高効率化コンピュータ通信システム
に関する研究

２００３年１２月

辻岡（つじおか）　哲夫（てつお）

内容梗概

本論文は，ネットワークの高速化とコンテンツの大容量化に対応可能な高効率化コンピュータ通信システムの実現を目的として，新しい通信端末の構成法と柔軟性の高いネットワーク制御法を提案するとともに，その性能評価についてとりまとめたものである．本論文は，序論と結論を含めて 6 章で構成した．

第 1 章は序論であり，研究の背景と目的，次章以降の概要について述べた．

第 2 章では，大容量データの高速入出力に適した通信端末のハードウェア構成法について提案した．ハードディスク装置の高速化を実現するために汎用パーソナルコンピュータの内部バス上に複数の RAID（Redundant Arrays of Inexpensive Disks）コントローラを実装し，これらを同時に制御することで二階層の並列化を実現し，高速化を図った．さらに，大容量データの一括入出力動作に適したファイルシステムを提案し，ソフトウェア処理のオーバヘッドを削減した．これらコンピュータ内通信の高速化・大容量化を図ったシステムを試作し，最高 105M バイト/秒の連続読出し速度を実験により確認した．

第 3 章では，第 2 章で提案したハードウェア構成を汎用ワークステーション上に実装する場合のソフトウェア構成法を提案した．断片化した処理ブロックを再構成する機能と，中間バッファメモリを介在させない新しいインタフェースを実装することで，コンピュータ内通信のボトルネックを解消し，ワークステーションを用いた通信端末において約 110M バイト/秒の連続読出し速度を実験により確認した．

第 4 章では，パケットごとの締切り時刻に基づいたネットワーク制御法を提案した．従来方式とは異なるコネクション管理を必要としない柔軟性の高い QoS （Quality of Service：サービス品質）保証について検討した．すなわち，パケットごとに締切り時刻を与え，これに基づいた優先制御と経路制御を行い，コンピュータ間通信における柔軟なエンド・エンド間での遅延保証を実現した．シミュレーションにより, 従来方式と比較して高負荷まで適用可能であることを確認した．

第 5 章では，高信頼性通信において，再送回数に応じたパケットごとの締切り時刻の割当てと多重再送を用いることで，送信完了までの所要時間を制御する方法を提案した．こ

の提案方式はネットワークへの影響が小さく，遅延制御に対して有効に機能することを確認した．

第 6 章は結論であり，本研究で得られた成果について総括した．

目 次

図 目 次

表 目 次

第1章
序論

近年のコンピュータ通信の普及により，ネットワークの高速化とマルチメディア化が急速に進んでいる．ネットワークの高速化に関しては，回線とルータ装置の高速化を目的として，多くの研究・開発が進められており，近年における LAN の回線速度は 1Gbits/s に達し，WAN においても 100Mbits/s を超える FTTH サービスが提供されている．しかし，送信元からあて先までエンド・エンド間でのデータ転送の高速化を行うには，回線やルータ装置だけでなく，サーバやクライアントなどの通信端末を含めた通信システム全体，すなわち，コンピュータ内通信とコンピュータ間通信の双方の高速化・高効率化が必要となる．

現行のコンピュータ通信では，通信端末として汎用のパーソナルコンピュータが広く利用されているが，そこで用いられる CPU と内部バスの高速化，主記憶装置であるメモリの高速化・大容量化についても急速に進んでおり，ネットワークの高速化に対応できる十分な性能を備えつつある．しかし，外部記憶装置であるハードディスク装置の読出し速度・書込み速度やディスクコントローラなどの個々のハードウェア性能の限界，オペレーティングシステム（OS: Operating System）におけるソフトウェア処理のオーバヘッドの制約から，従来構成のパーソナルコンピュータをギガビットクラスの高速ネットワークに接続したとしても，ファイル転送において十分なスループットを得ることが困難となっている．したがって，エンド・エンド間でのデータ転送の高速化を実現するには，高速化ネットワークに対応した新しいコンピュータ通信システムの構成法が必要不可欠となっている．

一方，マルチメディア化については，文字・音声・映像データ，プログラムなどのコンテンツを柔軟に扱うことのできる Web サービスを中心として研究・開発が活発であり，コンテンツの大容量化が進んでいる．ネットワークの高速化に伴いデータ転送時間が短縮化され，これまでリアルタイム系で提供されていた VoD(Video on Demand) などのサービ

スも非リアルタイム系のダウンロード型通信サービスとして提供されるように変化しつつある．このことから，ニュースや電話といった即時性が要求される通信を除いて，今後は，大容量コンテンツのダウンロード型通信が主流になるものと考えられる．

通信の高効率化のためには，時間軸と空間軸の双方での制御が求められる．具体的には，優先制御による空き時間の活用と経路制御による空き回線の活用である．リアルタイム系と非リアルタイム系のコンテンツを同じ通信ネットワーク上で効率よく送受するには，それぞれの要求条件に沿ったQoS(Quality of Service) 保証実現のための制御（以下，QoS 制御と称する）が不可欠であるが，その必要性はネットワークの速度と密接に関係している．すなわち，コンテンツの大容量化によりQoS 制御の必要性が増すが，ネットワークが高速化すればQoS 制御を行わないベストエフォート型の単純な制御でも十分な品質が得られるようになり，更に高速化したネットワークがコンテンツの大容量化を導くことになる．したがって，相対的な尺度ではなく絶対的な尺度に基づいた，より現実に即した柔軟性の高いQoS 制御が求められる．現在検討されているQoS には最低帯域の保証や最大ロス率の保証などがあり，インターネットにおいてはIntserv [1] に代表されるコネクションごとの優先制御やDiffserv [2] に代表されるクラスごとの優先制御などが検討されている．また，周波数領域における最小帯域幅ではなく，時間領域における最小遅延に基づいたQoS 制御についての検討も進められている [3].

以上のことから，ネットワークの高速化とコンテンツの大容量化のもとで高効率化コンピュータ通信システムを実現するには，サーバやクライアントなどの通信端末を含めたエンド・エンド間での高速化の検討と，柔軟性の高いQoS 制御の実現がポイントとなる．特に，コンピュータ内通信の高速化・大容量化は重要な課題の一つであり，通信端末において回線速度と同等の速度で大容量データを入出力できる外部記憶装置を構築することができれば，ネットワーク内に配備されるルータ装置において通信データを長時間待ち合わせることが可能となり，より広い範囲で空き時間を活用することができるようになると期待される．

そこで，筆者は上述の背景を踏まえて，本論文において「高効率化コンピュータ通信システム」の実現を目的として行った研究成果をまとめた．

なお，筆者はこれまで主として，

（1）衛星通信における多次元符号化変調方式に関する研究 [4], [5]

（2）変曲点抽出によるパルス検出法を用いた受信回路に関する研究 [6]

（3）ツイストペア線を用いた高速伝送方式に関する研究 [7], [8]

（4）大容量ファイル瞬時一括転送方式に関する研究 [9]〜[11]
（5）超高速メディア転送システムに関する研究 [12]〜[17]
（6）蓄積型ネットワークにおける優先制御・経路制御法に関する研究 [18]〜[23]
（7）光直交符号に関する研究 [24], [25]
（8）超広帯域無線通信に関する研究 [26]
に従事してきた．本論文は（5）（6）の研究成果を取りまとめたものである．

本論文における各章の概要は次のとおりである．

第 2 章では，大容量データの高速入出力に適した通信端末のハードウェア構成法について提案する．外部記憶装置であるハードディスクの高速化を実現する方式として RAID（Redundant Arrays of Inexpensive Disks）方式 [27] があるが，ホストコンピュータと接続するインタフェース速度と RAID コントローラの処理速度が性能のボトルネックとなっていた．そこで，汎用のパーソナルコンピュータの内部バス上に複数の RAID コントローラを実装し，デバイスドライバがこれらの RAID コントローラを同時に制御することで，RAID コントローラ上と内部バス上の二階層のストライピングによる読出し・書込み処理の並列化を行い，データ転送の高速化を図る．更に，大容量データの一括入出力動作に適したファイルシステムを提案し，ソフトウェア処理のオーバヘッドの削減を行う．これらコンピュータ内通信の高速化・大容量化を図ったシステムを実際に試作し，実験による性能評価を行う．

第 3 章では，コンピュータ通信端末の OS として UNIX [28] を用いる場合のデータ転送システムについて検討する．UNIX ではファイルシステムとして FFS（Fast File System）[29] や UFS（UNIX File System）[30] が広く用いられているが，大容量化に適さない処理ブロックの断片化を伴い，大容量ファイルの入出力時に，ファイルシステムからデバイスドライバに対する多くのコマンドリクエストが発生する．また，UNIX の仮想記憶のメモリ管理機構によりユーザ空間とカーネル空間でのデータの受け渡しに Buffer Cache と呼ばれる中間バッファが介在し，これによって生じるメモリ間コピーのソフトウェア処理がシステム性能のボトルネックとなっている．これまでに中間バッファを介在させない方式として VIA（Virtual Interface Architecture）[31] などが報告されているが，これはコンピュータネットワークにおけるメモリ間通信の効率化を目的としており，メモリとファイル間での入出力の検討は行われていなかった．ここでは，断片化した処理ブロックを再構成することでコマンドリクエスト数を削減する手法と，中間バッファを介在させずに直接ユーザ空間のメモリと DMA（Direct Memory Access）転送する手法を提案し，コンピュータ内

通信におけるボトルネックを解消する．提案方式の有効性を実験により検証するために，実際に評価システムを試作する．2 本の内部バスを有するパーソナルコンピュータを用いることで，ハードディスク側とネットワーク側の両方から同時にメモリアクセスを可能とするシステム構成とする．第 2 章で提案したハードディスクの高速化手法と併せて適用することで，仮想記憶のメモリ管理機構が実装された UNIX を利用している通信端末においても，ネットワーク速度と同等のファイル入出力速度が実現できることを実験的に示す．

第 4 章では，パケットごとの締切り時刻に基づいたネットワーク制御法を提案する．エンド・エンド間での遅延を保証する QoS 制御の代表的なものとして，WFQ（Weighted Fair Queuing）[32] を用いた帯域予約方式と，フローごとの締切り時刻を優先度とする EDD（Earliest Due Date）方式 [33], [34] や EDF（Earliest Deadline First）待ち行列 [35]〜[43] を用いた方式などがある．これらはいずれも優先制御方式であり，各中継ルータ装置における最大遅延を保証する技術である．通信を開始する前にユーザのコネクション要求に従った経路選択と中継ルータ装置ごとの帯域予約の設定を行うことで厳密な遅延保証が可能となるが，ルータ装置における処理負荷が大きいといった問題がある．ここでは，これらの問題を解決するためにコネクション管理を必要としない柔軟性の高い QoS について検討する．具体的には，パケットごとに締切り時刻を与え，その情報に基づいた優先制御と経路制御によって，コンピュータ間通信における柔軟なエンド・エンド間での遅延保証の実現を目指す．

第 5 章では，パケットごとの締切り時刻に基づくネットワークを用いた高信頼性通信について検討する．これまでは，送受信バッファが有限である場合のスループットの改善 [44]，再送回数の削減 [45], [46]，サーバの処理負荷の軽減などについて検討が進められてきたが，ここでは，再送回数に応じて優先度の高い締切り時刻を割当て，多重再送と組合せることで送信完了までの所要時間を制御する方法を提案する．高信頼性通信においては，再送回数が増すごとに送信パケット数が少なくなり，再送パケットに対して高優先度を与えたり多重化を行ってもネットワークへの負荷が小さいことから，提案方式の有効性が期待される．

第 6 章は結論であり，第 2 章から第 5 章までの総括となっている．本研究で得られた成果をまとめるとともに，今後に残された課題について述べる．

第2章
大容量データの一括入出力を考慮した通信端末構成法

2.1 まえがき

大容量化するマルチメディア情報に対応して，非リアルタイム系のダウンロード型通信が増加しつつある．これまでリアルタイム系の通信で問題となっていたシグナリングや回線接続（コネクション）の保持に伴うルータ装置の処理負荷を軽減することを狙って様々な試みが行われているが，その一つとして，大容量ファイル瞬時一括転送システム [10], [11], [47] が提案されている．このシステムでは，図 2.1 に示すように，経済的に有利な汎用のパーソナルコンピュータ（PC: Personal Computer）上に通信ファイルサーバを構築する．高速ファイルシステムは，外部記憶装置であるハードディスクからデータをホストメモリに短時間で読出し，同時に，NIC（Network Interface Card）を介してメモリからネットワークにデータを送信する．受信側では逆の操作でデータをハードディスクに書込む．パーソナルコンピュータの技術進展は著しく，その性能水準は年々高くなる一方であり，高速 CPU，高速大容量ホストメモリ，高速内部バスが標準仕様として実装されている．内部バスとして広く用いられている PCI バス（Peripheral Component Interconnection Bus）[48]〜[50] は，32bit/33MHz で動作する場合であっても最大133Mbytes/s（約 1Gbits/s）の転送速度を提供し，ギガビットネットワークをサポートできる能力を有する．また，汎用のパーソナルコンピュータのプラットフォームを用いることは，汎用性，拡張性などの点でも利点が多い．

一般に，ダウンロード型の通信サービスを提供する場合，システム上で最も大きな性能ボトルネックとなるのは，外部記憶装置とメモリ間の通信速度（ハードディスクの読出し速度と書込み速度）であり，大容量データをネットワークと同等の速度で高速に入出力可能な記憶装置の実現が必要不可欠となっている．

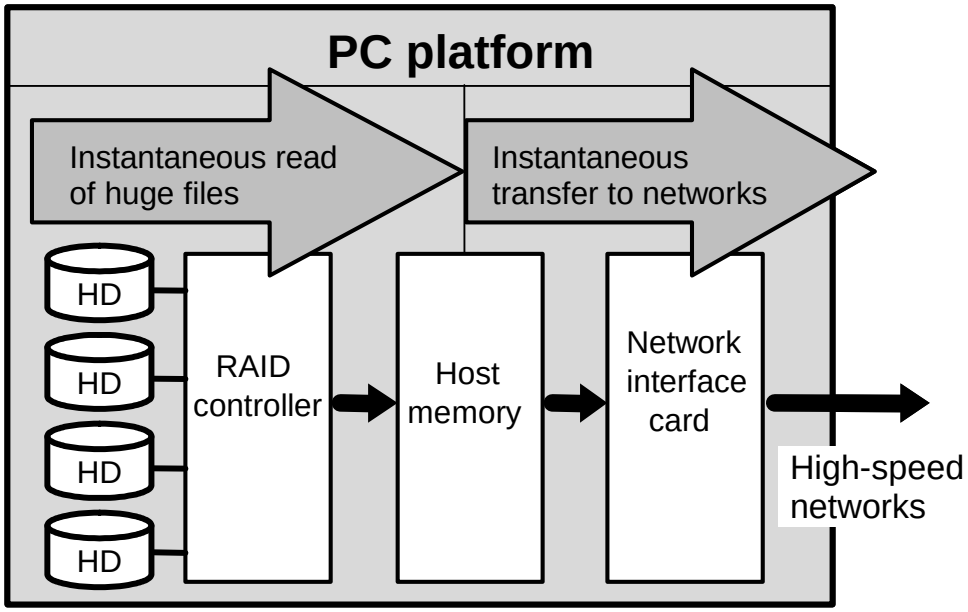


図 2.1　パーソナルコンピュータをプラットフォームとして用いた通信端末

本章では，大容量データの一括入出力を考慮した通信端末のハードウェア構成法について提案する．

大容量データを記憶する外部記憶装置としてハードディスクが広く用いられているが，データ転送速度はディスクの回転速度，記録密度，シーク時間，セットリング時間，回転待ち時間などの機械的な要因に大きく依存している．このため，プロセッサやメモリの性能向上と比較して，ハードディスクの性能向上の速度は緩やかであり，その差が顕在化しつつある．この問題を解決する方式として RAID 方式がある．RAID 方式は，データを複数のハードディスクに分配記録し，処理の並列化を行うことで，データ転送速度の改善を図る．信頼性が低下する問題を冗長ビットの付加と誤り訂正符号の適用で解決し，近年の高速記憶装置を構成する上での代表的な方式の一つとなっている．しかし，既存の RAID システムの多くは専用のプラットフォーム上に構築されるために経済的な問題を有している．また，SCSI や HIPPI，Fibre Channel などの汎用の接続インタフェースを経由してホストコンピュータから利用されるため，接続インタフェースの性能によってデータ転送速度が制限されるといった問題がある．

このような背景のもと，筆者はパーソナルコンピュータ上に高速 RAID ファイルサーバを構成する方法を提案する．図 2.1 に高速 RAID ファイルサーバのコンセプトを示す．パーソナルコンピュータの内部バス上に複数の RAID コントローラカード（PCI カード）を実装し，大容量ファイルをホストメモリに高速に一括して読出し，転送後のデータを同

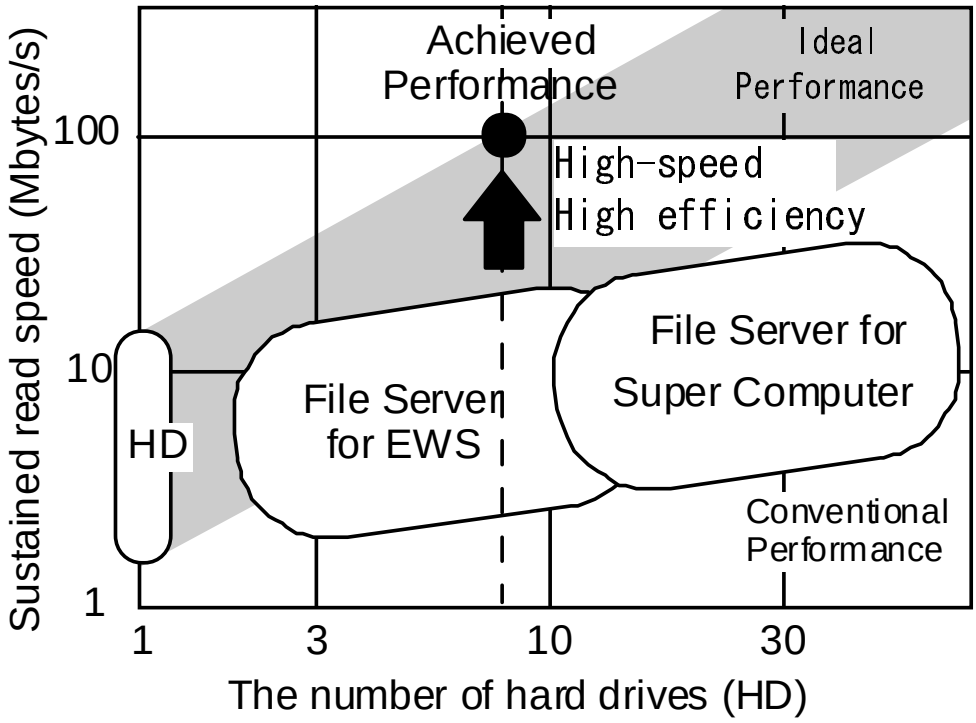


図 2.2　従来サーバのボトルネックと目標とする性能

じく内部バス上に実装される NIC などで利用する．このように，高速 RAID ファイルサーバは逐次処理を前提とした設計とし，大容量ファイルの連続読出し動作が主体となる用途をターゲットとしている．つまり，多ユーザに対して同時にデータ転送を行うシステムではなく，ユーザ一人一人に対して逐次処理でデータ転送を行うダウンロード型サービスを提供する送信側のシステムに適した設計とする．したがって，ランダムアクセス時や書込み時の性能よりも連続シーケンシャル読出し性能の改善に主眼をおいた最適化を行い，ハードディスク 1 台当りのデータ転送速度を改善し，システム全体の高速化・高効率化を目指す．なお，本サーバの応用例として，大容量ファイル一括転送サービス [10], [11], [47] のために開発した ATM-NIC を本システムに実装した通信端末としての利用を挙げる．

高速ネットワークを経由してダウンロード型の通信サービスを提供する場合，受信側のユーザ端末においてはハードディスクよりも高速な RAM ディスクやユーザプロセス上のバッファメモリ自体を最終あて先とする解もあり得るが，送信側のファイルサーバにおいては多くのユーザに対して多様なコンテンツを配信する必要があり，補助記憶装置の利用が不可欠である．したがって，ハードディスクの読出し速度の性能をネットワークと同等の速度にまで高めることが重要となる．

以上のことから，本章における研究の目的は，図 2.2 に示すように，従来，専用ハード

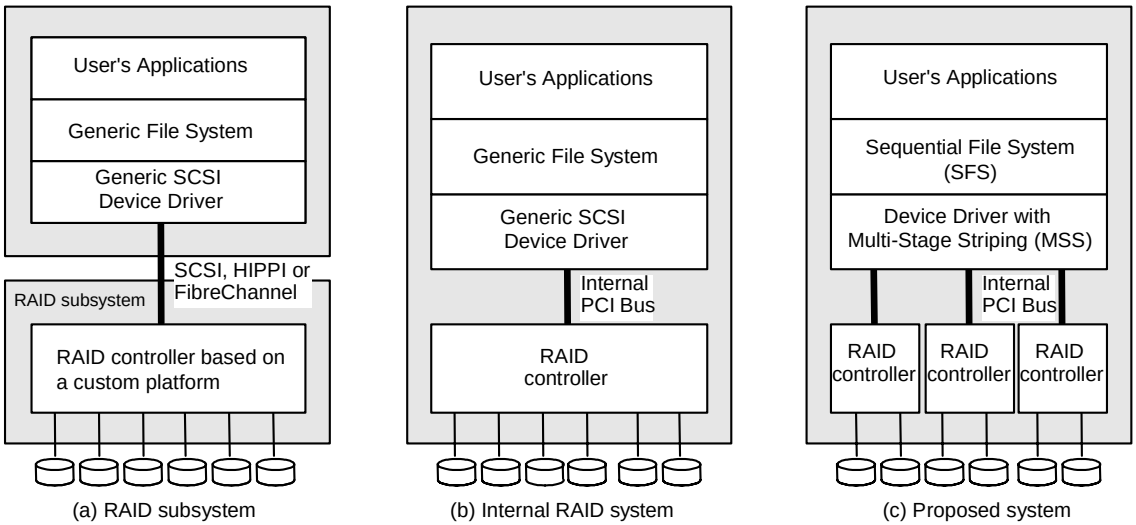


図 2.3　従来方式と提案方式の比較

ウェアを用いて数十台のハードディスク数という規模で実現していた 100Mbytes/s 以上のシーケンシャル読出し速度の性能を，汎用の PC プラットフォームを用いて少ないハードディスク数で実現することにある．この目標は，1Gbits/s-ATM，622Mbits/s-ATM のペイロード通信速度でのデータ送信に十分な性能として定めている．本章では，階層ストライピングとシーケンシャルファイルシステムの二つの手法を用いることによって複数の RAID コントローラを効率よく制御するハードウェア構成法を提案する．実際に汎用のパーソナルコンピュータを用いて高速 RAID ファイルサーバを試作し，実験により検討を行った結果を報告する．

2.2　高速 RAID ファイルサーバの構成法

2.2.1　従来方式

図 2.3（a），図 2.3（b）に従来の RAID 方式を適用したシステムのブロック図を示す．図 2.3（a）は，接続インタフェースを介してホストコンピュータに接続される RAID サブシステムの構成である．システムに特化した RAID コントローラを RAID サブシステム内に実装し，入出力データを一定長のブロック単位で複数のハードディスクに分配記録するストライピング処理（以下，ストライピングと称する）を行う．RAID サブシステムは，汎用の SCSI デバイスドライバで制御され，1 台の高速で容量の大きいハードディスクとして認識される．この見掛け上のハードディスクを論理 RAID ドライブと呼ぶこと

にする．この構成では，RAID コントローラを用いたストライピングによってハードディスク単体の性能限界の克服が可能となる．しかし，RAID サブシステムが良い性能を呈したとしても，データの入出力速度は接続インタフェースの速度に制限されることになる．

図 2.3（b）は内蔵型の RAID システムのブロック図である．この構成では，RAID コントローラは接続インタフェースを介さずに，ホストコンピュータの内部バスに直接接続されるため，接続インタフェースのボトルネックは排除される．しかし，RAID コントローラの性能限界が依然として存在し，システム全体の性能を制限する．また，単に複数の RAID コントローラを内部バス上に実装したとしても，それらは個別の論理 RAID ドライブとして認識されるに過ぎず，論理 RAID ドライブ単体の性能を改善することはできない．

更に，図 2.3（a），図 2.3（b）の従来方式のシステムでは，ファイルシステムにおけるホスト CPU のソフトウェア処理負荷がシステムの全体の性能を劣化させる問題がある．特に，ハードディスク上でのファイル配置と，ファイル配置テーブルの参照動作が，ソフトウェア処理のオーバヘッドの主な要因となる．一般的なファイルシステムでは，ファイル配置テーブルを用いてファイルの入出力を管理している．ファイルは一定長の管理ブロックに分割されてハードディスクに記録される．ファイル配置テーブルには，ハードディスク上でのファイルの格納位置を表すブロック番号リストが記録されている．上位のユーザプログラムがファイルデータを読出す場合，ファイルシステムは，まず，ファイル配置テーブルを参照し，要求されたファイルが記録されているブロック番号リストを得る．次に，このブロック番号リストに従って，1 ブロックずつ該当するブロックの内容を読出す．

例えば，MS-DOS®（Microsoft Disk Operating System）ファイルシステムにおいては，ファイル配置テーブルとして FAT（File Allocation Table）が用いられている．ファイルはクラスタと呼ばれるブロックに分割して管理される．クラスタはハードディスク上の複数のセクタから構成される．ファイルの格納位置を示すクラスタ番号リストが一方向リスト（Singly Linked List）形式で FAT に記録される．もし，クラスタ番号リストがシーケンシャルな連続番号となっていない場合は，ハードディスクへのアクセスはランダムアクセスとなり，シーク時間，セットリング時間，回転待ち時間などが原因となって，データ転送速度の大きな低下を招くことになる．よって，ファイルをハードディスク上にシーケンシャルに連続配置することが読出し速度の高速化を実現する上で重要となる．一括して読出すファイルのサイズが大きいほど連続配置によるデータ転送速度の改善効果が大きい

