多重プロトコル処理の
データ駆動型実現法に関する研究

工学研究科

筑波大学

2002 年 7 月

青木 一浩

内容梗概

本論文は，筆者が筑波大学大学院博士課程工学研究科電子・情報工学専攻在学中に西川研究室において遂行した，通信プロトコル処理の実時間多重処理のデータ駆動型実現法に関する研究をまとめたものであり，次の 5 章をもって構成している．

第 1 章では，序論として，将来の情報社会基盤としてのインフラストラクチャを実現するために，ネットワーク環境，特に通信プロトコル処理における従来研究の問題点を指摘する．次に，本研究の目的と，本研究によって得られた成果とについて概説する．

第 2 章では，実時間性を保証できる多重処理の実現法を明らかにするため，プロトコル TCP/IP (Transmission Control Protocol / Internet Protocol) の実時間多重処理のデータ駆動型実現法を検討している．本実現法では，プロトコル処理に本質的であるヘッダ処理と，データ処理を最大限に同時並行処理する構造を用いて，TCP/IP 処理をデータ駆動プロセッサ上に実現した．その結果，データ駆動プロセッサの持つ多重処理能力を活用すれば，ヘッダ処理がクリティカルパスとなる範囲のデータ長であれば，多重処理時においても，ターンアラウンドタイムがヘッダ処理に要する時間で維持できることを明らかにする．実時間性の保証には，いかなる処理状況下でも処理時間を保証できることが必須であり，本実現法によって得られた成果は，実時間多重処理の実現可能性を示すものである．次に，これらの成果を受けて，プロトコル処理向きデータ駆動プロセッサプロトタイプ CUE-p (Coordinating

Users' requirements and Engineering constraints - prototype) の実現法を検討について述べる．本実現法に用いたデータ駆動プロセッサアーキテクチャを踏襲するという制約のもと，TCP/IP over ATM (Asynchronous Transfer Mode) を実現するために，TCP/IP 処理のデータ駆動型実現法における各処理部のデータ通信量，ならびにボトルネックとなり得る処理の検証を通じて，2 チップのスーパーインテグレーション，ならびにプロトコル処理向き命令の追加により，TCP/IP over ATM が実現できることを明らかにする．

第 3 章では，前章で述べた TCP/IP 処理に加えて，分散オブジェクト環境 CORBA (Common Object Request Broker Architecture) におけるオブジェクト間通信プロトコル GIOP/IIOP (General Inter-ORB Protocol / Internet Inter-ORB Protocol) の実時間多重処理に関する実験的検討について述べる．前章で述べたデータ駆動プロセッサ CUE-p にメモリ処理機能の拡張を行った CUE-v1 (CUE-version1) を用いたデータ駆動プロセッサシステムに実現したプロトコル処理において，マルチメディア通信環境における多重処理性能を評価する．本検討では，マルチメディア通信環境として，動画像，音声，ドキュメント，および制御情報の 4 つのカテゴリにおける帯域，入力間隔を仮定し，それぞれが同時に入力される状況を想定し，ターンアラウンドタイムを最短に維持しながら，それぞれのメディアが多重処理される実験結果が得られた．特に，ヘッダ処理に要する時間内にデータ処理が完了できる動画像，音声，および制御情報に関しては，ヘッダ処理に要する時間として予測された時間で，処理が完了できていることも示す．

第 4 章では，前章までに述べたプロトコル処理のデータ駆動型実現法で得られた成果を実際の通信において活用するために検討した，ネットワーキングインタフェースのデータ駆動型実現法とその実験的検討について述べる．ネットワーキングインタフェースには，ネットワークの持つスループットを最大限に活用することが求められる．本実現法では，ATM，および，Fast Ethernet の持つ実効的な最大スルー

プットを充足するために，CUE-v1 と ATM および Fast Ethernet のコントローラ間のデータ転送を FPGA を用いてボトルネックなく実現した．本章では，このネットワーキングインタフェースのデータ駆動型実現法における構成，ならびにデータ転送処理の実現法を詳述する．転送能力に関する実験的検討では，このネットワーキングインタフェースが ATM の実効的な最大スループットを活用可能であることを示す．さらに，本検討で得られた知見をもとに，ネットワーキング向きデータ駆動プロセッサの実現に向けて，ネットワーキング向きデータ駆動プロセッサの入出力機構，特にネットワークインタフェースを提案する．

第 5 章では，結論として，本研究で得られた成果をまとめる．さらに，今後に残された課題として，より対話的な環境を想定した実験的検討，プロトコル処理層の上位層にあたるアプリケーション層のデータ駆動型実現法，ネットワーキング環境の適用分野の検討，および，携帯情報端末の普及に伴い課題とされつつある低消費電力化を含めた，ネットワーキングインタフェースの実現法について，要点と検討方針を述べる．

関連発表論文

1. Hiroaki Nishikawa and Kazuhiro Aoki, "Data-Driven Implementation of Protocol Handling," Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 430-437 (July 1998).

2. Hiroaki Nishikawa, Hiroshi Ishii, Ryousuke Kurebayashi, Kazuhiro Aoki and Naohisa Komatsu, "Data-Driven TCP/IP Multi-Processor Implementation with Optimized Program Allocation," Proceedings of the International Conference on Communication Systems, pp. 786-790 (Nov. 1998).

3. Kazuhiro Aoki, Hiroshi Ishii, Souichi Miyata and Hiroaki Nishikawa, "Super-Integrated Data-Driven Processor for TINA-kTN Protocol Handling," Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 998-1004 (June 1999).

4. Shinya Kudo, Kazuhiro Aoki and Hiroaki Nishikawa, "Super-Integrated Data-Driven Processor Architecture for Interoperable Networking Environment," Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 2167-2173 (June 2000).

5. Kazuhiro Aoki, Shinya Kudo and Hiroaki Nishikawa, “Data-Driven Protocol-Handling for Interoperable Networking Environment,” Proceedings of the 2001 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 243-249 (June 2001).

6. Hiroaki Nishikawa, Kazuhiro Aoki and Hiroshi Ishii,“Data-Driven Implementation of Efficient Protocol Handlers,” Proceedings of Scuola Superiore Guglielmo Reiss Romoli 2002w Computer & Internet Conference, CD-ROM (Jan. 2002).

7. Kazuhiro Aoki and Hiroaki Nishikawa, “Data-Driven Implementation of Protocol Handling and Networking Interface for Networking Environment,” Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (to be published in June 2002).

8. 西川博昭, 青木一浩, “プロトコル多重処理のデータ駆動型実現法とその実験的検討,” 電子情報通信学会論文誌 D-I J85-D-I 巻, 7 号, (2002 年 7 月掲載予定).

9. 青木一浩, 工藤慎也, 西川博昭, “ボトルネックのないレイヤ 2/3 間インタフェースのデータ駆動型実現法とその実験的検討,” 電子情報通信学会論文誌 B (投稿中).

目 次

内容梗概

関連発表論文

1 序論 **1**

2 プロトコルの実時間多重処理のデータ駆動型実現法 **9**

2.1 緒言 9

2.2 逐次処理プロセッサによる擬似的多重処理の問題点 10

2.3 データ駆動プロセッサアーキテクチャ 11

2.4 データ駆動型プロトコル処理のプログラム構造の検討 14

2.5 プロトコル処理向きデータ駆動プロセッサプロトタイプ 19

2.6 結言 21

3 プロトコルの実時間多重処理に関する実験的検討 **25**

3.1 緒言 25

3.2 マルチメディア通信環境の想定条件 26

3.3 多重プロトコル処理の実験的検討 30

3.4 結言 32

4 ネットワーキング向きデータ駆動プロセッサアーキテクチャ 35

4.1 緒言 35

4.2 ネットワーキングインタフェースのデータ駆動型実現法 36

4.3 転送能力に関する実験的検討 41

4.4 ネットワーキング向きデータ駆動プロセッサの実現に向けて 42

4.5 結言 45

5 結論 47

謝辞 51

参考文献 53

A GIOP/IIOP および TCP/IP 処理の仕様 63

B ネットワーキングインタフェースボードの仕様 73

図目次

2.1 高精度動画像処理向きデータ駆動プロセッサ 12
2.2 DDP および TAM におけるパケット構成 13
2.3 環状エラスティックパイプライン 14
2.4 IP 送信処理における処理構造による性能比較 15
2.5 TCP/IP 処理のデータ駆動型実現法における処理構造 16
2.6 データ駆動図式の構成要素 17
2.7 データ駆動プロセッサプロトタイプ CUE-p 22

3.1 CUE-v1 ボードおよび CUE-v1 のプロセッサ構成 28
3.2 GIOP/IIOP および TCP/IP プロトコル処理のデータ駆動型実現における処理構造 29
3.3 CUE-v1 ボードによるデータ駆動プロセッサシステム 30
3.4 TCP/IP 処理のプログラム配置 31
3.5 GIOP/IIOP 処理のプログラム配置 32
3.6 プロトコル多重処理の評価 34

4.1 データ駆動型ネットワーキングインタフェースボード 37
4.2 CUE-v1 と FPGA 間の送受信処理 39
4.3 FPGA およびデュアルポートメモリにおける転送制御処理 40

4.4 データ駆動型ネットワーキングインタフェースボードにおける送信処理 43
4.5 データ駆動実時間システムの開発支援環境 RESCUE 45

A.1 GIOP 受信処理部の仕様 65
A.2 GIOP 送信処理部の仕様 66
A.3 IIOP 受信処理部の仕様 67
A.4 IIOP 送信処理部の仕様 70
A.5 TCP/IP 処理部の仕様 71

B.1 ATM の送信プロセス概要 75
B.2 ATM の受信プロセス概要 76
B.3 Fast Ethernet の送信プロセス概要 77
B.4 Fast Ethernet の受信プロセス概要 78

表目次

2.1　TCP/UDP/IP 各処理部のターンアラウンドタイム　. 18

3.1　実験環境におけるメディアの想定条件　. 33

第 1 章

序論

次世代の情報社会基盤には，現在の公衆電話網が提供する音声サービスに加えて，複数のメディアによるサービスを同時に提供するために, プロセスを実時間で多重に扱えるシステムが望まれている [1]–[4]．また，携帯電話からのインターネット利用に代表される無線ネットワークと有線ネットワークの相互運用は，ユーザの要求と合致して急速に普及している [5], [6]．したがって，将来の情報社会基盤は，ユーザの多様な要求に応えるために，サービスの追加・変更などの柔軟性を高め，ネットワーク自身の自己発展的な展開が可能なシステムであるネットワーキング環境としての実現が求められている [7], [8]．このような複数のメディアを扱うネットワーキング環境の課題としては，QoS(Quality of Service) の保証，実時間処理の実現，メディア処理の高効率化，携帯端末における低消費電力の実現，ならびにプロトコルに代表されるネットワーク処理の高効率化が主に指摘されている． ここで，プロトコル処理とは，TCP(Transmission Control Protocol)，UDP(User Datagram Protocol)，および，IP(Internet Protocol) に代表される通信プロトコル処理を指している．

QoS の保証に関しては，エージェントやブローカなどの仲介を含めて，スケジュー

リングによって，できるだけ多くのメディアストリーム・データの品質を保証しようという研究が多くなされている [9]–[18]．実時間処理に関しては，ネットワーク自身においても多くの検討がなされているが [19], [20]，特にリアルタイム OS と呼ばれる実時間性の保証機能を持つオペレーティングシステムの研究が多く行われている [21]–[27]．QoS の保証および実時間処理の実現法の研究では，ともにハードウェアとして既存の逐次処理アーキテクチャのみを対象としているため，いかにプロセッサ資源を有効に活用し，かつ各メディアの要求を満足できるスケジューリング法を確立できるかという点に収束する．しかし，QoS や実時間性の保証が問題となるのは，不規則かつ同時に発生しうる要求に対し，逐次処理アーキテクチャを適用していることに起因している部分が大きい．筆者は，メディア，ならびにネットワークの特性を考慮したプロセッサアーキテクチャの適用，およびこれによる実時間多重処理の実現法をまず検討しなければならないと考えている．

メディア処理の高効率化に関しては，汎用プロセッサによる実現法 [28] だけでなく，メディア処理が信号処理の要素を多く持つことから，DSP(Digital Signal Processor) が従来より多く用いられきた [29], [30]．しかし，マルチメディア処理に要求されるスループット・処理時間が，DSP では満足できなくなると，PC クラスタ [31]，ならびに専用チップによってそれを満たしてきた [32]–[38]．近年になって，プログラマビリティを維持すると同時に，DSP の性能を超えるスループット・処理時間を満たすために，専用プロセッサである，メディアプロセッサが開発されている [39]–[41].

メディア処理では，メディアストリームの構成要素に対して画一的な処理を施すと同時に，構成要素間の依存関係は局所的であるため，並列処理が望ましい．また，積和演算を必要とする頻度が高いため，メディアプロセッサの多くは，並列処理の実現法として VLIW (Very Long Instruction Word) を採用している．しかし，メディアストリームのデータ値によって分岐しなければならない処理も多く，命令の複合化が必ずしも効果的であるとは言えない．VLIW と合わせてスーパーパイプ

ライン方式がメディアプロセッサで採用されているように，メディア処理には効率よくパイプライン処理可能な処理方式が望ましいと言える．また，携帯情報端末への発展を考慮すると，近年数多く研究されている低消費電力化も重要な要件である [42]–[45].

さらに，プロトコル処理に代表されるネットワーク処理についても，スループットや応答時間に対する要求が高まるのに伴い，専用ハードウェア，ならびにネットワークプロセッサと呼ばれる専用プロセッサによる実現法が研究されている [46], [47]. これらの研究成果では，データ長が比較的長い場合に高いスループットが得られる反面，短いデータ長では期待するスループットが得られていない．メディアを扱うネットワーキング環境では，ストリーミングなどを考慮すると，比較的短いデータ長の通信が多いと考えられる．したがって，短いデータ長のデータに対するネットワーク処理の効果的な実現法を検討する必要がある．

筆者らは，上述の要求を満足するためには，常にターンアラウンドタイムを維持できる多重処理方式が求められると考え，同時並行・パイプライン・多重処理のすべての並列処理性を自然に実現する，データ駆動プロセッサ CUE (Coordinating Users' requirements and Engineering constraints) によるマルチメディアネットワーキング環境の実現法を研究している [48]．また，ネットワーキング環境の具体例として研究開始当初より取りあげてきた TINA(Telecommunication Information Networking Architecture)[49], [50] では，プロトコル処理層から物理層に至るまでについても，多様な構成を許すことを想定している．筆者は，この TINA 環境のデータ駆動型実現法の検討 [51]–[53] の一環として，プロトコル多重処理のデータ駆動型実現法を検討している [54]–[61].

我々の研究室では，従来より流れ処理概念に基づくデータ駆動プロセッサを用いた，通信・メディア処理の実現法に関する研究を続けてきた．流れ処理概念は，データ駆動実行に必要なデータ転送・処理・記憶機能を自己同期型エラスティックパイ

プラインとして構成可能としている．データ駆動方式は，定常的なデータ流量が維持されている限り，処理遅延はほとんど問題にならないというパイプライン方式の利点を極限まで活用できる特徴を持っている．また，データ駆動発火規則によって，常に前進的な処理だけが実行に移され，フラッシングを必要としないことが保証されている．さらに，動的データ駆動方式を採用すれば，タグ付きトークン列としてストリームが並列処理可能な形式で実現されるため，スタベーション回避についても優れた特性を示す．したがって，メディアプロセッサをはじめとするメディア処理の効果的な実現法における要件として挙げているパイプライン処理に適した処理方式をもつプロセッサであると考えられる．

本研究は，プロトコル処理を具体例に，メディアに代表される短いデータ長を通信する環境において，すべてのデータの実時間性を保証する多重処理の実現法を明らかにすることを目的としている．QoS の保証や実時間処理の要件に挙げた，マルチメディアネットワーキング環境に適したプロセッサアーキテクチャに関しては，多重処理における文脈切換を駆動原理として持つ，我々のデータ駆動プロセッサを適用する．実時間性の保証には，多重処理時においてもターンアラウンドタイムを維持できる，さらには，そのターンアラウンドタイムを予測できる必要がある．なぜなら，実時間処理における時間制約の遵守を妨げる要因は，ターンアラウンドタイムの延びだからである．したがって，多重度によってターンアラウンドタイムが変化する方式を持ち込むのは本質的ではない．また，扱うデータ長の範囲において，データ長に関わらずターンアラウンドタイムが維持できる処理の実現法が必要となる．このとき，プロトコル処理では，ヘッダ処理はプロトコルの種類に関わらず本質的である．本研究では，メディアデータを扱う限り，プロトコル処理のターンアラウンドタイムをヘッダ処理に要する時間で維持する実現法を明らかにする．

さらに，本研究において実現するプロトコル処理をネットワークの持つスループットが十分に活用できる環境に適用するために，より下位層のネットワークイン

タフェースの実現法を実験的に検討する．上述したネットワークプロセッサに代表されるネットワーク処理の高効率化が検討されはじめたのは，元来外部からの入力および出力はプロセッサに比べて遅いとされ，既存の入出力機構を想定した入出力処理用いてきたことに起因している．ネットワークの高速化・広帯域化が進んだ結果，入出力機構，ならびに入出力処理がボトルネックとなっている．本研究では，既存のネットワークコントローラと，データ駆動型実現されたプロトコル処理との間で，ボトルネックのない入出力機構，ならびに入出力処理の実現法を実証する．

以下第 2 章では，実時間性を保証できる多重処理の実現法を明らかにするため，プロトコル TCP/IP (Transmission Control Protocol / Internet Protocol) の実時間多重処理のデータ駆動型実現法を検討している．本実現法では，プロトコル処理に本質的であるヘッダ処理と，データ処理を最大限に同時並行処理する構造を用いて，TCP/IP 処理をデータ駆動プロセッサ上に実現した．その結果，データ駆動プロセッサの持つ多重処理能力を活用すれば，ヘッダ処理がクリティカルパスとなる範囲のデータ長であれば，多重処理時においても，ターンアラウンドタイムがヘッダ処理に要する時間で維持できることを明らかにする．実時間性の保証には，いかなる処理状況下でも処理時間を保証できることが必須であり，本実現法によって得られた成果は，実時間多重処理の実現可能性を示すものである．次に，これらの成果を受けて，プロトコル処理向きデータ駆動プロセッサプロトタイプ CUE-p (Coordinating Users' requirements and Engineering constraints - prototype) の実現法を検討について述べる．本実現法に用いたデータ駆動プロセッサアーキテクチャを踏襲するという制約のもと，TCP/IP over ATM (Asynchronous Transfer Mode) を実現するために，TCP/IP 処理のデータ駆動型実現法における各処理部のデータ通信量，ならびにボトルネックとなり得る処理の検証を通じて，2 チップのスーパーインテグレーション，ならびにプロトコル処理向き命令の追加により，TCP/IP over ATM が実現できることを明らかにする．

第3章では，前章で述べたTCP/IP処理に加えて，分散オブジェクト環境CORBA (Common Object Request Broker Architecture)[62], [63] におけるオブジェクト間通信プロトコル GIOP/IIOP (General Inter-ORB Protocol / Internet Inter-ORB Protocol) の実時間多重処理に関する実験的検討について述べる．前章で述べたデータ駆動プロセッサ CUE-p にメモリ処理機能の拡張を行った CUE-v1 (CUE-version1) を用いたデータ駆動プロセッサシステムに実現したプロトコル処理において，マルチメディア通信環境における多重処理性能を評価する．本検討では，マルチメディア通信環境として，動画像，音声，ドキュメント，および制御情報の4つのカテゴリにおける帯域，入力間隔を仮定し，それぞれが同時に入力される状況を想定し，ターンアラウンドタイムを最短に維持しながら，それぞれのメディアが多重処理される実験結果が得られた．特に，ヘッダ処理に要する時間内にデータ処理が完了できる動画像，音声，および制御情報に関しては，ヘッダ処理に要する時間として予測された時間で，処理が完了できていることも示す．

第4章では，前章までに述べたプロトコル処理のデータ駆動型実現法で得られた成果を実際の通信において活用するために検討した，ネットワーキングインタフェースのデータ駆動型実現法とその実験的検討について述べる．ネットワーキングインタフェースには，ネットワークの持つスループットを最大限に活用することが求められる．本実現法では，ATM，および，Fast Ethernet の持つ実効的な最大スループットを充足するために，CUE-v1 と ATM および Fast Ethernet のコントローラ間のデータ転送を FPGA を用いてボトルネックなく実現した．本章では，このネットワーキングインタフェースのデータ駆動型実現法における構成，ならびにデータ転送処理の実現法を詳述する．転送能力に関する実験的検討では，このネットワーキングインタフェースが ATM の実効的な最大スループットを活用可能であることを示す．さらに，本検討で得られた知見をもとに，ネットワーキング向きデータ駆動プロセッサの実現に向けて，ネットワーキング向きデータ駆動プロセッサの入出

力機構，特にネットワークインタフェースを提案する．

第 5 章では，結論として，本研究で得られた成果をまとめる．さらに，今後に残された課題として，より対話的な環境を想定した実験的検討，プロトコル処理層の上位層にあたるアプリケーション層のデータ駆動型実現法，ネットワーキング環境の適用分野の検討，および，携帯情報端末の普及に伴い課題とされつつある低消費電力化を含めた，ネットワーキングインタフェースの実現法について，要点と検討方針を述べる．

