. of TINA'95, pp.39-54, Melbourne, Australia, Feb. 1995.

[64] Computational Modelling Concepts Version: 3.2, "http://www.tinac.com/specifications/specifications.htm".

[65] Network Resource Architecture Version: 3.0, "http://www.tinac.com/specifications/specifications.htm".

[66] Service Architecture Version: 5.0, "http://www.tinac.com/specifications/specifications.htm".

[67] P. Hellemans, H. Vanderstraeten, P. Lago, J. Yelmo, J. Villamor, and G. Canal, "TINA Service Architecture: From Specification to Implementation," Proc. of TINA'97, pp.174-183, Santiago, Chile, Nov. 1997.

[68] T. Eckardt, P. Kielhoefer, D. Guy, V. Perebaskine, and A. Akram, "A TINA Trial - Interworking Experience with the Legacy Telephone System," Proc. of TINA'97, pp.70-77, Santiago, Chile, Nov. 1997.

[69] 滝田亘, 安竹由起夫, 知加良盛, 宮岸修, "TINA国際デモにおける分散処理環境," 情報ネットワーク通信方式研究会予稿集, IN95-76, pp.37-42, Nov. 1995.

[70] 知加良盛, 中村光宏, 滝田亘, 佐藤文人, "テレコム95TINA国際デモマルチドメインコネクション管理システム," 情報ネットワーク通信方式研究会予稿集, IN95-77, pp.43-48, Nov. 1995.

[71] G. Spinelli, W. Takita, G. Martini, and S. Chikara, "TINA Connection Management implementation over Distributed Processing Platforms," Proc. of NOMS'96, pp.54-63, Kyoto, Japan, Apr. 1996.

[72] ACTS, "http://www.infowin.org/ACTS/actsmap.htm".

[73] REFORM Project, "http://www.infowin/ACTS/RUS/PROJECTS/ac208.htm".

[74] ATM Forum, "http://www.atmforum.com/".

[75] IETF, "http://www.ietf.org/".

[76] TM Forum, "http://www.tmforum.org/".

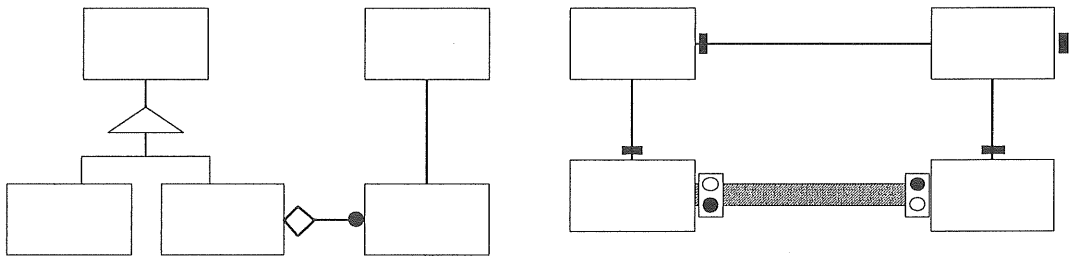
付録

A.1 TINA成果の概要

TINA仕様は後述のTINA-Cにおいて1993年から1997年まで検討が進められた．1998年からはよりTINAの製品化，ビジネス展開に重きをおいて新体制で活動を継続し，2000年で独自仕様化活動を終えた．その後は，協調関係にあるOMGやITU-T(International Telecommunications Union-Telecommunication standardization sector)[58]などの標準化団体で必要な標準化を継続し，TINA-Cの組織を解散し，TINA-ISC (International Scientific Committee)[5]を作り，TINAコンセプトの普及を目的にTINA国際会議を開催していくことになった．筆者はTINA-ISCのメンバーである．以下に，得られた主な成果の概要を示す[59]．