ことが予想される．また，シーケンシャルにファイルを配置したとしても，FAT を参照しながら一つずつクラスタにアクセスする動作は非効率であり，FAT の参照回数の削減も重要であるといえる．

これらの問題に対応するために，FAT の参照回数を削減し，RAID コントローラとメモリ間のデータ転送処理を DMA 転送で実行するなど，ホストコンピュータの性能を最大限に活用できる新しい方式の検討が課題となっている．

2.2.2 提案方式

2.2.1 節で述べた問題を解決するために，階層ストライピング（MSS: Multi-Stage Striping）とシーケンシャルファイルシステム（SFS: Sequential File System）の二つの手法を提案する．図 2.3（c）に提案方式を適用したシステムのブロック図を示す．階層ストライピングは複数の RAID コントローラを実装したハードウェア構成とデバイスドライバ内に実装する制御プログラムによって実現する．一方，シーケンシャルファイルシステムは従来ファイルシステムと置き換えて実装される新しいファイルシステムであり，ファイルの連続配置と連続 DMA 転送による高速読出し動作を行う．

2.2.2.1 階層ストライピング

RAID コントローラ単体の処理能力限界によるボトルネックを解消するために，階層ストライピングを提案する．入出力データは RAID コントローラのストライピングにより複数のハードディスクに分配記録されるが，階層ストライピングは，その更に上位層において複数の RAID コントローラに対してストライピングを行い，見かけ上の大きな論理 RAID ドライブを構成する手法である．この新しい論理 RAID ドライブは，従来の論理 RAID ドライブと同じ方法で利用することができる．

図 2.4 に階層ストライピングの概要を示す．階層ストライピングの制御プログラムが組み込まれたデバイスドライバ（以下，階層ストライピングデバイスドライバと称する）は，入出力データを複数の RAID コントローラに分配記録し，全体で，二階層のストライピングを行う．つまり，一階層目はそれぞれの RAID コントローラ上でのストライピングであり，二階層目はコンピュータの内部バス上でのストライピングである．内部バスとして PCI バスを用いた場合，その性能は，32bit/33MHz 動作時で約 1Gbits/s，64bit/66MHz 動作時で約 4Gbits/s であり，ハードディスクの単体性能，及び，RAID コントローラの単

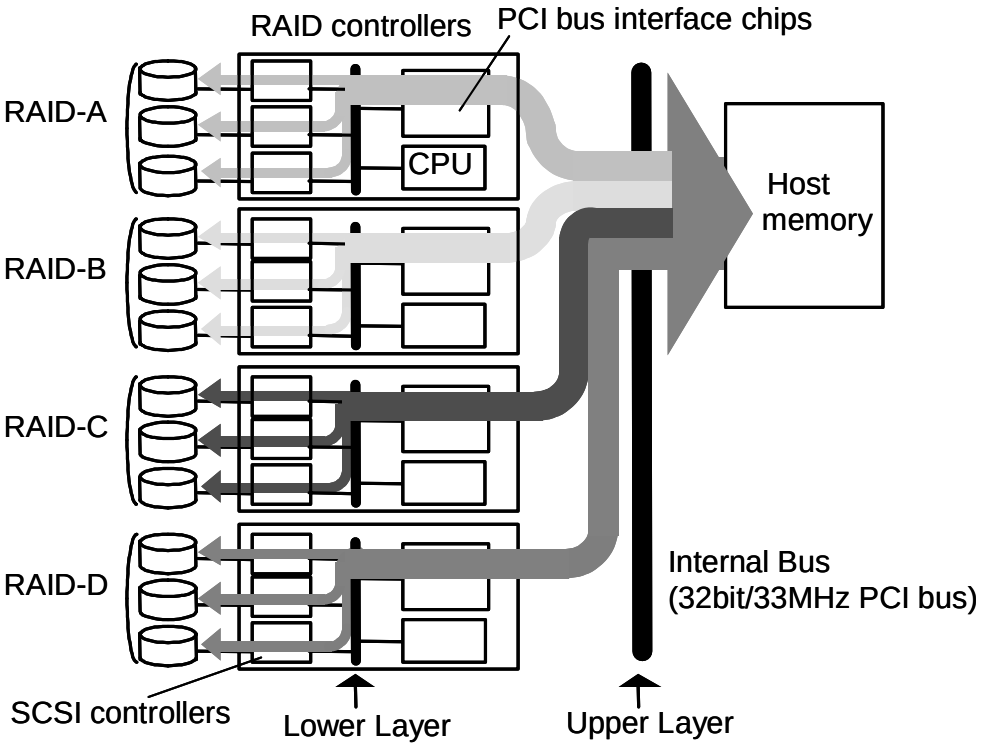


図 2.4　階層ストライピング（Multi-Stage Striping）

体性能よりも十分に高い．本提案手法の階層ストライピングにおいては，PCI バスを上位層のストライピングのプラットフォームとして活用し，RAID コントローラの性能限界によるボトルネックを解消することを目指している．階層ストライピングにより生成された論理 RAID ドライブは，一般の論理 RAID ドライブと同様に論理セクタ番号と領域サイズのリニアな一次元パラメータでアクセスすることができる．

なお，複数の RAID コントローラを個別の論理 RAID ドライブとしてシステムに認識させ，これらをソフトウェア RAID の既存手法で二階層のストライピングを行うことも可能である．しかし，この方式では，汎用の SCSI デバイスドライバを経由した Raw デバイスデータをストライピングするためにオーバヘッドが大きい．提案する階層ストライピングでは，高効率化のために，デバイスドライバのレベルで個別の論理 RAID ドライブのストライピングを行う点がソフトウェア RAID と大きく異なっている．

下位層のストライピングは各 RAID コントローラ上に実装される個々のローカル CPU によって制御・実行され，ホスト CPU による上位層のストライピングとは独立して並列に処理される．すべての RAID コントローラは PCI バスインタフェースチップ上にバスマスタ DMA 転送の機能を搭載し，ホスト CPU は RAID コントローラに対して DMA 転送の起動コマンドを発行するだけで，ソフトウェアの処理負荷の大きな増加を招くことな

く連続したDMA転送を行うことができる．すべてのRAIDコントローラに同時にDMA転送コマンドを発行し，複数のDMA転送を同時に実行する．各RAIDコントローラは連接する異なるメモリブロックの先頭アドレスに対してDMA転送を行うことで，入出力データのRAID-0（Pureストライピング）での分配記録を実現する．したがって，DMA転送のデータサイズは上位層のストライピングサイズと同じサイズとなる．ストライピングサイズが大きいほど，DMA転送コマンドの発行回数が減少し，動作効率が高まるが，小さいサイズのデータの入出力時において，複数のRAIDコントローラの同時動作が望めなくなり，逆に効率が低下することになる．ただし，DMA転送サイズが小さい場合の性能劣化を防ぐため，RAIDコントローラにはDMA転送のコマンドキューが搭載されており，トレードオフの関係がある点に注意する必要がある．

以上説明した階層ストライピングの適用によって，コンピュータの内部バスの利用効率を高め，システム全体の高スループット化を目指す．

2.2.2.2　シーケンシャルファイルシステム

シーケンシャルファイルシステムは，大容量ファイルの一括入出力時の性能改善に特化したファイルシステムである．階層ストライピングデバイスドライバが提供する論理RAIDドライブを高効率に最小限の劣化で利用することを目的として，ランダムアクセスの発生とソフトウェア処理のオーバヘッドを削減する．

図2.5に示すように，シーケンシャルファイルシステムは二つのコンセプトに基づいている．一つは，階層ストライピング論理RAIDドライブ上にファイルをシーケンシャルに配置することである．常にファイルを連続する論理セクタに配置することで，ランダムアクセスの発生をなくす．二つ目は，階層ストライピングデバイスドライバに発行するコマンド数の削減である．大量のコマンドリクエストはファイル配置テーブルの参照回数を招き，ソフトウェア処理のオーバヘッドを増加させる．

ファイルを必ず連続に配置することで，ファイルサイズと同じサイズの一つの大きなブロックにファイルが記録されていると扱うことができるようになる．これによって，コマンドリクエストの発行回数が1回で済むことになり，ファイルシステム経由のファイル入出力であっても階層ストライピングデバイスドライバの性能を劣化させることなく利用できるようになる．

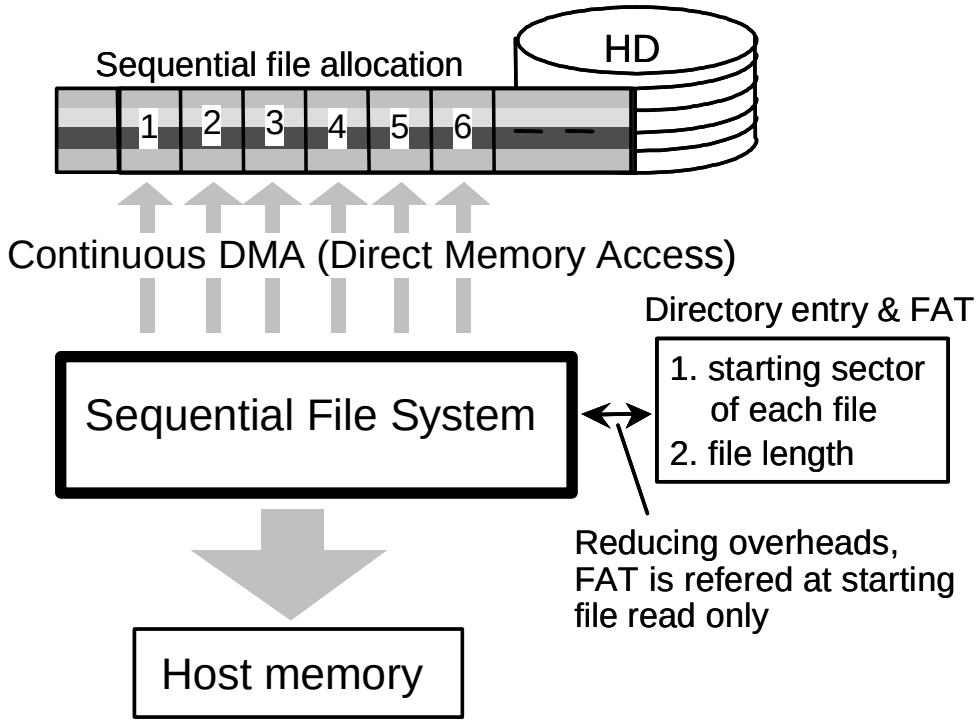


図 2.5　シーケンシャルファイルシステム（Sequential File System）

2.3　試作システム

実験により提案手法の有効性を確認するため，実際に高速 RAID ファイルサーバを試作した．図 2.6 に，その外観と諸元を示す．4 枚の RAID コントローラを汎用のパーソナルコンピュータの 32bit/33MHz 仕様の PCI バス上に実装する．ホスト CPU の OS として，組み込み向けの RTOS（Real Time Operating System）の一つである VxWorks® を利用した．この OS は仮想記憶管理の機能をもたないためプログラミングが容易であり，基本検討のための OS として適していることから採用した．ハードディスクについては容量 9GB（$= 9 \times 1024^3$bytes），回転速度 10000rpm，Ultra Wide SCSI-3 インタフェース（最高 40Mbytes/s）の仕様のものを用いた．各 RAID コントローラには 3 チャネルの SCSI コントローラ，ローカル CPU，DMA 転送時にイニシエータとなり得るバスマスタ機能を有する PCI バスインタフェースチップが搭載されている．試作システムで用いた RAID コントローラの基本性能を事前に評価した結果，各 SCSI チャネルに 2 台以上のハードディスクを接続してもデータ転送速度の飽和により性能の改善が見られなかったため，各 SCSI チャネルに 1 台ずつ，すなわち，各 RAID コントローラに 3 台ずつのハードディスクを

接続する構成としている．RAID コントローラ上のローカル CPU は，ホスト CPU が発行する DMA 転送コマンドを受けた後，3 チャネルの SCSI コントローラを独立して制御することでハードディスクとホストメモリ間の DMA 転送を同時に実行することができる．DMA 転送の並列実行中はマザーボード上のチップセットが PCI バスの調停処理を行い，各 RAID コントローラ上の PCI バスインタフェースチップ間でのアクセス競合を調整する．

図 2.7 は，試作システムにおける階層ストライピングとシーケンシャルファイルシステムの実装について示したものである．

図中の RAID-A から RAID-D は，下位層の論理 RAID ドライブを表す．これら 4 台の論理 RAID ドライブを束ねて，1 台の大きな論理 RAID ドライブを構成する．各 RAID コントローラ上で実行される下位層のストライピングについては，そのストライピングサイズを 128KB（$= 128 \times 1024$bytes）と設定する．これは，事前のベンチマークにおける最速の設定値である．平均故障間隔時間の短縮やエラーの増加などを招き信頼性が犠牲となるが，高速化のため，上位層のストライピングと同様に下位層のストライピングについても RAID-0 の Pure ストライピングで動作するように設定する．上位層のストライピングサイズについては任意の値を設定できるが，2.4 節で述べる実験において設定値を変えながら性能を評価することにする．なお，上位層のストライピングにおける 1 ブロックの入出力は 1 回の DMA 転送で行うとし，この結果，上位層のストライピングサイズと DMA 転送のブロックサイズは同じ値となる．例えば，DMA 転送のブロックサイズが 2MB（$= 2 \times 1024^2$bytes）の場合，ファイルデータを 2MB ずつのブロックに分割し，ファイルデータの先頭から 2MB を RAID-A に，次の 2MB を RAID-B に，同様にそれ以降 2MB ずつの領域を RAID-C，RAID-D，RAID-A，RAID-B，…と分配記録する．続いて，下位層の各 RAID コントローラは 2MB のブロックを 128KB 単位でそれぞれのハードディスクに更に分配記録する．

シーケンシャルファイルシステムについては，構造が単純な MS-DOS® ファイルシステムを修正したものを用いた．MS-DOS® ファイルシステムの場合，ファイルデータの先頭クラスタ番号とファイルサイズの情報はディレクトリエントリに，全体のファイル配置を示すクラスタ番号リストは FAT に格納されている．ファイルの書込み時には，上位層ストライピングの境界クラスタ位置からファイルデータの記録を開始し，必ず連続するクラスタに配置する．常にハードディスク上にファイルが連続配置されることから，ファイルの読出し時には，従来のように FAT を参照しながらクラスタを一つずつ読出すのではな

く，ディレクトリエントリを参照してファイルデータの先頭クラスタ番号とファイルサイズを得た後，すぐに連続シーケンシャル読出しの動作に移行することができる．FAT は空きクラスタの情報を得るために書込み時に参照・更新されるが，読出し時に参照されることはない．ディレクトリエントリと FAT は，ハードディスクの内周の末尾クラスタにまとめて配置する．一般に内周のセクタは外周のセクタよりも転送速度がわずかに遅いが，参照頻度が少ないため，ファイルサイズが大きい場合は相対的なアクセス時間の割合は小さく，ほとんど無視することができるようになる．

連続シーケンシャル読出しの動作は，連続 DMA 転送によって実行される．この転送処理は，デバイスドライバが階層ストライピングの処理と併せもって行う．このとき，添字が同じ上位層のブロック（A_i, B_i, C_i, D_i）（＝グループ i）のそれぞれをグループ化して扱い，各 RAID コントローラの同じ論理セクタ領域に格納し，これによって，セクタ管理と入出力制御の簡単化を図る．すなわち，同じグループ i に属するブロックについては常に同時に読出し/書込みコマンドを発行する．具体的な連続 DMA 転送の読出し手順は，まず（A_1, B_1, C_1, D_1）のブロック領域を同時に DMA 転送で読出し，続いて（A_2, B_2, C_2, D_2）,（A_3, B_3, C_3, D_3），…のようになり，上位層ストライピングのブロック単位で DMA 転送を連続して実行する．ファイルデータの先頭位置を上位層ストライピングの境界セクタの先頭位置に揃えた理由は，境界が揃っていないことによる例外処理をなくし，ファイルデータを先頭から読出す場合に，グループ 1 の初回アクセスから連続 DMA 動作に入れるようにするためである．

#RAID controllers	: 4
Hard Drives (HD)	: Seagate Cheetah-9 (9GB, 10000rpm)
SCSI channel	: Ultra Wide SCSI-3 (max. 40Mbytes/s)
HD Internal Speed	: 16.8Mbytes/s (outer) and 11.3Mbytes/s (inner)
#HDs for RAID-0	: 8 (Total Capacity = 72GB)
Mother Board	: DOS/V PC mother board
Host CPU	: Pentium 200MHz
Host Memory	: EDO-DRAM (512MB)
Memory Speed	: 133Mbytes/s (72bit, 16.67MHz)
#PCI buses	: 1
Dimensions	: 213mm(W) x 510mm(H) x 445mm(D)

図 2.6　試作システムの外観と諸元

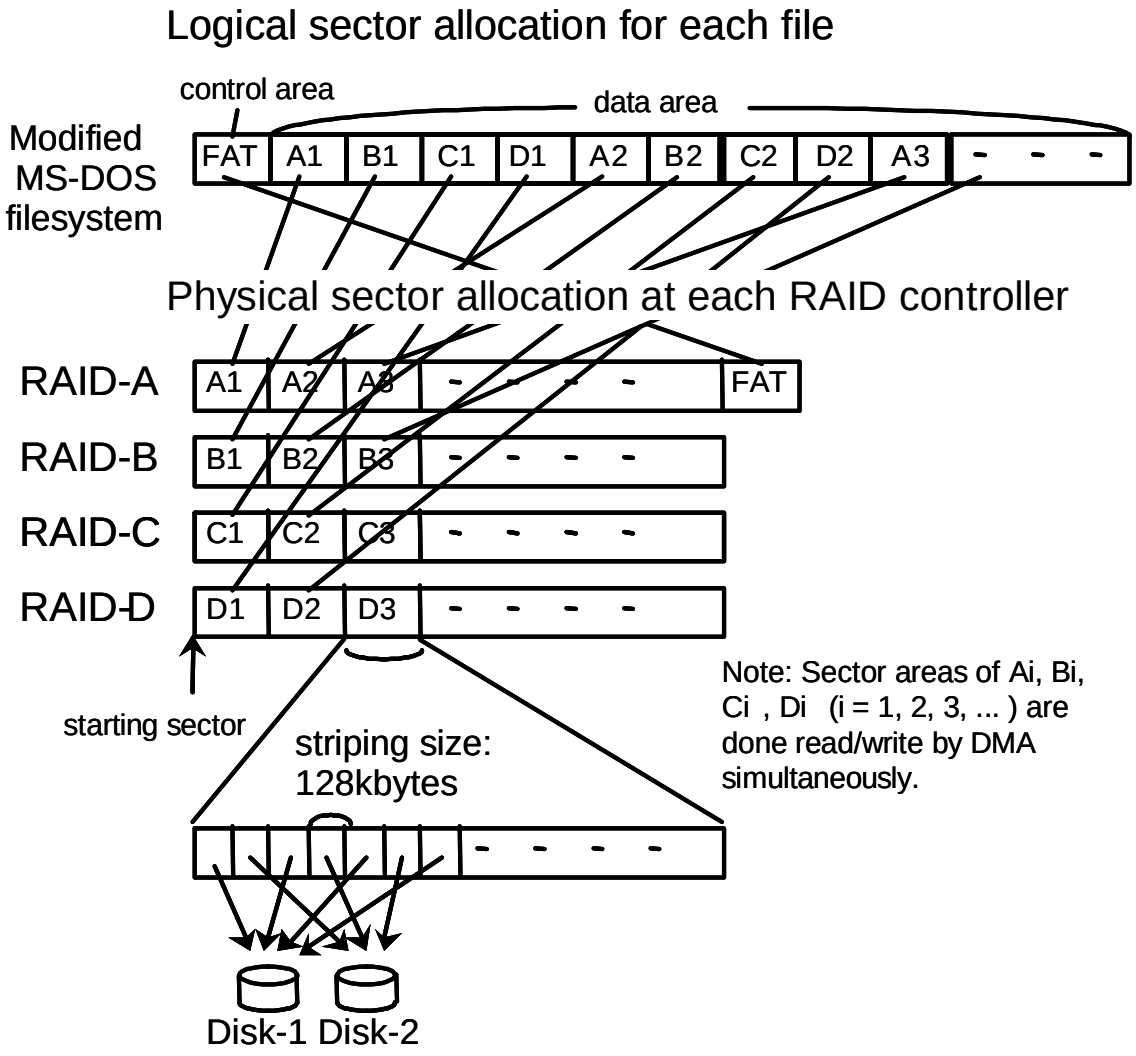


図 2.7　試作システムにおける階層ストライピングとシーケンシャルファイルシステムの実装

2.4 実験による性能評価

本節では，試作した高速RAIDファイルサーバについて，実験により性能評価を行う．

評価内容は，

（a） ハードディスクからホストメモリへの連続シーケンシャル読出し速度

（b） ホストメモリからハードディスクへの連続シーケンシャル書込み速度

（c） RAIDコントローラー上のキャッシュメモリからホストメモリへの読出し速度

の三項目である．

ここでいう連続速度とは，瞬間のピーク転送速度ではなく，持続的な平均転送速度を表す．また，シーケンシャル読出し/書込みとは，ランダムアクセスを伴わない読出し/書込み動作を表す．

測定に用いたファイルのサイズは256MBであり，ファイルデータの先頭から末尾までの全体の読出し/書込みを1回のシステムコールで実行し，それに要した時間を測定することによって速度を求める．なお，ハードディスクの最外周付近の入出力速度が高速なセクタにファイルを配置する．

ストライピングしたブロックに対して複数回のDMA転送に分けてアクセスすることも考えられるが，速度劣化を伴うため，本性能評価では，上位層ストライピングサイズとDMA転送サイズを同じ値とする．RAIDコントローラ数が1〜4枚の条件について，DMA転送サイズを32KBから3MBまで変化させながら性能を測定し，上位層のストライプサイズの最適値を導出する．各RAIDコントローラには2台ずつのハードディスクを接続する．よって，1〜4のRAIDコントローラ数に対するハードディスク数は2，4，6，8となる．

測定結果を図2.8〜図2.11にそれぞれ示す．まず，全体を通して（a）の連続シーケンシャル読出し速度の特性を見ると，DMA転送サイズが128KB未満の領域で速度が急激に落ち込んでいることがわかる．この理由は，上位層のストライピングサイズが下位層のストライピングサイズの128KBよりも小さくなると，各RAIDコントローラ上のローカルCPU処理によるバッファリング処理が発生し，オーバヘッドが増加するためと考えられる．

図2.8のRAIDコントローラ数が1枚の結果は，階層ストライピングが適用されず，シーケンシャルファイルシステムによる連続DMA転送だけが適用された場合の特性である．同じ条件のハードウェア構成で，従来デバイスドライバと従来MS-DOS® ファイルシステムを用いてファイルの読出し速度を測定すると約15Mbytes/sであった．図2.8の

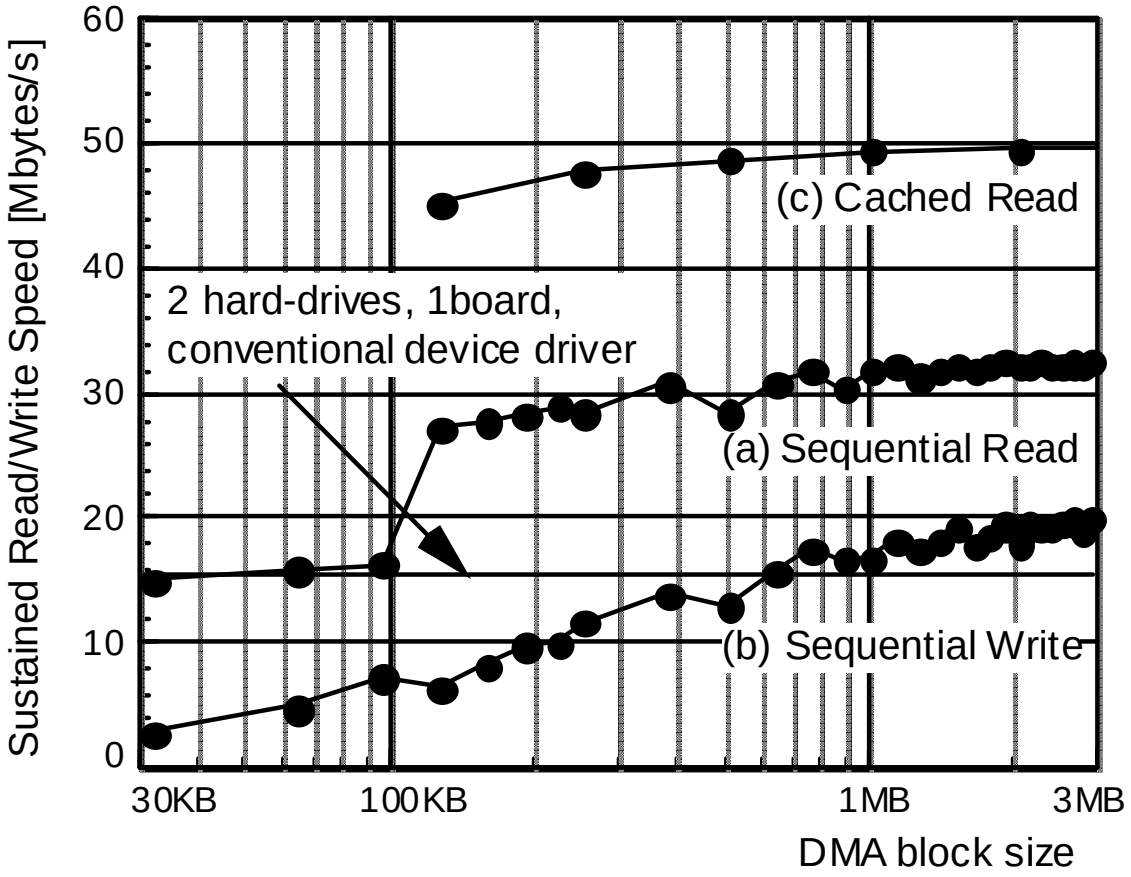


図 2.8　ファイルの連続シーケンシャル読出し速度/書込み速度の測定結果
（RAID コントローラ数： 1 枚）

（a）の特性と比較すると，シーケンシャルファイルシステムによる連続 DMA 転送によって FAT の参照回数が削減され，約 33Mbytes/s にまで特性が改善されていることがわかる．また，図中の（c）は RAID コントローラ上のキャッシュに常にヒットする条件下での連続読出し速度を示している．具体的には，同一グループ（A_1, B_1, C_1, D_1）のハードディスク領域を繰返し読出すことで測定した性能である．この特性を見ると，DMA 転送サイズが十分大きい 1MB 以上の領域では約 50Mbytes/s となっている．以上のことから，2 台のハードディスクが接続された RAID コントローラの性能限界は約 33Mbytes/s であり，ハードディスクとその入出力処理を除いた RAID コントローラ単体の処理能力の限界は約 50Mbytes/s であることがわかる．