第 2 章

プロトコルの実時間多重処理のデータ駆動型実現法

2.1 緒言

本章では，通信プロトコル TCP/IP の実時間多重処理のデータ駆動型実現法について詳述する．実時間多重処理の実現において，逐次処理プロセッサによる実現法では，擬似的多重処理によって生じる文脈切替のオーバヘッドが実時間性を保証するための妨げとなる．したがって，本研究は，駆動原理が多重処理を内在しているデータ駆動プロセッサによる実現法を検討している．本章では，逐次処理プロセッサによる擬似的多重処理の問題点を指摘する．次に，本実現法に用いたデータ駆動プロセッサアーキテクチャについて述べる．このデータ駆動プロセッサによるプロトコル TCP/IP 処理の実現法として，ヘッダ処理とデータ処理の並列化により，ヘッダ処理がクリティカルパスとなるデータ長では，ターンアラウンドタイムがヘッダ処理に要する時間で維持されることを示す．さらに，本実現法をもとに検討した，プロトコル処理向きデータ駆動プロセッサプロトタイプの実現法について述べる．

2.2 逐次処理プロセッサによる擬似的多重処理の問題点

筆者らは，本研究の基礎的検討として，逐次処理プロセッサ上に実現されたTCP/IPの多重処理を実験的に検証した．実験では，構成として2台のワークステーションの一方をサーバ，もう一方をクライアントとし，これをEthernetで接続した．また，測定は次のように行った．まず，クライアント側でスレッドを生成する．各スレッドはそれぞれTCP/IPコネクションを設定し，サーバに多重に処理を要求する．サーバでは各クライアントスレッドからの要求が到着するごとにスレッドを生成する．このクライアントスレッドの数を増加させたときの，サーバ側での処理時間，および，TCP/IPによるデータ転送時間を測定した結果，逐次処理プロセッサでは，文脈切り換えのオーバヘッドのため，多重度の増加に従い，プロセス当たりの処理時間が増大することが明らかとなっている[53]．さらに，各プロセスの処理を高速化して，満足できる時間制約の上限を広げるために，ハードウェアによるTCP/IP処理の実現が報告されている[46], [47]．例えば，文献[47]のTCP/IP通信ボードの評価では，IPデータグラム長が長くなるにつれてスループットが向上し，IPデータグラム長が6kバイト以上と，比較的長い場合に目標のスループットが得られている．逆に，IPデータグラム長が短かくなるとスループットは減少し，2kバイト以下の場合には，既存のTCP/IP処理と比較してもスループットの向上は得られていない．IPデータグラム長が長くなるにつれてスループットが向上する原因は，TCP/IP処理では，TCPにおけるチェックサムを除き，ヘッダの処理が主であり，データグラム長が長くなれば，相対的にヘッダの割合が減少するからである．しかし，目標とするメディア通信において，メディアストリームをデータとするデータグラムは，例えばMPEG2における通信時に用いられるトランスポートストリームパケットが188バイトであるように，150〜200バイト程度の長さであることが予想される[64]．故に，実時間多重処理が要求されるプロトコル処理の実現には，実行時のオーバヘッ

ドのない多重処理が実現できるプロセッサアーキテクチャを用いるべきであると考えられる．

2.3 データ駆動プロセッサアーキテクチャ

本研究において，プロトコル処理のデータ駆動型実現法は，図 2.1に示すデータ駆動プロセッサを用いて検討した．このデータ駆動プロセッサは，あらゆるメディア処理の中で，最もスループットを要求する，高精度動画像の実時間処理を既に実現している．ネットワーキング環境の構築には，他のメディア処理とともに，通信に必須となるプロトコル処理も，本データ駆動プロセッサに基づく検討が不可欠であると考え，図に示す DDP(Data-Driven Processor) および TAM (Tag Addressable Memory) による実現法を検討した．図 2.2は，DDP および TAM におけるパケット構成を示している．DDP および TAM において，トークンは，命令実行に必要な情報を全て含んだ，パケットという形で実現されている．パケットは，2 ワード (1 ワード 32 ビット) で構成されており，12 ビット× 2 のオペランドデータ (図中 データ 0，データ 1)，24 ビットの世代と呼ばれる文脈識別子 (図中 世代)，ならびに 13 ビットの行き先プロセッサ・ノード情報 (図中 PE 番号, エントリ番号) を持つ．ここで，PE(Processing Element) とは DDP および TAM における処理要素の総称である．パケット中の PE 番号は，マルチチップ構成をとった場合でも，どのチップのどのプロセッサ (PE) かを指定可能としている．また，ノード情報は，パケットが存在する位置に応じて，入力先を示すエントリ番号，もしくは，ノード番号となる．パケットの持つ情報のうち，特にアプリケーションの仕様に影響するオペランドデータおよび世代は，上述した高精度動画像処理において，それぞれ色情報，位置情報を扱うのに必要十分なビット幅を確保している．

DDP および TAM に集積されている各処理要素は，異なる命令セットを持つ．

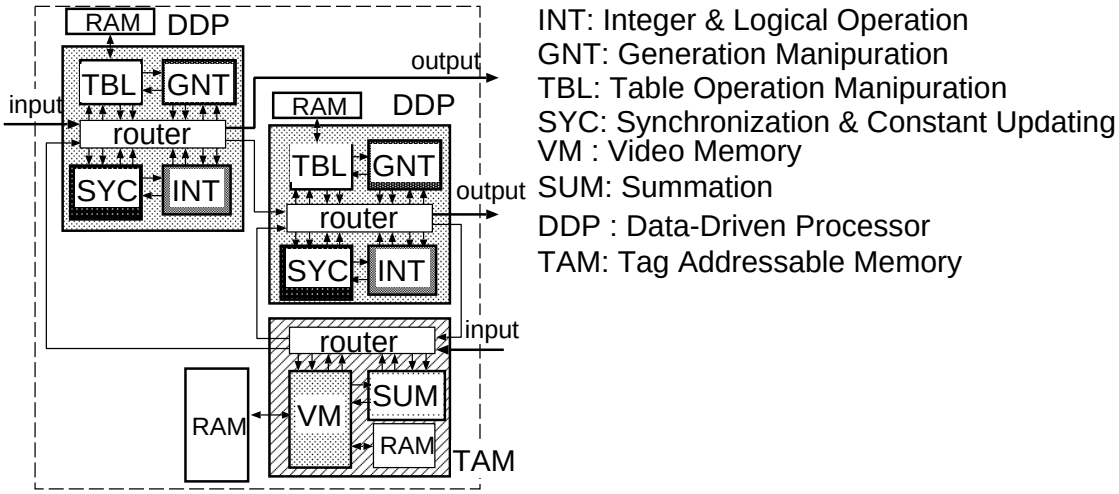

図 2.1: 高精度動画像処理向きデータ駆動プロセッサ

これは，1 個の処理要素に割り当てられるハードウェア量を斉一な命令セットを持たせた場合より少なくし，チップ面積を有効利用するためである．DDP における各処理要素の処理内容は，INT(INTeger & logical operation) がオペランドデータの処理，GNT(GeNeraTion manipulation) が世代の処理，TBL (TaBLe operation manipulation) が，動画像処理における逆非線型変換処理を実現する履歴依存処理，SYC(SYnchronization & Constant updating) が，並列性向上のために，2 つのオペランドデータを 1 パケット化する同期処理のための命令セットをそれぞれ保持している．なお，メモリ処理に特化したデータ駆動プロセッサ TAM では，VM (Video Memory) が履歴依存処理を，SUM(SUMmation) が積和演算処理を行う．TAM では，メモリを世代によってアクセスすることにより，異なる世代間での履歴依存処理の多重処理を実現している．

図 2.3は，DDP および TAM を構成する PE の機能要素，ならびに各機能要素を形成するエラスティックパイプラインの構造を示している．図 2.3 の左下は，図 2.1に示した DDP の構成である．DDP を構成する PE は，図 2.3の左上に示すよう

図 2.2: DDP および TAM におけるパケット構成

に，発火制御部 FC (Firing Control), 関数処理部 FP (Funcltional Processor), プログラム記憶部 PS (Program Storage) といった機能要素を環状に接続した環状パイプラインによって構成されている．これは，TAM においても同様である．さらに，これらの機能要素，およびルータ (図中 router) は，すべて自律分散制御型のエラスティックパイプラインにより実現されている．このエラスティックパイプラインはステージ間のハンドシェークによって動作する．図 2.3の右に，エラスティックパイプラインにおけるステージ間のハンドシェーク機構を示す．パイプラインの形成するデータラッチ間のパケットの移動は，局所的なクロックを生成する C 素子間における，次のステージへの移動要求である FWD 信号と当該ステージが移動可能状態であることを示す ACK 信号の送受 (ハンドシェーク) によって実現されている．ハンドシェークはプロセッサ，ならびにチップをまたいだステージ間でも行われるため，付加的なハードウェアを用いることなく，マルチプロセッサシステムが容易に実現可能である．この特性を利用して，DDP，および TAM ではマルチプロセッサの 1 チップ集積を行っている．

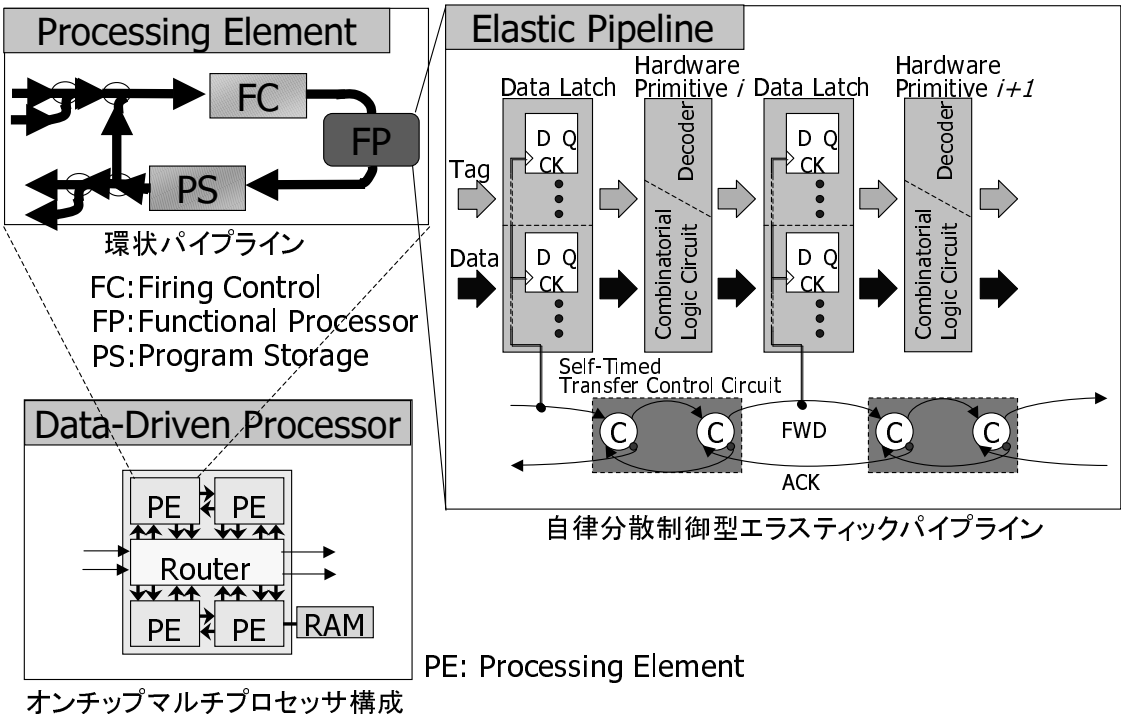

図 2.3: 環状エラスティックパイプライン

2.4 データ駆動型プロトコル処理のプログラム構造の検討

TCP/IP 処理は，逐次処理プロセッサアーキテクチャの影響を受け，メモリを介した逐次的な処理構造で実現されてきた．前節で述べたとおり，我々のデータ駆動プロセッサは，環状パイプラインで構成されているため，逐次的な処理は，スループットの低下とターンアラウンドタイムの増加を招く要因となる．図 2.4に示すように，IP 送信処理の一部について，実際に逐次的な処理構造で実現した結果，並列な処理構造と比較して，ターンアラウンドタイムが延びると同時に，スループットも低下することが明らかになっている [52]．本節では，CUE プロセッサの持つ細粒度

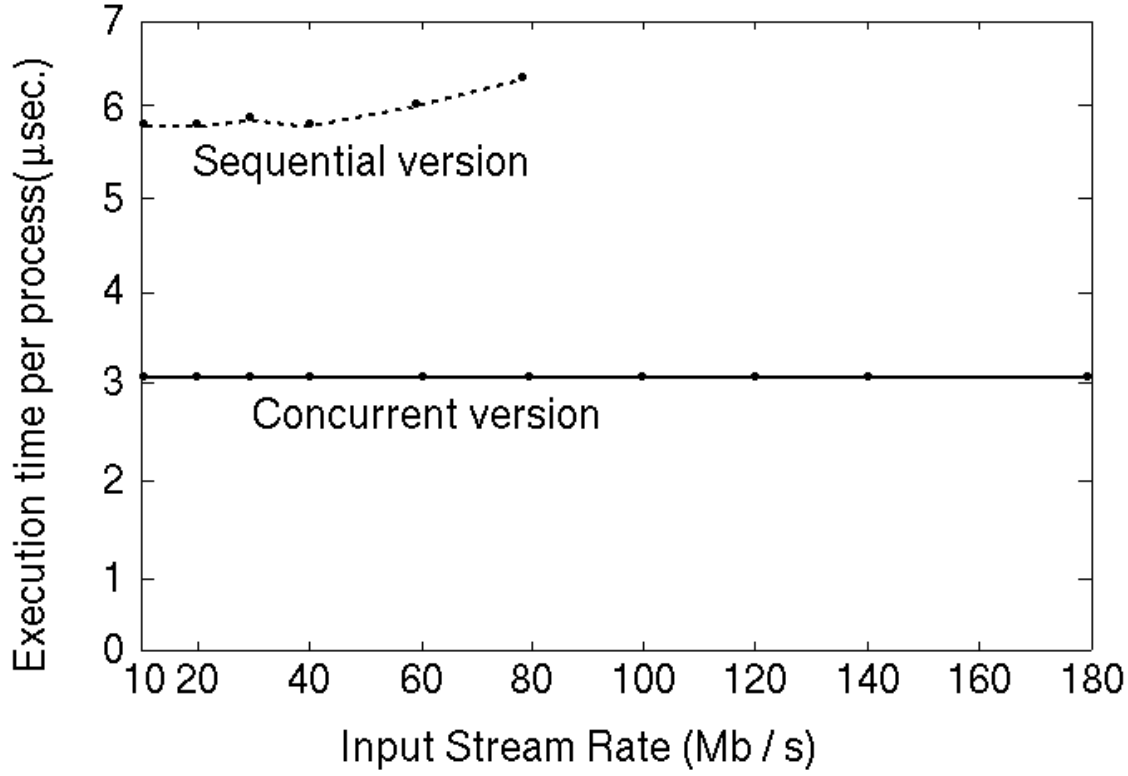

図 2.4: IP 送信処理における処理構造による性能比較

並列処理性を最大限に活用するために，IP データグラムをトークン列として扱い，できる限り並列な処理構造で実現した，TCP/IP 処理のデータ駆動型実現法について議論する． 図 2.5は，本実現法における TCP/IP 処理のプログラム構造を表す仕様である．これは我々の研究室で検討・実現している，データ駆動プロセッサ CUE のための実時間実行システム RESCUE(Realtime Execution System for CUE series data-driven processors)[65]–[67] による仕様記述の画面を切り取ったものである．ここで，図 2.5の仕様は，図 2.6に示すように，仕様記述あるいは部分仕様を記述するためのブロック（図 2.6(a)），機能要素を示すノード（図 2.6(b1)），ノードの入出力を示すポート，外部との入出力を示すソース（図 2.6(b2)）・シンク（図 2.6(b3)），

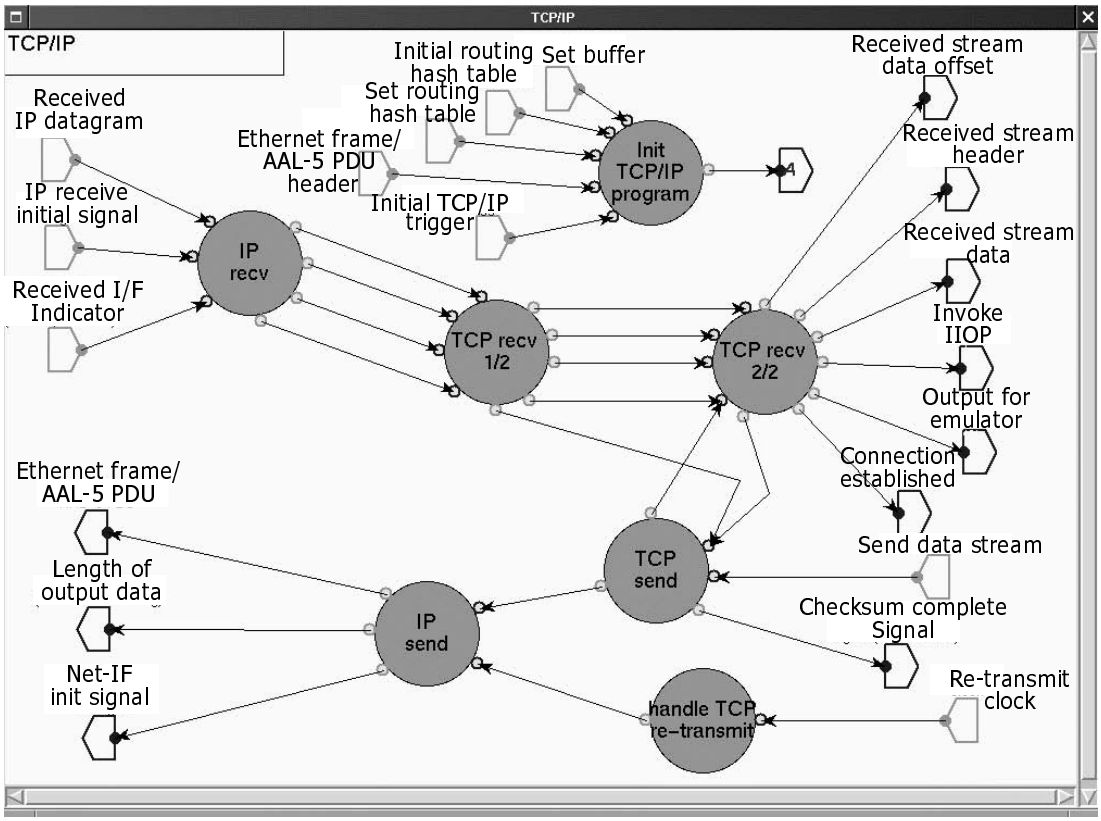

図 2.5: TCP/IP 処理のデータ駆動型実現法における処理構造

データ依存性を示すアーク（図 2.6(c)），およびそれらの名前からなるデータ駆動図式によって記述されている．

図 2.5は，ネットワークから IP データグラムを受け取り，アプリケーション層にデータを送る受信処理 (図中 IP recv, TCP recv 1/2, TCP recv 2/2)，ならびに，アプリケーション層より送られてきたデータおよび受信した TCP セグメントに対する返答をネットワークに送る送信処理 (図中 TCP send, IP send) の流れを示している．なお，図中のノード “init TCP/IP program” はルーティングテーブル (図中 routing hash table) やバッファ (図中 buffer) の設定を行う初期化処理である．この他にも，再送処理 (図中 handle TCP re-transmit) に代表される例外処理が TCP/IP

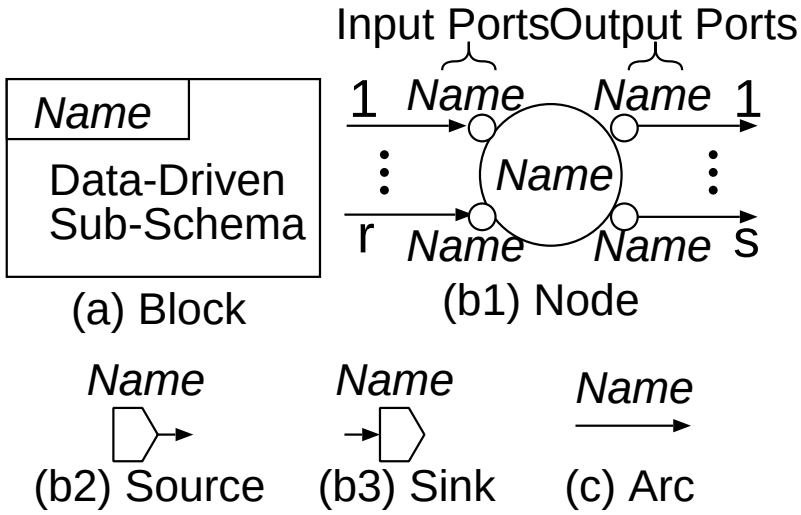

図 2.6: データ駆動図式の構成要素

には存在するが，ネットワーク障害などによる外部要因によってタイムアウト，再送などが起こった場合には，プロトコル処理自体によって実時間性を保証するのは難しい．本実現法ではこれらの処理についても実現しているが，実時間多重処理に関しては，上述した受信，ならびに送信処理を中心に検討を行った．

ネットワークより送られてくる IP データグラム (図中 Received IP datagram) は，ネットワークインタフェースからの受信信号 (図中 IP receive initial signal) およびインタフェース識別子 (図中 received I/F indicator) とともに IP 受信部 (図中 IP recv) に入力される．このとき，IP データグラムは，データ駆動プロセッサに入力される際に 1 バイト毎のパケット列として入力される．このパケット列の順序は世代によって保持される．IP 受信部では，ヘッダ検査が行われると同時に，データ部のバッファリングが行われる．IP 受信部からは，格納したデータ部のアドレス，ヘッダの IP アドレス情報，データ長，チェックサム結果がそれぞれ並列に出力される．それらを入力とする TCP 受信ヘッダ検査部 (図中 TCP recv 1/2) では，TCP ヘッ

表 2.1: TCP/UDP/IP 各処理部のターンアラウンドタイム

処理部	IP recv	TCP recv (1/2,2/2)	TCP send	IP send
≤251 バイト	18μsec.	31μsec.	25μsec.	14μsec.
1.4k バイト	18μsec.	40μsec.	33μsec.	14μsec.
処理部	IP recv	UDP recv	UDP send	IP send
≤1.4k バイト	18μsec.	12μsec.	10μsec.	14μsec.