(1)共通的仕様

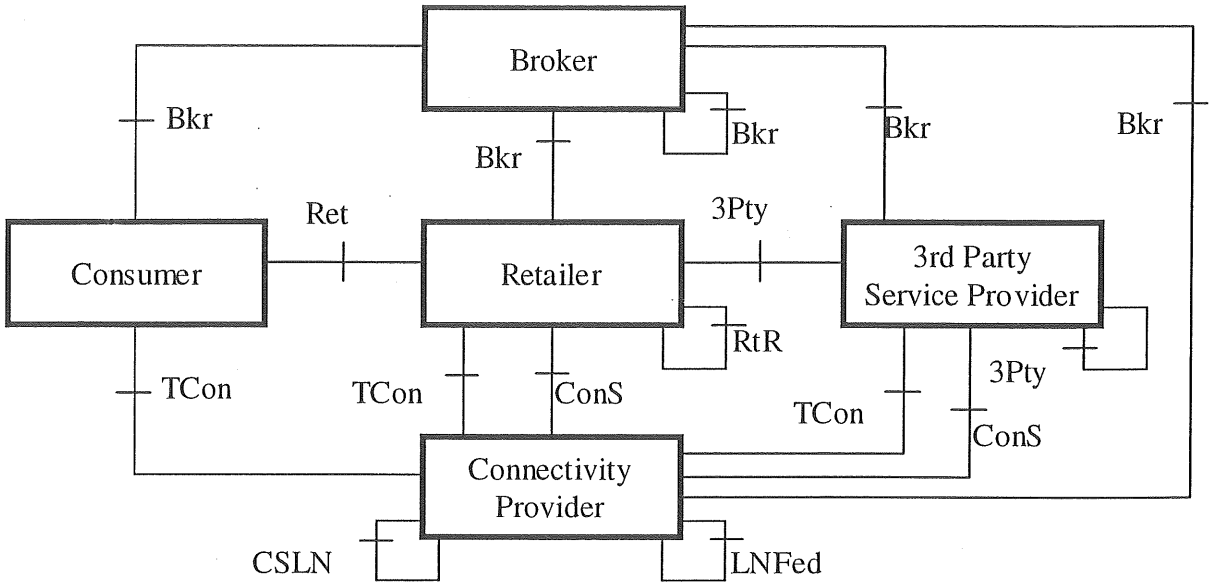
・モデリング手法の確立：ISO/ODPの手法にもとづき，TINAシステムの静的な関係を示す情報モデル，機能間の関係を示すコンピュテーショナルモデル，機能を実際のシステムに配備するエンジニアリングモデルの3モデリング手法の，サービス・マネジメントを統合するTINAシステムへの適用法を明らかにした（図A.1.1）．



図A.1.1 モデリング手法

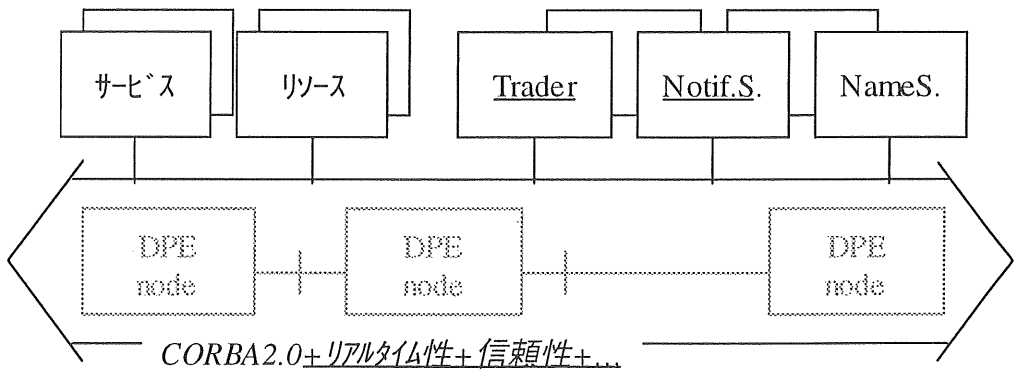
・ODL (Object Definition Language)：TINAオブジェクト間インタフェースを記述するODLの仕様を完成しマニュアルを作成した[60]．

・ビジネスモデル：各種のマルチメディアサービス実現にTINAシステムを適用する時のTINAシステムを運用する時の役割分担の参照モデルとそのインタフェースを定めたビジネスモデルを完成した[61][62](図A.1.2)．