図 2.9 は，RAID コントローラ数が 2 枚の場合の特性であり，シーケンシャルファイルシステムに加えて，階層ストライピングが適用されている．(a）の連続シーケンシャル読出し速度は約 55Mbytes/s, (c）のキャッシュ読出し速度は約 99Mbytes/s に達している．

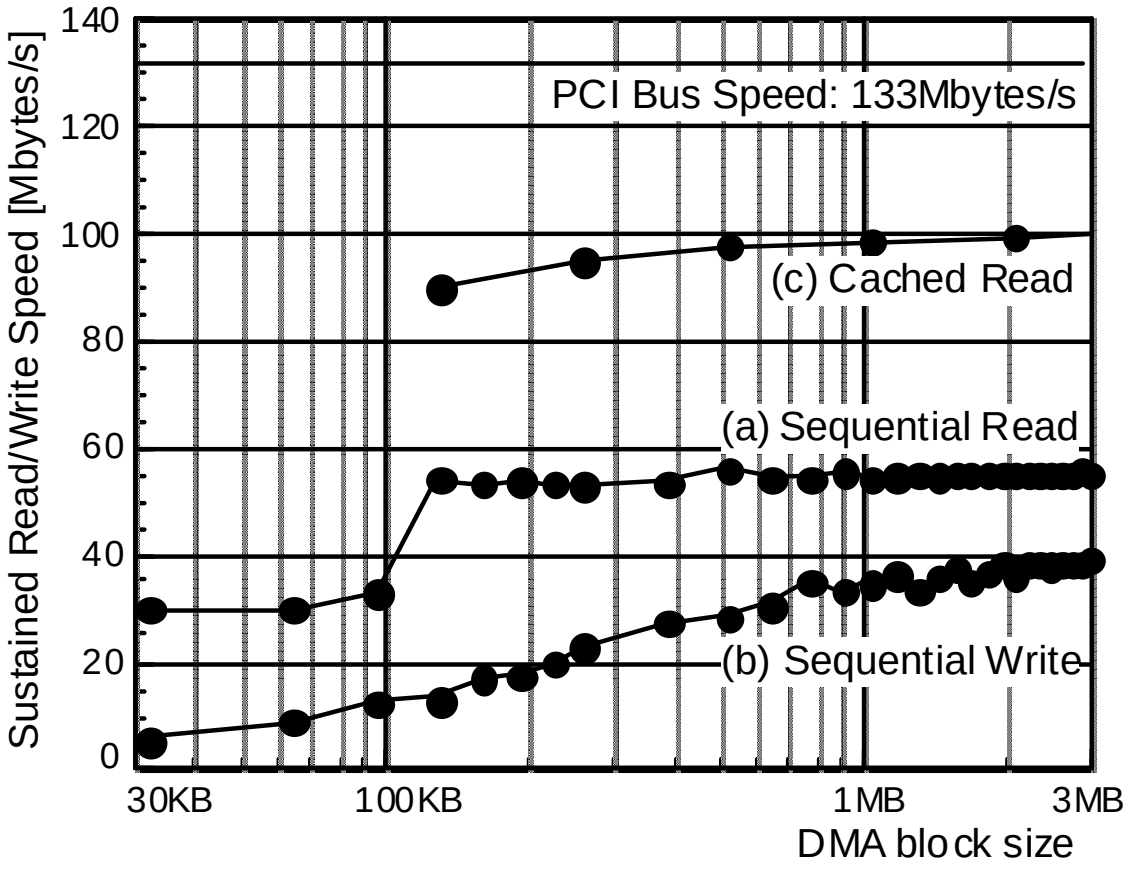


図 2.9　ファイルの連続シーケンシャル読出し速度/書込み速度の測定結果
（RAID コントローラ数： 2 枚）

図 2.8 と比較すると，PCI バスの調停処理による若干の性能劣化が見られるが，およそ 2 倍の性能を呈している．

図 2.10，図 2.11 は RAID コントローラ数が 3 枚，及び，4 枚の場合の特性である．DMA 転送サイズが 128KB のとき，(a）の連続シーケンシャル読出し速度にピークが見られ，最大の性能が得られている．これは，DMA 転送サイズと下位層ストライピングサイズが同じ場合，ローカル CPU における分配記録の処理が低減されるためと考えられる．

DMA 転送サイズが 128KB のときの最大の連続シーケンシャル読出し速度は，RAID コントローラ数が 3 枚，ハードディスク数が 6 台の場合で 80.6Mbytes/s であり，また，RAID コントローラ数が 4 枚，ハードディスク数が 8 台の場合で 105Mbytes/s であった．すなわち，当初に目標としていた 1Gbits/s-ATM，622Mbits/s-ATM のペイロード通信速度と同等の 100Mbytes/s 以上の連続シーケンシャル読出し速度を実現できた．

以上により，本試作システムが大容量ファイルの瞬時一括読出しに十分な性能を有していることがわかった．例えば，CD-ROM 1 枚分の 600MB のコンテンツデータファイルの

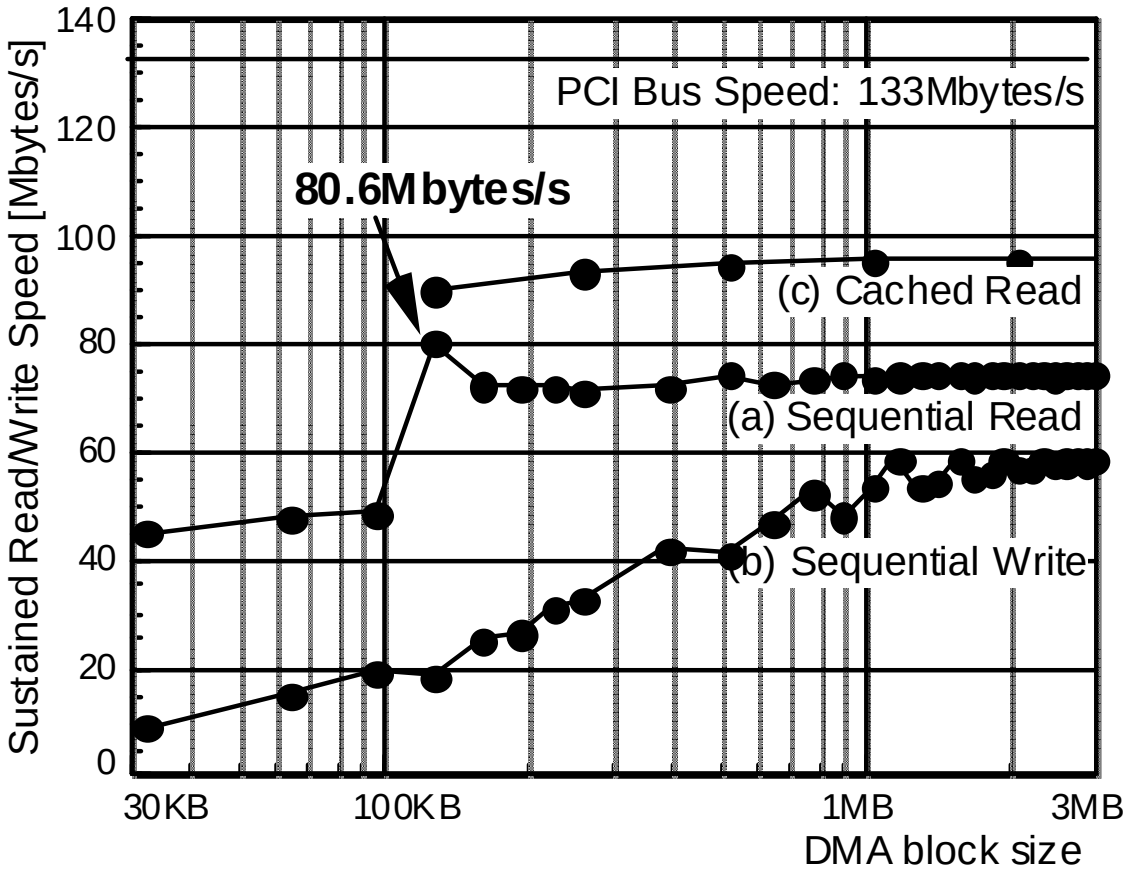


図 2.10　ファイルの連続シーケンシャル読出し速度/書込み速度の測定結果
（RAID コントローラ数： 3 枚）

読出しに 6 秒程度，約 5 分の MPEG-1 形式の動画ファイルの場合では，0.5 秒程度しか必要としない．

続いて，(b）の連続シーケンシャル書込み速度について見ると，RAID コントローラ数にほぼ比例して速度が増加していることがわかる．また，DMA 転送サイズが大きくなるほど良い性能が得られている．これは，RAID コントローラが Write Through モードのキャッシュ設定で動作しているため，書込み時のコマンドキューの利用が抑制され，コマンドリクエストを受信するごとに書込み動作が行われているためであると考えられる．最大の連続シーケンシャル書込み速度は，RAID コントローラ数が 4 枚，ハードディスク数が 8 台の場合に得られ，DMA 転送サイズが 3MB の場合で約 92Mbytes/s であった．

最後に，読出し速度のセクタ位置依存性について調べる．試作システムで用いたハードディスクは，内部転送速度の仕様が外周のセクタにおいて 16.8Mbytes/s，内周のセクタにおいて 11.3Mbytes/s となっている．そこで，試作システムにおいて階層ストライピン

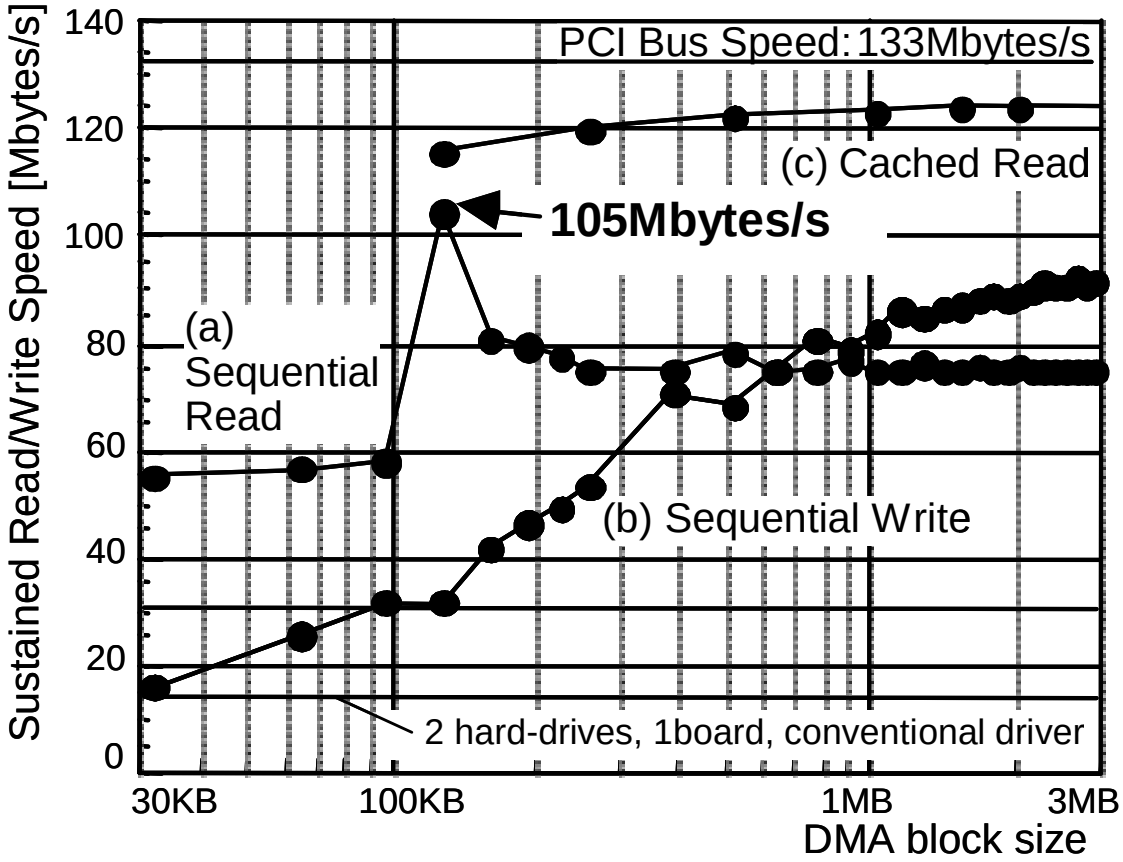


図 2.11　ファイルの連続シーケンシャル読出し速度/書込み速度の測定結果
（RAID コントローラ数： 4 枚）

グによって論理ドライブを構成した場合，内周セクタと外周セクタでどの程度の差異が見られるのかを調べた．RAID コントローラ数が 4 枚，ハードディスク数が 8 台，DMA ブロックサイズが 128KB の場合について，ファイルを記録するセクタ配置を外周セクタから内周セクタに移動させながら連続シーケンシャル読出し速度を測定した．図 2.12 に測定した結果を示す．図から明らかなように，セクタ依存性はなく，どのセクタ位置においても 100Mbytes/s を超える速度が得られていることがわかる．これは，試作システムではハードディスク単体の性能限界がボトルネックとなっていないことを示しており，システム性能がハードディスク単体の性能ではなく，別の要因で制限されていることを示唆している．ハードディスク数が少ない場合や，内部バスや RAID コントローラの性能向上によってハードディスクの性能が相対的に低下する場合においては，ハードディスク単体の性能がシステム性能を支配するようになり，外周セクタと内周セクタで性能に差異が生じる可能性が高い．このようなセクタ位置依存性が問題となる場合は，1 台当りのスループットが低下することになるが，ハードディスク数を増やすことで解決を図ることが可能

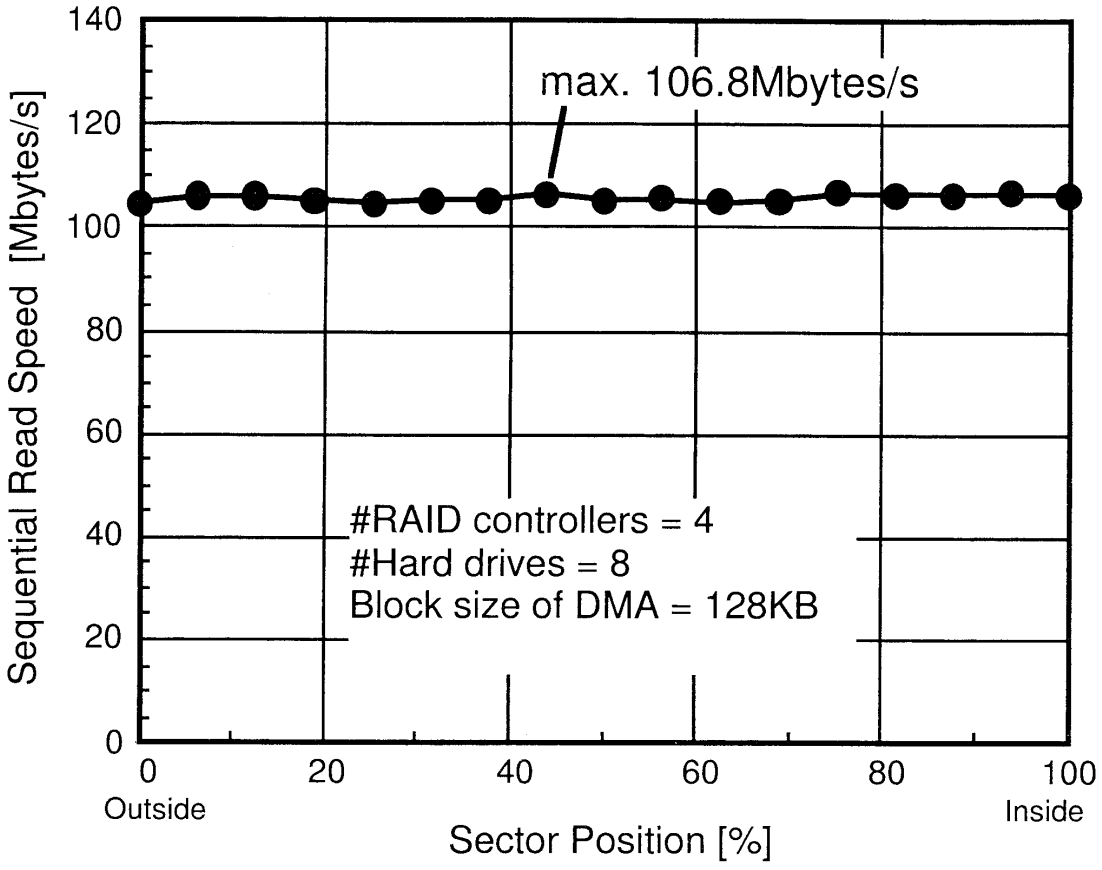


図 2.12　記録セクタ位置に対するファイルの連続読出し速度の変化

であり，図 2.12 の結果のように差異をなくすことができると考えられる．

2.5　関連研究

これまでに RAID 方式の性能改善やファイルシステムの高効率化に関する多くの研究が行われている．著者の研究と関連するものとして，RAID II システム [51] と TickerTAIP アーキテクチャ [52] がある．RAID II システムは，専用の RAID コントローラを試作・実装することで性能改善を図り，更に，ファイルシステムとして LFS（Log-Structured File System）[53] を用いている．これによって，大容量ファイルに対して約 21Mbytes/s の読出し速度を実現したことが報告されている．LFS は日誌（Log）を書くように，様々なファイルに対する書込みデータを一旦仮の領域にシーケンシャルに記録し，ハードディスクのアクセス負荷が小さいときに本来記録すべき位置にデータを移動させる．このようにハードディスクの非動作時間を活用することで，小さいサイズでのランダムなファイルアクセ

スが多い場合のシーク時間が短縮化され，性能改善を図ることができる．しかし，本システムのように，持続的な連続読出し/連続書込み時の高効率化を図る目的においては，入出力動作が二重となる LFS は適しているとはいえない．

一方，TickerTAIP アーキテクチャは本提案と同様に RAID の並列化を行うものである．しかし，RAID II と TickerTAIP は，両者とも RAID サブシステムとして実現されており，RAID サブシステムとホストコンピュータの間の接続インタフェースの速度が，システム性能のボトルネックとなる．これらの従来方式とは対照的に，筆者が提案する方式では複数の RAID コントローラをコンピュータの内部バス上で束ねて並列化している点が異なり，更に，小さいファイルの速度改善，信頼性，ディスクスペースの利用効率を犠牲にして，大容量ファイルの連続シーケンシャル読出し速度の性能を改善することを目指している点が大きく異なっている．

2.6 むすび

大容量ファイルの入出力に適した通信端末のハードウェア構成について検討した．汎用のパーソナルコンピュータの内部バス上に複数の RAID コントローラを実装し，デバイスドライバがこれらを同時に制御し，RAID コントローラ上と内部バス上の二階層のストライピングによる並列化を行い，データ転送の高速化を図った．更に，大容量ファイルの一括入出力動作に適したシーケンシャルファイルシステムを提案し，ソフトウェア処理のオーバヘッドの削減を行った．実際に試作したシステムを用いて，実験による性能評価を行った．性能測定の結果，最高 105Mbytes/s の連続シーケンシャル読出し速度を 8 台のハードディスクと 4 枚の RAID コントローラを用いて実現することができた．また，同じハードウェア構成時の連続シーケンシャル書込み速度は最高 92Mbytes/s であった．

残された課題として，複数の内部バスの活用や，RAID コントローラの改良，MS-DOS® 以外のファイルシステムへの適用，一般の UNIX への適用などが挙げられる．これらの課題の一部については，第 3 章で検討を行う．

第3章
UNIXシステムにおける通信端末構成法

3.1 まえがき

第 2 章において，大容量ファイルの一括入出力 [10], [11], [47] に適した通信端末のハードウェア構成法について提案を行った．ここでは，まず，大容量ファイルのシーケンシャル読出し速度の性能の向上に主眼を置いて検討を行った．汎用のパーソナルコンピュータの内部バス上に複数の RAID コントローラを実装し，階層ストライピングデバイスドライバによる並列制御によって内部バスを活用した二階層の階層ストライピングを実現し，高速な論理 RAID ドライブを構成した．更に，常に連続なファイル配置を行うことで連続 DMA 転送動作を可能とするシーケンシャルファイルシステムを適用し，100Mbytes/s を超える連続シーケンシャル読出し速度を実現した．この高速 RAID ファイルサーバにおける問題点は，MS-DOS® をベースとしたファイルシステムであるために汎用性，拡張性に乏しく，また，OS として RTOS（VxWorks®）を用いているためにネットワーク機能に対する親和性，拡張性が良くないことである．このことから，より汎用性の高い UNIX システム上での実現が検討課題として挙げられていた．

そこで，本章では，パーソナルコンピュータ用の UNIX の一つである OpenBSD システム [28] を用いた高速 RAID ファイルサーバについて検討し，UNIX システム特有のソフトウェア処理問題の解決を図る．ファイルシステムとして UNIX で広く用いられている UFS との互換性を考慮した高速 RAID ファイルサーバを UNIX システム上に構築する．

3.2 UNIXシステムにおける入出力ボトルネック

UNIX システム上に高速 RAID ファイルサーバを構築するに際して，階層ストライピング技術を適用した UNIX デバイスドライバを新たに設計・作成し，RTOS（VxWorks®）

上で実現している 100Mbytes/s を超える性能と同クラスの性能をもつシステムを実現することを目標と定める．

RTOS とは異なり UNIX では仮想記憶のメモリ管理機構が用られており，i386 アーキテクチャの場合，4KB のページ単位でのページングが行われている．このため，仮想メモリ空間で連続しているメモリ領域であっても物理メモリ空間では 4KB 単位で不連続となっている場合が存在し，RAID コントローラが DMA 転送を行う前に，DMA 転送アドレスに関して，仮想アドレスから物理アドレスへの変換処理や，不連続なページに対する DMA 転送の分割処理などの処理が必要となる．実際には，これらのアドレス変換処理は DMA 転送処理のバックグラウンドで実行され，オーバヘッドはそれほど大きくないが，仮想記憶を考慮したシステム設計が高効率化のキーポイントの一つとなる．

UNIX システム上にファイルサーバを構築する場合，最も重要な問題となるのは，高速なファイルシステムをどのようにして実現するのかということである．デバイスドライバが提供する Raw デバイスの入出力性能が高速であったとしても，UNIX の実装構造そのものに関連する性能劣化要因が二つ存在する．

一つ目の劣化要因は，中間バッファメモリの介在である．UNIX はマルチタスク OS であることから，プロセスごとに割当てられるユーザメモリは個別の仮想メモリ空間に確保される．このため，ハードディスクとユーザメモリ間のデータ転送は Buffer Cache（バッファキャッシュ）と呼ばれる中間バッファメモリを介して実行される．Buffer Cache はいわゆる通常のキャッシュの機能に加えて，各種の入出力デバイスとユーザメモリ間で DMA 転送を実行する場合の一時的な受け皿としての機能を提供する．入出力デバイスと Buffer Cache 間のデータ転送には DMA 転送が利用されるが，Buffer Cache とユーザメモリ間のデータ転送についてはホスト CPU がメモリコピーを実行し，ソフトウェア処理で行われる．このような実装となっている理由は，Buffer Cache はカーネルモードの仮想メモリ空間に位置し，常に物理メモリが割当てられていることが保証されているが，ユーザメモリはユーザモードの仮想メモリ空間に確保されることから，ページが部分的に Swap-out されている場合が存在するためである．DMA 転送は物理メモリ空間におけるデータ転送手段であり，仮想メモリ空間の Page Fault に対して関与できないことから，仮想メモリ空間で動作している CPU にデータ転送を任せている．これに起因して，ハードディスクの入出力速度は CPU のメモリコピー速度に制限され，ボトルネックが生じることになる．例えば，Pentium Pro 200MHz プロセッサを用いたメモリコピーの速度は約 45Mbytes/s 程度にとどまる．このソフトウェア処理のオーバヘッドを削減する方法として，最近の研

究では non-buffer copy 実装が注目を集めている [31], [54]．本研究では，この実装をどのようにファイルシステムに取り込むかを検討する必要がある．

二つ目の劣化要因は，UFS の管理ブロック（以下，UFS ブロックと称する）のサイズに上限値が存在することである．シーケンシャルに配置された大容量ファイルを読出す場合，デバイスドライバへの読出し命令要求コマンド/書込み命令要求コマンド（以下，リクエストと称する）は出来るだけ少ない回数であった方が効率が良くなる．しかし，UFS ブロックサイズは 64KB 以下に制限されているため，最大でも 64KB 単位にデータを分割してデバイスドライバにリクエストを発行することになる．また，Buffer Cache におけるバッファサイズの上限値も UFS と同様に 64KB となっている．これらの上限値は初期の UNIX における設計思想に起因することであり，この値を変更することは OS 全体に影響が及ぶことになり，互換性の問題や Buffer Cache との親和性の問題から好ましいとはいえない．よって，互換性を保持しながら大きなブロックサイズを扱えるようにし，リクエスト数の削減を可能とする新しい方式が性能改善のために不可欠である．

3.3 提案システム

3.2 節で述べたボトルネックを解決するために，ソフトウェア構成に関連する二つの手法の提案を行う．

一つはリクエスト数削減手法である．ユーザプロセスからファイルの入出力のシステムコールが呼出された場合，ファイルシステムにおいて UFS ブロックが連続している部分を一つの大きなブロックに結合し，見かけ上のブロックサイズを増加させる操作を行う．これにより，ファイルシステムがデバイスドライバに発行するリクエスト数を削減し，ソフトウェア処理負荷を軽減させる．

もう一つは，ダイレクトアクセス手法である．Buffer Cache を介在させずにデータの入出力を行うコマンドエントリを，階層ストライピングデバイスドライバに実装する．これにより，RAID コントローラからユーザプロセスのメモリ空間に対して直接 DMA 転送することを可能とする．更に，このコマンドエントリを呼出すように UFS を変更し，階層ストライピングデバイスドライバと結合させることで，ファイルの入出力速度の改善を図る．

これら二つの提案手法によって，UNIX ファイルシステム上に高速 RAID ファイルサーバを構築することを目指す．

3.3.1 設計方針

本システムの設計方針を次に示す.