ダ検査が行われる．ヘッダ検査で異常がなければ，返信に必要なヘッダ情報を TCP 送信部に送る．同時に，データ長やコネクション管理における状態フラグが TCP 受信出力部 (図中 TCP recv 2/2) に送られる．この仕様では，上位層に CORBA のオブジェクト間通信プロトコルである GIOP/IIOP が想定されているので，それらのヘッダ，データ，オフセットおよび起動トリガが出力される．TCP 送信部 (図中 TCP send) では，送るべきデータをバッファリングすると同時に，TCP ヘッダならびに疑似ヘッダを生成し，IP 送信部 (図中 IP send) に出力する．IP 送信部では，疑似ヘッダをもとに IP ヘッダを生成した後，IP データグラムであるフレームまたは PDU(Protocol Data Unit)(図中 Ethernet frame/AAL-5 PDU)，データ長，ネットワークインタフェースへのトリガを出力する．TCP/IP の送受信処理におけるヘッダ生成・検査は，チェックサムを除きほぼ独立に処理できるので，並列化して実現している．また，チェックサムを除いたヘッダ生成・検査はデータのバッファリングとも並列に処理可能である．したがって，データのバッファリング，ならびに TCP チェックサムがヘッダ生成・検査より短い時間で処理できる間は，ターンアラウンドタイムはヘッダ生成・検査処理に要する時間で一定となる．

表 2.1は，RESCUE を用いたターンアラウンドタイムの評価である．この評価

は，第 3 章で後述する実験的検討と比較できるよう，第 2.4 節で述べる CUE-p を対象プロセッサとしている．表に示すように，TCP/IP ヘッダを含めて約 250 バイトまでのデータ長では，ターンアラウンドタイムが維持できている．このとき，例えば時間制約の厳しい動画像では， MPEG2 のストリーミング用のトランスポートストリームパケットの長さが 1 パケット 188 バイトである [64]．音声のデータ長はより短かくなることが容易に予測できるため [68]，メディアを扱うためには十分な長さであると言える．データ長が 250 バイトより長くなるとデータ処理に要する時間がヘッダ生成・検査に要する時間より長くなる．即ち，データ長に依存してターンアラウンドタイムは線形に延び，表に示す 1.4k バイトのデータ長では，TCP 処理のターンアラウンドタイムの延びが顕著に現れる．同じく表 2.1に，データ駆動プロセッサ上に実現した UDP/IP の評価結果を示す．評価に用いた UDP では，TCP におけるチェックサム処理の影響を検証するため，チェックサムを無効化している．このとき，表 2.1に示すターンアラウンドタイムは約 1.4k バイトのデータにおいても変化しないという評価結果が得られた．したがって，データ処理に要する時間を延ばす要因は，チェックサムにある．また，データ受信時のバッファリングにも当然時間を要するが，IP 処理に要する時間は上位のプロトコルに関わらず同じであり，UDP においてもヘッダ処理に時間を要するためターンアラウンドタイムを維持できるデータ長が長くなると考えられる．

2.5 プロトコル処理向きデータ駆動プロセッサプロトタイプ

TCP/IP 処理のデータ駆動型実現法では，8 個の DDP と 4 個の TAM を使用した．TCP/IP 処理を TCP 送信，TCP 受信，IP 送信および IP 受信と大きく 4 つの処理部に分類したとき，各処理部は，図 2.1 に示した DDP2 個，TAM1 個で実現され

ている．このとき，各処理部内のチップ間通信がボトルネックとなり，最大スループットが約 80Mbit/sec に制限されることが明らかとなった．各処理部間は，データ部はメモリを介して渡すので，基本的にそのメモリアドレスとヘッダ生成・検査の完了通知を通信するのみであるのに対し，各処理部内では，ヘッダ部の各要素，およびデータ部がすべて通信対象となるためである．検討開始当初，本検討の成果に基づきデータ駆動プロセッサプロトタイプ CUE-p を実現する際には，TCP/IP over ATM が実現できることを目標としていた．また，CUE-p は，DDP のデザインルール 0.6μm に対し，0.35μm であるため，DDP2 個，あるいは，DDP1 個と TAM1 個を 1 チップに集積できることが分かっていた．さらに，開発期間の関係から，DDP，TAM のパイプライン構成は変えずに，単純な相互結合による 1 チップ集積を目指すこととした．

これらの制約の下で目標である ATM の実効的な最大スループット 135Mbit/sec を満足するために，本検討ではまず，各処理部内のどの通信路がボトルネックとなるかを検証した．このとき，2 個の DDP 間の通信が約 80Mbit/sec であり，DDP と TAM 間の通信においては，約 140Mbit/sec のスループットが保持できることが明らかとなった．TAM は，メモリ容量を必要とするバッファリング，ならびにコネクション管理における状態の保持に用いているため，DDP 内の TBL にてチェックサムを行っている．また，プログラムメモリの容量の制約から，1 個の DDP 内でチェックサムを実現できないため，ヘッダ検査・構成のための通信に加えて，チェックサムに必要な要素のパケットがすべて通信される．したがって，2 個の DDP 間の通信がボトルネックとなっていた．一方，DDP と TAM 間は，バッファリング，ならびにバッファリングしたデータの読み出し以外は，ヘッダ検査・生成時の通信がないため，目標に対するボトルネックとはならないことが分かった．したがって，2 個の DDP を 1 チップに集積することを決定した．

次に，DDP 間のチップ間通信を解消することによって，チェックサム計算にお

けるスループットを向上する必要があった．上述のとおり，チェックサム計算にはTBL メモリを用いるため，メモリアクセス遅延が生じる．評価の結果，チェックサム計算のスループットは約 120Mbit/sec であることが明らかとなった．このとき，チェックサム計算における累積加算時のメモリアクセス遅延を極小化すれば，目標のスループットは十分に達成できることから，CUE-p は，チェックサム計算を，メモリを用いずに実現するための累算器を設けた．図 2.7(a) および (b) は，CUE-p を 4 個，TAM を 2 個搭載した CUE-p ボード，ならびにその構成を示している．CUE-p ボードは，ネットワークインタフェースカードと共に用いることによって，CUE-p をネットワーキング環境にて活用することを想定して実現されている．同時に，マルチメディア通信の実現を想定し，IEEE1394 インタフェースも搭載している．CUE-p の構成を図 2.7(c) に示す．上述のとおり，CUE-p は DDP2 個にあたる 8 個のプロセッサが 1 チップに集積されている．さらに，累算器を持つプロセッサ MUL(MULtiplex operation manipulation) を DDP における SYC と置換した．加えて，上記の累算器によるチェックサム計算命令をはじめとするプロトコル処理向きの命令を追加した．

CUE-p による TCP/IP 処理のスループットの評価では，ATM の実効的な最大スループットを満足する，142Mbit/sec に達した．本検討では，ボトルネックを適確に追求し，それらの解消のために限られた条件下でのオンチップマルチプロセッサ化，および命令セットの適正化を活用すれば，目標のスループットを達成するための効果が得られることを示した．

2.6 結言

本章では，プロトコル TCP/IP の実時間多重処理のデータ駆動型実現法について述べた．まず，実時間多重処理の実現において，逐次処理プロセッサによる実現法では，擬似的多重処理によって生じる文脈切替のオーバヘッドが実時間性を保証

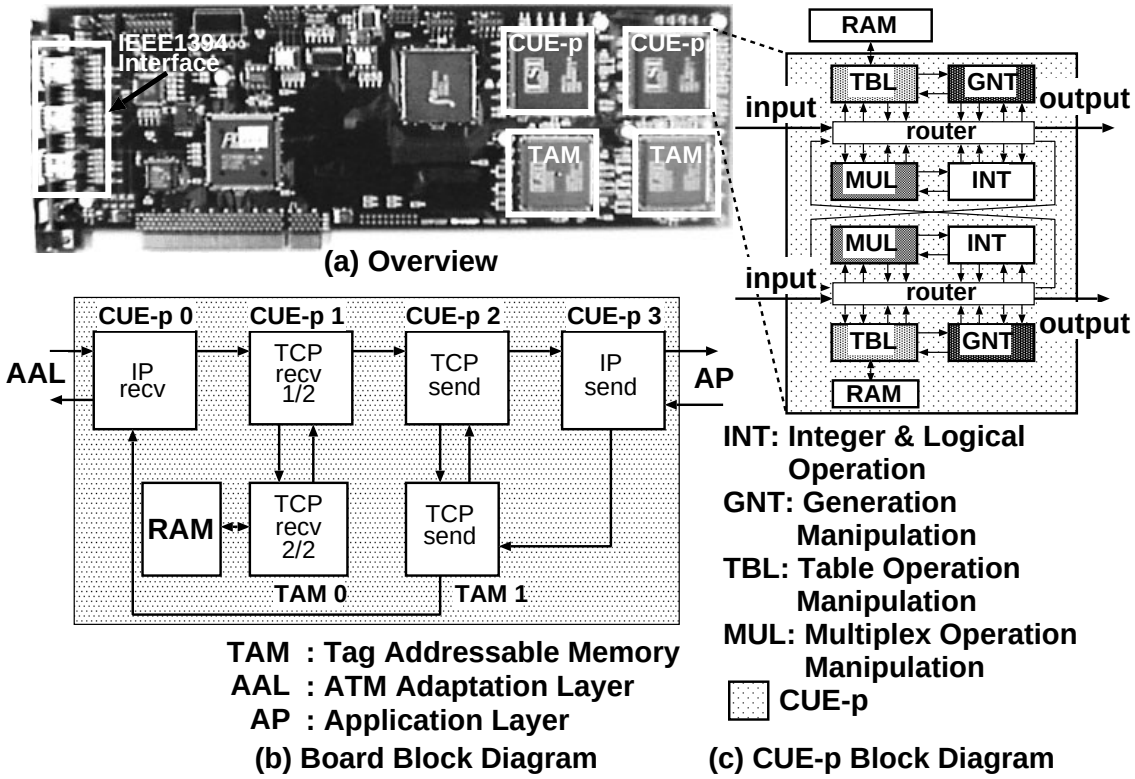

図 2.7: データ駆動プロセッサプロトタイプ CUE-p

するための妨げとなることを示した．次に，本実現法に用いたデータ駆動プロセッサアーキテクチャについて，ヘテロジニアスなプロセッサのスーパーインテグレーションによるプロセッサ構成，機能要素のエラスティックパイプラインによる実現について詳述した．このデータ駆動プロセッサによるプロトコル TCP/IP 処理の実現法として，ヘッダ処理とデータ処理の並列化により，ヘッダ処理がクリティカルパスとなるデータ長では，ターンアラウンドタイムがヘッダ処理に要する時間で維持されることを，ターンアラウンドタイムの見積りによる検証により明らかにした．さらに，本実現法をもとに検討した，プロトコル処理向きデータ駆動プロセッサプロトタイプの実現法について，工学的制約のもとで，TCP/IP over ATM が実現で

きるスループットを得るために，2 チップのスーパーインテグレーションとプロトコル処理向き命令の追加を行い，要求されるスループットが得られたことを示した．次章では，本実現法の有効性を，マルチメディア通信環境を想定した実験において実証する．

第 3 章

プロトコルの実時間多重処理に関する実験的検討

3.1 緒言

本章は，前章で詳述した TCP/IP 処理に加えて，分散オブジェクト環境である CORBA のオブジェクト間通信プロトコル GIOP, IIOP のデータ駆動型実現法の有効性の検証を目的とした実験的検討について述べている．本実験では，マルチメディア通信環境を想定したデータ入力条件のもと，TCP/IP, GIOP ならびに IIOP の実時間多重処理性能を評価する．本章ではまず，マルチメディア通信環境として，動画像，音声，ドキュメント，および制御情報の送受信を想定し，それぞれの帯域，入力間隔を規定する．次に，実験環境として，本実験に用いたデータ駆動プロセッサ CUE-v1，ならびに CUE-v1 を 7 個搭載した CUE-v1 ボードによるデータ駆動プロセッサシステムについて説明する．これらの条件および環境における実験において，それぞれのメディアを扱ったときのプロトコルの多重処理が，ターンアラウンドタイムを一定に維持できることを示す．最後に，本実験における考察を述べる．

3.2 マルチメディア通信環境の想定条件

本検討は，マルチメディアネットワーキング環境におけるプロトコル多重処理において，データ駆動型実現法の有効性を検証するために行った．本検討では，図 3.1 に示す CUE-v1 ボードを使用した．CUE-v1 ボード 3 枚，および評価結果を受け取る制御用 PC と接続するためのパケット入出力ボードにより構成される，図 3.3のデータ駆動プロセッサシステムに，第 2.3 節で実現法を述べた TCP/IP，ならびに CORBA におけるオブジェクト間通信プロトコル GIOP/IIOP を実装した．筆者らは，ネットワーキングの実現に不可欠な相互運用性を確立するための具体案として，分散オブジェクト指向技術である CORBA の利用を検討しており，そのオブジェクト間通信プロトコルである GIOP/IIOP のデータ駆動型実現を行っている [57]．本検討では，これらのプロトコル多重処理性能を上記の環境で評価した．

GIOP/IIOP を含めたプロトコル処理のデータ駆動型実現における処理構造を，図 3.2 に示す．図 3.2の左側がアプリケーションオブジェクトから入力されたメッセージをネットワークへ出力する送信処理，右側がネットワークから入力されたメッセージをオブジェクトへ出力する受信処理である．GIOP/IIOP， TCP/IP 処理はともにプロトコル処理であることから，転送の対象であるアプリケーションデータに対して行われるデータ処理と，転送の制御のためのヘッダに対して行われるヘッダ処理に大別できる．図 3.2の，網かけされた部分がデータ処理であり，それ以外はヘッダ処理を示している．図 3.2 のノード “チェックサム” に示すチェックサム計算処理は，送受信処理のどちらにおいても，IP ヘッダの生成・検査処理との間にデータ依存関係が存在しない．本実現法では，こうしたデータ依存関係のない処理を可能な限り並列化した．その結果，再送などの例外的な処理が伴わなければ，十分短いデータ長において，全体のターンアラウンドタイムはヘッダ処理に要する時間で一定となる．したがって，本実現法では，ヘッダ処理がクリティカルパスであ

り，ヘッダ処理の効果的な実現が重要となる．また，GIOP/IIOP 処理のデータ駆動型実現法では，IP データグラムや GIOP メッセージなどのストリームの，データ駆動プロセッサのパケットによる表現形式として，24bit の世代の上位 12bit をストリームの識別子として使用し，下位 12bit をストリーム中の要素の識別子として使用する．その結果，多重処理される複数のストリームを構成する個々の要素を一意に識別することが可能となり，要素単位の細粒度並列処理を実現している．1 つのパケットに格納される要素のサイズは，1 バイトとした．これは，GIOP/IIOP，ならびに TCP/IP において，バイト単位の通信を行うためである．これら GIOP/IIOP，ならびに TCP/IP の処理の詳細は，付録 A に示している．

図 3.4，ならびに図 3.5は，そのプログラム配置を示している．図 3.4，図 3.5ともに，図左に RESCUE による仕様記述，図右に CUE-v1 ボードのブロック構成を示している．CUE-v1 ボードは，7 個の CUE-v1 を搭載しているが，うち 3 個はチップ間の接続が入力，あるいは出力処理に特化しており，残り 4 個が互いのチップ間の行き来が可能な構成になっている．したがって，プログラムは，処理間相互のデータ転送が可能となるよう，4 個のプロセッサを中心に配置している．配置は TCP/IP ，ならびに GIOP/IIOP 処理の流れに沿うものとなっている．具体的には，ボードの入力ポートに近いプロセッサより，IP 受信部，TCP 受信部，TCP 送信部，IP 送信部としている．このとき，TCP 受信には 2 チップ使用するため，IP 送信部はボードの出力ポートに直結するチップに配置した．GIOP/IIOP 処理でも同様に，ボードの入力ポートに近いプロセッサより，IIOP 受信，GIOP 受信，GIOP 送信，IIOP 送信としている．このように，本実験において TCP/IP，および GIOP/IIOP 処理に 2 つの CUE-v1 ボードを用いているが，実際には 9 個のチップにて処理を行っている．

マルチメディアネットワーキング環境は次のように想定した．クライアント・サーバモデルにおいて，サーバでは，クライアント同士が通信する，あるいはクライアン

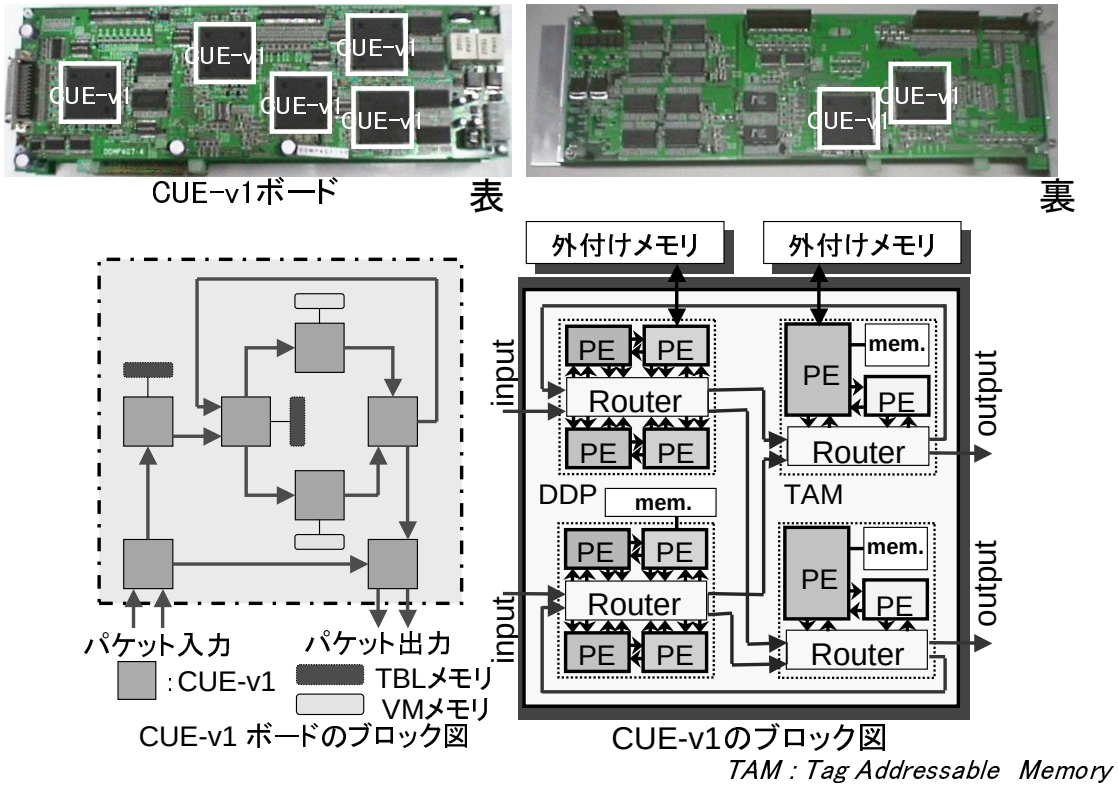

図 3.1: CUE-v1 ボードおよび CUE-v1 のプロセッサ構成

トからの要求される情報を提供するために動画像，音声，ドキュメントおよび制御情報といったメディアの同期や配信が行われる．本実験では，実験環境におけるメディア，帯域，データ長，入力間隔を表 3.1 のように仮定した．動画像，音声には帯域と 1 メッセージあたりのデータ長，ドキュメント，制御情報にはそれぞれのデータ長を仮定した．このとき，動画像は MPEG4, 音声は PCM(Pulse Code Modulation)，ドキュメントには，静止画を含めた典型的な Web ページのデータサイズを考慮した．また，ドキュメントにおける 1 メッセージあたりのデータ長は，ローカルエリアネットワークの構築に一般的に用いられる Ethernet の MTU (Maximum Transfer Unit) である 1500 バイトから，プロトコルのヘッダ分を除いた 1408 バイトとした．同様