Broker: 各種ｴﾝﾃｨﾃｨの位置情報等を提供する役割。
Consumer: ｻｰﾋﾞｽを最終的に「消費」する役割。個人ﾕｰｻﾞ、大小ﾋﾞｼﾞﾈｽﾕｰｻﾞを含む。
Retailer: Retailer/3rd Party SP/Connectivity Providerによって提供されるｻｰﾋﾞｽを Consumerに「小売り」する役割。
3rd Party Service Provider: 情報転送ｻｰﾋﾞｽを除く各種ｻｰﾋﾞｽを提供する役割。Content Providerを含む。
Connectivity Provider: ﾄﾞﾒｲﾝ間の各種情報（音声・画像・制御など）の転送ｻｰﾋﾞｽを提供する役割。

図A.1.2　ビジネスモデル



TINAシステムを構成する分散オブジェクト間の通信サポート等の機能仕様

図A.1.3　コンピューティングアーキテクチャ

(2)コンピューティングアーキテクチャ

DPE機能仕様（OMGのCORBAを基本仕様としてテレコム特有で必須な機能の拡張）をほぼ完成し，OMGに採用を働きかけた（一部は採用済み)[63][64](図A.1.3).

(3)リソースアーキテクチャ

下位の伝達技術の違いを隠蔽した情報転送機能を提供するための規定であり，コネクションの設定･解放･QoS制御，ネットワークの構成管理などの仕様を完成した[65].

(4) サービスアーキテクチャ

各種サービス（通信系･情報系）のための共通部分の規定であり，ドメインへのコンタクト，サービスの起動･一時停止･終了，パーティの追加･削除，ストリームの追加･一時停止･削除，ネゴシエーション，ユーザモビリティのサポート，ユーザの認証，嗜好の反映など，コネクションを活用し，多様な形態のマルチメディアサービスを実現する，コネクションより上位の概念であるセッションの機能レベルの仕様を完成した[66][67].

A.2 TINA-Cの概要

ここではTINA仕様を標準化してきたTINA-Cについて述べる.

(1) 1993年－1997年のTINA-C

40社を越える，ネットワーク事業者／通信機器ベンダー／コンピュータベンダー／ソフトウェアベンダーで構成されていた．TINA-CにおけるTINA仕様検討は，米国ニュージャージー州のレッドバンクにあるベルコア拠点内に設けた研究環境に,各中心メンバー社が研究員を派遣し,結成されたコアチームによる組織的な研究推進と,各参加メンバー社独自の補助プロジェクトによる研究の補完／強化,加えて国際デモシステム[68]-[71]によるアーキテクチャ検証，により進められた.