- UFS からデバイスドライバに発行されるリクエスト数を削減することで，大容量ファイルの連続入出力速度の改善を図る.
- 互換性を維持するため，UFS の記録形式を変更しない．従来の UFS を用いてマウントして利用できるようにする.
- ファイルシステムの処理負荷を軽減する．ファイルシステム経由のファイル入出力であっても，ユーザプロセスからデバイスドライバの単体性能を最小限の性能劣化で利用できるようにする.
- 中間バッファである Buffer Cache を介在させないデータ転送を実現する．すなわち，ユーザプロセス上のバッファメモリとハードディスク間で直接 DMA 転送できるような仕組みを実装する.
- UNIX システム用の階層ストライピングデバイスドライバを実装する．各 RAID コントローラ上と内部バス上における二階層のストライピングによって，DMA 転送の並列化を行う.
- 100Mbytes/s 以上の連続シーケンシャル読出し速度を達成させる.

本章の提案手法であるリクエスト数削減手法とダイレクトアクセス手法について，次節以降で述べる.

3.3.2 リクエスト数削減手法

ファイルシステム経由でユーザプロセスがデバイスドライバの性能を最小限のロスで無駄なく利用できるようにするためには，リクエスト 1 回当りの要求サイズ（以下，リクエストサイズと称する）を可能な限り大きくし，リクエスト数を削減することが必要とされる．UFS は UFS ブロックのサイズ単位でファイル配置を管理している．UFS ブロックのサイズはファイルシステムの初期化コマンドの実行時に与えることができ，UNIX システムにおける newfs コマンドの b オプションで指定する．この値はファイル配置の他にデータ転送にも関係し，UFS は UFS ブロックのサイズ単位にデータを分割しながらデバイス

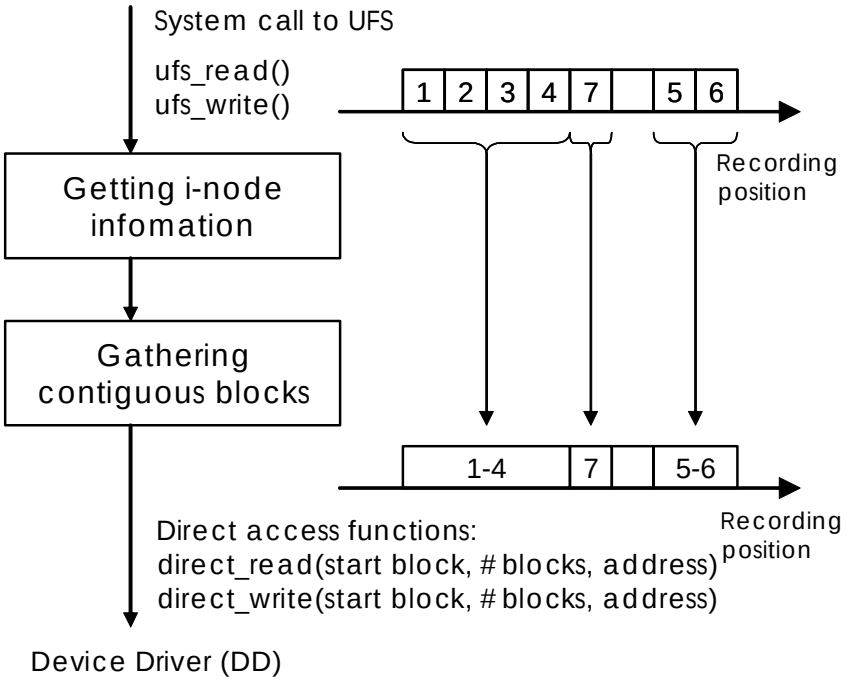


図 3.1　リクエスト数削減手法

ドライバに対してリクエストを発行する．したがって，リクエストサイズは UFS ブロックのサイズ以下に制限されることになる．UFS ブロックのサイズは 64KB が上限値となっており，大容量ファイルの入出力時に非常に多くのリクエストが発生する．単純にこの上限値を大きくすることでリクエストサイズを増加させ，リクエスト数を削減することが可能であるが，既存の UFS との互換性が失われることになる．また，関連するすべてのユーザプログラム（fsck，newfs など）やカーネル内のパラメータを変更する必要があり，システムのソフトウェアを再構築しなければならない問題が生じる．そこで，UFS の記録形式の変更を避け，連続する UFS ブロックを一つの大きなブロックに結合する機能を UFS 内に実装する．UFS からデバイスドライバに対して UFS ブロック単位でリクエストを発行するのではなく，部分的に連続するブロックごとにリクエストを発行する．これをリクエスト数削減手法と呼ぶ．

図 3.1 はリクエスト数削減手法の動作例を示したものである．図に示した例においては，ファイルが 7 ブロックに分割されてディスク上に記録されている．このうち，ブロック 1〜4，及び，ブロック 5〜6 が連続しているとする．ユーザプロセスがファイル全体（ブロック 1〜7）の内容を読出す場合，read() 関数経由でシステムコールが呼出された後，従来の UFS では，UFS ブロック単位で合計 7 回の読出しリクエストがデバイスドライバに

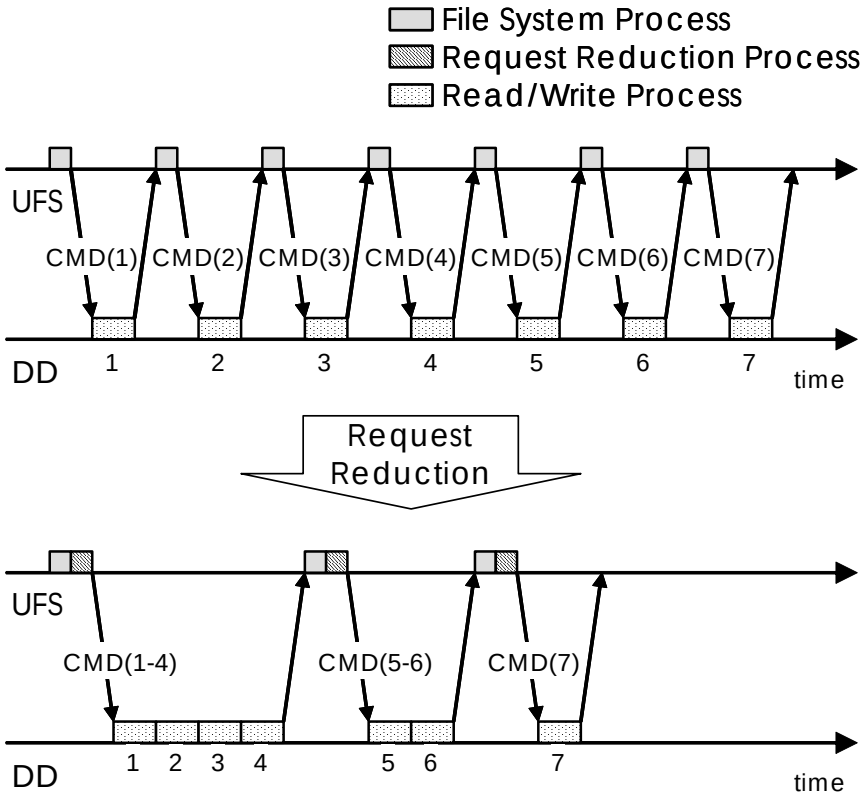


図 3.2　コマンドシーケンスの比較

発行される．しかし，UFS にリクエスト数削減手法を適用した場合は，システムコールが呼出された後，次以降の UFS ブロックの配置を調べ，配置が連続している範囲の UFS ブロックを一つの大きなブロックに結合した後，リクエストをデバイスドライバに対して発行する．このとき，リクエストの引数は，先頭ブロック番号，ブロック数，メモリアドレス（実際の実装においては先頭セクタ番号，セクタ数，メモリアドレス）となる．この手法を用いることによって，最大 UFS ブロックサイズが 64KB であっても，ファイルがある程度連続して配置されていればデバイスドライバに対するリクエストサイズを大きくすることが可能となり，従来の UFS の記録形式を変更することなく，見かけ上の UFS ブロックサイズを増加させることができる．

図 3.2 のコマンドシーケンスに示すように，リクエスト数削減手法によって CPU の処理負荷がわずかに増加するが，リクエスト数削減による性能改善の効果の方が大きいと期待される．なお，本手法が有効に機能するためにはファイルが連続配置 [55] となっていることが重要である．本研究においては，この問題を解決するために，専用の UFS のデフラグプログラム（Defrag Program）を開発し，ファイル配置の整列化を行う．ただし，

UFS では，ディレクトリエントリ，i-node と呼ばれる管理ブロック [56]，並びに，i-node に入りきらないファイル配置リストが論理ドライブ上に分散配置され，ファイルの完全なシーケンシャル配置を行えないことが多い．この不連続性によるシーケンシャル読出し/書込みの分割は止むを得ないものとし，デフラグプログラムを定期的に実行させることで，ある程度のファイルの連続配置を維持する．

3.3.3 ダイレクトアクセス手法

CPU によるメモリコピーのソフトウェア処理を回避するために，Buffer Cache を介在させない新しいコマンドフローとデータフローを UFS とデバイスドライバに実装する．具体的には，デバイスドライバに対して直接アクセスするための新しいコマンドエントリを実装し，UFS から呼出せるようにする．この手法をダイレクトアクセス手法と呼ぶ．

従来の処理スタックとダイレクトアクセス手法を用いた処理スタックの比較について図 3.3 に示す．図中，灰色の矢印はコマンドフローを，黒色の矢印はデータフローをそれぞれ表す．

図 3.3（a）の従来の処理スタックにおいては，Buffer Cache がユーザメモリ空間とカーネルメモリ空間のデータ転送の受け皿として利用される．RAID コントローラと Buffer Cache 間では DMA 転送が行われ，Buffer Cache とユーザメモリ間では CPU によるメモリコピーのソフトウェア処理でデータ転送が行われる．このデータ転送における Buffer Cache の介在がシステムの性能を低下させる要因となっている．

一方，図 3.3（b）の提案する処理スタックにおいては，階層ストライピングデバイスドライバが適用され，更に，リクエスト数削減の処理機能とダイレクトアクセスのコマンドエントリが実装されている．ファイルシステムが読出しのリクエストを受けると，ファイルの配置情報を得て，連続ブロックを結合してリクエストを作成した後，階層ストライピングデバイスドライバのダイレクトアクセスのコマンドエントリを呼出す．デバイスドライバがこの呼出しを受けると，各 RAID コントローラに対して DMA 転送の設定を行う．この時，DMA 転送の対象として任意のメモリ領域を与えることができるようにし，ユーザプロセス上のメモリ領域の指定を可能とする．

書込みのリクエストの場合の処理については，ファイル配置の処理が新たに必要になること以外は同様の処理の流れとなる．

ファイルデータ転送以外の頻度の少ない操作（ディレクトリ操作，スーパブロックの

参照など）については，OS の他の機能との整合性の問題から，従来処理と同様に Buffer Cache 経由でアクセスする．ファイルサイズが大きい場合は，ファイルデータ転送以外の操作による影響を無視することができる．

3.3.4 階層ストライピングデバイスドライバ

本システムで用いるデバイスドライバにおいても，第 2 章と同様に，階層ストライピングの手法を適用する．階層ストライピングによって，複数の RAID コントローラを同時に並列制御し，RAID コントローラ上における下位層のストライピングとコンピュータの内部バス上における上位層のストライピングの二階層のストライピングを実現し，RAID コントローラ単体の性能のボトルネックを解消する．

仮想記憶を用いない RTOS の場合とは異なり，UNIX システムを考慮した設計が必要とされ，UNIX 用の階層ストライピングデバイスドライバを新たに開発する．更に，このデバイスドライバには 3.3.3 節で述べたダイレクトアクセスのコマンドエントリも実装する．

コマンドの引数は，コマンド種別（読出し/書込み），先頭セクタ番号，セクタ数，バッファメモリアドレス，仮想アドレスと物理アドレスの区別である．メモリアドレスとしてユーザメモリ空間の仮想アドレスが与えられている場合，DMA 転送の実行前に，仮想アドレスから物理アドレスへの変換処理や転送対象のメモリ領域が swap-out していないかどうかのチェック処理が行われ，swap-out している場合は swap-in の処理によって物理メモリが割当てられる．これらの準備を終えてはじめて，RAID コントローラからユーザメモリ空間の領域に対してハードディスクから読出したデータを直接 DMA 転送することができるようになる．

なお，従来手法であるソフトウェア RAID を用いれば，Raw デバイスの上位で上位層のストライピングを行うことができるが，この場合は，リクエスト数削減手法，および，ダイレクトアクセス手法の適用が困難であり，ボトルネック解消の観点からメリットが小さいといえる．

3.3.5 仮想メモリ空間と物理メモリ空間

UNIX システムは仮想記憶のメモリ管理機構を実装している．i386 アーキテクチャの場合，4KB 単位のページングによって，仮想メモリ空間でメモリ領域が連続であっても，物理メモリ空間において連続であるとは限らない．すなわち，DMA 転送の際に仮想メモリ

アドレスから物理メモリアドレスへのアドレス変換処理は，4KB 単位で行われることを意味している．DMA 転送回数を削減するために，事前に連続するページを 1 回の DMA 転送で扱うように結合しておく必要がある．この結合処理は Scatter Gather 処理と呼ばれている．

なお，ページが不連続であっても，RAID コントローラが有する連鎖 DMA 転送（Chaining DMA Transfer）の機能を利用することで，ある程度までは，性能劣化を抑えることができる．連鎖 DMA 転送は，ほとんどの PCI バスインタフェースチップが有している機能である．

以上の処理は小さい負荷で実行される必要があり，OS 内の適切な箇所で，最小限のプログラムで最適に行わなければならない．どの部分で担当するべきかを吟味した結果，ファイルシステムとデバイスドライバ間のリクエスト数の削減という観点から，前述のとおり最下位のデバイスドライバ内に実装する．

図 3.4 は UNIX（i386 アーキテクチャ版 OpenBSD）のカーネルモードにおけるメモリマップである．デバイスドライバやファイルシステムなどカーネル空間のプログラムを CPU が実行しているときは，システムの状態はカーネルモードにあり，システムコールを呼出したユーザプロセスのユーザメモリ空間とカーネルメモリ空間の両方が同一の仮想メモリ空間上にマップされている．したがって，RAID コントローラからユーザメモリ空間の領域に DMA 転送を実行する場合と，カーネルメモリ空間の領域に DMA 転送を実行する場合の違いは，単純に DMA 転送の対象アドレスの違いだけである．ただし，前者のユーザメモリ空間の領域は swap-out されていることがあるため，既に述べたように，DMA 転送の前にあらかじめ物理メモリを割当てる処理が必要となる．一般的な UNIX では，物理メモリの割当て処理を行う mlock() 関数が提供されており，DMA 転送前にバッファ領域の物理メモリをロック（以下，mlock と称する）しておくことで，swap-out しないメモリ領域を確保することができる．DMA 転送後，munlock() 関数を呼出すことで設定を解除（以下，munlock と称する）する．

Buffer Cache のメモリ領域はカーネルメモリ空間に確保されているため，OS の起動時から物理メモリがロックされた状態にあり，swap-out が発生することがない．したがって，従来のデータフローにおいては mlock/munlock の処理は不要である．これが DMA 転送の受け皿として Buffer Cache が用いられる理由の一つとなっている．

なお，mlock/munlock の操作はシステム性能の劣化要因となりうる危険性がある．後述する試作システムの場合，mlock の処理に 10MB 当り約 200ms の時間を必要とし，read()

関数/write() 関数の呼出し毎に mlock/munlock を行うことは効率的であるとはいえない．そこで，ioctl() 関数を使った新たなファイル読出し/書込み関数を併設する．ioctl() 関数の第 1 引数はファイルデスクリプタ番号，第 2 引数は機能番号であるが，機能番号として FIOBULKREAD または FIOBULKWRITE を与えることで read() 関数/write() 関数と同様にファイルの読出し/書込みを実行できるようにする．なお，メモリアドレス，ファイルサイズなどの情報は第 3 引数で与える．ioctl() 関数経由でシステムコールが呼び出されると，カーネルではファイルデスクリプタが確認され，read() 関数/write() 関数のときと同様に該当するファイルシステムが呼び出される．この ioctl() 関数を用いてファイルの読出し/書込みを実行する場合，ファイルシステム内ではなく，あらかじめユーザプロセス側でバッファ領域を mlock しておく．共通のバッファメモリをユーザメモリ上に確保し，ioctl() 関数の呼出しごとではなく，ユーザプロセスの起動時と終了時にだけ mlock と munlock を実行するようにプログラム作成を行えば，オーバヘッドを削減することが可能となる．

しかしながら，既存のユーザプログラム（例えば，cat，dd などのコマンド）に対しても，ソースの書き換えや再コンパイルを必要とせずに高速化を図る目的で，read() 関数/write() 関数によるシステムコールにおける提案手法の適用も重要であり，ファイルシステム内で mlock/munlock を実行する機能も実装する．ただし，この場合のオーバヘッドの存在は止むを得ないものとする．

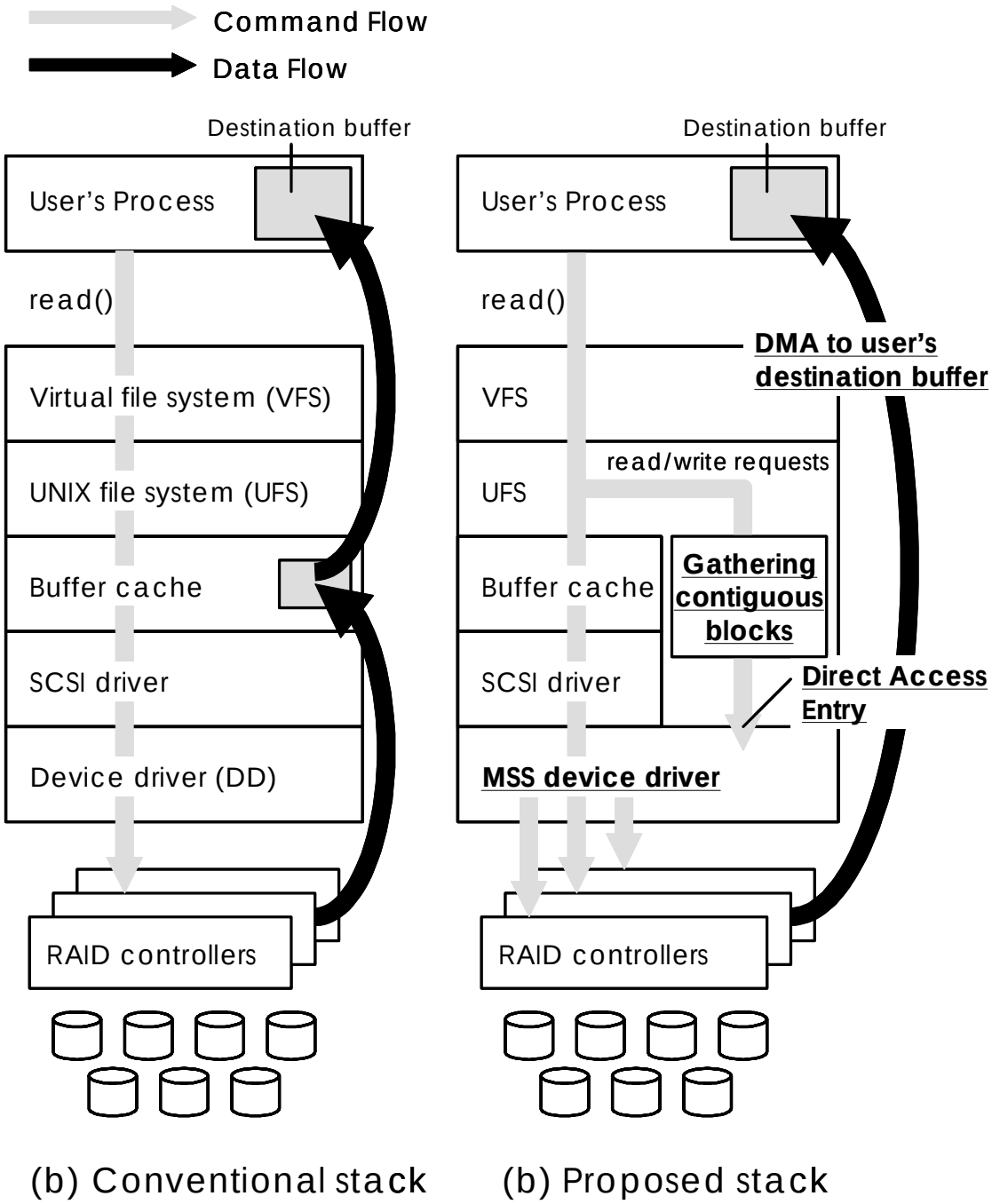


図 3.3　処理スタックの比較

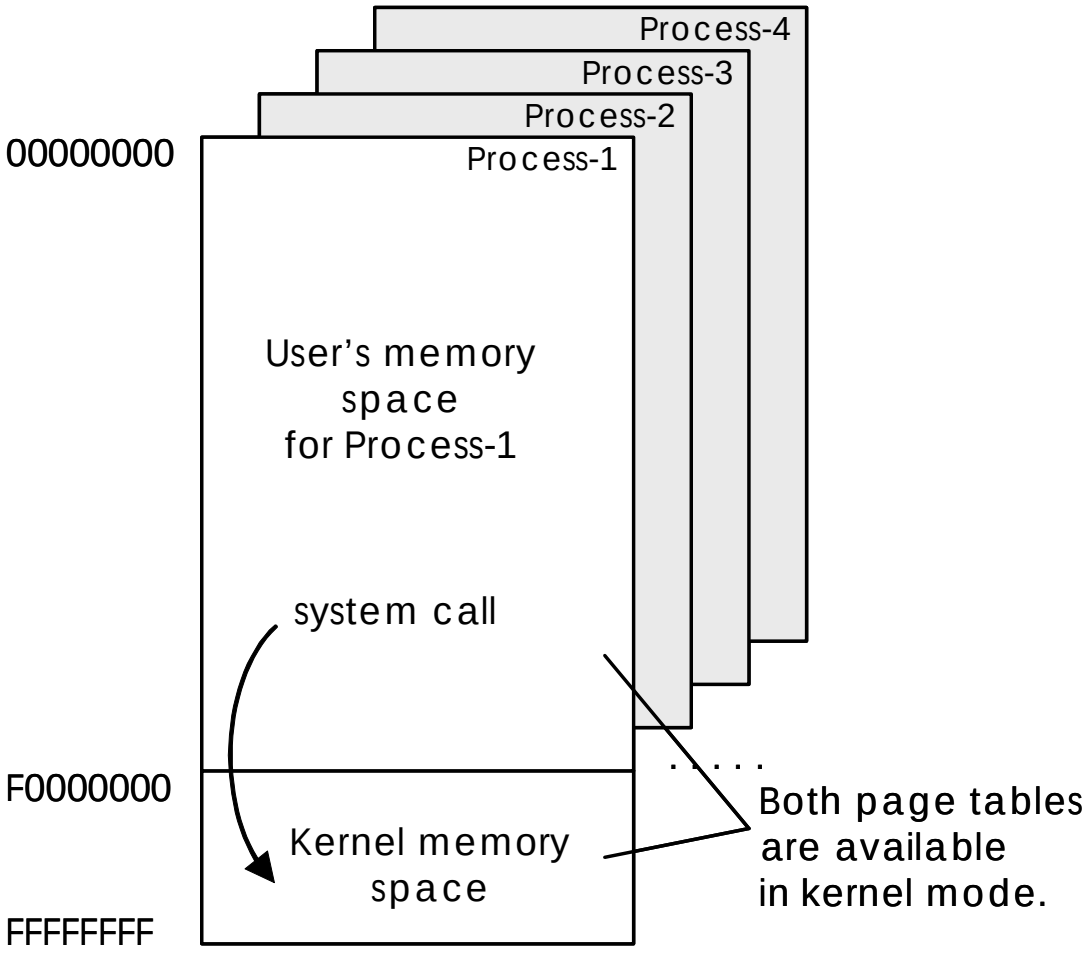


図 3.4　UNIX のカーネルモードにおけるメモリマップ

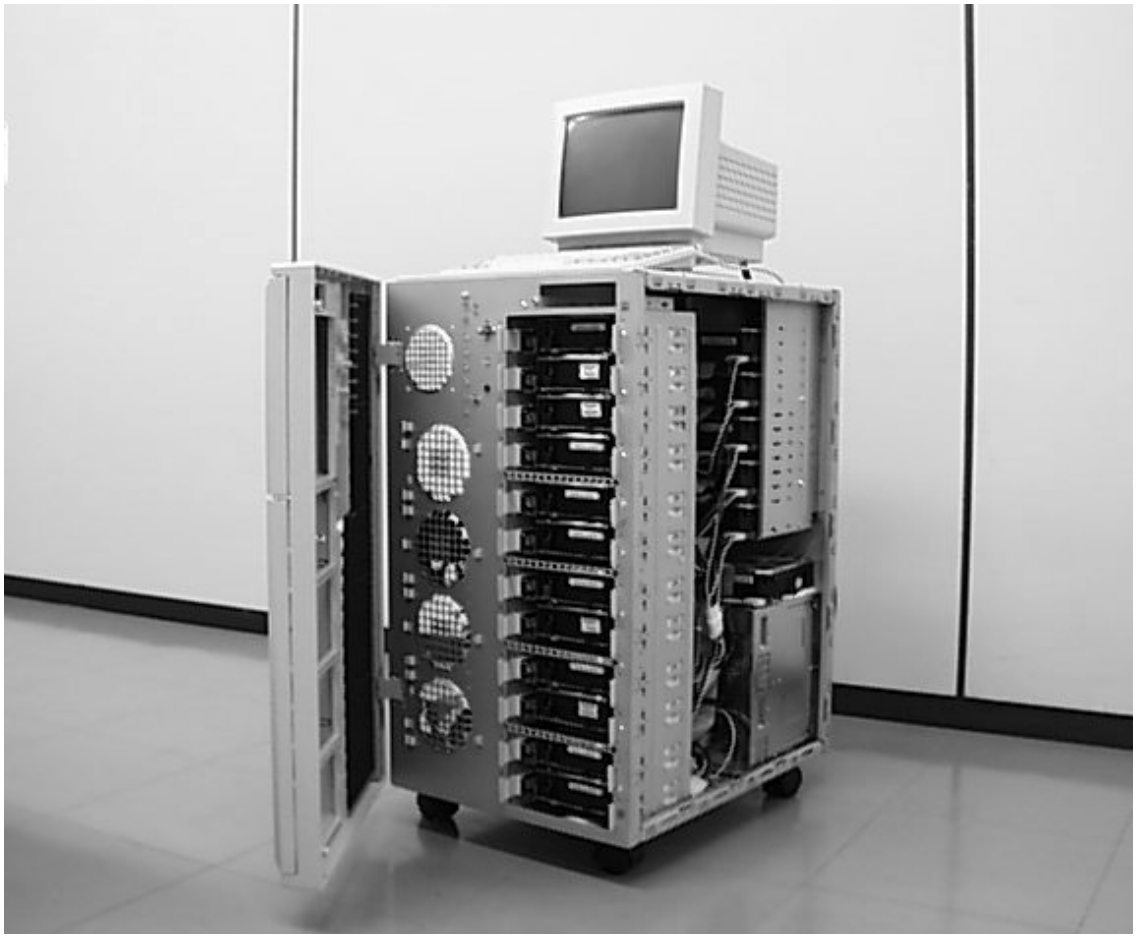


図 3.5　試作システムの外観

3.4　試作システム

提案手法を適用した高速RAIDファイルシステムを試作し，実験により性能評価を行った．図 3.5 に試作システムの外観を示し，また，表　3.1 に試作システムの諸元を示す．CPUはi386アーキテクチャのPentium Pro 200MHzであり，マザーボード上に，1GBのメモリと3枚のRAIDコントローラ，18台のハードディスクを実装する．UNIXとしてOpenBSD version 2.1を用いる．