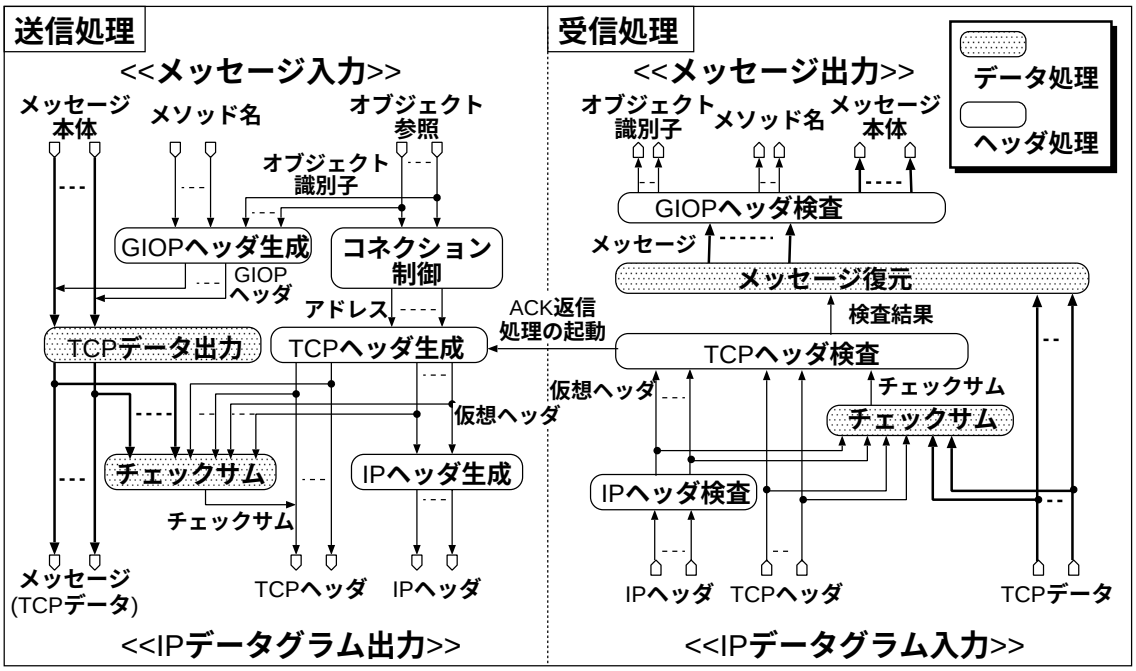

図 3.2: GIOP/IIOP および TCP/IP プロトコル処理のデータ駆動型実現における処理構造

に，入力間隔も動画像，音声についてはそれぞれ MPEG4，PCM におけるスループットを考慮した．制御情報に関しては，最も頻繁な状態を想定し，人間が応答できるおおよその最小時間を間隔とした．このとき，多重度を制約する要因はネットワークの持つスループットとなる．CUE-p で実現できる TCP/IP over ATM を想定すると，実効的な最大スループットは 135Mbit/sec である．ユーザ 1 人あたりの帯域は，表 3.1に示す各メディアの帯域に，各プロトコルのヘッダのオーバヘッドを考慮すると，約 3.5Mbit/sec となることから，最大ユーザ数は約 35 となる．本検討では，最大ユーザ数 35 までの各メディアにおける TCP/IP の多重処理性能を評価した．図 3.3 のデータ駆動プロセッサシステムにおけるデータ生成部では，本実験を行うために，各メディアに見合うデータ長のデータグラムの生成を行うととも

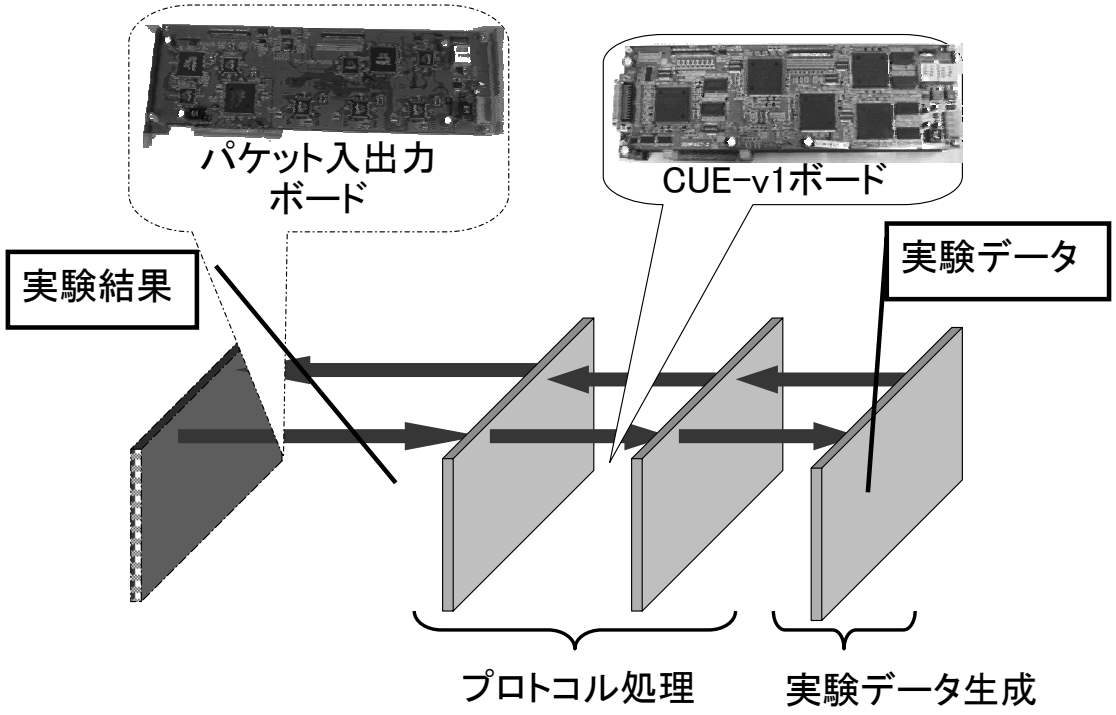

図 3.3: CUE-v1 ボードによるデータ駆動プロセッサシステム

に，帯域，入力間隔に合わせて，プロトコル処理部に送出するプログラムを配置した．このとき，マルチメディア通信を想定しているので，各メディアは並列に入力される．本実験では，これらのメディアデータを受信し，応答を送信するまでの処理に要するターンアラウンドタイムを 1 つのメディアデータごとに測定した．

3.3 多重プロトコル処理の実験的検討

図 3.6は，想定条件のもとで行ったプロトコル多重処理性能の評価結果である．動画像 (図中 (a) 動画像) をはじめ，4 つのメディアすべてにおいて，多重度の増加に関わらずターンアラウンドタイムがほぼ一定に維持できている．さらに，音声 (図

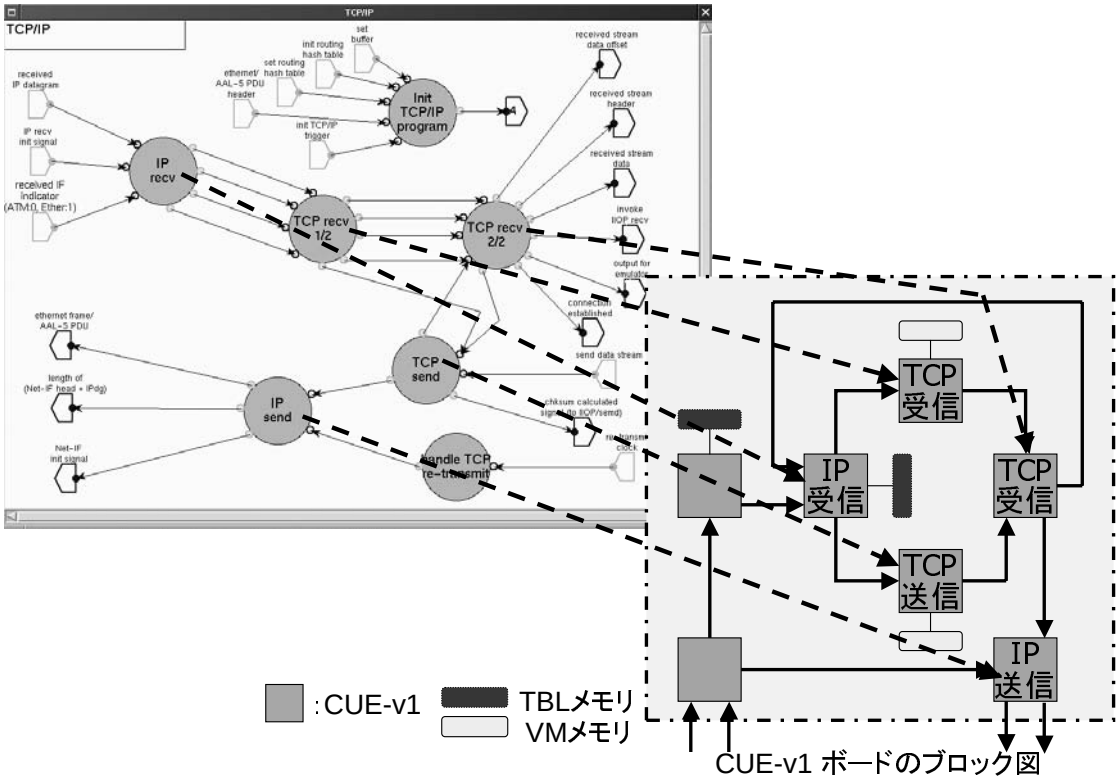

図 3.4: TCP/IP 処理のプログラム配置

中 (b) 音声)，制御情報 (図中 (d) 制御情報) は，すべてのプロトコルのヘッダを含めても 250 バイト以下となるため，2.3 節で評価した，本実現法における最短のターンアラウンドタイムで処理されている．ドキュメント (図中 (c) ドキュメント) に関しては，他の 3 つと比較して，ターンアラウンドタイムが約 3 倍となっているが，他の 3 つに比べて時間制約が緩いため，支障はないと考えられる．しかし，多重度の増加はターンアラウンドタイムをわずかに増加させている．これは，多重度が増加すると特定のプロセッサにおいて瞬時的に過負荷が生じ，そのプロセッサでの処理時間が延びるためと考えられる．この問題に関しては，今後のプロトコル処理向きデータ駆動プロセッサアーキテクチャの検討において，RESCUE による負荷の予測

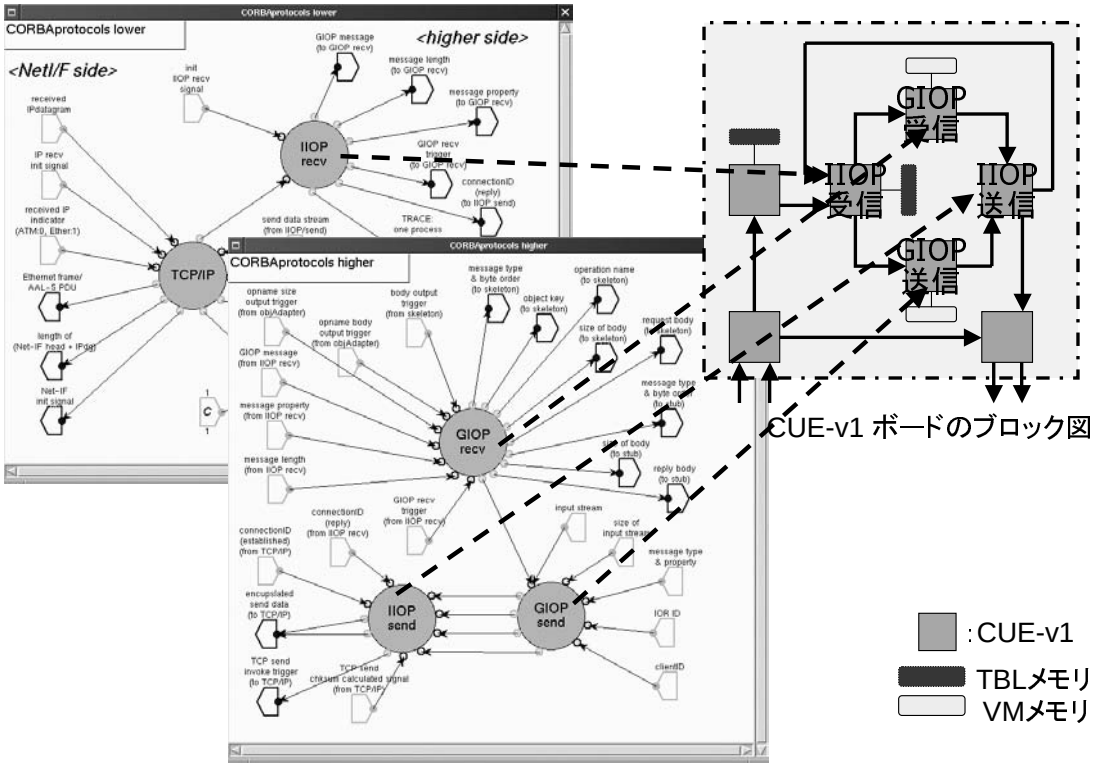

図 3.5: GIOP/IIOP 処理のプログラム配置

機能 [65]–[67]，RESCUE におけるエミュレータによる負荷の検証 [69]，ならびに，負荷をより均衡化させるためのプログラム配置支援 [70] を活用して，瞬時的な負荷に対応できるパイプライン構成を明らかにしたいと考えている．

3.4 結言

本章では，TCP/IP 処理，ならびに CORBA のオブジェクト間通信プロトコル GIOP，IIOP のデータ駆動型実現法の実験的検討を通じて，本実現法の有効性を実証した．マルチメディア通信環境を想定したデータ入力条件のもと，TCP/IP, GIOP

表 3.1: 実験環境におけるメディアの想定条件

	帯域	データ長 (1 メッセージあたり)	入力間隔
動画像	2Mbit/sec.	— (250 byte)	1 msec.
音声	64 kbit/sec.	— (40 byte)	5 msec.
ドキュメント	—	300 kbyte (1408 byte)	—
制御情報	—	100 byte (100 byte)	100 msec.

ならびに IIOP の実時間多重処理性能を評価した結果，それぞれのメディアを扱ったときのプロトコルの多重処理が，ターンアラウンドタイムを一定に維持できることを示した．特に，前章で示した 250 バイト以下の IP データグラムである，動画像，音声，制御情報では，ヘッダ処理に要する時間でターンアラウンドタイムが維持できることを示した．実時間性の保証には，いかなる処理状況によっても処理時間を保証できることが重要である．したがって，本実験における評価結果は，本実現法の実時間多重処理への有効性を実証するものと考えている．次章では，ネットワーキング向きデータ駆動プロセッサアーキテクチャの検討の一環として行ってきた，ネットワーキングインタフェースのデータ駆動型実現法とその実験的検討について詳述する．

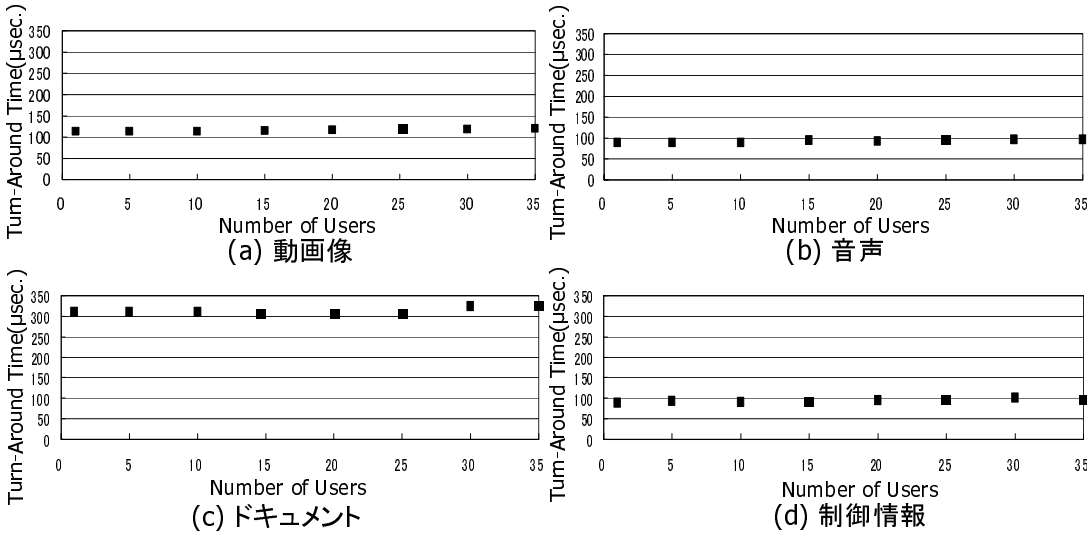

図 3.6: プロトコル多重処理の評価

第 4 章

ネットワーキング向きデータ駆動プロセッサアーキテクチャ

4.1 緒言

本章は，前章までに述べたプロトコル処理のデータ駆動型実現法で得られた成果を実際の通信において活用するために行った，ネットワーキングインタフェースのデータ駆動型実現法とその実験的検討について述べている．さらに，本検討を通じて，ネットワーキング向きデータ駆動プロセッサアーキテクチャを提案している．ネットワーキングインタフェースには，ネットワークの持つスループットを最大限に活用することが求められる．本実現法では，ATM，および，Fast Ethernet の持つ実効的な最大スループットを充足するために，CUE-v1 と ATM および Fast Ethernet のコントローラ間のデータ転送を FPGA を用いてボトルネックなく実現した．本章では，このネットワーキングインタフェースのデータ駆動型実現法における構成，ならびにデータ転送処理の実現法を詳述する．転送能力に関する実験的検討では，このネットワーキングインタフェースが ATM の実効的な最大スループットを活用可

能であることを示す．最後に，本検討を通じて，ネットワーキング向きデータ駆動プロセッサの入出力機構，特にネットワークインタフェースに関する提案を行う．

4.2 ネットワーキングインタフェースのデータ駆動型実現法

筆者らは，ネットワーキング向きデータ駆動プロセッサの研究の一環として，ネットワーキングインタフェースのデータ駆動型実現法を検討している．図 4.1は，ネットワーキング環境におけるネットワーキングインタフェースの要件であるネットワーク用 LSI とのデータ転送機構を，CUE-p の後継である CUE-v1 を用いて実現したデータ駆動型ネットワーキングインタフェースボードを示している．図 4.1上は，本ボードの全体像である．ネットワーク用 LSI である ATM SAR and PHY(図中 ATM SAR and PHY), FastEthernet PCI bus コントローラ (図中 FastEthernet PCI buw Controller)，および，データ駆動プロセッサ CUE-v1 が 2 個搭載されている．また，ネットワーク用 LSI と CUE-v1 との間のデータ転送におけるデータフォーマット変換処理に，FPGA (Field Programmable Gate Array) を用いている．ATM SAR and PHY には，NEC μ pd98405 を用いた．FastEthernet PCI bus コントローラには，Intel 82558 を用いた．さらに，FPGA には ALTERA EPF10K100 を使用している．図 4.1中および下が，本ボードの接続構成である．図の左側がネットワークへの入出力，右側が，TCP/IP 処理を実現する CUE-v1 システムへの入出力である．ネットワーク用 LSI は，ATM, FastEthernet ともに PCI バスへの接続を前提として設計されているため，FPGA との間に PCI バスを設けている．デュアルポートメモリは，CUE-v1 からネットワーク用 LSI へのデータ転送時に必須となる，データグラムの構成処理をパイプライン処理可能とするために用いている．