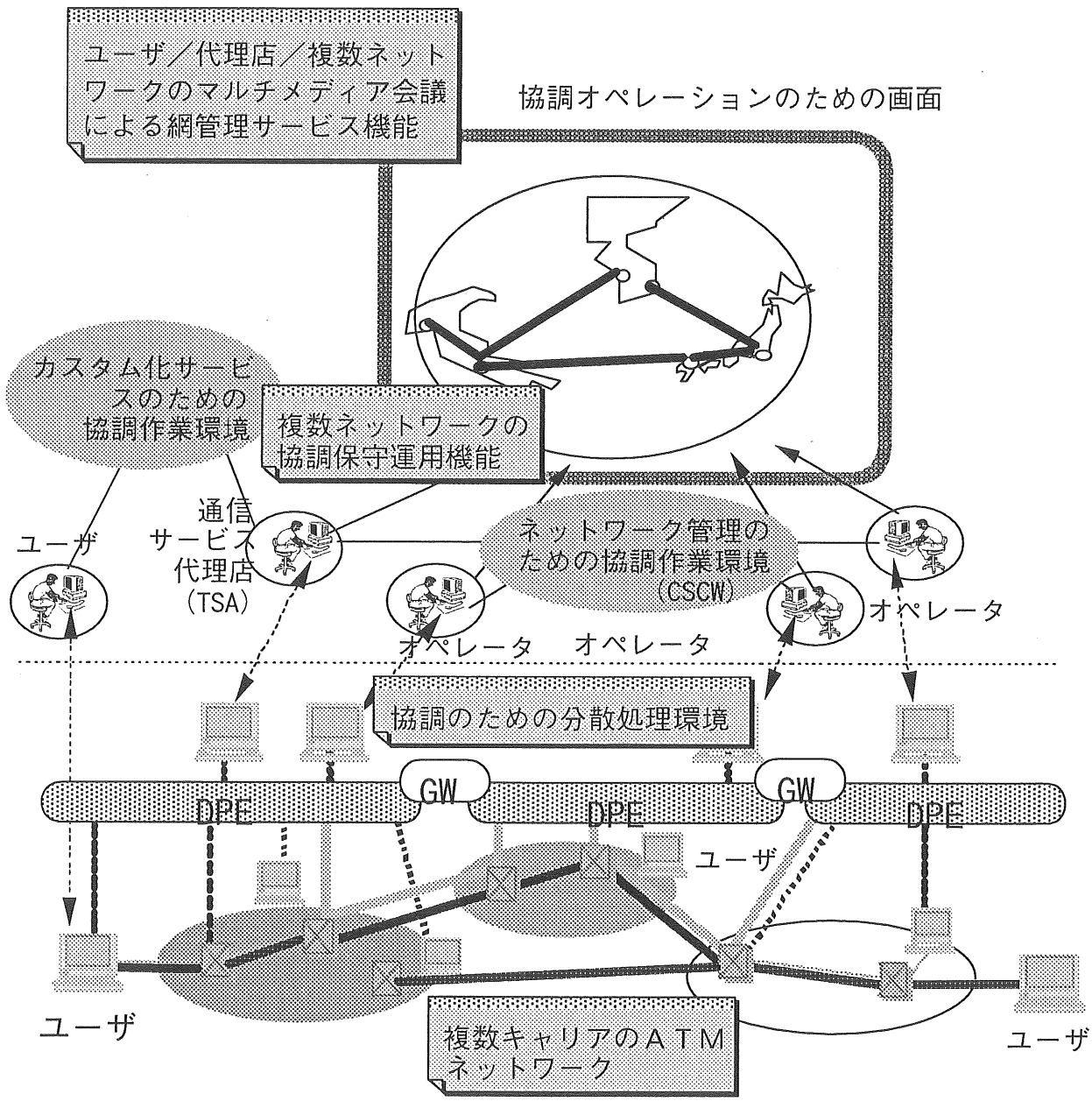
筆者の属するNTTは，コアチームへの研究者派遣，２つの補助プロジェクトの提案,推進,国際デモシステム(TINA-WWD: WorldWide Demonstration)[69]-[71]を通した実証と,いずれの面からも精力的にTINAの検討を進め，筆者もその中心的役割を務めてきた.

NTT,イタリアCSELT，富士通，日本DEC（現日本コンパック)，韓国テレコム，ETRIで検討し，1995年のTelecom95にて展示説明を行ったWWDシステムの構成図と，デモにおける制御画面を，それぞれ図A.2.1，図A.2.2に示す.