図 3.6 に試作システムのブロック図を示す．内部バスとして64bit/33MHz動作のコンカレントPCIバスを搭載するマザーボードを選択している．コンカレント動作とは，デュアルポートメモリに2本のPCIバスを接続するような構成ではなく，十分に高速な通常のシングルポートメモリに対してブリッジを介して2本のPCIバスを接続し，ブリッジ内に十分な緩衝バッファを設け，擬似的な同時アクセスを実現することである．これによって，ホストメモリを共有メモリとして活用した，通信端末装置を構成するプラットフォー

表 3.1　試作システムの諸元

Processor	Pentium Pro 200MHz
Chipset	Intel 450GX/KX
Host Memory	1GB interleaved EDO-DRAM
PCI buses	Two PCI buses with 3 slots each
RAID controller	AMI MegaRAID 434 with tuned firmware
RAID level	RAID 0, 1, 3 and 5 supported
#Controllers	3
Hard Drive	Seagate Cheetah-9 (9GB) with internal transfer rate of 122-177Mbps
#Drives	18 (6 drives at each controller)
OS	OpenBSD version 2.1
Dimensions	370mm(W), 680mm(H), 510mm(H)

ムが提供される．

本試作システムでは，一次側の PCI バスに記憶装置用の RAID コントローラを，二次側の PCI バスにネットワークに接続される NIC を実装した通信端末構成とする．なお，ネットワーク接続が不要で記憶装置のみの構成の場合は，一次側と二次側の PCI バスに RAID コントローラカードを分散して実装する構成としてもよい．

RAID コントローラカードは 32bit/33MHz 動作仕様の PCI カードであり，3 チャネルの Ultra Wide SCSI-3 インタフェースを搭載し，それぞれに 2 台のハードディスクが接続される．システム全体では，18 台のハードディスク数となる．3 枚の RAID コントローラはデバイスドライバによって並列に制御され，ホストメモリとハードディスク間で高速なデータ転送を実現する．このとき，3 枚の RAID コントローラは RAID-0 で制御される．これによって，デバイスドライバによる上位層，RAID コントローラによる下位層の二階層ストライピングを実現する．

それぞれの RAID コントローラは，直接接続された 6 台のディスクを用いて下位層の論理 RAID ドライブを構成する．設定に応じて，RAID-0，RAID-1，RAID-3，RAID-5 のいずれかの RAID レベルでディスクアレイ動作をさせることが可能であるが，本試作システムでは効率的な動作を狙い，各 RAID コントローラの設定内容を統一する．すなわち，論理ドライブを構成するハードディスク数，RAID レベル，ストライプサイズを同一とする．なお，設定によって，RAID コントローラに接続された 6 台うち任意の台数のハードディ

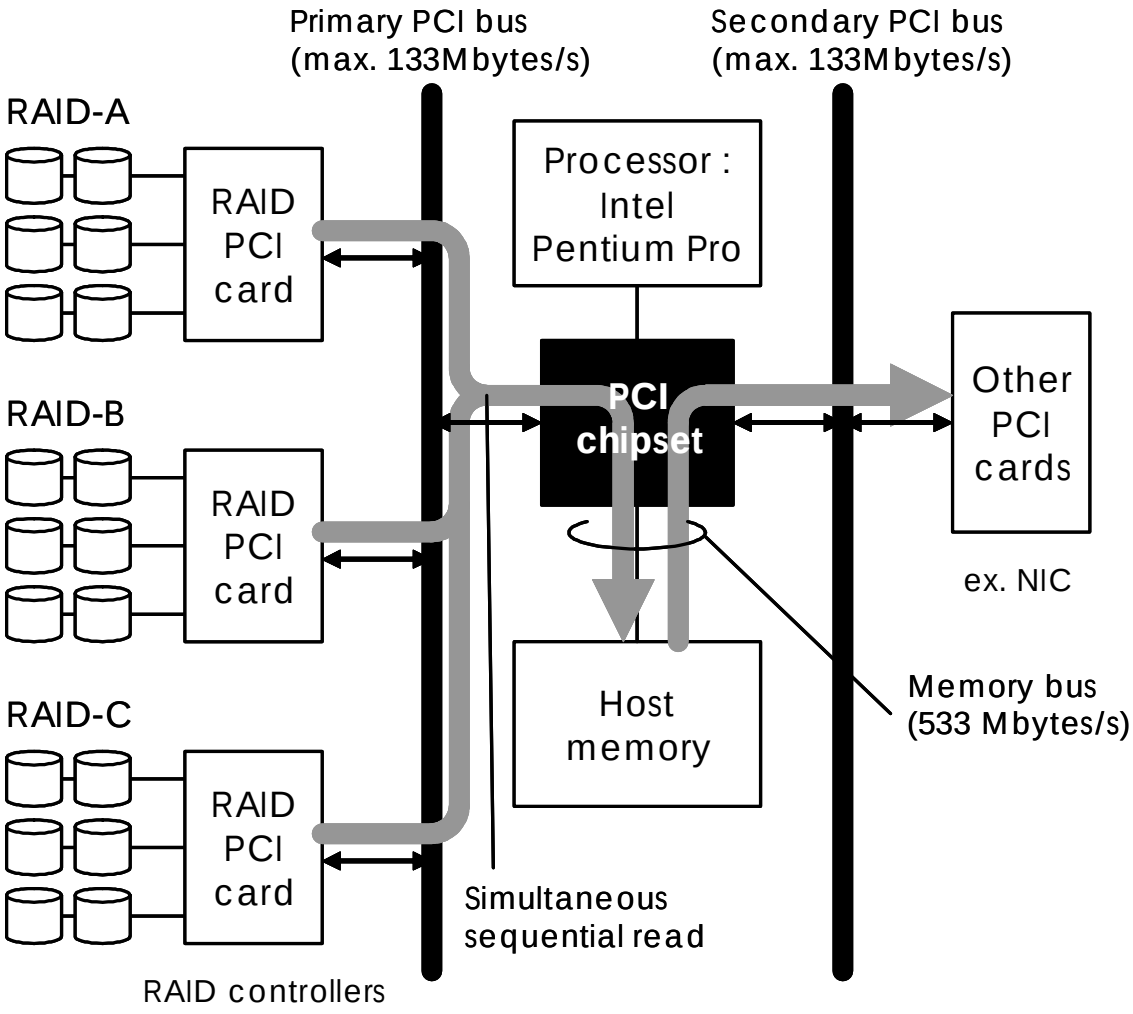


図 3.6　試作システムのブロック図

スクを用いてRAID論理ドライブを構成することが可能となっている．ただし，RAID-5の論理ドライブを構成するためには，RAIDコントローラ当り最低3台のハードディスク数が必要となる．

メモリバスの最大転送能力は533Mbytes/s（=4Gbits/s，16.67MHz，72bit幅EDO-DRAM，2バンクのインタリーブ動作による）であり，64bit/33MHz動作の2本のPCIバスからの同時連続メモリアクセスに耐えられるようになっている．

表 3.2　デバイスドライバの連続読出し速度の測定結果（NR と LS に対する依存性の確認）

RAID 0		Sequential read speed (Mbytes/s)		
Lower stripe size (LS)	Request size	#RAID controllers (NR)		
		1	2	3
32 KB	1 MB	20.2	39.9	50.8
	50 MB	20.4	40.8	55.0
64 KB	1 MB	59.5	87.5	92.9
	50 MB	61.8	97.9	109.5
128 KB	1 MB	32.4	59.7	78.6
	50 MB	34.5	67.9	90.0

Note: #Hard drives (ND) = 18, and the upper stripe size (US) = 64 KB.

3.5　実験による検討

3.5.1　最適な設定パラメータの導出

本節では，RAID コントローラ数と下位層のストライピングサイズがシステム性能に与える影響を検討し，最適な設定パラメータを導出する．一般に，RAID 方式の設計においてはランダムアクセスなどの性能を改善するために Stripe Allocation [57] についても考慮されるべきであるが，本システムでは連続シーケンシャルアクセス時の性能改善を目指しており，また，RAID コントローラのコマンドキューによる緩衝効果が期待されることから，単純なリニアアロケーションによる Pure ストライピングとしている．

試作システムにおける最適な設定パラメータの導出に関しては数値解析が困難なため，以下における議論では，実験により検討を行うこととする．ここで，NR を RAID コントローラ数，LS を下位層のストライピングサイズ，US を上位層のストライピングサイズ，ND をシステム全体のハードディスク数としたとき，本節での目的は最適な NR，LS，US の各値を見出すことである．最適の判断を行うための評価基準として，デバイスドライバ単体の連続シーケンシャル読出し速度を用いる．これまでの議論と同様に，ダイレクトアクセスのコマンドリクエストの引数で与える読出しまたは書込みの要求サイズをリクエストサイズと呼ぶことにする．これは，図 3.1 に示したダイレクトアクセスのコマンドエントリである direct_read() 関数，direct_write() 関数の引数で与えるサイズに相当する．

表 3.3　デバイスドライバの連続読出し速度の測定結果（ND に対する依存性の確認）

RAID 0		Sequential read speed (Mbytes/s)	
#hard drives (ND)	Request size	#RAID controllers (NR)	
		2	3
6	1 MB	83.1	83.3
	50 MB	82.3	82.8
8	1 MB	86.6	-
	50 MB	93.9	-
9	1 MB	-	93.1
	50 MB	-	109.1
10	1 MB	87.3	-
	50 MB	96.9	-

Note: The upper and lower stripe size (US and LS) = 64 KB.

まず，最適な LS の値について調べる．$LS = 32$KB，64KB，128KB の場合の性能を，$NR = 1$〜3，リクエストサイズが 1MB，50MB の条件について測定した．ハードディスク単体の性能の影響を受けないように $ND = 18$ とした．測定結果を表 3.2 に示す．

表から，$LS = 64$KB，$NR = 3$ のとき，最も高速な連続シーケンシャル読出し速度となり，リクエストサイズ 50MB に対して 109.5Mbytes/s の性能が得られた．よって，以下の検討においては，下位層ストライピングサイズとして $LS = 64$KB の設定を最適値として用いる．更に，$NR = 1$ のときの速度は $ND = 18$ であっても 61.8Mbytes/s となっているが，これは試作システムで用いた RAID コントローラの性能限界が 61.8Mbytes/s であることを示している．このことから，RAID コントローラの性能限界を超える 100Mbytes/s 以上の性能を得るには，$NR \geqq 2$ の階層ストライピングの手法が不可欠であることがわかる．

続いて，ND の値を減らしながらハードディスク数がシステム性能に与える影響について調べた．表 3.3 は $NR = 2$，$NR = 3$ の場合について，$ND = 6$〜10 の範囲で連続シーケンシャル読出し速度を測定した結果である．なお，各 RAID コントローラで同じハードディスク数を用いると制約を定めたので，ND が NR の倍数であることに留意する．表を見ると，$ND = 9$，$NR = 3$ のとき，100Mbytes/s を超える性能が得られている．よっ

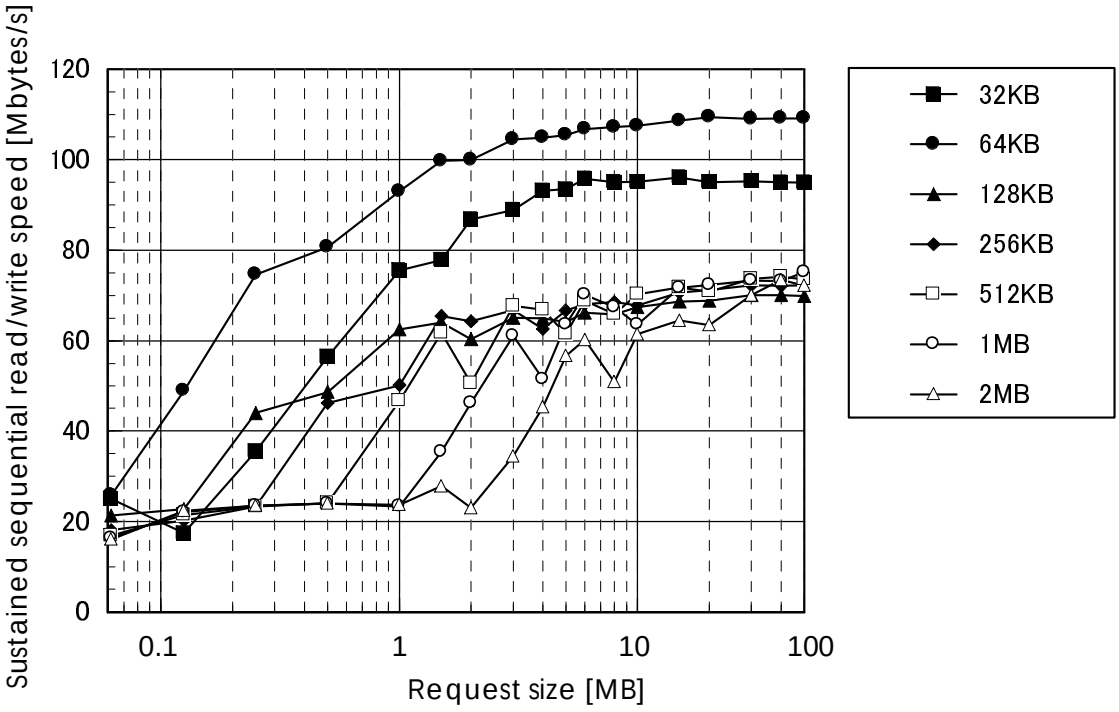


図 3.7　デバイスドライバの Raw デバイス性能（$NR = 3$，$LS = 64$KB，$ND = 9$，RAID-0，$US = 32$KB〜2MB）

て，以下の検討においては，最適な RAID コントローラ数として $NR = 3$ を用いる．なお，ハードディスク 1 台当りの速度という観点から見ると，$ND = 6$ のときの 82〜83Mbytes/s（1 台当り約 14Mbytes/s）の方が性能が良い．このことは，ハードディスク単体の性能が向上すれば，RAID コントローラ数が $NR = 2$ の場合においても 100Mbytes/s を超える可能性を示唆しており，設計検討については，各要素の単体性能を吟味して行われなければならないことを示している．

最後に，上位層のストライピングサイズ US について検討する．US の値は，RAID コントローラの並列動作の効率に関係している．$NR = 3$，$LS = 64$KB，$ND = 9$，RAID-0 の条件において，上位層のストライピングサイズを $US = 32$KB〜2MB，リクエストサイズを 64KB〜100MB の範囲で変化させた場合について，連続読出し速度を測定した結果を図 3.7 に示す．リクエストサイズを増加させるに従って読出し速度が増加するが，ある値を境として鋭く立ち上がっていることがわかる．US の値が大きいほど立ち上がりに必要なリクエストサイズが大きくなっている．これは，US の値の増加に比例して RAID コントローラの効率的な並列動作に必要な最小のリクエストサイズが増加することを表しており，概ね $US \geqq NR \times RS$（RS はリクエストサイズ）が条件となっている．更に，

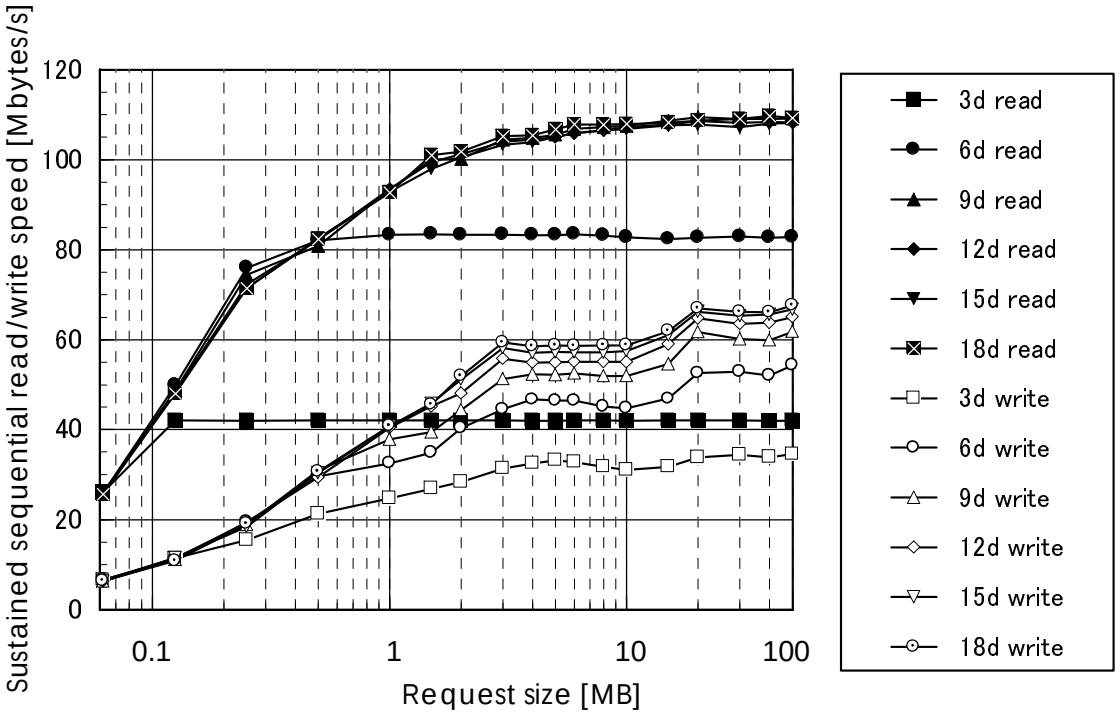


図 3.8　デバイスドライバの Raw デバイス性能（$NR = 3$，$LS = 64$KB，$US = 64$KB，$ND = 3$〜18，RAID-0）

$US = 32$KB のときは，$US < LS$ となるため，バッファリング処理のオーバヘッドによる劣化が見られる．これらのことから，試作システムにおける最適な US の値は $US = 64$KB であるといえる．

なお，図 3.7 の $US = 64$KB の結果を見ると，100Mbytes/s を超える連続速度を得るためには最低でも 1.5MB 以上のリクエストサイズを必要としている．この結果は，UFS ブロックのサイズが 64KB に制限されている従来の UFS を用いた場合，並列化の効率が低く，十分な性能が得られないことを示しており，リクエスト数削減手法の必要性を裏付けるものである．

3.5.2　デバイスドライバの Raw デバイス特性

まず，試作システムにおいてデバイスドライバの Raw デバイス性能を評価する．連続シーケンシャル読出し速度/書込み速度について，RAID-0 及び RAID-5 の場合の性能を測定した．RAID-0 のときの測定結果を図 3.8 に，RAID-5 のときの測定結果を図 3.9 にそれぞれ示す．ここで，$NR = 3$，$LS = 64$KB，$US = 64$KB である．図中における “3d”

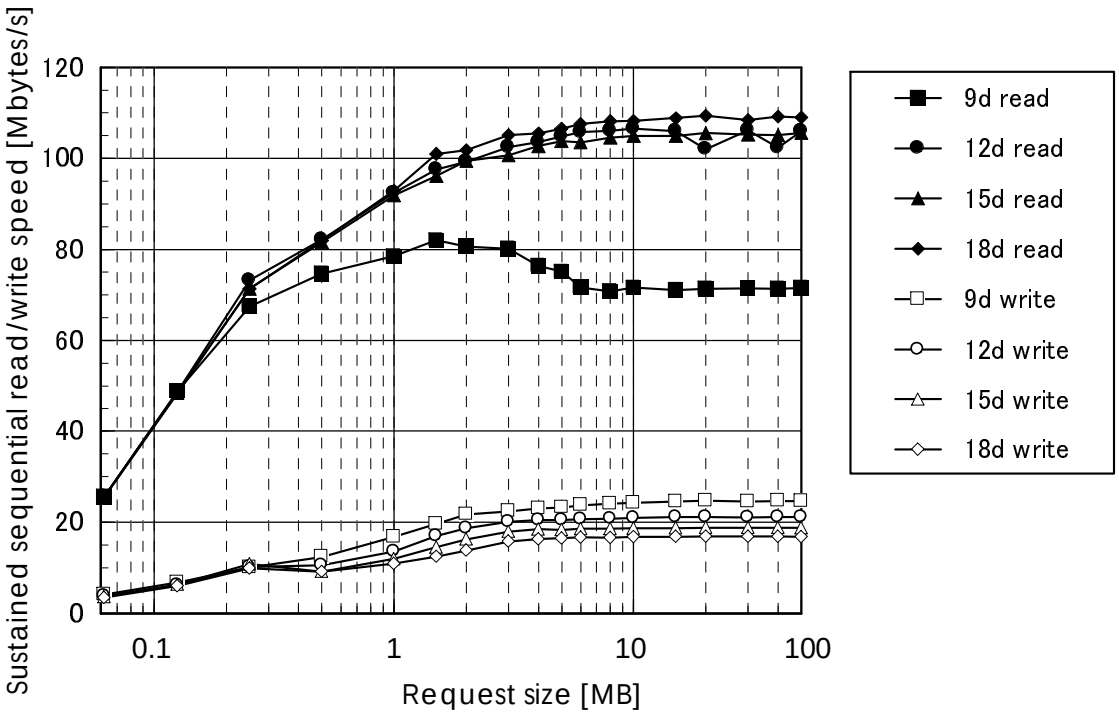


図 3.9　デバイスドライバの Raw デバイス性能（$NR = 3$，$LS = 64$KB，$US = 64$KB，$ND = 9$〜18，RAID-5）

の表記は $ND = 3$ の場合の結果を表し，以下同様に，$ND = 18$ の場合までの特性を示している．

連続シーケンシャル読出し速度の特性を見ると，RAID-0 の場合は $ND \geq 9$ で性能が飽和し，1.5MB 以上のリクエストサイズの場合に 100Mbytes/s を超え，最高で約 110Mbytes/s の連続シーケンシャル読出し速度が得られている．一方，RAID-5 の場合は各 RAID コントローラにつき 1 台，合計 3 台のハードディスクがパリティ用に利用されるため，$ND \geq 12$ のときに性能が飽和する．飽和時の特性は，RAID-0 と RAID-5 でほぼ一致しており，RAID-0 の場合と同様に 1.5MB 以上のリクエストサイズに対して 100Mbytes/s を超える連続シーケンシャル読出し速度が得られた．

試作システムで用いた RAID コントローラ単体の性能限界は 61.8Mbytes/s であり，PCI バスにおける調停処理による劣化がないと仮定すれば，$NR = 3$ のとき，最大で約 185Mbytes/s の性能となる．このことから，約 110Mbytes/s で飽和する原因は，133Mbytes/s の PCI バス速度の制限によるものと考えられる．