図 4.2に示すデータ駆動図式は，TCP/IP 処理を実現している CUE-v1 と本ボー

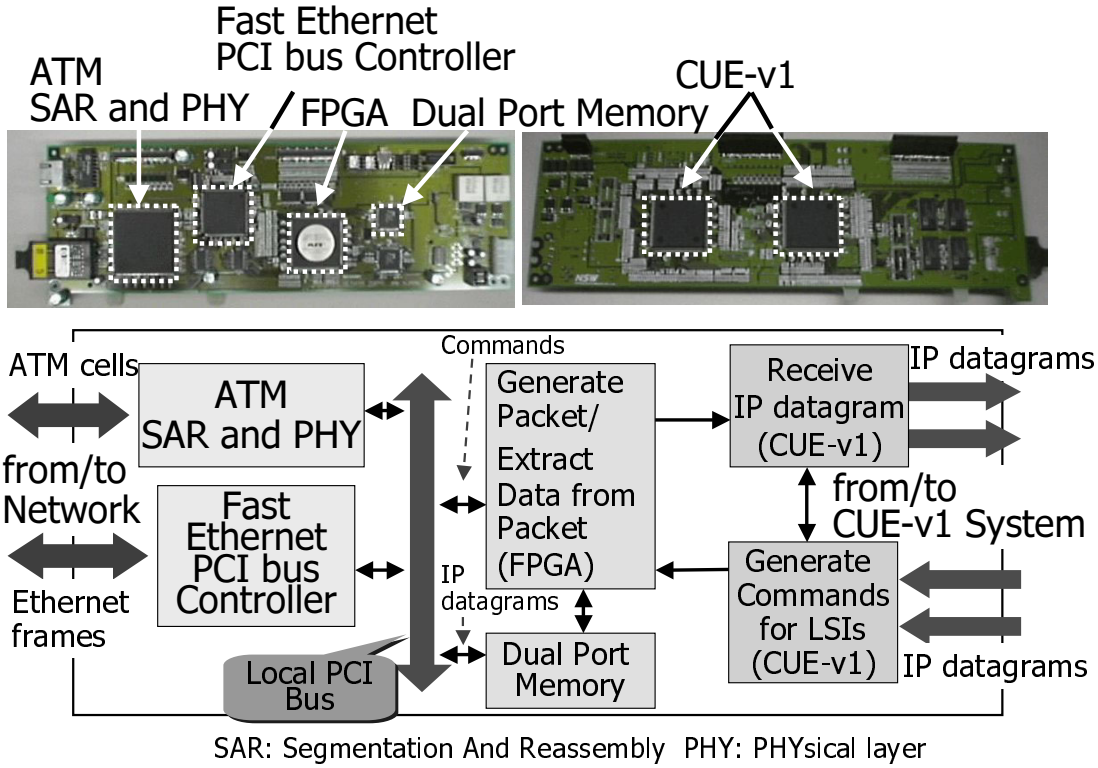

図 4.1: データ駆動型ネットワーキングインタフェースボード

ド上の FPGA 間の送受信処理の仕様である．すなわち，図 4.1に示す FPGA より右側の部分で行われる処理の仕様である．図 4.2に示す仕様は，データ駆動図式に基づいており，矢印がデータの流れ，楕円が処理を示している．図 4.2に示す仕様のデータの流れの方向は，図 4.1に合わせて，左方向がネットワーク側，右方向が CUE-v1 側としている．送受信ともに，IP 処理の入出力データである IP データグラムが入力されると，行き先となるバッファのアドレス指定処理 (図中 set buffer address) が行われる．さらに，CUE-v1 から FPGA への送信時には，ATM におけるチャネルやデータ長などの情報を持つパケットディスクリプタ (図中 PD) がその出力処理 (図中 output PD) にて，ネットワーク用 LSI へのバッファ参照要求や送信要求などの

コマンドがその発行処理 (図中 issue command) にて，それぞれ送信される．また，データ受信通知やエラーについても，エラー内容の識別処理 (図中 interpret status, および check error) を行った後，CUE-v1 に送信される．正常に通信が行われている IP データグラムの実時間多重処理を実現するために，エラーに伴う処理は CUE-v1 で他の処理と多重に行う．実時間多重処理を実現し，ネットワークの転送能力を最大限に活用するために，本実現法では，CUE-v1 から FPGA への送信処理，ならびに FPGA から CUE-v1 への受信処理を同時並行に処理可能とするとともに，送受信すべきデータである IP データグラムの送受信と，ATM のチャネルや IP データグラム長などの処理に必要なデータの送受信を独立に処理し，IP データグラムの送受信を阻害しないよう実現している．

また，図 4.3に示す図式は，本ボード上のネットワーク用 LSI，FPGA ならびにデュアルポートメモリにおけるデータ転送制御処理の仕様である．すなわち，図中の網かけの部分が図 4.1に示すボード中の ATM LSI で行われる処理であり，その他の部分が，FPGA，デュアルポートメモリおよび CUE-v1 で行われる処理である．図 4.3に示す仕様に用いた図式ならびにデータの流れの方向は，図 4.2と同様である．ATM セルを受信すると，ネットワーク用 LSI にて，セルヘッダ検査 (図中 check cell header) やチャネルの識別 (図中 notify channel & error) を行った上で IP データグラムを構成する (図中 restore IPdg)．このとき，セルヘッダ検査にてエラーがあった場合には，エラー通知処理 (図中 notify channel & error) から割り込み発行処理 (図中 issue interrupt) を経て，CUE-v1 に通知される．また，ネットワークからの IP データグラムの受信では，IP データグラムはネットワーク用 LSI にて既に構成されているので，CUE-v1 に入力するまでに IP データグラムをバッファに書き込む必要はない．しかし，ネットワーク用 LSI がノイマン型プロセッサのもとで用いられることを前提としているため，図 4.3 では，IP データグラムの受信においてバッファアドレスの指定処理 (図中 allocate receive buffer) が存在する．本実現法では，

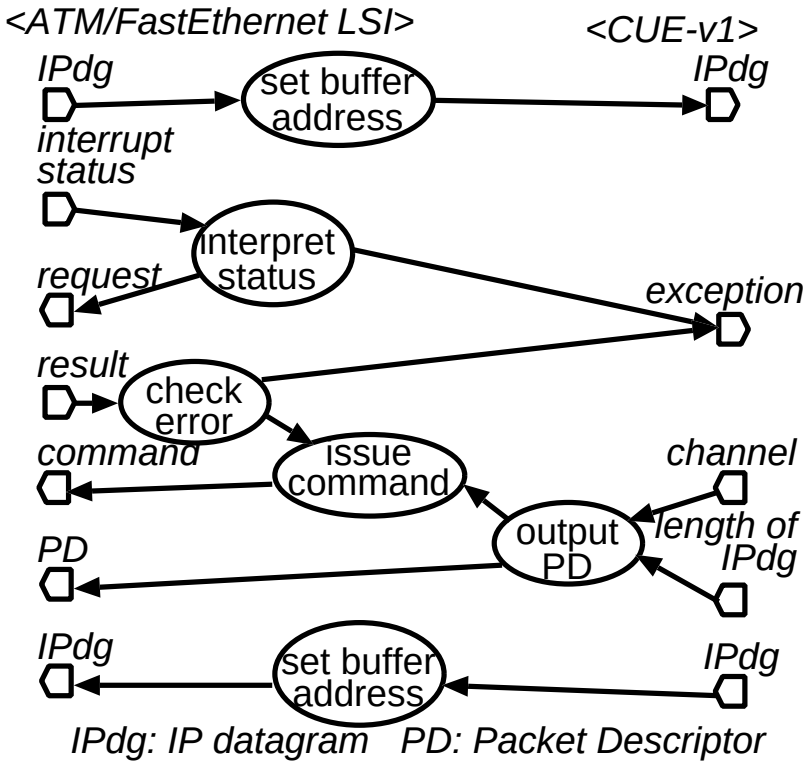

図 4.2: CUE-v1 と FPGA 間の送受信処理

上記のアドレス指定処理において，IP データグラムが FPGA から CUE-v1 に入力されるアドレスを予め決めておき，そのアドレスを IP データグラムに与えている．一方，送信時において，CUE-v1 からネットワーク用 LSI へ送信される IP データグラム，およびパケットディスクリプタ (図中 PD) は，バッファに格納される (図中 write IPdg，write PD)．このとき，IP データグラムの格納バッファは デュアルポートメモリ であり，パケットディスクリプタの格納バッファは FPGA 内部のメモリである．CUE-v1 からの IP データグラムの書き込み，ならびに，ネットワーク用 LSI の IP データグラムの読み出しが同時並行に行うために，本ボードは IP データグラムのバッファとして デュアルポートメモリを採用した．IP データグラム，お

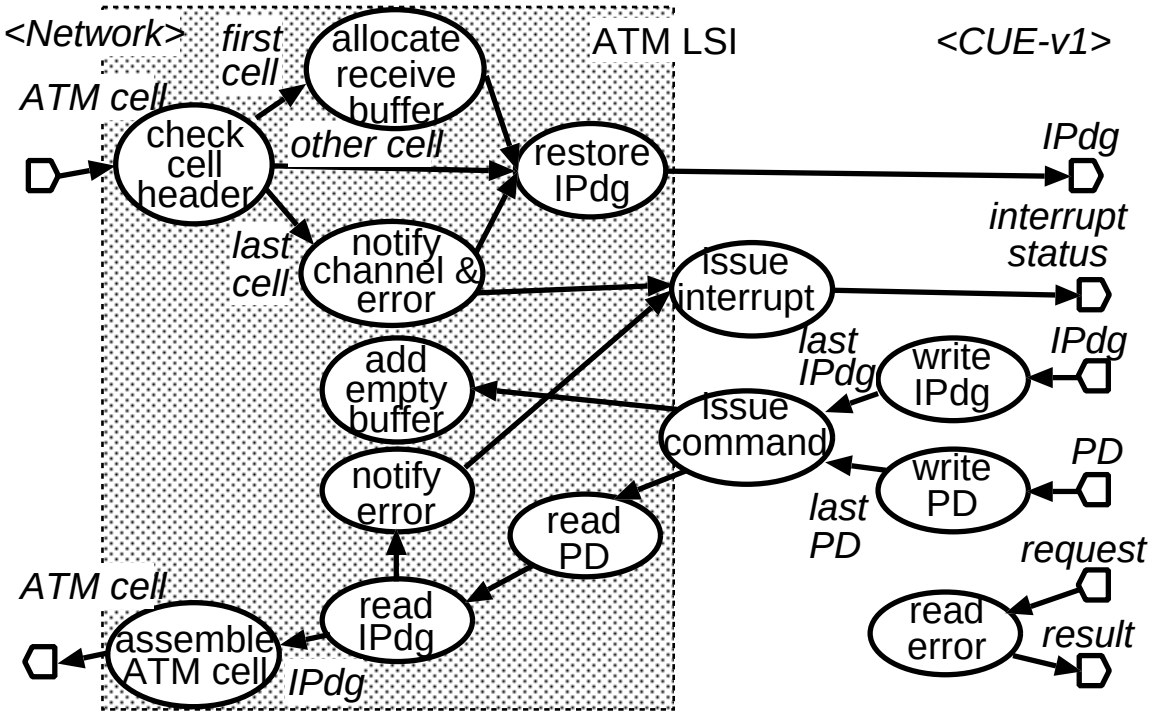

図 4.3: FPGA およびデュアルポートメモリにおける転送制御処理

よび パケットディスクリプタ を構成する最後のパケットが格納された後， ネットワーク用 LSI へコマンドが発行され (図中 issue command)，ネットワーク用 LSI はそれに従い パケットディスクリプタ，ならびに IP データグラムをバッファより読み出す (図中 read PD，read IPdg)．読み出された IP データグラムは，順次 ATM セルとして構成される (図中 assemble ATM cell)．IP データグラムの読み出しまでにおけるエラーについては，エラー検出 (図中 notify error) を経て，エラー通知が発行される．エラーに伴う処理については，CUE-v1 と FPGA 間の送受信処理と同様である．なお，図 4.2，および 4.3では，ATM を例としているが，FastEthernet の場合も処理は同様である．

4.3 転送能力に関する実験的検討

本節では，ネットワーキングインタフェースボードによって達成される性能を検証する．PCI のクロック周波数は 33MHz で，バス幅は 32bit であるため，PCI の帯域は 1056 Mbit/s となる．PCI がバスであることや，アドレス線とデータ線が共有されていることを考慮しても ATM の最大実効速度である 135Mbit/s に対して十分大きく，PCI がボトルネックとはならない．また，受信時においては，データグラムをバッファを介さずに直接データ駆動プロセッサへ送出するため，受信に関しては物理層の転送速度を活用できる．したがって，送信時に行うデュアルポートメモリへの書き込み動作のスループットが ATM の最大実効速度である 135Mbit/s より大きければ，ネットワーキングインタフェースボード上にボトルネックがなく，物理層の転送速度を活用できることが実証できる．

図 4.4はデータ転送時におけるデータ駆動ネットワーキングインタフェースボード上でのデータの流れ，ならびに各部の転送速度を示している．IP データグラムやパケットディスクリプタの構成要素となるパケットは，多重に，かつ順序が守られることなく CUE-v1 から FPGA へ送出される．ここでパケットとは，CUE-v1 におけるトークンの実現法であり，命令実行のために必要な情報をすべて含んでいる．このとき，ネットワーキングインタフェースボードにおいて，CUE-v1 からデュアルポートメモリへの書き込み時間は 1 パケットあたり 90nsec である．このとき，IP データグラムの長さを a バイトとすると，その IP データグラムをデュアルポートメモリへ書き込むには，図 4.4 上部に示すように，45$a$nsec 要する．上述したように，ATM SAR and PHY のデュアルポートメモリ参照は，CUE-v1 からデュアルポートメモリへの書き込みと独立にできるので，図 4.4 上部に示す 60nsec/32bit はスループットに対するボトルネックとはならない．一方，CUE-v1 から FPGA 内部のメモリへのアクセス時間は，図下部に示すように 1 パケットあたり 150nsec である．一

連の送信動作で FPGA 内部のメモリへは，20 バイト分のアクセスが行われる．また，このアクセスに必要な時間は，1 パケットあたり 2 バイトのデータを保持するため，1.5μsec となる．したがって，IP データグラムを 1 個送信するのにかかる時間は，送信する IP データグラムの大きさを x とすると，次式で示される．

$$150nsec \times \frac{20}{2} + \frac{x\ バイト}{2} \times 90nsec.$$

さらに，IP データグラムを ATM の最大実効速度である 135Mbit/s で送信するための条件は，次式で表される．

$$1.5\mu sec + \frac{x\ バイト}{2} \times 90nsec \leqq \frac{x\ バイト}{135Mbit/s},$$

$$x \geqq 105\ バイト.$$

以上より，IP データグラムの大きさが 97 バイト以上であれば，ATM の最大実効速度である 135Mbit/s で通信できる．TCP/IP では，少なくとも 40 バイト はヘッダであるため，実質的なデータは 57 バイトとなる．これは動画や音声に代表されるメディアのデータ長としても十分妥当である [68].

4.4 ネットワーキング向きデータ駆動プロセッサの実現に向けて

前節までに述べたデータ駆動型ネットワーキングインタフェースボードは，筆者らがこれまで検討してきたプロトコル処理のデータ駆動型実現法を活用するとともに，ATM SAR and PHY，および FastEthernet PCI bus コントローラを利用してネットワーキングインタフェースを実現している．本ボードでは，自己同期である CUE-v1 からクロック同期である FPGA へのデータ出力部が実現するのに最も難しい部分であった．これは，コストや期間などの面から，既存の ATM SAR and

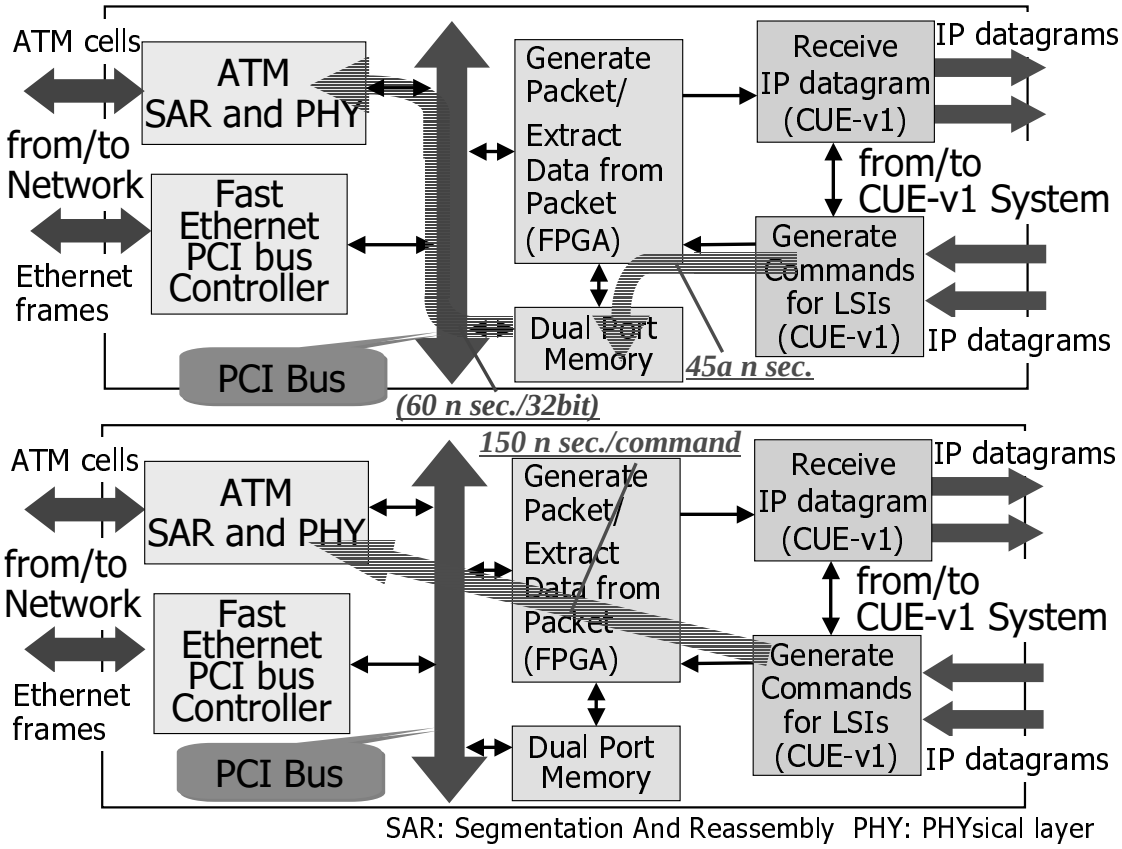

図 4.4: データ駆動型ネットワーキングインタフェースボードにおける送信処理

PHY ならびに FastEthernet PCI bus コントローラを利用したためであるが，将来的には，この部分も自己同期式の CUE プロセッサで実現したいと考えている．なぜなら，通信が元来非同期であると考えるからである．ATM はその名の通り非同期であるし，FastEthernet も昨今はスイッチングハブが普及するなどバス構造によるスループット低下を回避する方向にある．これまでプロセッサと比較して遅いという仮定のもとで実現されてきた入出力機構，および入出力処理が，ネットワークが高速化・広帯域化されたためネットワーク処理に対するボトルネックとなりつつある．本研究で適用しているデータ駆動プロセッサは，マルチプロセッサ構成を設計

当初より視野に入れた，入出力処理向きのプロセッサである．マルチメディアネットワーキング環境のデータ駆動型実現を目指す上で，ネットワークインタフェース処理のデータ駆動型実現法，ならびにそのプロセッサアーキテクチャの検討は不可欠である．このとき，本ネットワーキングインタフェースボードの転送処理の実現法が応用できると考えられる．

また筆者らは，本研究と並行して，ネットワーキング環境に適したデータ駆動プロセッサアーキテクチャを明らかにするために，図 4.5に示すデータ駆動実時間システムの開発支援環境 RESCUE (Realtime Execution System for CUE series data-driven processors) を実現している [67]．RESCUE における仕様記述環境では，プロセッサアーキテクチャからプログラムの仕様までを一貫したデータ駆動パラダイムにおいて記述し，機能検証ならびに性能予測を行う．さらに，プログラム実行時の性能検証を，パイプラインタクト水準でエミュレーションできるデータ駆動プロセッサエミュレータを CUE-v1 システム上に実現している．仕様記述環境は，既存のパーソナルコンピュータ，およびワークステーションの GUI を利用して実現している．したがって，仕様記述環境とデータ駆動プロセッサエミュレータの相互運用には，CUE-v1 システムとの通信が必須である．RESCUE では，仕様記述環境とエミュレータを接続するために，データ駆動型ネットワーキングインタフェースボードを活用している．

さらに，これまでの検討より，ネットワーキング環境におけるメディア処理・プロトコル処理では，逐次処理が本質的な処理部が存在することが明らかとなっている．データ駆動型実現法では，これらの処理がボトルネックとならないよう，これらの処理部を逐次処理プロセッサのインライン実行によって実現することを検討している．データ駆動プロセッサと逐次処理プロセッサ間のデータ通信には，本ネットワーキングインタフェースボードにおいて FPGA で実現したデータ転送機構が応用できると考えている．

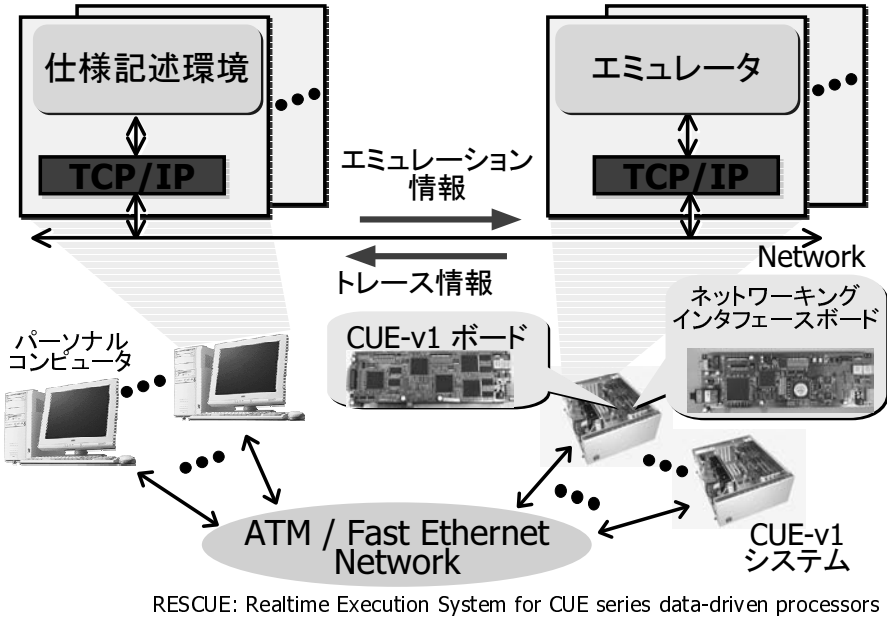

図 4.5: データ駆動実時間システムの開発支援環境 RESCUE

4.5 結言

本章では，ネットワーキングインタフェースのデータ駆動型実現法とその実験的検討について述べた．本実現法では，ATM，および，Fast Ethernet の持つ実効的な最大スループットを充足するために，CUE-v1 と ATM および Fast Ethernet のコントローラ間のデータ転送を FPGA を用いてボトルネックなく実現した．データ転送においてもできる限り同時並行な処理構造で実現した結果，転送能力に関する実験的検討では，このネットワーキングインタフェースが ATM の実効的な最大スループットを活用可能であることを示した．最後に，本検討を通じて，ネットワーキング向きデータ駆動プロセッサの入出力機構，特にネットワークインタフェースに関して，ネットワークコントローラも含めたプロセッサの実現が，ネットワーキング向きデータ駆動プロセッサアーキテクチャとして望ましいことを示した．次章

では，本研究の結論を述べるとともに，今後の展開を示す.