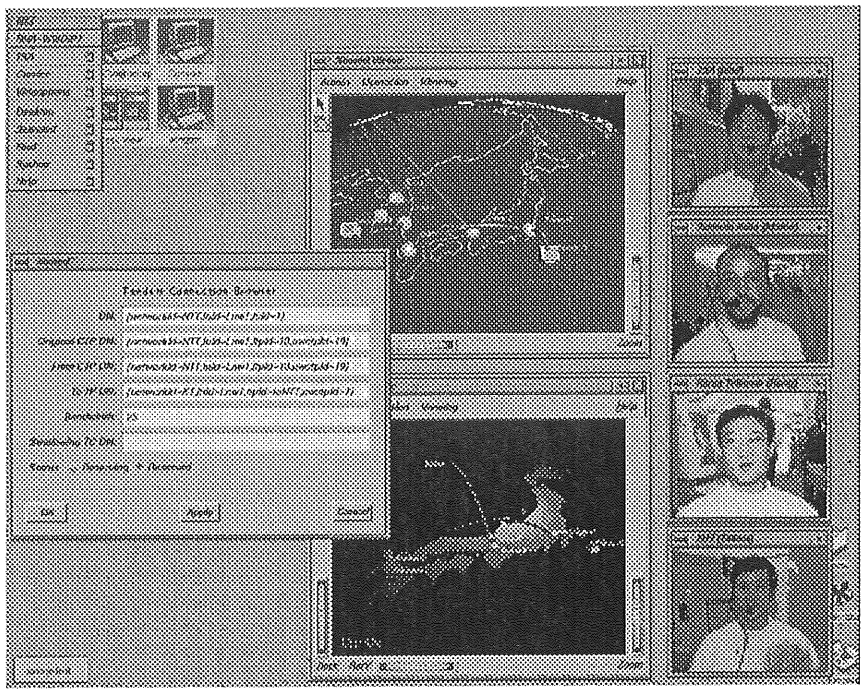
(2) 1998年－2000年のTINA-C

1998年４月１日より,TINA-Cは最高運営代表(CEO)と最高技術代表(CTO)を擁する有限会社として再出発した．第１フェーズで行った国際デモをふまえ、より商用展開を明確にするTTT（The TINA Trial)を遂行した．従来メンバー会社のみに閉じていた技術情報，仕様は一般に開示されホームページから取得することができる[5]．TINA-Cの成果は，NTT，アルカテル，ドイツテレコム，等のTINAメンバ各社の貢献もあり，様々な標準化機関での標準仕様にも反映された．主な標準化機関としては，分散オブジェクト環境とこの上でのサービスの標準化を進めているOMG（Object Management Group)，第３世代移動通信の標準化を進めている3GPP，国際電気通信連合（ITU）傘下の第10研究委

員会（通信ソフトウエア）や第11研究委員会（信号方式），等があげられ，TINA-Cで開発された様々な仕様の反映が進められた．



図A. 2. 1　国際デモシステム構成



図A.2.2　国際デモシステム操作画面

A.3 今後

TINA-WWDを通じて，TINA仕様が机上のものではなく，現実のシステムに適用可能であることが実証され，既に多くの準拠製品を持つCORBA仕様をDPE仕様のベースとし，実用化に向けた技術面でのハードルを低くしたと考えられる．

1995年9月に開始されたACTS（情報通信分野におけるヨーロッパ共同体主催の研究開発プログラム）[72]においても，TINA関連プロジェクトが多数採用された．TINA仕様の近未来の普及を見越して，TINA-C参加の欧州企業が準拠ソフトの開発を加速しつつあり，その際の仲間作りの場として，ACTSの枠組みを利用していたと見ることができる．筆者もACTSプログラムのひとつであるATM障害管理機構の検討であるREFORMに1996年から2000年まで参加し，TINA仕様の実用化の面で検討を行った[73]．

ATMF(ATM-Forum)[74]やIETF(Internet Engineering Task Force)[75] などの伝達技術の標準化，OMGなどのプラットフォーム技術の標準化，TMF(Telecommunication Management Forum)[76]などの網管理という特定分野でのアプリケーションの検討とは異なり，より高位のアプリケーションまでを含むデファクト標準化を行う場としてのTINA-Cの位置づけは重要であったと考えられる．

このように，TINAの利用の環境条件は整いつつある状況下で，TINA-Cは，仕様開発や実験室，プロトタイプレベルのシステム化から，実用ビジネスへの展開を強く指向して

きた．この目標を現実のものとするために，各参加企業は，WG活動を主体として，個々の階層のインタフェース仕様開発のみならず，必要な仕様を垂直統合したシステム化を進めてきた．また多くの関連標準化団体でTINA-C仕様をもとにした標準化が進められている．TINA-Cが解散した現在，明示的にTINA仕様と標榜するしないに関わらず，通信と情報処理が融合したシステムアーキテクチャは現在の常識となり，TINA-Cはその役目を十分に果たしたと総括できる．