リクエストサイズが 1MB 以下の場合，RAID コントローラの並列動作の効率が低下し，性能が急激に落ち込んでいる．これは，$LS = 64$KB，$US = 64$KB であることから $ND = 9$

の RAID-0 の場合，576KB（$= 9 \times 64$KB）以上のリクエストサイズでなければ 9 台すべての同時ディスクアクセスが発生しないためである．リクエストサイズが 64KB の場合を考えると，リクエスト対象のセクタ領域の境界が LS，及び，US の境界と一致するならば，1 回のコマンドリクエストで 1 台のハードディスクしか動作しないことになる．試作システムで用いたハードディスク単体での連続読出し速度は約 14Mbytes/s であり，最悪の場合，この速度にまで性能が劣化することになるが，図を見ると約 25Mbytes/s の性能が得られている．この理由は，RAID コントローラが備えているコマンドキューと先読みの機能によって，特性が改善されるためと考えられる．

続いて，連続シーケンシャル書込み速度の特性について見ると，図 3.8 の RAID-0 のとき，$ND \geqq 12$ で性能が飽和している．また，リクエストサイズの増加とともに性能が改善され，リクエストサイズが 30MB 以上のとき，60Mbytes/s 以上の連続シーケンシャル書込み速度が得られる．

一方，図 3.9 の RAID-5 のときは，リクエストサイズが大きい場合であっても高々 20Mbytes/s の速度しか得られていない．これは，誤り訂正のためのパリティ生成とその割当て処理に時間を要するためである．

3.5.3 ファイルシステムのファイル入出力特性

本節では，リクエスト数削減手法とダイレクトアクセス手法を適用した UFS の性能評価を行う．ファイルシステム経由でファイルデータの入出力を行った場合の連続シーケンシャル読出し速度/書込み速度を実験により測定する．この評価においては，図 3.8，及び，図 3.9 に示したデバイスドライバの Raw デバイス性能が基準となり，性能劣化をどの程度小さく抑えることができるかがポイントとなる．

図 3.10 は，測定に用いた連続シーケンシャル読出し用のベンチマークプログラムである．64KB〜100MB のファイルサイズの条件に対して，300MB 分のファイル数を読出すための所要時間を測定し，速度を求めた．例えば，10MB ファイルの場合，測定用の 30 個の 10MB ファイルを UFS 上に作成し，30 個のファイルそれぞれに対して 10MB の read() を実行し，所要時間を測定する．実際の速度を評価する理由からデフラグプログラムによる整列を敢えて行わない状態で測定する．ただし，条件を揃えるため，毎回 newfs コマンドでファイルシステムを初期化した後にベンチマークプログラムを実行する．なお，連続シーケンシャル書込み速度の測定についても，同様のベンチマークプログラムと手順で測

Benchmark for file read of *filesize*

```
buff = malloc(filesize);
mlock(buff, filesize);

N = 300MB / filesize;

prepare_files(filename, filesize, N);

start_timer(); /* for measurements */
for (i = 0; i < N; i++) {
    fd = open(filename[i], O_RDWR);
    bulk_read(fd, buff, filesize); /* this calls ioctl() */
    close(fd);
}
stop_timer();

munlock(buff);
```

図 3.10　ファイルの連続シーケンシャル読出し速度の測定に用いたベンチマークプログラム

定する．

ファイルシステムの初期化時の UFS のブロックサイズは 64KB，フラグメントサイズは 64KB，シリンダグループのシリンダ数は 1024 であり，それぞれ，newfs コマンドの b, f, c オプションで指定する．これらのファイルシステムのパラメータは高効率化のために最大値を与えている．

ハードディスク数 $ND = 3$〜18，ファイルサイズ 64〜100MB の場合について，ファイルの連続シーケンシャル読出し速度を測定した．図 3.11，図 3.12 は，RAID-0，RAID-5 の場合の測定結果である．従来の UFS を用いた場合の結果も重ねて示している．

従来の UFS では，読出し速度が約 25Mbytes/s で特性が飽和しているが，これは Buffer Cache の介在によって，メモリコピーの処理速度がボトルネックとなっていることが原因である．試作システムで用いた Pentium Pro 200MHz の CPU ではメモリコピーの処理速度は約 45Mbytes/s であり，この性能を超える性能は得られない．

一方，提案手法を適用した場合の特性を見ると明らかなように，提案手法によってボトルネックが解消され，従来の UFS の場合を大きく上回る性能が得られている．ファイルサイズが 50MB を超えると 100Mbytes/s 以上の連続シーケンシャル読出し速度が得られる．このことから，UNIX システム上のファイルシステム経由のファイル入出力であって

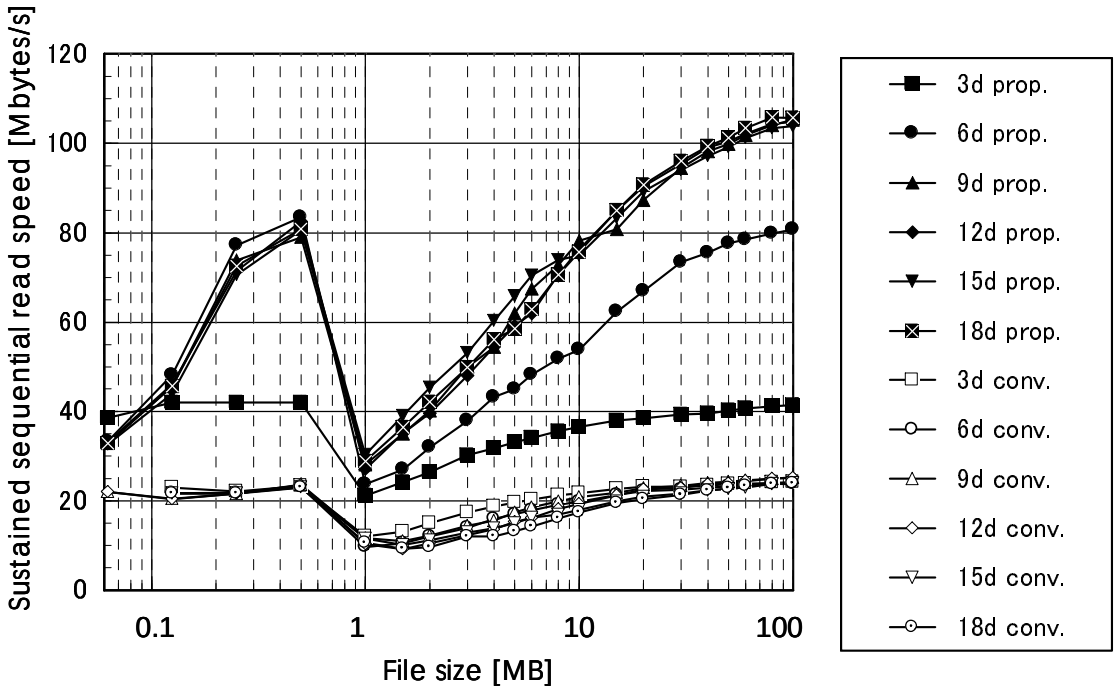


図 3.11　ファイルの連続シーケンシャル読出し速度の測定結果（RAID-0）

も，ファイルサイズが大きい場合は，デバイスドライバの Raw デバイス性能を劣化させることなく利用することが可能であることを実験により示すことができた．

なお，ファイルサイズが 512KB のときに性能のピークがあり，また，1MB で急激な性能の落ち込みが見られる．この理由は，i-node のブロック管理方法にある．i-node では先頭の 12 個の UFS ブロック配置の情報を i-node 内に格納し，13 個目以降の UFS ブロック配置の情報は一次間接リストとして外部に別ブロックを確保して格納する [56]．この外部ブロック利用の境界が 64KB×12 = 768KB にあり，ファイルサイズが 1MB〜50MB の領域では無視できないオーバヘッドとなっている．

RAID-0 の場合の連続シーケンシャル書込み速度の測定結果を図 3.13 に示す．読出し速度の実験結果と同様に，提案手法を実装した UFS は従来の UFS におけるボトルネックを解消し，性能改善の結果，約 60Mbytes/s の連続書込み速度が得られている．一方，従来の UFS の連続書込み速度はハードディスク数 $ND = 12$ であっても，約 7Mbytes/s で性能が飽和している．

以上の実験結果から，提案したリクエスト数削減手法とダイレクトアクセスは有効に機能し，ファイルサイズが大きい場合，階層ストライピングデバイスドライバの性能劣化を極めて小さく抑えられることを確認することができた．

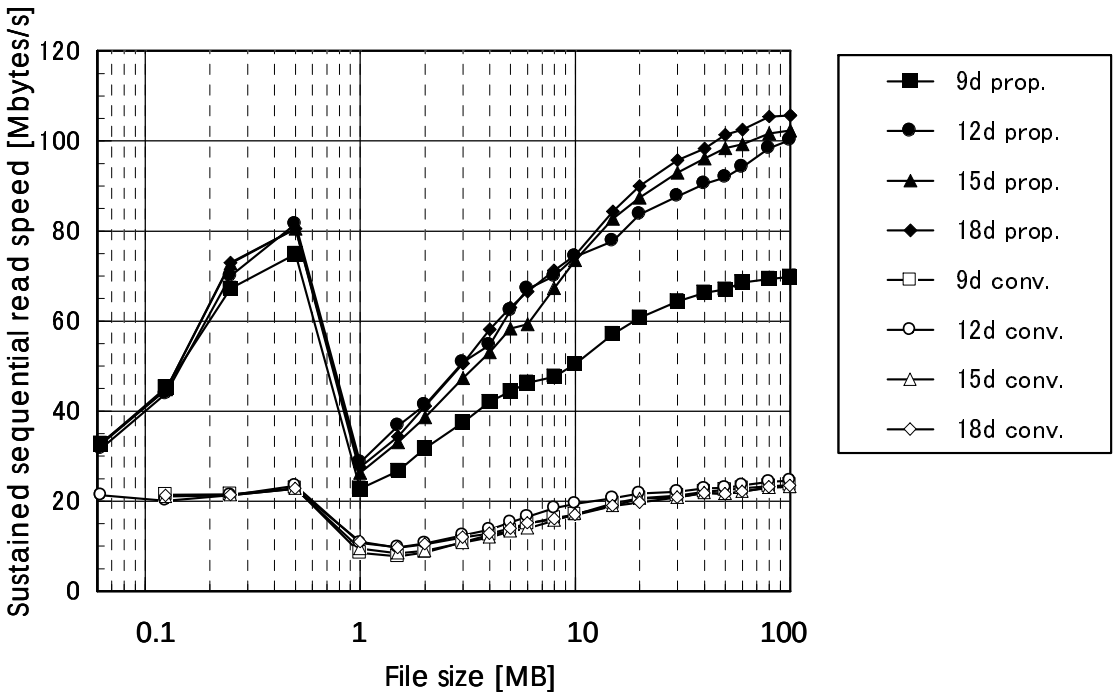


図 3.12　ファイルの連続シーケンシャル読出し速度の測定結果（RAID-5）

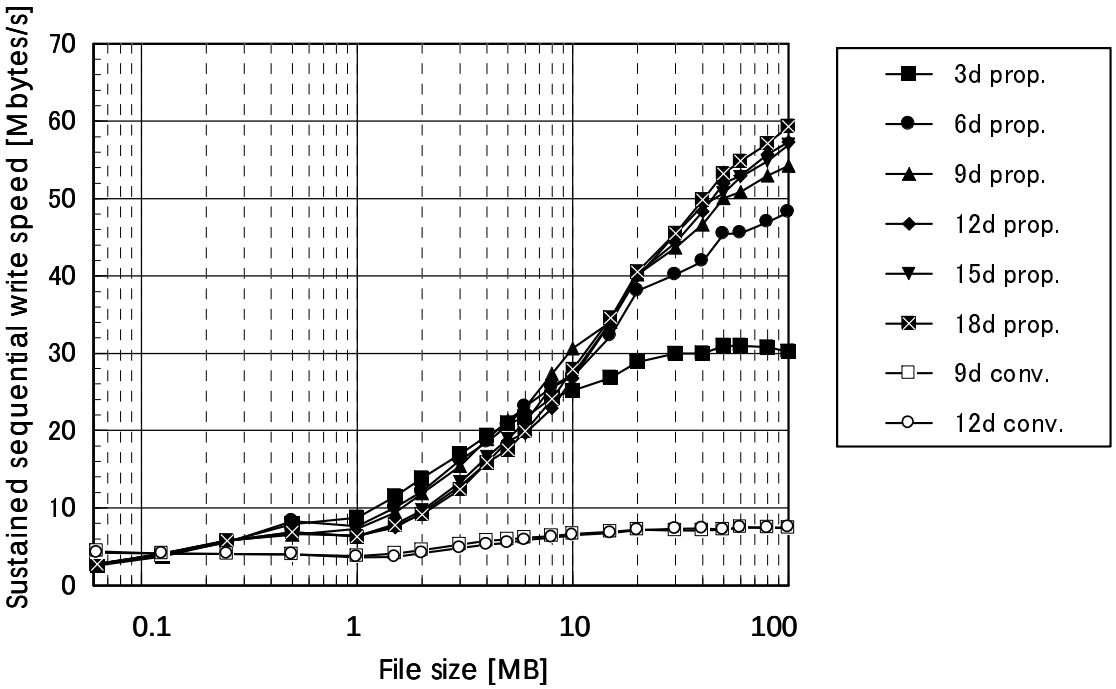


図 3.13　ファイルの連続シーケンシャル書込み速度の測定結果（RAID-0）

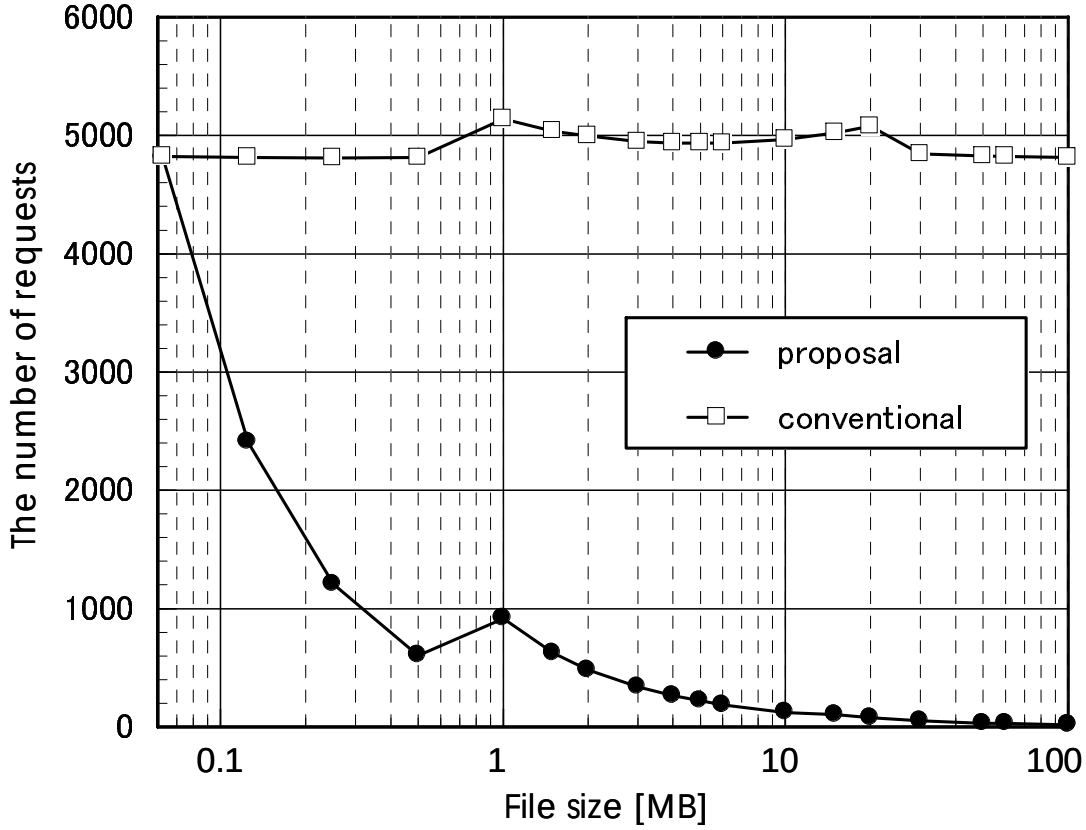


図 3.14　リクエスト数削減手法の効果

3.6　リクエスト数削減手法の効果

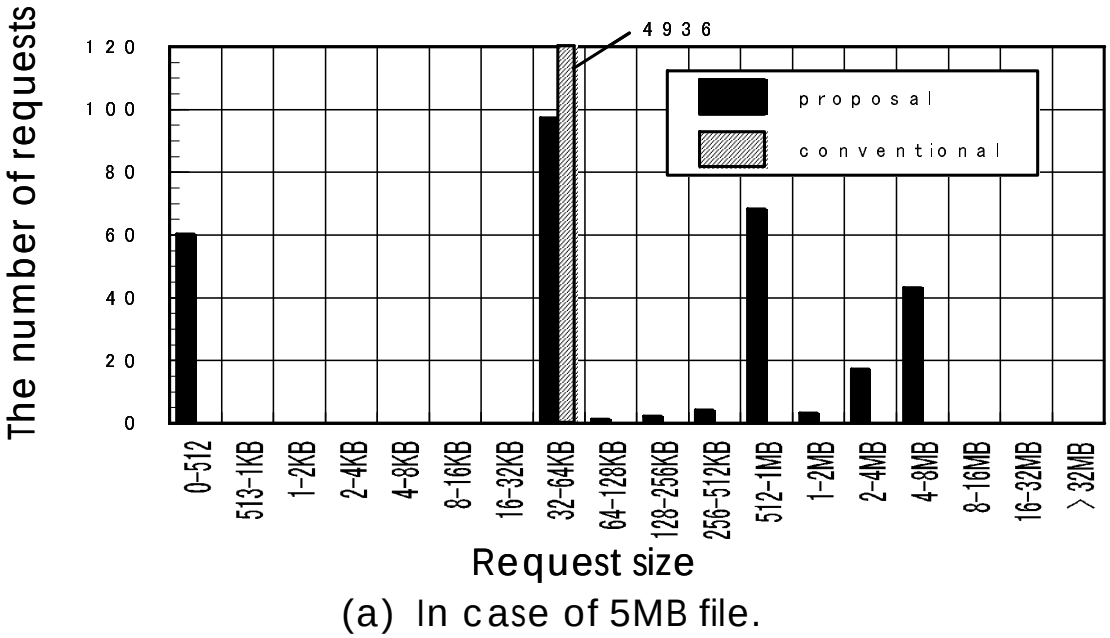
本節では，リクエスト数削減手法の効果について検討する．まず，リクエスト数とファイルサイズの関係について調べる．図 3.10 のベンチマークプログラムを用いて，図 3.11 のファイル読出し速度を測定した場合，ファイルシステムからデバイスドライバに発行されるリクエスト数を観測する．ファイル配置に関係するファイルシステムの初期化条件は，3.5.3 節における測定時と同じである．測定結果を図 3.14 に示す．図から明らかなように，ファイルサイズが 64KB の場合を除いてリクエスト数削減手法が有効に機能し，リクエスト数を削減していることがわかる．ファイルサイズが大きいほどその効果が大きい．ファイルサイズが 1MB のとき，わずかな増加が見られるが，これは 3.5.3 節で述べた i-node の一次間接ブロックの利用開始に伴うリクエスト発生によるものである．

最後に，リクエストサイズの度数分布について，図 3.15 に示す．図 3.15（a）は 60 個の 5MB ファイルを読出した場合，図 3.15（b）は 3 個の 100MB ファイルを読出した場合の結果であり，図 3.10 のベンチマークプログラムを実行させたときのファイルシステム

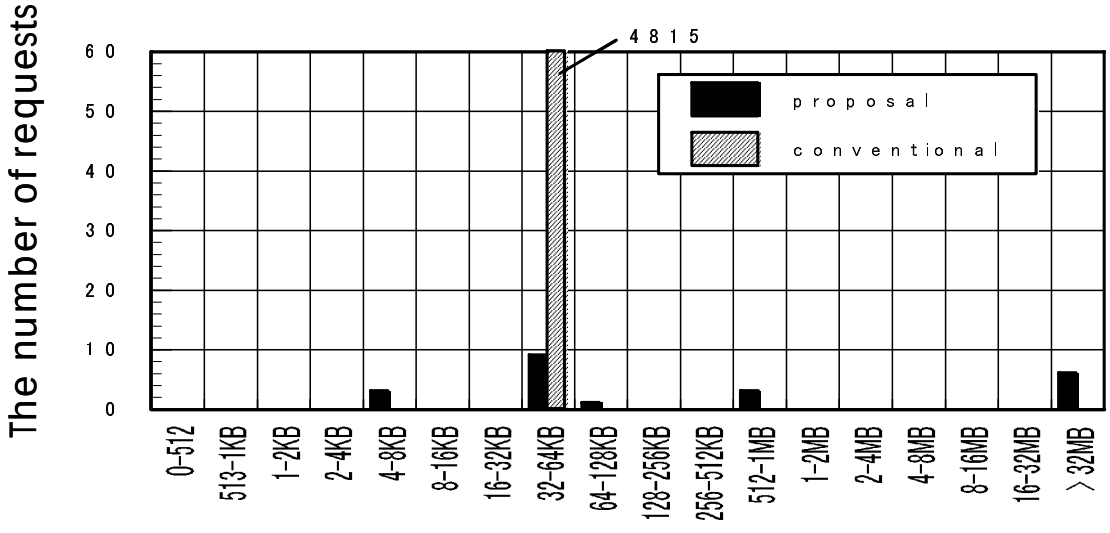
からデバイスドライバに発行されたリクエストを観測し，引数のリクエストサイズごとの頻度を測定した．

従来の UFS では UFS ブロックサイズと同じ 64KB のリクエストしか存在しないが，提案手法を実装した UFS では，64KB を超える多くのリクエストが発行されていることがわかる．特に，図 3.15（b）のファイルサイズが 100MB の場合，32MB を超えるリクエストサイズを引数とするリクエストが 6 回発行されていた．図 3.15（a），図 3.15（b）を比較すると明らかなように，ファイルが大容量になるほど効果が大きく，このことが高速な連続シーケンシャル読出し速度の結果に結び付いている．なお，提案手法を適用した場合に 64KB 未満のリクエストの発生が見られるが，これはリクエスト削減手法においてブロックの連続性を調べるために i-node の管理情報を読出したときに発生したものであり，その頻度はわずかである．64KB 以下のリクエスト数を比較すると，従来の UFS では約 4800 回であるのに対して，提案手法を実装した UFS では，ファイルサイズが 5MB の場合は約 160 回，ファイルサイズが 100MB の場合は 20 回以下である．

以上の実験結果から，提案手法のリクエスト数削減手法の有効性を確認することができた．



(a) In case of 5MB file.



(b) In case of 100MB file.