第 5 章

結論

本論文では，マルチメディアネットワーキング環境における多重プロトコル処理のデータ駆動型実現法について述べた．

第 2 章では，通信プロトコル処理の具体例として取り上げ検討した，TCP/IP の実時間多重処理のデータ駆動型実現法について述べた．本実現法では，プロトコル処理に本質的であるヘッダ処理と，データ処理をデータ依存性のない限り同時並行に処理する構造で，TCP/IP 処理をデータ駆動プロセッサ上に実現した．その結果，データ駆動プロセッサの持つ多重処理能力を活用すれば，ヘッダ処理がクリティカルパスとなる範囲のデータ長である 250 バイト以下のデータ長では，多重処理時においても，ターンアラウンドタイムがヘッダ処理に要する時間で維持できることを明らかにした．実時間性の保証には，いかなる処理状況下でも処理時間を保証できることが必須であり，本実現法によって得られた成果は，実時間多重処理の実現可能性を示すものである．次に，これらの成果を受けて，プロトコル処理向きデータ駆動プロセッサプロトタイプ CUE-p の実現法を検討について述べた．本実現法に用いたデータ駆動プロセッサアーキテクチャを踏襲するという制約のもと，TCP/IP over ATM を実現するために，TCP/IP 処理のデータ駆動型実現法における各処理

部のデータ通信量，ならびにボトルネックとなり得る処理の検証を通じて，2 チップのスーパーインテグレーション，ならびにプロトコル処理向き命令の追加により，TCP/IP over ATM が実現できることを明らかにした．

第 3 章では，前章で述べた TCP/IP 処理に加えて，分散オブジェクト環境 CORBA におけるオブジェクト間通信プロトコル GIOP/IIOP の実時間多重処理に関する実験的検討について述べた．前章で述べたデータ駆動プロセッサ CUE-p にメモリ処理機能の拡張を行った CUE-v1 を用いたデータ駆動プロセッサシステムに実現したプロトコル処理において，マルチメディア通信環境における多重処理性能を評価した．本検討では，マルチメディア通信環境として，動画像，音声，ドキュメント，および制御情報の 4 つのカテゴリにおける帯域，入力間隔を仮定し，それぞれが同時に入力される状況を想定し，実験の結果，ターンアラウンドタイムを維持しながら，それぞれのメディアを多重処理できることを明らかにした．特に，ヘッダ処理に要する時間内にデータ処理が完了できる動画像，音声，および制御情報に関しては，ヘッダ処理に要する時間として予測された時間で，処理が完了できていることも示した．

第 4 章では，前章までに述べたプロトコル処理のデータ駆動型実現法で得られた成果を実際の通信において活用するために検討した，ネットワーキングインタフェースのデータ駆動型実現法とその実験的検討について述べた．本実現法では，ATM，および，Fast Ethernet の持つ実効的な最大スループットを充足するために，CUE-v1 と ATM および Fast Ethernet のコントローラ間のデータ転送を FPGA を用いてボトルネックなく実現した．転送能力に関する実験的検討では，このネットワーキングインタフェースが ATM の実効的な最大スループットを活用可能であることを示した．さらに，本検討で得られた知見をもとに，ネットワーキング向きデータ駆動プロセッサの実現に向けて，ネットワーキング向きデータ駆動プロセッサの入出力機構に関して，ATM や FastEthernet のコントローラを含めた実現法を提案した．

これまでの研究を通じて，定常状態におけるマルチメディア通信に対する，プロ

トコルの実時間多重処理のデータ駆動型実現法の有効性が実証できた．しかしながら，目標とするマルチメディアネットワーキング環境では，ユーザから複数の対話の要求が同時に起こり得るため，不定期に起こる通信要求に対する検討は必須と考えられる．さらに，通信のみならず，放送や配信といったいわゆるマルチキャスト性についても検討が必要である．本実現法では，過負荷状態に陥らない限り，要求が瞬時的に増加してもターンアラウンドタイムを最短に維持できると考えられるが，このような環境における実証は不可欠である．本論文における実験的検討では，マルチメディア通信について各メディアの帯域や入力間隔は仕様から一定の帯域，入力間隔を仮定していた．より実際の環境に近い状態で評価するには，この定常状態に加えて，インタラクティブな環境を想定した不定期の要求を起こし，その時の処理性能を評価する必要がある．この不定期の要求は，要求仕様に応じて確率モデルを定義することによって，実現できると考えている．

本研究における実験的検討では，マルチメディア通信を仮定した，複数のメディアの実時間多重処理について検討してきたが，アプリケーション層はメディアデータの受け取り，ならびに送信要求の発行のみである．ユーザとの接点であるアプリケーションの実現法の検討は，今後取り組まなければならない重要な課題の 1 つである．今後は，マルチメディアコンテンツ処理 [71] を含む，マルチメディアネットワーキング環境のデータ駆動型実現法の優位性を実験的に検証したい．また，現状では，通信・放送の分野は娯楽的な領域で発展しているきらいがある．しかしながら，社会人教育をはじめとする生涯教育・研究の重要性を考慮すると，教育・研究の分野への通信・放送を含めたマルチメディアネットワーキング環境の適用は必須である．マルチメディアネットワーキング環境を用いた教育・研究システムの構築は，時間・距離の克服のみならず，動画像や音声を用いた教材の活用，ならびに講師・受講者間のコミュニケーションの確立など，理解度の促進に貢献できると考えられる．今後，情報社会基盤としてのマルチメディアネットワーキング環境のアプリ

ケーションの具体例として，教育・研究システムのためのアプリケーションのデータ駆動型実現法を検討したい．

さらに，本論文で述べたデータ駆動型ネットワーキングインタフェースボードに用いたネットワーク用 LSI はクロック同期であり，ノイマン型プロセッサとの相互運用においても本実現法の適用部分があると考えられる．また，ネットワーキング環境では，パーソナルコンピュータに代わり携帯電話に代表される携帯情報端末が端末として普及すると予想される．携帯情報端末では，動作時とともに待機時の消費電力が問題となるが，データ駆動プロセッサは，自己同期型のエラスティックパイプラインで構成されているため，待機時にはほとんど電力を消費しないという特長がある．今後は，プロトコル・メディア処理を実現する CUE プロセッサによるネットワーキング環境の実現法，ならびにこれらの環境を実現するためのプロセッサアーキテクチャを検討する予定である．今後，本検討で実現したデータ駆動型ネットワーキングインタフェースボードの機能を 1 チップで実現するデータ駆動プロセッサアーキテクチャについても検討したいと考えている．

謝辞

本研究の全過程を通じて, 終始適確かつ熱心な御指導・御鞭撻を賜った筑波大学電子・情報工学系教授 西川博昭先生には, 心から深く感謝の意を表するとともに，厚く御礼を申し上げる．

本論文の審査にあたり，懇篤なる御指導を頂くとともに種々の御高配を賜った，筑波大学 電子・情報工学系 教授 海老原義彦先生，同教授 田中二郎先生，同助教授 安永守利先生，ならびに，同助教授 朴泰祐先生に深謝の意を表する．

ネットワーキングアーキテクチャに関して多大な御教示と御助言を頂いた，NTT 情報流通プラットフォーム研究所 石井啓之博士に心から感謝する．

データ駆動型プロセッサのアーキテクチャやシミュレータ・エミュレータ環境に関して多大な御支援を頂いたシャープ株式会社 宮田宗一博士，木原誠一郎氏，
紫竹リカルド毅史氏ならびに芳田眞一氏に心より御礼申し上げる．

本論文の作成にあたり適切な御助言と御示唆を頂いた，筑波大学電子・情報工学系講師 冨安洋史先生に心から感謝する．

西川研究室の同期である，有限会社 情報基盤研究所 我孫子泰祐博士には，本研究を進めるにあたり多大な御助言と御協力を頂いた．ここに記して深く感謝する．

西川研究室の卒業生である工藤慎也氏には，プロトコル処理のデータ駆動型実現，ならびにその実験的検討にあたってご協力いただいたことを心から感謝する．

工学研究科 5 年次 榑林亮介氏，システム情報工学研究科 3 年次 伊藤伸也氏，同

2 年次 高橋徹氏，安村康平氏には，ネットワーキング環境のデータ駆動型実現法に関して，常日頃から有益な議論を頂いた．ここに記して感謝する次第である．

参考文献

[1] 村井 純,“次世代インターネット技術,” 電子情報通信学会誌, Vol.84, No.1, pp.2-9, Jan. 2001.

[2] 若原俊彦, “マルチグループ通信システムの構成法の提案と評価,” 電子通信学会論文誌 B, vol. J82-B 巻，no. 1, pp. 31-42, Jan. 1999.

[3] 萩島功一, 小柳 恵一, 北見 憲一, 山口 治男, “インフラ系ネットワークの変革,” 電子情報通信学会誌, Vol.81, No.4, pp.371-379, Apr. 1998.

[4] 岸本 了造, “次世代のマルチメディア通信トランスポートネットワークについて,” 電子情報通信学会論文誌, Vol. J79-B-I, No.5, pp.195-206, May 1996.

[5] 山尾 泰, 梅田 成視, 大津 徹, 中嶋 信生, “第 4 世代移動通信の展望 — 無線システムを中心とした課題について —,” 電子情報通信学会誌, Vol.J83-B, No.10, pp.1364-1373, Oct. 2000.

[6] 松下 温, 屋代 智之, “ITS の通信基盤の展望と課題,” 電子情報通信学会論文誌, Vol. J82-B, No.11, pp.1950-1957, Nov. 1999.

[7] 須田 達也, 板生 知子, 中村 哲也, 松尾 真人, “サービス創発のための適応型ネットワークアーキテクチャ,” 電子情報通信学会論文誌, Vol. J84-B, No.3, pp.310-320, Mar. 2001.

[8] 岸本 了造, “なぜ，やわらかい通信ネットワークの研究を行うのか,” 電子情報通信学会論文誌, Vol. J79-B-I, No.5, pp.173-184, May 1996.

[9] Chao-Ju Jennifer Hou, “Routing Virtual Circuits with Temporal QoS Requirements in Virtual Path-Based ATM Networks,” IEEE Transactions on Computers, Vol. 48, No. 11, pp.1228-1243 , Nov. 1999.

[10] 小菅 昌克, 山崎 達也, 荻野 長生, 松田 潤, “マルチエージェントによる適応的 QoS 制御,” 電子情報通信学会論文誌，Vol. J82-B, No. 5, pp. 702-710, May 1999.

[11] 阿部 睦, 嶋野 淳子, 渥美 幸雄, “端末におけるサービス品質保証方式の提案と評価,” 電子情報通信学会論文誌，Vol. J82-B, No. 5, pp. 711-721, May 1999.

[12] 安家 武, 西村 正寿, Andreas Sucahyono, 中西 正洋, 戸出 英樹, 池田 博昌, “スクリーニングサービスを含めた通信サービス仕様エミュレータの構築及び評価,” 電子情報通信学会論文誌, Vol. J82-B, No.5, pp.858-867, May 1999.

[13] 西 和彦, 片山 泰男, 正村 和由, 持田 康典, “計算処理能力に応じた QOS 概念に基づく動画像符号化処理方式,” 電子情報通信学会論文誌，Vol. J82-B, No. 1, pp. 19-30, Jan. 1999.

[14] Jurg Bolliger and Thomas Gross, “A Framework-Based Approach to the Development of Network-Aware Applications,” IEEE Transactions on Software Engineering, Vol. 24, No. 5, pp. 376-390, May 1998.

[15] 河内谷清久仁, “マルチメディア処理の動的 QoS 制御のフレームワーク,” 電子情報通信学会論文誌，Vol. J80-B-I, No. 6, pp. 465-471, June 1997.

[16] Jan Gecsel, “Adaptation in Distributed Multimedia Systems,” IEEE Multimedia, pp. 58-66, April-June 1997.

[17] Klara Nahrstedt and Jonathan M. Smith, “The QoS Broker,” IEEE Multimedia, Vol. 2 No. 1, pp. 53-67, 1995.

[18] Martin W. Sachs, Avraham Leff and Denise Sevigny, “LAN and I/O Convergence: A Survey of the Issues,” IEEE COMPUTER, pp. 24-32, Dec. 1994.

[19] 陳 紅兵, 木村 成伴, 海老原 義彦, “異なる CPU 処理能力を持つリアルタイム通信システムの平均応答時間と入力トラヒック,” 情報処理学会論文誌, Vol.37, No.11, pp.2066-2071, Nov. 1996.

[20] 海老原 義彦, 張 興周, “ホモジニアスなリアルタイム通信システムの上限付き平均応答時間とトラヒック,” 情報処理学会論文誌, Vol.34, No.8, pp.347-355, Aug. 1993.

[21] 佐藤 秀雄, 矢向 高弘, “時間制約の厳しいマルチメディアのためのリアルタイム通信機構,” 情報処理学会論文誌, Vol.42, No.2, pp.189-196, Feb. 2001.

[22] 鮫島 康則, 宮崎 敏明, 深沢 友雄, 寺元 光生, 松広 一良, “リアルタイムパケットフィルタ,” Vol. J83-B, No. 10, pp. 1419-1427, Oct. 2000.

[23] 竹内 理, 岩嵜 正明, 中原 雅彦, 中野 隆裕, “連続メディア処理向き OS の周期駆動保証機構の設計と実装,” 情報処理学会論文誌, Vol. 40, No. 3, pp. 1204-1215, Mar. 1999.

[24] 足高正訓，大久保英嗣，“リアルタイムトランザクションのための時刻印方式に基づく動的な同期プロトコルとスケジューリング手法,” 電子通信学会論文誌，vol. J82-D-I，No.4，pp. 560-570, Apr. 1999.

[25] 緒方 正暢, “リアルタイム OS の研究動向,” 情報処理, Vol. 36, No. 8, pp. 743-744, Aug. 1995.

[26] 箱森 聰, 谷口 秀夫, “分散型リアルタイム OS: DIROS,” 情報処理, Vol. 36, No. 8, pp. 751-754, Aug. 1995.

[27] 竹山 寛, 坂村 健, “制御用リアルタイム OS 体系 ITRON 仕様の設計思想とその実現,” 情報処理, Vol. 30, No. 5, pp. 532-543, May. 1989.

[28] 宮崎孝, 黒田一郎, “汎用プロセッサを用いたビデオコーデック,” 電子情報通信学会論文誌, Vol.J83-A, No.12, pp.1339-1348, Dec. 2000.

[29] Rolf Fiedler, “Beyond ILP-Trends for Programmable DSP Machines,” The MathWorkds International DSP Conference, May 2001.

[30] Kwok Wah Yeung, “A Data-driven Multiprocessor Architecture for High Throughput Digital Siginal Processing,” Ph.D Thesis, University of California at Berkeley, May 1995.

[31] 有田 大作, 濱田 義雄, 米本 聡, 谷口 倫一郎, “PC クラスタを利用した実時間並列画像処理環境 PRV,” Vol. J84-D-II, No. 6, pp. 965-975, June 2001.

[32] Margarida F. Jacome, Helvio P. Peixoto, “A Survey of Digital Design Reuse,” IEEE Design & Test of Computers, Vol.18, No.3, pp.98-107, Mar. 2001.

[33] ウィチャイ ブンクムクラオ, 三木 信弘, “帯域消去フィルタの FPGA による実現,” 電子情報通信学会論文誌, Vol.J83-D2, No.11, pp.2171-2179, Nov. 2000.

[34] Margarida F. Jacome, Gustavo de Vaciana, “Design Challenges for New Application-Specific Processors,” IEEE Design & Test of Computers, pp.40-50, April-June 2000.

[35] Sujit Dey, Clark N. Taylor, Debashis Panigrahi, Krishna Sekar, Li Chen, Pablo Sanchez, “Using a Soft Core in a SoC Design: Experiences with picoJava,”

IEEE Design & Test of Computers, Vol.17, No.3, pp.60-71, Mar. 2000.

[36] 星合 隆成, パトリック ツィナニー, 佐藤 規男, “要求応答型ストリームインタフェースの提案とその実装方式,” 電子情報通信学会論文誌, Vol. J82-D-I, No. 8, pp. 1080-1092, Aug. 1999.

[37] Hoon Coi, Jong-Sun Kim, Chi-Won Yoon, In-Cheol Park, Seung Ho Hwang, Chong-Min Kyung, “Synthesis of Application Specific Instructions for Embedded DSP Software,” IEEE Transactions on Computers, Vol.48, No.6, pp.603-614, June 1999.

[38] Vincent Michael Bove, Jr. and John A. Watlington, “Cheops: A Reconfigurable Data-Flow System for Video Processing,” IEEE Transactions on Circuit and Systems for Video Technology, 5, pp.140-149, Apr. 1995

[39] 高橋 真史, “携帯電話用マルチメディアSoCの開発,” 電子情報通信学会誌, Vol.84, No.11, pp.790-795, Nov. 2001.

[40] Chris Basoglu, Karl Zhao, Keiji Kojima, Atsuo Kawaguchi, “The MAP-CA VLIW-based Media Processor From Equator Technologies Inc and Hitachi Ltd.,” Equator Technologies, Inc.

[41] 中村 浄重, 酒居 敬一, 阿江 忠, “VLIW ハードウェアスタックプロセッサを用いたマルチメディアデータ処理,” 電子情報通信学会論文誌, Vol.J81-D-I, No.1, pp.21-27, Jan. 1998.

[42] 室山 誠一, 松島 敏雄, 村上 直樹, “情報通信用電源システムの課題と動向,” 電子情報通信学会論文誌, Vol. J83-B, No.5, pp.829-839, May 2001.

[43] 内山 邦男, “ディジタル民生用マイクロコンピュータにおける低電力設計,” 電子情報通信学会論文誌, Vol. J83-C, No.6, pp.447-453, June 2000.

[44] 大隈 孝憲, 石原 亨, 安浦 寛人, “可変電源電圧プロセッサに対するリアルタイムタスクスケジューリング手法,” Vol. J83-C, No. 6, pp. 454-462, June 2000.

[45] 中本 幸一, 辻野 嘉宏, 都倉 信樹, “携帯機器の 2 次電池の最長利用を目的とするリアルタイムスケジューリングアルゴリズム,” 電子情報通信学会論文誌, Vol. J83-D-I, No. 1, pp. 125-133, Jan. 2000

[46] 丸山充，中野治，西村一敏，“多重度の向上を目指したプロトコル処理技術の提案と評価,” 電子情報通信学会技術報告，IN93-113, Jan. 1994.

[47] 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.

[48] Hiroaki Nishikawa and Souichi Miyata, “Design Philosophy of Super-Integrated Data-Driven Processors: CUE,” Proc. of the 1998 Int. Conf. on Parallel and Distributed Processing Techniques and Applications, pp. 415-422, Las Vegas, U.S.A., July 1998.

[49] Marcel Mampaey, “TINA for Services and Advanced Signaling and Control in Next-Generation Networks,” IEEE Communication Magazine, pp.104-110, Oct. 2000.

[50] Gunnar Nilsson, Fabrice Dupuy and Martin Chapman, “An Overview of the Telecommunications Information Networking Architecture,” TINA95, Melbourne pp.1-12, Feb 1995.

[51] Hiroshi Ishii, Hiroaki Nishikawa, and Yuji Inoue, “Data-Driven Implementation of Highly Efficient TCP/IP Handler to Access the TINA Network,” IEICE Trans. Commun., vol. E83-B, no. 6, pp. 1355-1362, June 2000.

[52] Hiroaki Nishikawa, Souichi Miyata, Shin’ichi Yoshida, Tsuyoshi Muramatsu, Hiroshi Ishii, Yuji Inoue, and Ken’ichi Kitami, “Data-Driven Implementation of TINA kernel Transport Network,” Proc. of the TINA’97 Conf., IEEE, pp. 184-192, Santiago, Chile, Nov. 1997.

[53] 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,” 電子通信学会論文誌 (B-I), vol. J80-B-I, no. 6, pp. 457-464, June 1997.

[54] Hiroaki Nishikawa and Kazuhiro Aoki, “Data-Driven Implementation of Protocol Handling,” Proceedings of the 1998 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 430-437 (July 1998).

[55] Kazuhiro Aoki, Hiroshi Ishii, Souichi Miyata and Hiroaki Nishikawa, “Super-Integrated Data-Driven Processor for TINA-kTN Protocol Handling,” Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 998-1004 (June 1999).

[56] Shinya Kudo, Kazuhiro Aoki and Hiroaki Nishikawa, “Super-Integrated Data-Driven Processor Architecture for Interoperable Networking Environment,” Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 2167-2173 (June 2000).

[57] Kazuhiro Aoki, Shinya Kudo and Hiroaki Nishikawa, “Data-Driven Protocol-Handling for Interoperable Networking Environment,” Proc. of the 2001 Int.

Conf. on Parallel and Distributed Processing Techniques and Applications, pp. 243-249, Las Vegas, U.S.A., June 2001.

[58] Hiroaki Nishikawa, Kazuhiro Aoki and Hiroshi Ishii,"Data-Driven Implementation of Efficient Protocol Handlers," Proceedings of Scuola Superiore Guglielmo Reiss Romoli 2002w Computer & Internet Conference, CD-ROM (Jan. 2002).

[59] Kazuhiro Aoki and Hiroaki Nishikawa, "Data-Driven Implementation of Protocol Handling and Networking Interface for Networking Environment for Interoperable Networking Environment," Proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (to be published in June 2002).

[60] 西川博昭, 青木一浩, "プロトコル多重処理のデータ駆動型実現法とその実験的検討," 電子情報通信学会論文誌, Vol. J85-D-I, No. 7, (2002 年 7 月掲載予定).

[61] 青木一浩, 工藤慎也, 西川博昭, "ボトルネックのないレイヤ 2/3 間インタフェースのデータ駆動型実現法とその実験的検討," 電子情報通信学会論文誌 (投稿中).

[62] 小野沢 博文, "分散オブジェクト指向技術 CORBA," ソフト・リサーチ・センター, 1996.

[63] The Object Management Group, "CORBA/IIOP 2.2 Specification," Object Management Group, Inc. Publications, July 1998.

[64] 工藤誠也, 高木英明, 濱田元, 久保田文人, "ATM 網における圧縮動画像トラヒックの自己相似性," 電子通信学会論文誌, vol. J81-B-I, No. 9, pp. 549-556, Sep. 1998.

[65] 西川博昭, 我孫子泰祐, “タグ操作を許すデータ駆動プログラムの開発・再利用支援手法,” 電子情報通信学会論文誌 D-I J85-D-I 巻, 3 号, pp. 294-302, (2002 年 3 月).

[66] Yasuhiro Wabiko and Hiroaki Nishikawa, “Performance Prediction and Verification Environment for Super-Integrated Data-Driven Processors; RESCUE,” Transactions of the Society for Design and Process Science: Journal of Integrated Design and Process Science, Vol.5, No.4, pp.55-76, (Dec. 2001).

[67] Yasuhiro Wabiko and Hiroaki Nishikawa, “A Data-Driven Paradigm to Develop and Tune Data-Driven Realtime System,” Proc. of the 2001 Int. Conf. on Parallel and Distributed Processing Techniques and Applications, pp. 350-356, Las Vegas, U.S.A. June 2001.

[68] 今井恵一, “VoIP 実現上の課題,” 電子情報通信学会誌, Vol.83, No.4, pp.295-301 (2000).

[69] 浦田卓治, 樗林亮介, 西川博昭, “オンチップマルチプロセッサ型データ駆動アーキテクチャの評価手法とその実験的検討,” 情報処理学会論文誌, 42 巻 SIG 9(HPS 3) 号, pp. 135-144 (2001 年 8 月).

[70] Hiroaki Nishikawa, Hiroshi Ishii, Ryousuke Kurebayashi, Kazuhiro Aoki and Naohisa Komatsu, “Data-Driven TCP/IP Multi-Processor Implementation with Optimized Program Allocation,” Proceedings of the International Conference on Communication Systems, pp. 786-790 (Nov. 1998).

[71] Lixin Tao, “Shifting Paradigms with the Application Service Provider Model,” IEEE COMPUTER, vol. 34, no. 10, pp. 32-39, Oct. 2001.

付録 A

GIOP/IIOP および TCP/IP 処理の仕様

本章では，CORBA のオブジェクト間通信プロトコルである GIOP/IIOP，ならびにこれらの基幹プロトコルとなる TCP/IP 処理の実現法について，RESCUE を用いて作成した仕様を用いて概説する．まず，GIOP 受信処理部の仕様を図 A.1に示す．図に示す仕様では，オブジェクト間のメッセージ通信に不可欠な，メソッドの起動に用いるリクエストメッセージ，およびその応答であるリプライメッセージの処理を実現している．まず，後述する IIOP 受信処理部で行われるメッセージの復元処理が終了し，送られてきた GIOP メッセージは，そのメッセージのプロパティ，メッセージ長とともにバッファリングされる (図中 buffer message & set data)．並行して，GIOP メッセージのヘッダにあるメッセージタイプを参照し，送信したメッセージに対するリプライ (図中 Reply) であるのか，もしくは送信元からのリクエストメッセージ (図中 Request) なのかによって分岐する．送信したメッセージに対するリプライである場合には，図下部に示すように，リプライの状態を検査し (図中 check reply status)，リクエスト ID(図中 reqID) をもとに GIOP メッセージのデー

タ部 (図中 reply body)，メッセージタイプ，バイトオーダ，クライアント ID(図中 message type & byte order & clientID)，ならびにデータ部の長さ (図中 size of reply body) を上位層のオブジェクトに出力する (図中 output reply body)．送信元からのリクエストメッセージである場合には，まずメッセージのヘッダ部を参照し，後述する GIOP 送信処理において，リプライメッセージを生成するために必要なリクエスト ID(図中 requestID) を出力する (図中 prepare for sending Reply message)．リプライメッセージの生成と並行して，サーバまたはエージェントがオブジェクトを特定するための識別子であるオブジェクトキーを出力し (図中 output object key)，そのオブジェクトキーをもとにオペレーション名とリクエストメッセージのデータ部を出力する (図中 output operation name & reqbody)．ノード “output operation name & reqbody” に入力されるトリガ (図中 opname_size output trigger, opname_body output trigger, および body output trigger) は，メッセージのバッファリング時に生成される．ここで，オペレーションとは，オブジェクトの特定の実装から独立した，抽象的なサービスを表現するものである．このとき，オペレーション名，リクエストメッッセージのデータ部と同時に，上位層での処理に必要となるメッセージタイプ，バイトオーダ (図中 message type & byte order)，およびデータ部のサイズ (図中 size of request body) も出力される．

GIOP 送信処理部の仕様を図 A.2に示す．ノード “initialize GIOP send” は，GIOP 送信処理部の初期化処理である．GIOP 送信処理におけるヘッダ処理は，オブジェクトのデータ構造を示すインタオペラブル・オブジェクト・リファレンス ID(図中 IOR ID)，メッセージタイプ，プロパティ (図中 message type & property)，ならびにクライアント ID(図中 client ID) を入力し，初期化処理 (図中 initialize) およびインタオペラブル・オブジェクト・リファレンス ID の出力 (図中 copy) を行った後，これらの入力とメッセージ長 (図中 size of input stream) をもとに，メッセージ長 (図中 size of message, msgsize)，ならびにインタオペラブル・オブジェクト・リファ

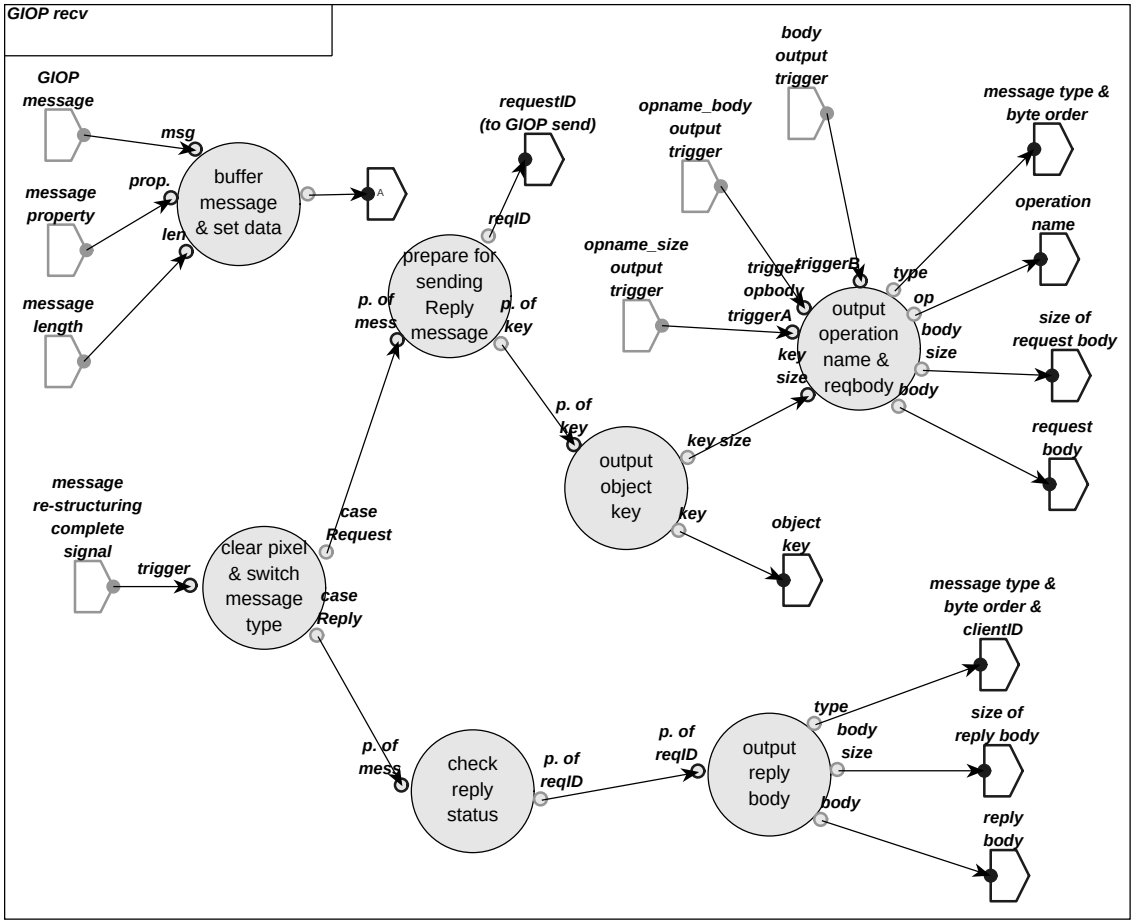

図 A.1: GIOP 受信処理部の仕様

レンス ID(図中 IOR ID) を生成し，出力するとともに，ノード “assemble message (header-side)” においてメッセージのヘッダ部を構成する．一方データ部は，図中のソース “input stream (tailer-side of message)” から入力される．入力されたデータ部の各要素は，ノード “add offset” において，本実現法メッセージの構成要素を識別する世代番号がヘッダの長さをもとに付与され，ヘッダ部とともに IIOP 送信部に出力される．

IIOP 受信処理部の仕様を図 A.3に示す．図上部のノード “initialize IIOP restruct

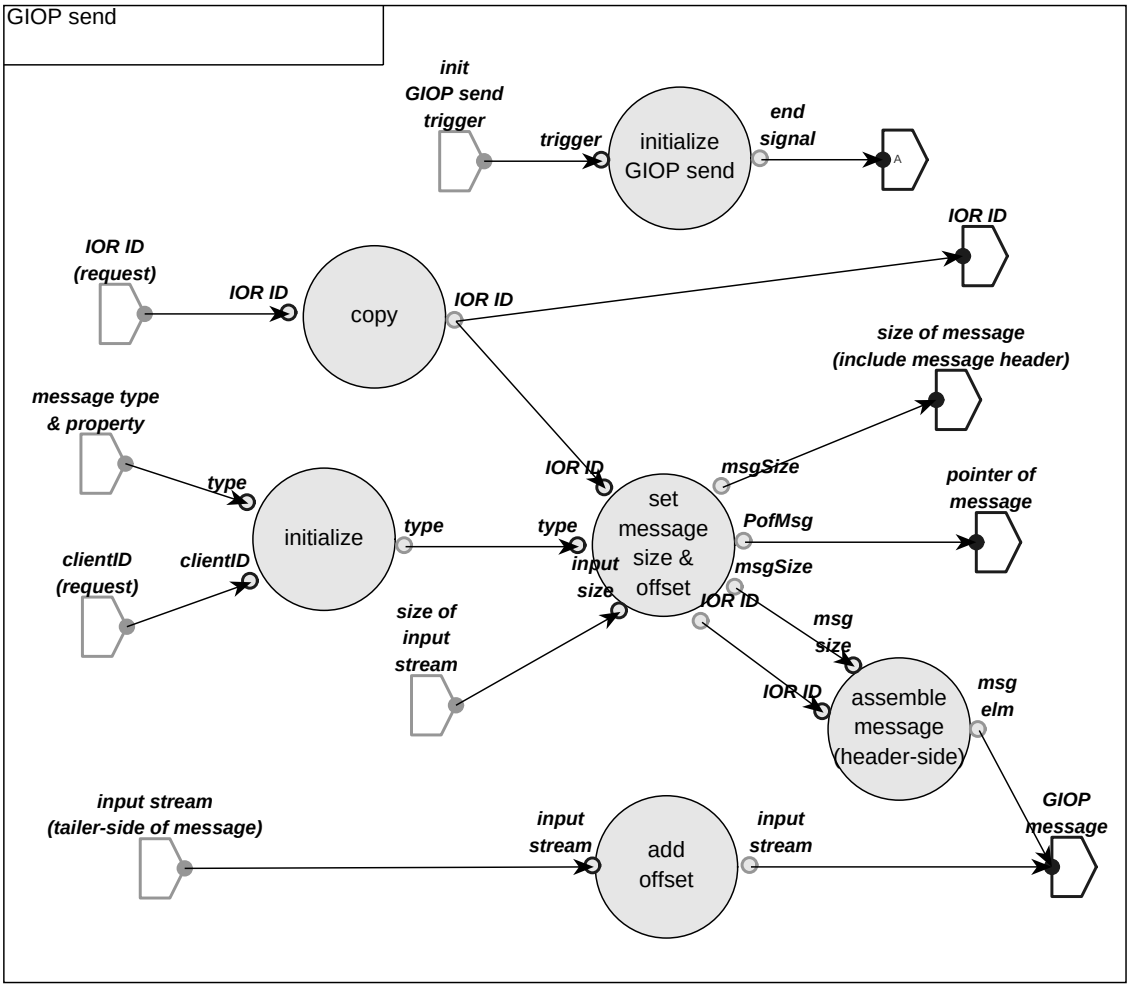

図 A.2: GIOP 送信処理部の仕様

profile” は IIOP 受信処理の初期化処理である．後述する TCP 受信処理部から，IIOP 処理要求 (図中 output demand) が送られると，ノード “check restructuring” において，メッセージの再構成が必要かどうかを検査する． メッセージの再構成が必要ない場合には，メッセージヘッダの検査 (図中 check message header) を経て，検査結果が正常ならば，メッセージ長 (図中 message size)，ならびにプロパティ (図中 message property) を GIOP 受信部に出力する (図中 set msgID to line)．このとき，

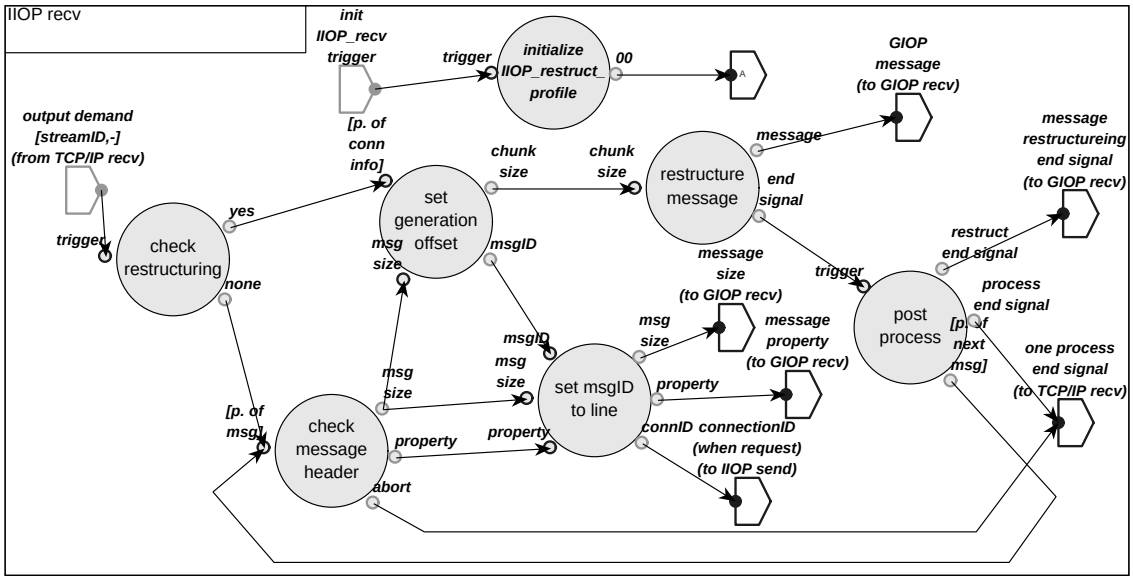

図 A.3: IIOP 受信処理部の仕様

メッセージの識別子であるメッセージ ID を，本実現法においてメッセージの識別に用いている世代に格納する．ヘッダ検査でエラーがあった場合には，TCP 受信部に通知する．メッセージの再構成が必要な場合にはまず，TCP セグメントごとに異なる世代をメッセージごとに統一する (図中 set generation offset)．その後，再構成後のメッセージ長をもとにメッセージを再構成し，再構成の完了通知と，再構成した GIOP メッセージを GIOP 受信部に出力するとともに，IIOP のヘッダ検査を並行して行う (図中 restructur message, および post process).

IIOP 送信処理部の仕様を図 A.4に示す．まず，上位層のオブジェクトよりメッセージ送信要求があった場合，IIOP 送信処理部では，オブジェクトより送られるインタオペラブル・オブジェクト・リファレンス (図中 IOR) を格納するとともに，その内容から ID を取り出し，オブジェクト・アダプタに出力する (図中 “store & interpret IOR)．さらに，GIOP 送信処理部から送られてくるインタオペラブル・オブジェクト・リファレンス ID(図中 IOR ID)，および TCP/IP より送られるコネ

クション ID(図中 connectionID) をもとに，コネクションの状態を検査する (図中 check connection status)．このとき，TCP/IP 処理に必要なアドレスなどの情報 (図中 encapsulated address information) を TCP/IP 送信部に出力する．また状態を検査済のコネクション ID(図中 connID) は，ノード “output message” に送られる．GIOP 送信処理部から入力されたメッセージ (図中 message) も，メッセージ長 (図中 message length) をもとにノード “buffer message” でバッファリングされ，ノード “output message” に送られる．ノード “output message” では，メッセージからセグメントへの変換処理が行われる．変換されたセグメント (図中 encapsulated data) は，送信要求 (図中 invoke trigger) とともに TCP 送信部へ出力される．なお，ノード “unlock accm (TCP chksum)” は，本実現法において TCP チェックサムに用いるアキュムレータへのロックの解除である．また，図下部のノード “init IIOP send” は IIOP 送信処理の初期化処理である．

TCP/IP 処理部の仕様を図 A.5に示す．仕様自体は，第 2 章に示した図 2.5と同様であるが，ここでは，GIOP メッセージの処理を含めて説明する．図上部のノード “init TCP/IP program” はルーティングテーブル (図中 routing hash table) やバッファ(図中 buffer) の設定を行う初期化処理である．また，図下部のノード “handle TCP re-transmit” は，再送処理である．ネットワークより送られてくる IP データグラム (図中 Received IP datagram) は，ネットワークインタフェースからの受信信号 (図中 IP receive initial signal) およびインタフェース識別子 (図中 received I/F indicator) とともに IP 受信部 (図中 IP recv) に入力される．IP 受信部では，ヘッダ検査が行われると同時に，データ部のバッファリングが行われる．IP 受信部からは，格納したデータ部のアドレス，ヘッダの IP アドレス情報，データ長，チェックサム結果がそれぞれ並列に出力される．それらを入力とする TCP 受信ヘッダ検査部 (図中 TCP recv 1/2) では，TCP ヘッダ検査が行われる．ヘッダ検査で異常がなければ，返信に必要なヘッダ情報を TCP 送信部に送る．同時に，データ長やコネ

クション管理における状態フラグが TCP 受信出力部 (図中 TCP recv 2/2) に送られる．TCP 受信出力部からは，IIOP のヘッダ，データ，オフセットおよび起動トリガが出力される．なお，IIOP のデータ部には，IP と TCP の関係と同様に，GIOP のヘッダ部が含まれている．TCP 送信部 (図中 TCP send) では，送るべきデータをバッファリングすると同時に，TCP ヘッダならびに疑似ヘッダを生成し，IP 送信部 (図中 IP send) に出力する．IP 送信部では，疑似ヘッダをもとに IP ヘッダを生成した後，IP データグラムであるフレームまたは PDU(Protocol Data Unit)(図中 Ethernet frame/AAL-5 PDU)，データ長，ネットワークインタフェースへのトリガを出力する．