図 3.15　リクエストサイズの度数分布

3.7 関連研究

UNIX システムに関する研究において，ファイルシステムは重要な研究対象の一つであり，活発な検討が行われている．最も広く用いられているファイルシステムは FFS [29]，あるいは，これを改良した UFS [30] であり，20 年以上に渡って使われ続けている [58]．本検討で指摘したように初期の設計パラメータが現在のハードウェア性能や利用規模に適合しない点が増加しつつあり，最近では，LFS（Log-structured File System）[53] や XFS（eXtended File System）[59] などの様々なファイルシステムが提案され，UNIX システムに実装されている．LFS はジャーナルファイルシステム（Journaling File System）とも呼ばれ，小さいサイズの書込みリクエストに対する性能改善に効果的であり，このアイデアは，XFS をはじめ，Linux システムの ext3 ファイルシステム（3rd Extended File System）[58], [60] などで採用されている．しかし，大容量ファイルの入出力時は，LFS と UFS の性能はほとんど同じであることが文献 [61] で指摘されており，より効率の良いファイルシステムの開発が検討課題となっている．

一般に，ファイルシステムは，様々なファイルサイズ，アクセスパターンに柔軟に対応できるように万能な設計がなされていなければならない．XFS はこの点で有力なファイルシステムといわれており，大容量ファイルの入出力とソフトウェア RAID が考慮された設計となっている．前者については，ファイルのシーケンシャル配置と DMA 転送の効率化が図られている．また，後者については，二つ以上の Raw デバイスを一つの論理デバイスとして束ねる機能がファイルシステムのレベルで提供されている．しかし，XFS ではボリュームマネージャ（Volume Manager）がソフトウェア RAID の管理に介在するため，オーバヘッドを避けることができない．本章で提案した手法はベースとなるファイルシステムの記録形式を変更しないで適用可能であり，他のファイルシステムへの応用も可能である．したがって，さらに検討を進めることによって，XFS にも適用可能であると考えられる．

また，最近の研究の方向とし，分散ストレージネットワーク（Distributed Strage Network）がある．文献 [62] では，Disk-Direct I/O 方式が提案され，ハードディスクへのアクセスパターンに性能が依存しない特徴をもちながら最高 33.5Mbytes/s のシステム性能が得られたことが報告されている．更に，文献 [63] で提案されている Sawmill は RAID II システム [51] をベースとしたネットワークファイルシステムである．専用のハードウェアを設計し，ハードディスクとネットワーク間に Direct Path を設けることによって，約 21Mbytes/s の読出し速度を達成したことが報告されている．

以上の関連研究における本提案の位置付けをまとめると，次のとおりとなる．

1. ベースとなる UFS の記録形式を変更することなく高速化を図ることができる．
2. 専用のハードウェアを必要とせず，汎用のパーソナルコンピュータと UNIX システムを用いて構成可能である．
3. 分散ストレージネットワークではなくスタンドアロンのシステムであっても 9 台のハードディスクと 3 枚の RAID コントローラを用いて最高 110Mbytes/s の連続ファイル読出し速度を達成できる．

3.8 むすび

高速ネットワークと同等の連続読出し速度を有するファイルサーバを UNIX システム上に構築するために，UFS の記録形式を変更しない設計方針のもと，リクエスト数削減手法とダイレクトアクセス手法を提案した．これらのソフトウェア構成手法の有効性を確認するため，実際に試作システムを製作し，実験による検討を行った．

デバイスドライバの Raw デバイス性能の測定結果から，RAID コントローラ数 $NR = 3$，ハードディスク数 $ND = 9$，RAID-0 の構成のとき，リクエストサイズが 1.5MB 以上の場合において 100Mbytes/s を超える連続読出し速度を達成することができた．また，連続書込み速度については 60Mbytes/s 以上の性能を実現した．

ファイルシステム経由のファイル入出力速度の測定結果から，ファイルサイズが 50MB 以上の場合において 100Mbytes/s を超える連続シーケンシャル読出し速度が得られることを確認した．連続シーケンシャル書込み速度については，最高 60Mbytes/s の性能であった．

以上の実験結果から，提案手法によって，UNIX システムにおいて，階層ストライピングデバイスドライバを最低限の性能劣化で利用可能なファイルシステムを構築できることを確認した．

第4章
パケットごとの締切り時刻に基づいたネットワーク制御法

4.1 まえがき

近年のIPネットワークの普及に伴い，リアルタイムデータからバルクデータまで，ネットワークを介して様々な情報交換が可能となっている．利用アプリケーションの多様化によって，通信品質とネットワーク機能に対する要求が増し,「高速化」「遅延保証」「マルチキャスト」がキーワードとなっている．本章では，遅延保証，すなわち，指定された時間内に相手先にパケットを送り届ける技術を扱う．

なお，本章以降における検討では，ルータ装置と通信端末を総称してノードと表記することにする．

IPネットワークにおいてエンド・エンド間での遅延を保証する技術として，多くの方式が検討されている．代表的なものとして，WFQを用いた帯域予約方式（以下，WFQ方式と称する）と，フローごとの締切り時刻を優先度とするEDD方式やEDF待ち行列などがある．これらはいずれも優先制御技術であり，各中継ノードにおける最大遅延を保証する技術である．エンド・エンド間での遅延保証を行うには，通信を開始する前にユーザのコネクション要求に従って経路を選択し，中継ノードごとに帯域予約，あるいは，許容遅延の設定を行う．それぞれの中継ノードでは，コネクションごとに設定・管理された要求帯域/許容遅延に従ってパケットの優先制御を行う必要があり，厳密な遅延保証が実現できる反面，ノード負荷が大きいといった問題があった．

本章では，ノード負荷の軽減のため，次に挙げる方針に従ったシンプルで柔軟な優先制御・経路制御法を検討し，ソフトなエンド・エンド間での遅延保証の実現を目指す．

- コネクション管理を行わない．
- パケット上にあて先ノードにおけるデッドラインをラベル付けし，パケットごとに

独立した優先制御・経路制御を行う．

- パケットの送達順序を保持しない．トランスポート層において並べ換えが可能であるとする．デッドラインまでの時間が長い場合では，バルク系のコンテンツ配信などでの利用を想定している．

- 複数経路を用いる．デッドラインを守れる範囲で迂回経路を柔軟に選択することで負荷分散を図るとともに，空きリンク帯域を活用する．

- ルーチングテーブル上に集約された遅延パラメータ情報を使ってあて先ノードまでの遅延予測を行う．

本方針では，コネクション管理を行わないため，ネットワークサービス種別の識別が困難となるが，フローごとに制御する方式と比較してスケーラビリティに関して有利である．

受信側でパケットの送達順序が保持されないことについては，順序の並べ換え [69] や順序の逆転を考慮したトランスポートプロトコル [47] を用いて対策が可能である．アプリケーション層までを含めた実際のスループットについては，並べ換え処理のオーバーヘッド，遅延ゆらぎの影響，プロトコルの再送処理などによって性能が左右されるが，本検討ではパケットごとの遅延保証を目的としているため，パケットレベルでの性能をデッドライン違反率によって評価する．

なお，ソフトな遅延保証の実現とは，従来のようにコネクション管理とトラヒックシェーピングによって厳密な遅延保証を行うのではなく，与えられた要求遅延が満足されないパケットの割合を一定値以下に抑える確率サービス [3] を提供するという意味である．ソフトな遅延保証に関連する従来検討 [36], [37], [41]〜[43] では，EDD 方式と同様に，あて先までの単一経路上の各ノードに EDF 待ち行列を配する方式が用いられている．EDF 待ち行列が用いられる理由はノード内スケジューリングが最適となるためであり，準最適な WFQ と比較して性能が良い [42], [43]．本章では，EDF 待ち行列と複数経路を用い，パケットごとにラベル付けされたデッドラインに基づいて柔軟に負荷分散を行う方式を提案し，優れた遅延特性を得ることを図る．

4.2 関連技術

4.2.1 遅延保証

これまでの研究における遅延保証については二つのアプローチがあった．一つは帯域割当てであり，もう一つは中継ノードにおける許容遅延の設定である．これらの技術では，主にリアルタイム系のトラヒックにおける遅延保証のために利用することを目指しており，コネクション設定（帯域予約）が必要とされ，場合によってはトラヒックシェーピングが併用される．

帯域割当てに基づく WFQ 方式では，各中継ノードに複数の待ち行列を用意し，設定された重みレートに従って待ち行列の先頭からパケットを送出する．待ち行列は，コネクションごと，あるいは，コネクションを集約したサービスクラスごとに用意される．Leaky Bucket との組合せによって最大ノード遅延が保証されることが知られている．

一方，許容遅延に基づく EDD 方式では，コネクション設定時にユーザから要求されたエンド・エンド間の許容遅延から経路遅延を差し引いた遅延余裕を各中継ノードに分配することで各中継ノードの許容遅延を設定し，デッドラインのスケジューリングを行う．各中継ノードでは EDF 待ち行列を用いて優先制御を行う．パケットがノードに到着するごとに，それぞれのパケットに対して到着時刻に許容遅延を加算したローカルデッドラインを割当てる．出力バッファの待ち行列では，ローカルデッドラインの早いパケットから順に送出する優先制御を用い，最大ノード遅延を設定された許容遅延以下に制限することができる．なお，従来研究の中にはエンド・エンド間での遅延保証方式を広義の意味で EDF 方式と称するものもあるが，ここでは，EDF は FCFS（First Come First Serve）と同様に待ち行列の制御手法を表す用語として扱うこととする．

これらの方式による遅延保証が厳密に機能するためには，いわゆる「スケジューリング」が必要とされ，コネクション設定時に遅延保証が可能かどうか，すなわちスケジューラブル条件を満たすかどうかを確認し，条件を満たさないコネクション要求は拒否される．

EDD 方式で問題となるのは，遅延余裕の分配法である．すなわち，EDF 待ち行列における優先度の割当て方がエンド・エンド間での遅延保証の性能を左右し，コネクション設定時ではなくネットワークの状況変化に動的に対応可能な分配法が検討課題となっている．

4.2.2 複数経路を用いた経路制御法

複数経路を用いて負荷分散を行う従来技術として，OSPF（Open Shortest Path First）プロトコルにおけるECMP（Equal-Cost Multi-path）ルーチング[70], [71]，MPLS（Multi Protocol Label Switching）におけるOMP（Optimized Multi-path）ルーチング[72]，適応型マルチパスルーチング（AMR: Adaptive Multi-path Routing）[73]〜[75]などがある．これらの技術はリンク利用率が等しくなるように最適化を図る．例えば，文献[74], [75]で提案されている適応型マルチパスルーチングでは，ある中継ノードにおいてパケットの経路選択を行う場合，あて先までのコストが等しいK本の経路候補が存在し，経路i $(i = 1, \ldots, K)$に対する自ノードの出力バッファの待ち行列長をN_iとするとき，次の確率P_iで経路iを選択する．

$$P_i = \frac{e^{-\beta N_i}}{\sum_{j=1}^{K} e^{-\beta N_j}} \quad (i = 1, \ldots, K) \tag{4.1}$$

ここでβ $(\beta > 0)$は負荷分散の感度を決めるパラメータである．式（4.1）からわかるように，利用率の低いリンクの選択頻度を高め，すべてのリンク利用率が等しくなるように負荷分散を行う．このように，従来の適応型マルチパスルーチングは自ノードの待ち行列長の情報を使う簡単な経路制御法であるが，あて先ノードまでのパス長や，途中の中継ノードにおける混雑具合（遅延）については考慮されないため，遅延保証のための経路制御法としては不十分である．

4.3 提案方式

4.3.1 アプローチ

従来方式におけるエンド・エンド間での遅延保証は，遅延が最小のスケジューラブル条件を満たす単一経路に対してコネクション設定を行い，コネクションごとに各ノードにおける許容遅延を定め，管理する必要があった．更に，コネクション設定時にエンド・エンド間の遅延が守れるかどうかを判断し，守れないコネクション要求を拒絶することで厳密な遅延保証を実現していた．

本章では，複数経路を用いたパケットごとの処理によるソフトな遅延保証のしくみを提案する．具体的には，各パケットにあて先ノードにおけるデッドラインをラベル付けし，各中継ノードは，そのデッドラインに基づいて優先制御，経路制御を行う．デッドライン

表 4.1　提案方式と従来方式の比較

方式	経路制御		優先制御		フロー管理（受付制御）	送達順序の保証
	経路	経路制御基準	方式	優先度基準		
SPF-FCFS	Single Path	最小メトリック（最小遅延）	FCFS	到着順で処理される	不要	可
SPF-EDF	Single Path	最小メトリック（最小遅延） ※ スケジューラブルなパス	EDF	コネクション設定時に各ノードの許容遅延を設定する	必要	可
AMR-FCFS	Multi Path	ホップ数と自ノードの出力バッファの待ち行列長に基づいて経路選択を行う	FCFS	到着順で処理される	不要	不可
提案方式	Multi Path	あて先までの平均遅延時間とデッドライン違反率の予測値に基づいて経路選択を行う	EDF	動的に各ノードにおける締切時刻を設定する	不要	不可

については，例えばIPv6パケットの場合は，中継ノードに対する制御情報を記述するために規定されたHop-by-hop optionsヘッダに記述する．

表 4.1 に，提案方式と従来方式との比較について整理する．表中のSPF-FCFS（Shortest Path First and FCFS）方式は，通常の最短経路選択による経路制御とFCFS待ち行列による優先制御を行う方式である．SPF-EDF方式は，便宜上，平均遅延が最小の経路をスケジューラブル条件を満たす経路とみなした場合のEDD方式，あるいは，単一経路を用いてソフトな遅延保証を試みた方式に相当する．AMR-FCFS方式は，4.2.2 節で説明した適応型マルチパスルーチング方式である．提案方式では，パケットの送達順序が保持されないことを許容し，デッドラインが守れる範囲で柔軟に迂回経路を選択し，トラヒックの負荷分散の効果を得ることを図る．これによって，ソフトな遅延保証が実現できる負荷範囲の拡大が期待されることになる．

4.3.2　ノード構成

図 4.1 にノード構成について示す．まず，入力リンクから到着したパケットのヘッダ内容を確認する．デッドラインが記述されていない従来パケットについては，ルーチングテーブルを参照して最短経路を求め，FCFS待ち行列に送られる．デッドラインが記述されている場合は，デッドラインに基づく経路制御と優先度割当てを行い，EDF待ち行列に送られる．これら2種類の待ち行列から，WRR（Weighted Round Robin）方式に従って設定帯域比率でパケットを取り出し，次ノードに送出する．

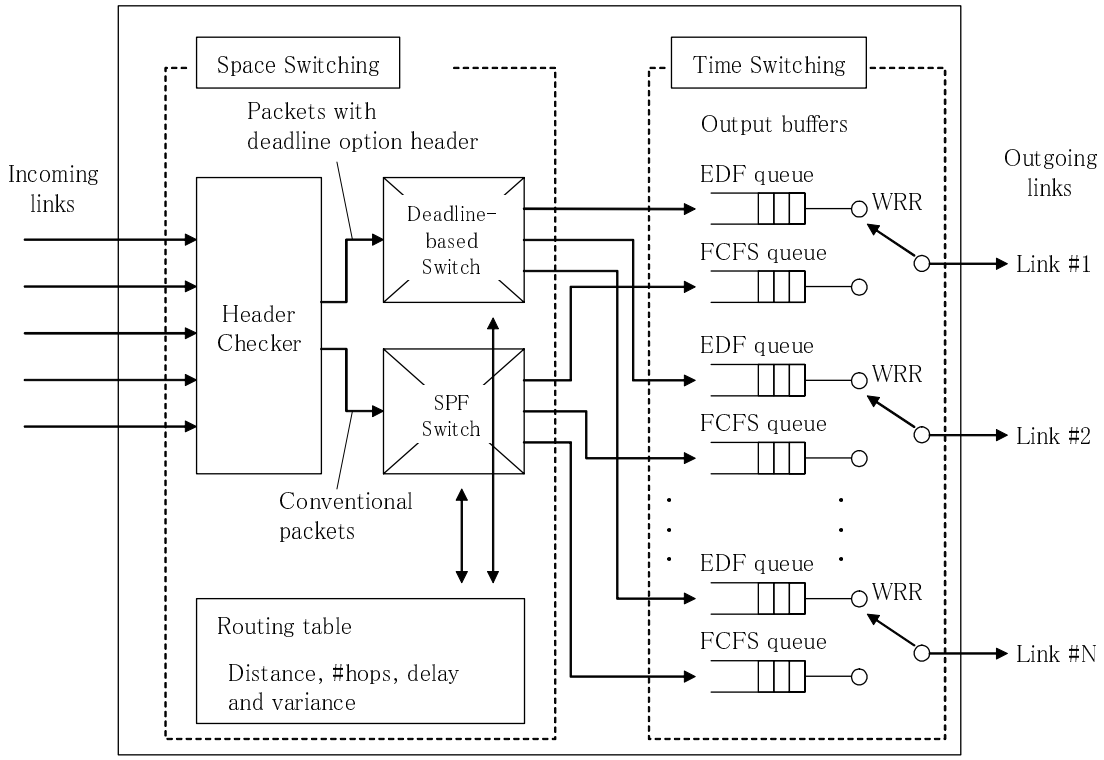


図 4.1　ノード構成

4.3.3　遅延予測のためのルーチングテーブル

遅延保証のためには，中継ノードにおけるあて先ノードまでの遅延予測が必要となる．本方式では，各ノードで定期的に計測されたリンクの出力バッファにおける平均待ち時間 W，リンク速度 μ，リンク遅延 d の遅延パラメータを交換してルーチングテーブルを構築し，この情報を用いて遅延予測を行う．いま，リンク i を経由して次ノードに到着するまでのノード間の平均遅延 D_i は次式で与えられる．

$$D_i \;=\; W_i + 1/\mu_i + d_i \tag{4.2}$$

ここで，添字の i はリンク i に関する遅延パラメータであることを示す．実際のルーチングプロトコルにおいては，リンクのコストとして，IGP（Interior Gateway Protocol）の OSPF ではリンク速度の逆数 $1/\mu$ を基準とした値が用いられ，EGP（Exterior Gateway Protocol）の BGP（Border Gateway Protocol）ではリンク遅延 d を基準とした値が用いられる．これは，物理的規模が小さいネットワークを扱う IGP ではリンク速度がノード間遅延を支配し，逆に物理的規模が大きい EGP ではリンク遅延がノード間遅延を支配するためである．本章では，遅延保証を目指しているため，すべての遅延パラメータを考慮した式（4.2）の値を用いる．

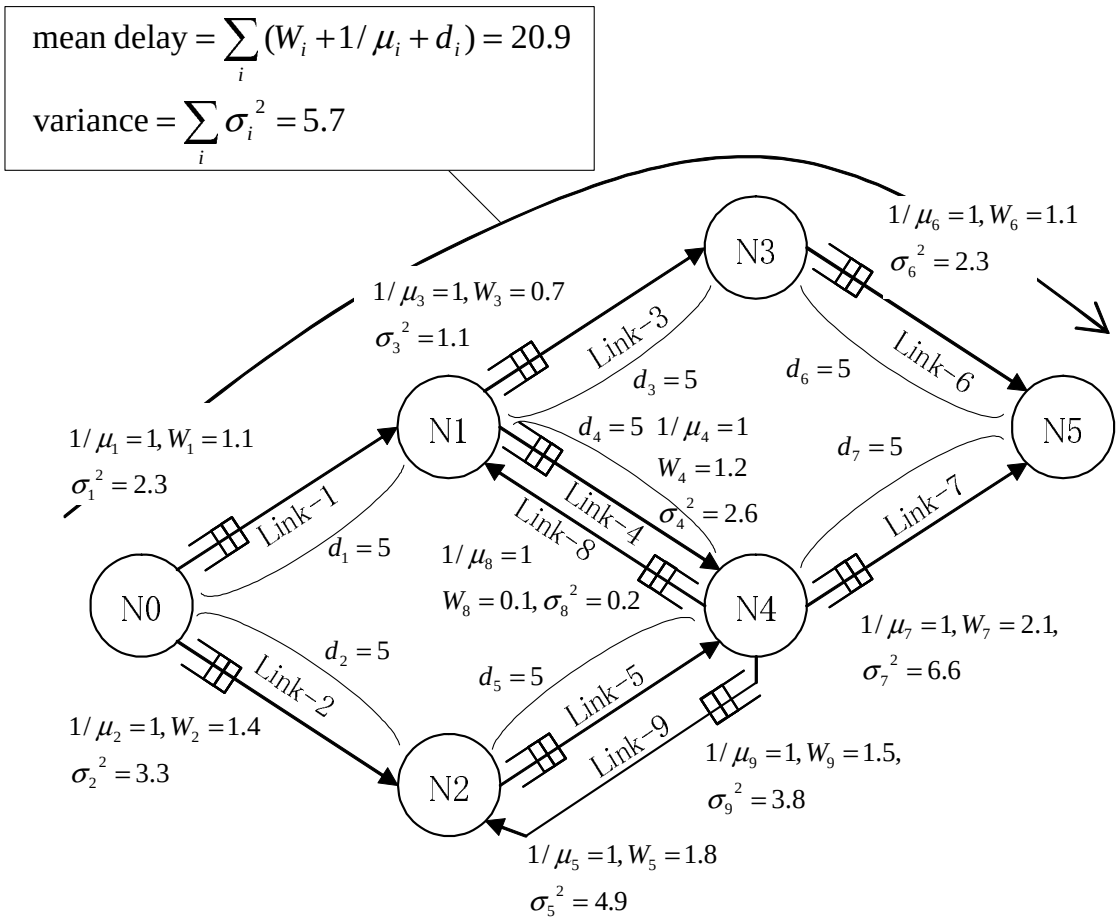


図 4.2　ネットワーク構成例と遅延パラメータ

ネットワーク上のすべてのリンクの遅延パラメータを交換・収集することは困難であるため，パラメータの集約を行う．具体的には，あて先ノードまでの平均遅延が最小となる経路上での D と σ^2 の積算値とホップ数（以下，積算遅延パラメータと称する）を，各出力リンクごとにルーチングテーブルに登録する．定期的に隣接ノードのルーチングテーブルの情報を得て，これに自ノードの遅延パラメータを加算し，遅延予測のためのルーチングテーブルを構築・更新する．理解のために，図 4.2 に示したネットワーク構成例について考える．ノード N0 におけるルーチングテーブルを構築した結果を表 4.2 に示す．図 4.2 中の σ_i^2 は，リンク i 宛の出力バッファにおける待ち時間の分散である．なお，遅延パラメータを 1 パケットの平均送信時間で正規化している．いま，ノード N0 がノード N5 宛のパケットを処理する場合を考える．表 4.2 のルーチングテーブルを参照することで，リンク 1 経由でパケットを送出すれば平均遅延 20.9 の経路が最小平均遅延経路であり，そのパスに沿った分散の合計は 5.7，ホップ数は 2 であることがわかる．同様に，リンク 2 経由でパケットを送出した場合は，平均遅延 23.3，分散 14.8，ホップ数 2 が最小平

表 4.2　ノード N0 におけるルーチングテーブルの例

Dest. node	Output link #	#Hops	Distance	Minimum mean delay path		
				Delay	Variance	#Hops
N1	1	0	5	7.1	2.3	0
	2	2	15	21.3	8.4	2
N2	1	2	15	21.8	8.7	2
	2	0	5	7.4	3.3	0
N3	1	1	10	13.8	3.4	1
	2	3	20	28.0	9.5	3
N4	1	1	10	14.3	4.9	1
	2	1	10	15.2	8.2	1
N5	1	2	15	20.9	5.7	2
	2	2	15	23.3	14.8	2

均遅延経路の積算遅延パラメータとなる．これらの情報を用いて，リンク 1，リンク 2 の経路選択の判断と，出力バッファの EDF 待ち行列における優先度の割当てを行う．

4.3.4　経路制御法

本方式では，デッドラインが守れる範囲でトラヒックの負荷分散を行うために，経路制御において複数の送出リンクの候補を評価する．

図 4.3（a）はあて先ノードまでに 2 ホップの中継ノードを経由する場合の各ノードにおける到着時刻の分布を表している．横軸は送信ノードからの経過時間，縦軸は PDF（Probability density function）である．ノードをホップするごとに遅延分散が増加し，ノード遅延が独立であるとき，中央極限定理によって正規分布に漸近する．実際の IP ネットワークにおいては，エンド・エンド間の遅延分布は，ネットワーク負荷，ホップ数，バーストトラヒックの影響などに大きく左右され，裾野の広い分布となるが，本検討では，簡単のため，正規分布で近似できると仮定する．このことから，あて先ノードにおけるデッドライン違反率の予測値を，平均遅延と分散の積算値をもとに誤差関数を使って計算すること

ができる．本方式では，ノードの出力バッファにEDF待ち行列を用いて優先制御を行う．パケット上に記述されたデッドラインに基づいて各ノードのEDF待ち行列のデッドラインを設定しながら，あて先ノードまでの動的な優先制御を行う．すべてのノードにおけるデッドラインを逐次守ることによって，結果としてエンド・エンド間のデッドラインが守られることになる．このことから，制御基準を，エンド・エンド間のデッドラインから，ノードごとのデッドラインに落し込む必要がある．図 4.3（b）に，次ノードにおける遅延分布の予測について示す．エンド・エンド間の遅延を1ホップ当りの尺度で捉え，遅延分布をエンド・エンド間の場合と同様に正規分布で近似する．ノード到着時刻からデッドラインまでの残時間より，経路の平均遅延時間を差し引いた値を遅延余裕と呼ぶことにする．本方式では，文献[34]の比較対象方式と同様に，各ノードにおいて遅延余裕を均等に分配してEDF待ち行列のデッドラインを設定する．ただし，従来方式ではコネクション設定時に遅延余裕を均等に設定するのに対し，本方式ではパケットがノードを経由するごとに動的に配分計算を行う点が異なる．以上のことから，次ノードにおけるEDF待ち行列のデッドライン違反率 P_{v} は相補誤差関数（erfc）を用いて次式で与えられる．

$$\begin{aligned} P_{\mathrm{v}} &= \frac{1}{2}\mathrm{erfc}\left(\frac{\frac{(t_{\mathrm{deadline}}-t)-\sum_{i=1}^{h+1} D_i}{h+1}}{\sqrt{2\,\frac{\sum_{i=1}^{h+1}\sigma_i^2}{h+1}}}\right) \\ &= \frac{1}{2}\mathrm{erfc}\left(\frac{(t_{\mathrm{deadline}}-t)-\sum_{i=1}^{h+1} D_i}{\sqrt{2(h+1)\sum_{i=1}^{h+1}\sigma_i^2}}\right) \end{aligned} \tag{4.3}$$

ここで，t_{deadline} はパケットヘッダに記述されたあて先ノードにおけるデッドライン，t はノード到着時刻（現在時刻），h は自ノードを含まないあて先までの残ホップ数であり，$\sum_{i=1}^{h+1}$ は平均遅延が最小となる経路上での合計を表し，D_i と σ_i^2 は，その経路上の i 番目のリンクに関する遅延パラメータである．$\sum_{i=1}^{h+1} D_i$，$\sum_{i=1}^{h+1}\sigma_i^2$ の値については，ルーチングテーブルに集約済みの値を利用する．

本検討では，ルーチングテーブルを参照し，平均遅延が最小となる出力リンクを第1候補（主経路），2番目に最小となる出力リンクを第2候補（迂回経路）として，これら2経路を用いた負荷分散について評価する．それぞれの出力リンクについて求めた次ノードにおけるデッドライン違反率を P_{v1}，P_{v2} とし，出力リンクの経路選択率を r_1，r_2 とするとき，それぞれの経路経由でのデッドラインを守れないパケット数が等しくなるように制御する．このとき，r_1，r_2 は次式で与えられる．

$$r_1 = P_{\mathrm{v2}}/(P_{\mathrm{v1}}+P_{\mathrm{v2}}) \tag{4.4}$$

$$r_2 = P_{\mathrm{v1}}/(P_{\mathrm{v1}}+P_{\mathrm{v2}}) \tag{4.5}$$

これらの確率で第 1 候補リンク，第 2 候補リンクの経路選択を行い，負荷分散と遅延保証のバランスをとる．第 2 候補を利用する迂回によってネットワーク全体でのトラヒックが増加することになるが，これについては，平均待ち時間の増加として反映され，経路選択率へのフィードバックの結果，最適点で平衡状態となる．なお，デッドライン t_{deadline} までの残時間 $t_{\mathrm{deadline}} - t$ よりも経路の平均遅延の方が大きい出力リンクについては，あて先ノードにおけるデッドラインを守れない確率が高いため，候補リンクから除外する．また，デッドラインを守れるリンクが一つも存在しない場合は，そのパケットを破棄してネットワーク負荷を低減する手法も考えられるが，本検討では破棄せずに，最小の平均遅延の出力リンクに送出することとする．