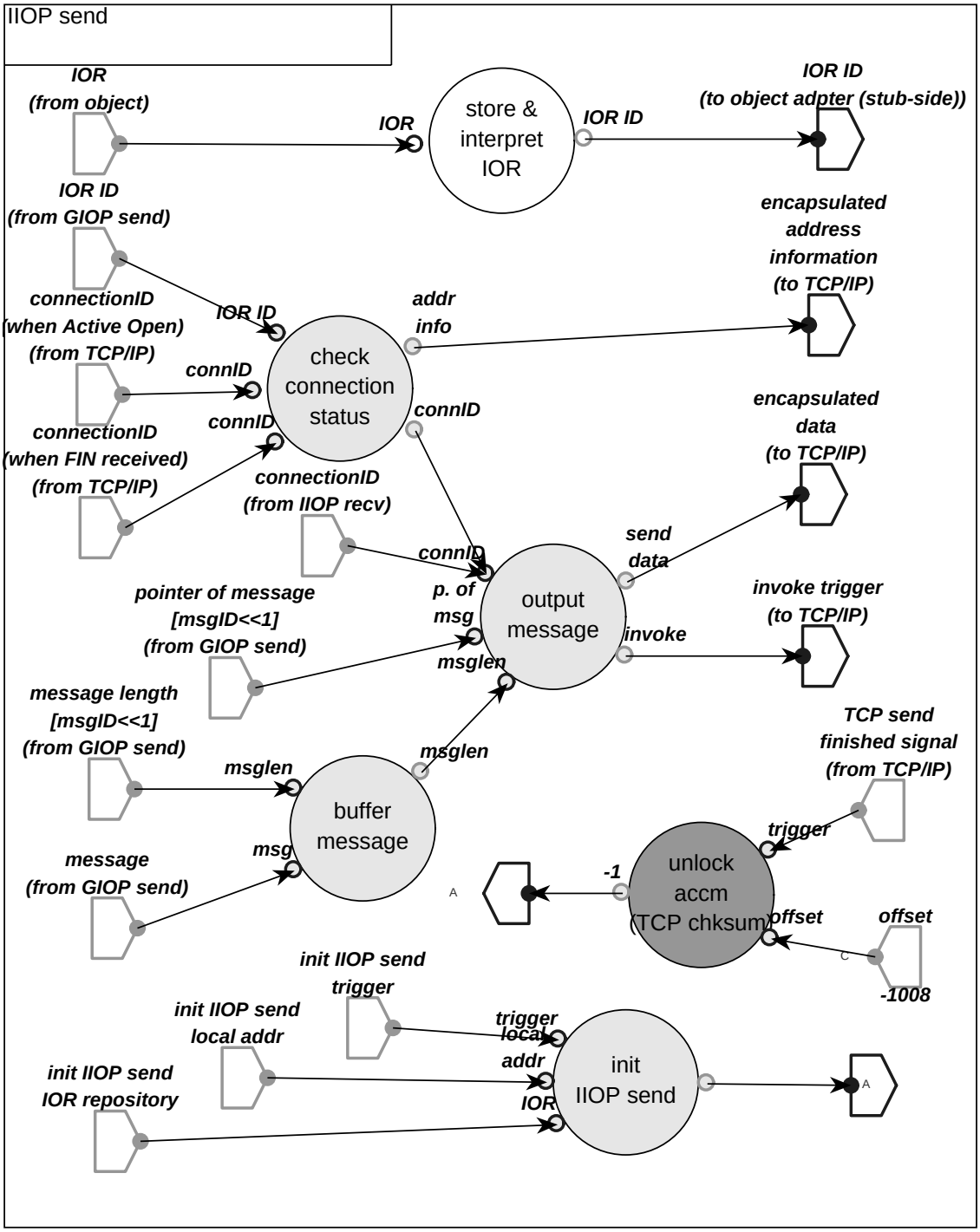

図 A.4: IIOP 送信処理部の仕様

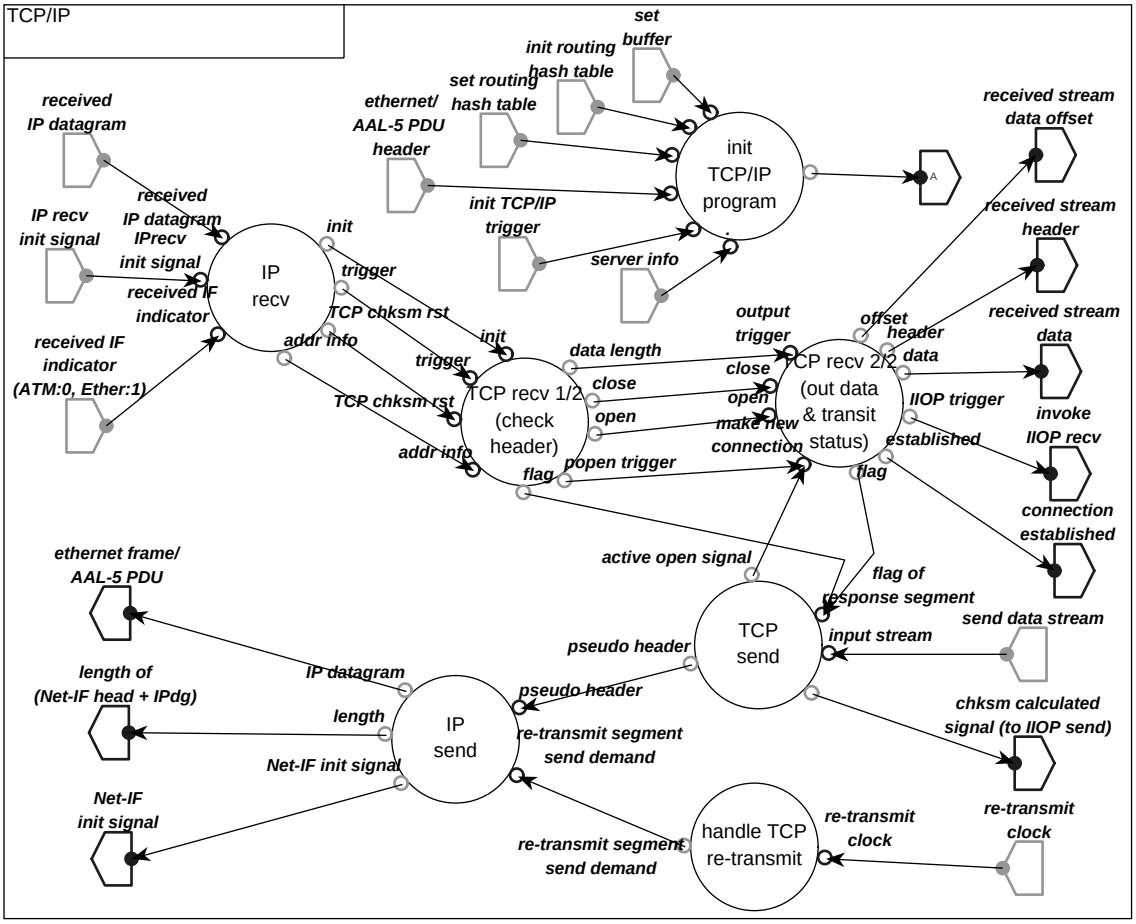

図 A.5: TCP/IP 処理部の仕様

付録 B

ネットワーキングインタフェースボードの仕様

本章は，CUE-v1 を用いて実現したネットワーキングインタフェースボードにおける，ATM および Fast Ethernet の送受信処理の仕様について概説している．本章で示す送信および受信プロセスの概要におけるデータの流れの方向は，左方向がネットワーク側，右方向が CUE-v1 側としている．送受信ともに，IP 処理の入出力データである IP データグラムが入力されると，行き先となるバッファのアドレス指定処理が行われる．さらに，CUE-v1 から FPGA への送信時には，ATM におけるチャネルやデータ長などの情報を持つパケットディスクリプタが出力される．また，ATM および Fast Ethernet コントローラへのバッファ参照要求や送信要求などのコマンドが送信される．また，データ受信通知やエラーについても，エラー内容の識別処理を行った後，CUE-v1 に送信される．正常に通信が行われている IP データグラムの実時間多重処理を実現するために，エラーに伴う処理は CUE-v1 で他の処理と多重に行う．実時間多重処理を実現し，ネットワークの転送能力を最大限に活用するために，本実現法では，CUE-v1 から FPGA への送信処理，ならびに FPGA か

ら CUE-v1 への受信処理を同時並行に処理可能とするとともに，送受信すべきデータである IP データグラムの送受信と，ATM のチャネルや IP データグラム長などの処理に必要なデータの送受信を独立に処理し，IP データグラムの送受信を阻害しないよう実現している．

まず，ネットワークインタフェースに ATM を選択した際の送信動作を図 B.1 に示す．送信する IP データグラムをバッファであるデュアルポートメモリ (図中 DPM: Dual Port Memory) に，ポインタや大きさを送信キューである FPGA 内部の RAM に保存する．その後 ATM コントローラ (図中 ATM-LSI) のコマンドレジスタへ送信キューに新規データグラムが追加されたことを通知すれば，ATM コントローラがセルを生成し物理層へ出力する．送信終了後，ATM コントローラは割り込みを発行するので，CUE-v1 は送信キューを読み込んで送信完了を確認する．例外処理として，このときエラーの発生がないか確認するとともに，エラーがあればその内容を参照する．一連の送信・受信動作は，ATM/Fast Ethernet の区別に関わらず多重に処理できる．

ATM における受信動作を図 B.2に示す．ATM コントローラ (図中 ATM-LSI) は，IP データグラムの先頭セルを受信すると，FPGA 内蔵 RAM よりバッファのアドレスを取得する．以降，セルを受信するたびに受信データをバッファに出力する．FPGA は，受信データをパケットに変換し CUE-v1 へ出力する．最終セルを受信すると，ATM コントローラはデータグラムを受信したチャネル番号を FPGA 内蔵 RAM に書き込み，割り込みを発行する．CUE-v1 は，FPGA 内蔵 RAM を読み込み，受信に使用したバッファが解放されたことを ATM コントローラに通知する．例外処理として，エラーが発生した場合には，ATM コントローラはエラーの情報を FPGA 内蔵 RAM に書き込み，割り込みを発行するので，CUE-v1 は FPGA 内蔵 RAM を読み込み，エラーの発生とその内容を確認する．

ネットワークインタフェースに FastEthernet を選んだ際の，送信動作を図 B.3に

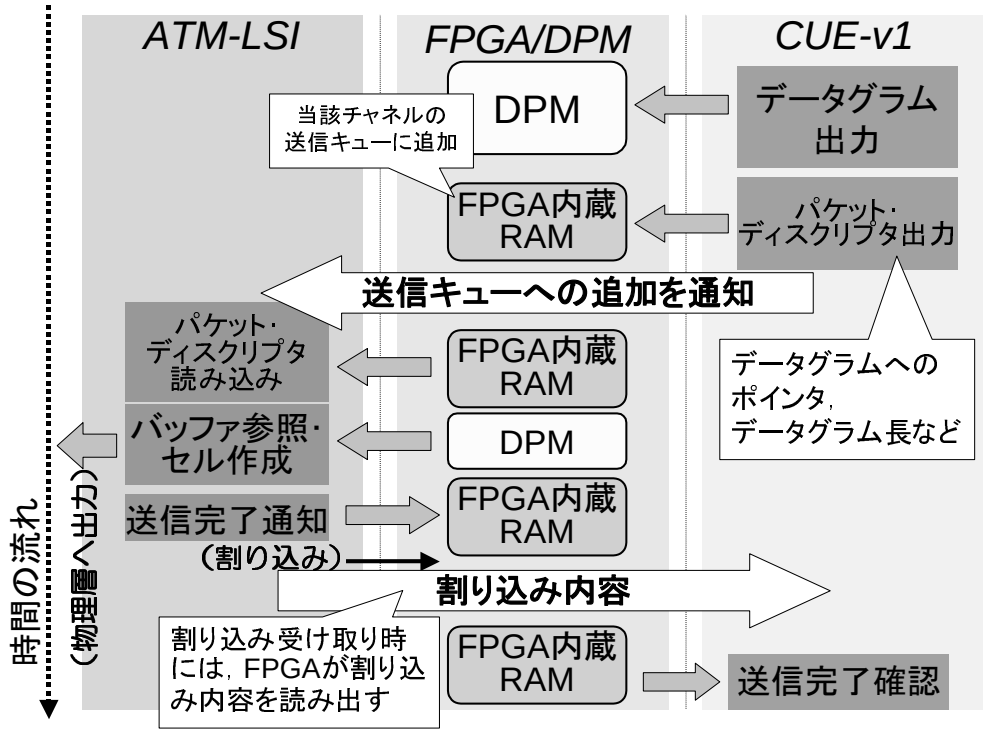

図 B.1: ATM の送信プロセス概要

示す．まず，CUE-v1 が Ethernet フレームを送信バッファであるデュアルポートメモリ (図中 DPM) に出力する．また，Ethernet フレームのポインタやフレーム長を含む送信コマンドをコマンドリストである FPGA 内蔵 RAM へ出力する．その後，Fast Ethernet コントローラ (図中 Ether-LSI) のコマンドレジスタへ送信コマンドが追加されたことを通知する．Fast Ethernet コントローラは，送信コマンドを受けて物理層へフレームを出力する．例外処理として，エラーがあった場合には，エラー情報をコマンドリストへ書き込んで割り込みを発行する．CUE-v1 は，コマンドリストを読み込み，エラーの発生ならびに内容を確認する．

Fast Ethernet における受信動作を図 B.4に示す．Fast Ethernet コントローラ (図

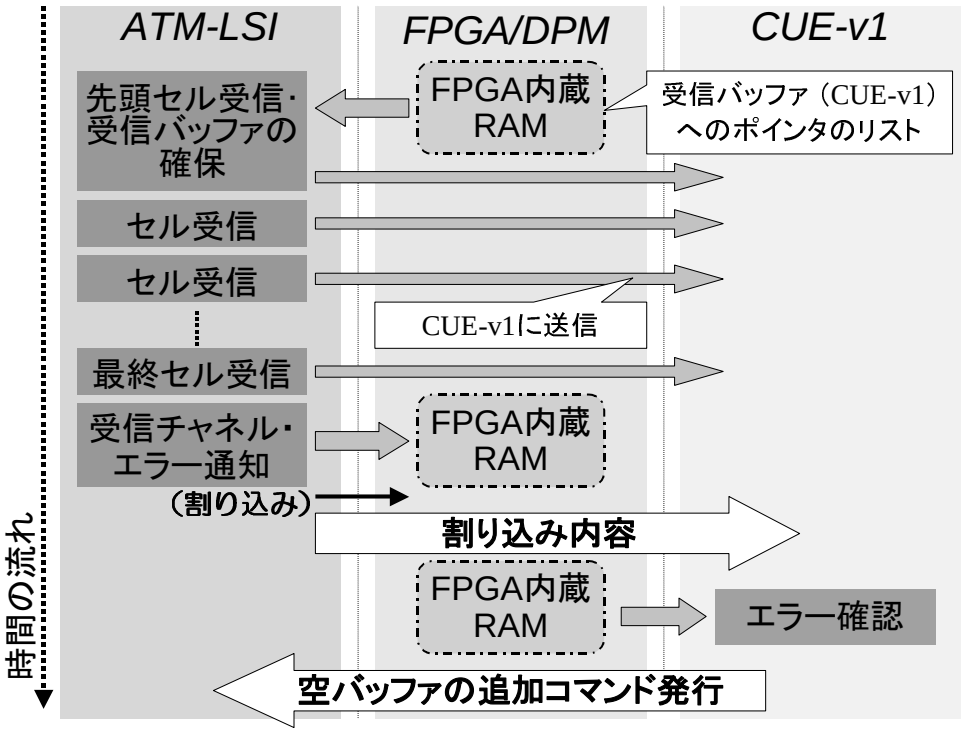

図 B.2: ATM の受信プロセス概要

中 Ether-LSI) は，フレームを受信すると RFD (Receive Frame Descriptor) より受信バッファのアドレスを得て，受信フレームをバッファへ出力する．FPGA は受信フレームをパケットに変換して CUE-v1 へ出力する．Fast Ethernet コントローラはフレーム受信完了通知として，その情報を RFD に書き込んで割り込みを発行する．CUE-v1 は RFD を参照し，受信に使用したバッファ領域を解放する．例外処理として，エラーが発生した場合には，Fast Ethernet コントローラはエラーの情報を RFD に書き込んで割り込みを発行する．CUE-v1 は RFD を読み込んでエラーを確認するとともに，受信に使用したバッファ領域を解放する．

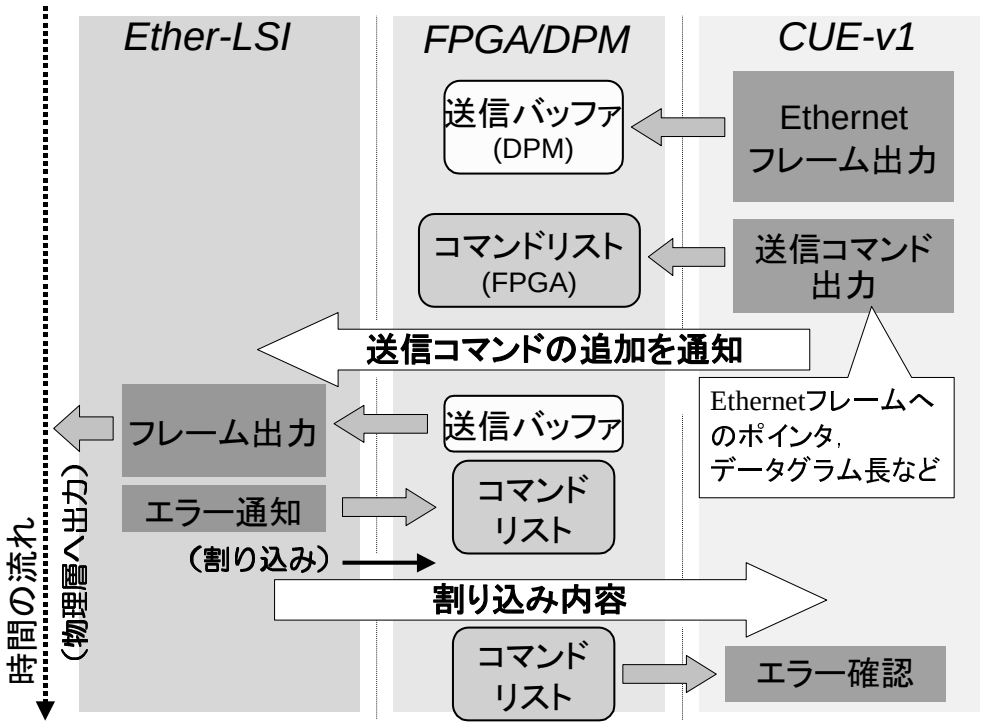

図 B.3: Fast Ethernet の送信プロセス概要

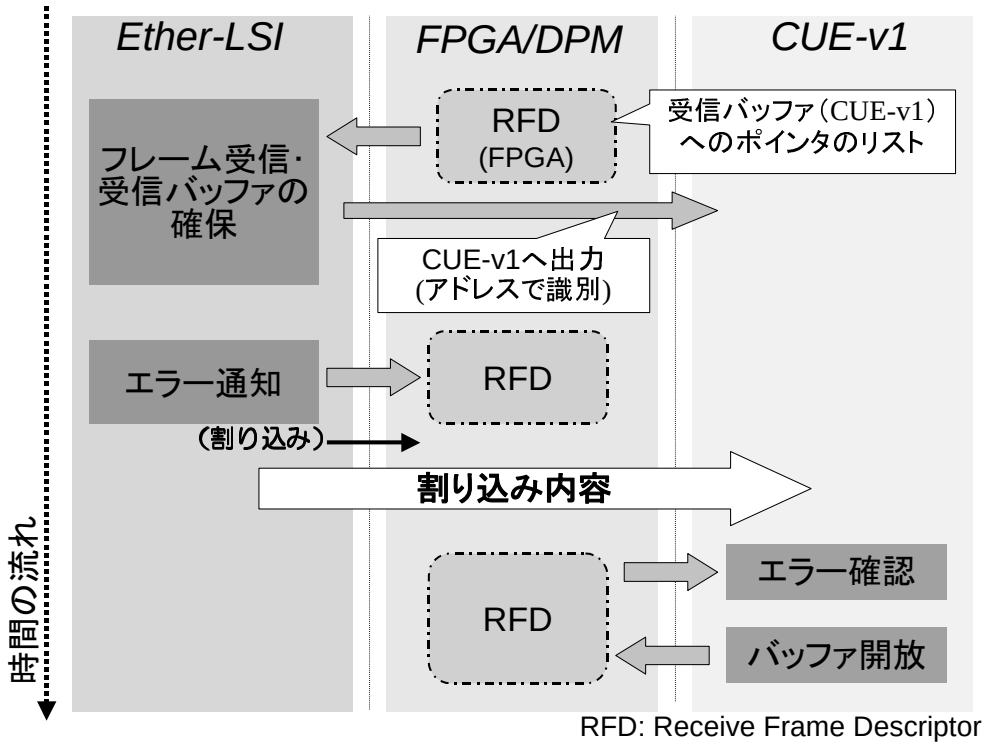

図 B.4: Fast Ethernet の受信プロセス概要