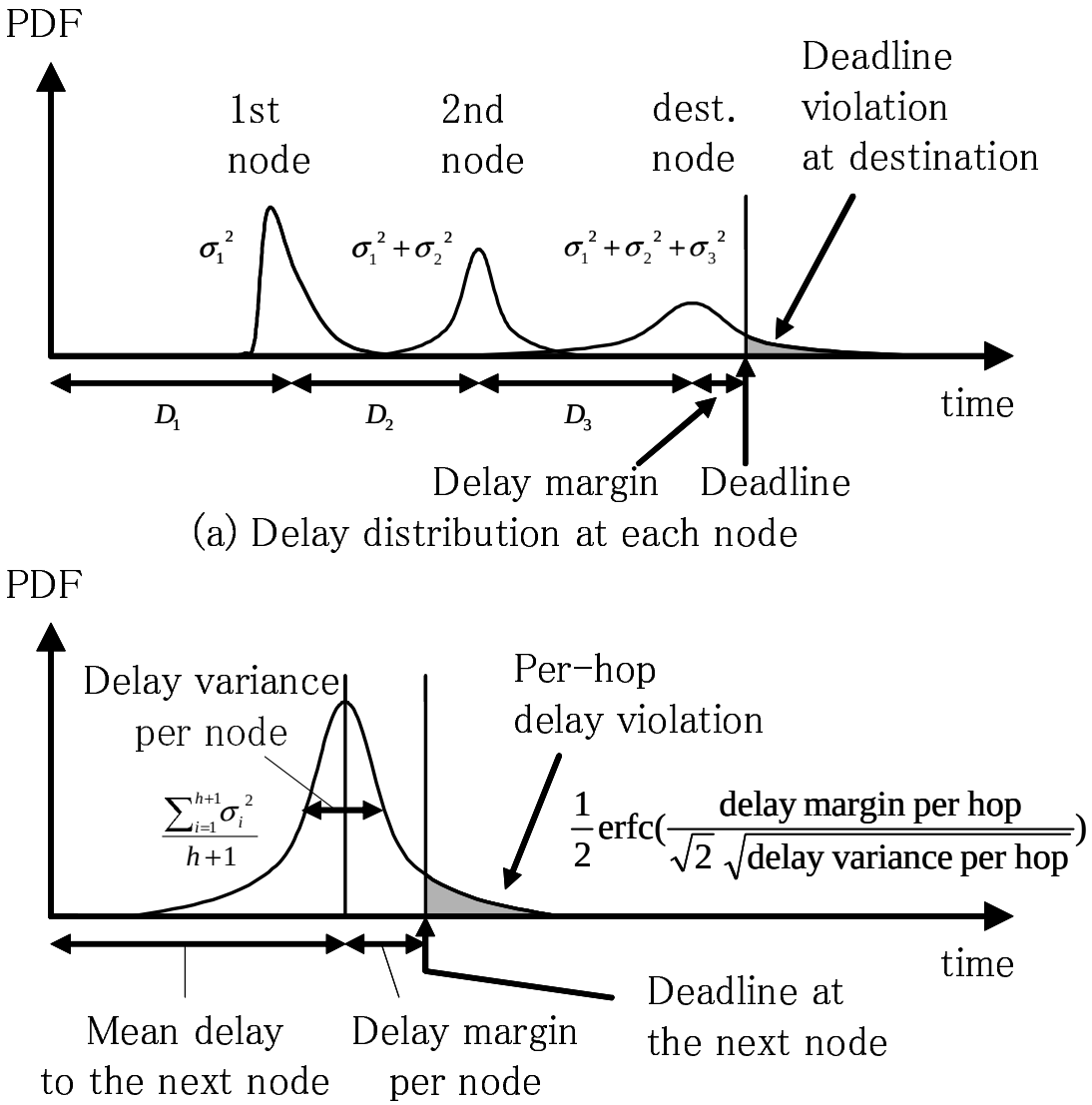


図 4.3　遅延分布とデッドライン違反率

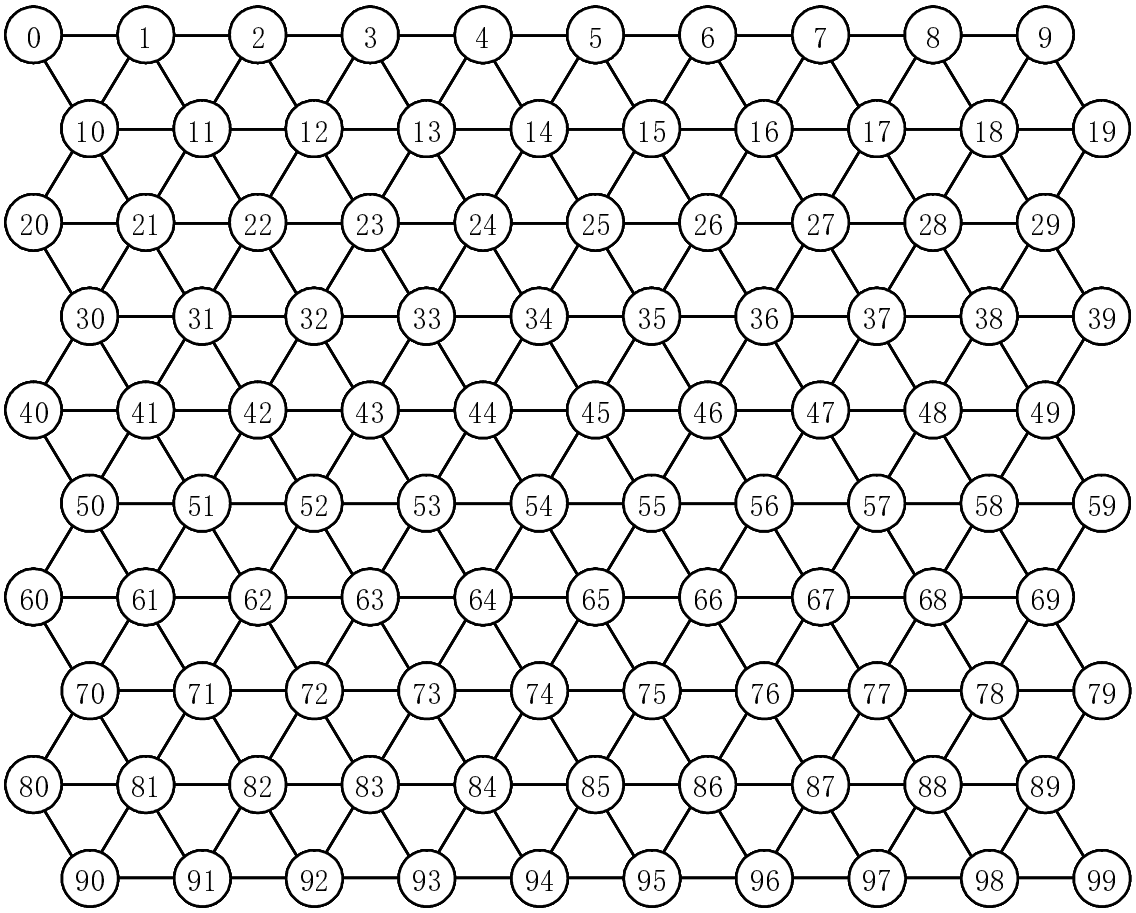


図 4.4　評価ネットワーク

4.4　性能評価

4.4.1　シミュレーション条件

提案方式を表 4.1 に示した従来方式と比較するため，計算機シミュレーションにより性能評価を行った．評価ネットワークとして，図 4.4 に示したノード数が 100，ノード当りのリンク数が 6 のメッシュネットワークを用いた．パケット長は一定，リンク速度はすべて等しいとし，以下，1 パケットの送信時間で時間を正規化する．簡単のため，すべてのリンク遅延 d を 5，リンク速度 μ を 1，ノードの最大バッファ長を 1000 パケットとした．各ノードにおいて，一様の生起率 λ でパケットがポアソン生起し，あて先ノードについては一様乱数で定めた．このとき，メッシュネットワークの中央部ではリンク利用率が高く，周辺部では利用率が低いトラヒック分布となる．遅延パラメータの計測とルーチングテーブルの更新の間隔は 100 とし，更新は理想的に行われるとする．なお，バッファあふれによって破棄されたパケットは計測の対象としない．本検討の条件ではトラヒック変動

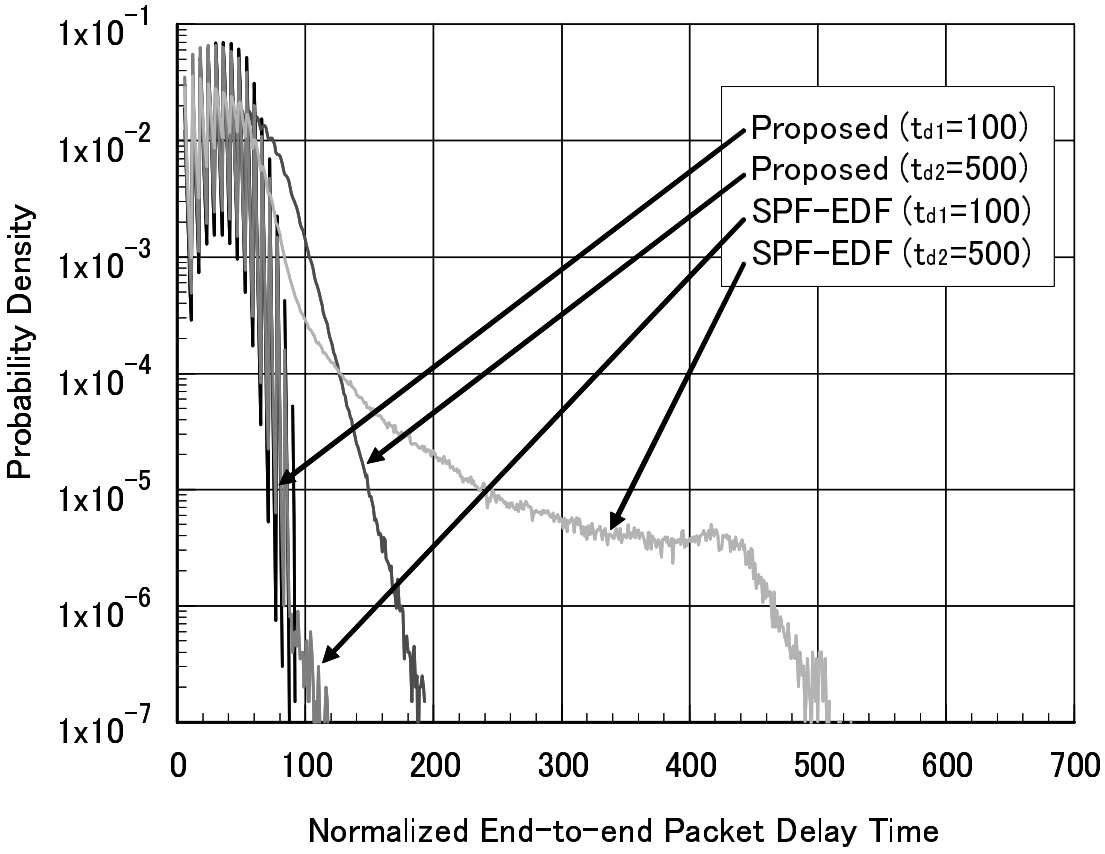


図 4.5　全体の遅延分布特性（提案方式，SPF-EDF 方式，$t_{\mathrm{d1}} = 100$，$t_{\mathrm{d2}} = 500$，$\lambda = 0.4$）

を与えないため，ルーチングテーブルの積算遅延パラメータは時間の経過とともに収束する．シミュレーション時間は 2×10^6 とし，開始から半分の結果を捨てて，積算遅延パラメータが収束した後の特性を得た．また，本検討では基本特性を把握するために，全ノードが提案方式の処理機能をもち，すべての生起パケットに対してデッドラインを与えた．

4.4.2　エンド・エンド間の遅延分布特性

まず，生起パケットの 50%（ランダムに選択）に対して (生起時刻) + t_{d1}（$t_{\mathrm{d1}} = 100$）のデッドラインを与え，残りの 50%のパケットに対して (生起時刻) + t_{d2}（$t_{\mathrm{d2}} = 500$）のデッドラインを与えた場合の特性を求めた．t_{d1}，t_{d2} の値は，送信元ノードとあて先ノードの距離に関係なく一定とする．

ノード生起率 $\lambda = 0.4$ の場合のエンド・エンド間の遅延時間分布を図 4.5 に，$\lambda = 0.6$ の場合のエンド・エンド間の遅延時間分布を図 4.6 にそれぞれ示す．図 4.5 及び図 4.6 の提案

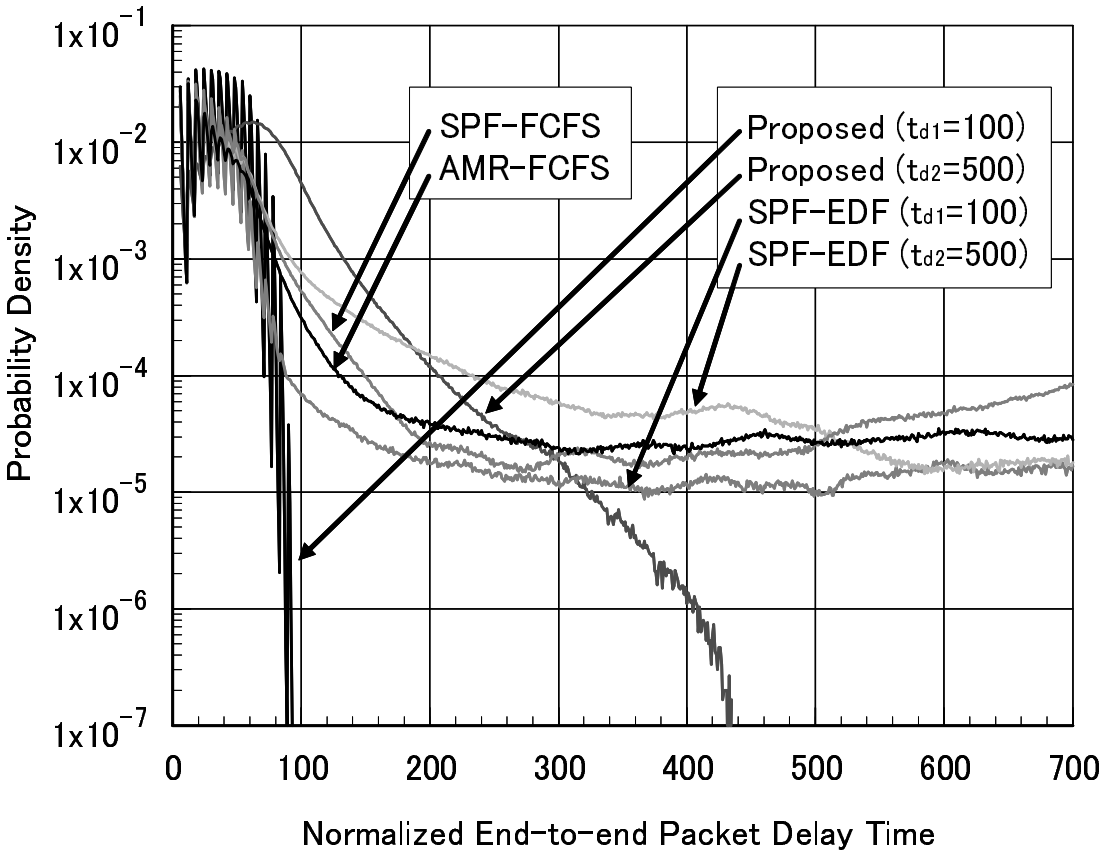


図 4.6　全体の遅延分布特性（提案方式，SPF-EDF 方式，SPF-FCFS 方式，AMR-FCFS 方式，$t_{\mathrm{d1}} = 100$，$t_{\mathrm{d2}} = 500$，$\lambda = 0.6$）

方式の遅延分布特性を見ると，$t_{\mathrm{d1}} = 100$ のパケットについてはエンド・エンド間の遅延時間が約 95 以下に，$t_{\mathrm{d2}} = 500$ のパケットについては $\lambda = 0.4$ の場合で約 190 以下，$\lambda = 0.6$ の場合で約 440 以下に収まり，デッドラインが守られていることがわかる．一方，従来方式の SPF-EDF 方式については，単一経路であることから輻輳を回避できず，$\lambda = 0.4$ の場合では若干のパケットが，$\lambda = 0.6$ の場合では更に多くのパケットがデッドラインを守ることができない．実際の EDD 方式においては，このようなスケジューラブル条件を満たさない通信要求は拒絶されるため，更に小さい λ の範囲でしか利用されることはない．図 4.6 のデッドラインを用いない SPF-FCFS 方式と AMR-FCFS 方式の遅延分布特性を見ると，慢性的な遅延が発生し，エンド・エンド間の遅延時間が大きいパケットが存在することがわかる．以上のことから，提案方式では，複数経路を用いたデッドラインが守れる範囲内での柔軟な迂回による負荷分散が効果的に機能しているといえる．

次に，図 4.6 の結果から，発信元がノード 21，あて先がノード 78 のパケットに関する遅延分布だけを抜き出したものを図 4.7 に示す．ノード 21 からノード 78 までの最小経路

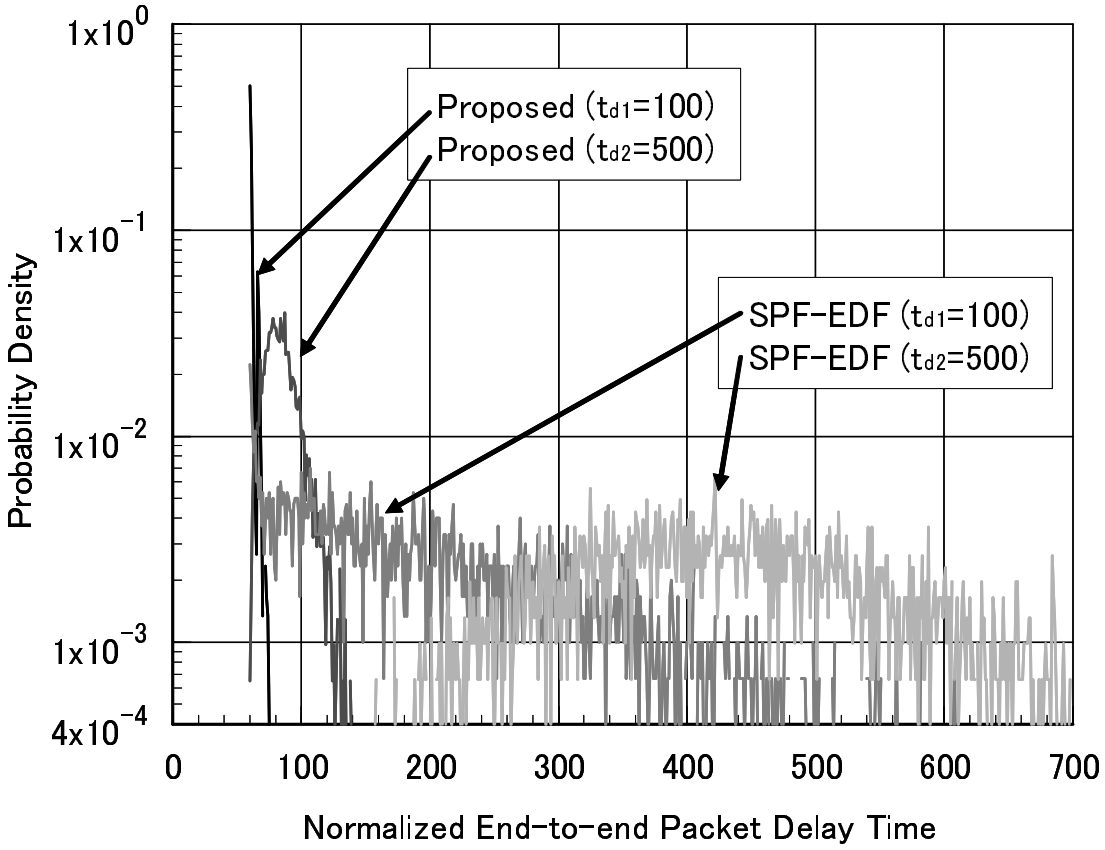


図 4.7　ノード 21→78 の遅延分布特性（提案方式，SPF-EDF 方式，$t_{\mathrm{d1}} = 100$，$t_{\mathrm{d2}} = 500$，$\lambda = 0.6$）

遅延（$\sum_i (1/\mu_i + d_i)$）は 60 である．図 4.7 において，提案方式の遅延分布特性を見ると，$\lambda = 0.6$ の高負荷でありながら，$t_{\mathrm{d1}} = 100$ のパケットの約半数が最小経路遅延に近い遅延で届き，$t_{\mathrm{d2}} = 500$ のパケット遅延は 60〜140 の範囲に分布している．一方，SPF-EDF 方式については，もはやデッドラインを守ることができない．このことから，本方式を用いることにより負荷に対する耐性が高まり，遅延保証が可能な負荷の範囲を広げることが可能となり，有効性を確認することができる．

4.4.3　デッドライン違反率とロス率特性

本方式の適用範囲を確認するために，ノード負荷 λ とデッドライン違反率の関係を計算機シミュレーションによって求めた．ここで，デッドライン違反率を，全送信パケット数に対するデッドラインを守れなかったパケットの割合と定義する．図 4.5，図 4.6 と同じ条件で，λ の値を 0.1〜0.8 まで変化させながら，デッドライン違反率とバッファあふれ

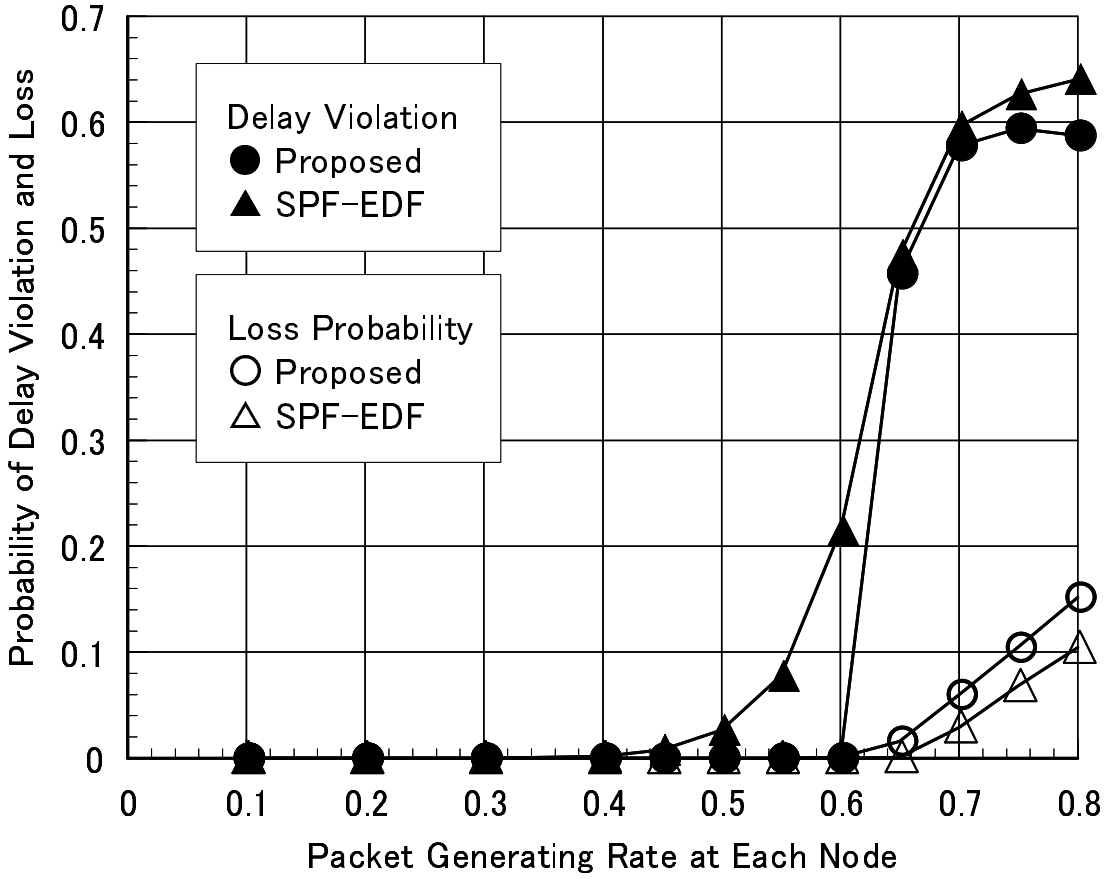


図 4.8　デッドライン違反率とパケットロス率の特性（提案方式，SPF-EDF 方式，$t_{\mathrm{d1}} = 100$，$t_{\mathrm{d2}} = 500$）

によるパケットロス率の特性を求めた．図 4.8 に結果を示す．図から，SPF-EDF 方式では $\lambda = 0.4$ 付近から違反率が上昇しているのに対して，提案方式では $\lambda \leqq 0.6$ の範囲までデッドライン違反率が 0 となっている．このことから，提案方式では高負荷まで遅延保証が実現できていることがわかる．なお，注意すべき点は，$\lambda = 0.6$ 時の提案方式の違反率が 0 であることを保証できないことであり，確率は非常に小さいがデッドラインを守れないパケットがわずかに存在する．これが「ソフトな遅延保証」と呼ぶ理由である．単一経路を用いた従来検討では違反率の上界値が導出されているが，提案方式のように複数経路を用いる場合は解析が困難であるため，本検討ではシミュレーションで性能比較を行った．なお，SPF-EDF 方式は $\lambda \leqq 0.4$ で低いデッドライン違反率を示しているが，これは単一経路を用いてソフトな遅延保証を試みた場合の適用領域に相当し，実際の EDD 方式では，最悪ケースの遅延でスケジューラブル条件が判断されるため，厳密な保証が行える反面，適用範囲は更に狭くなる．

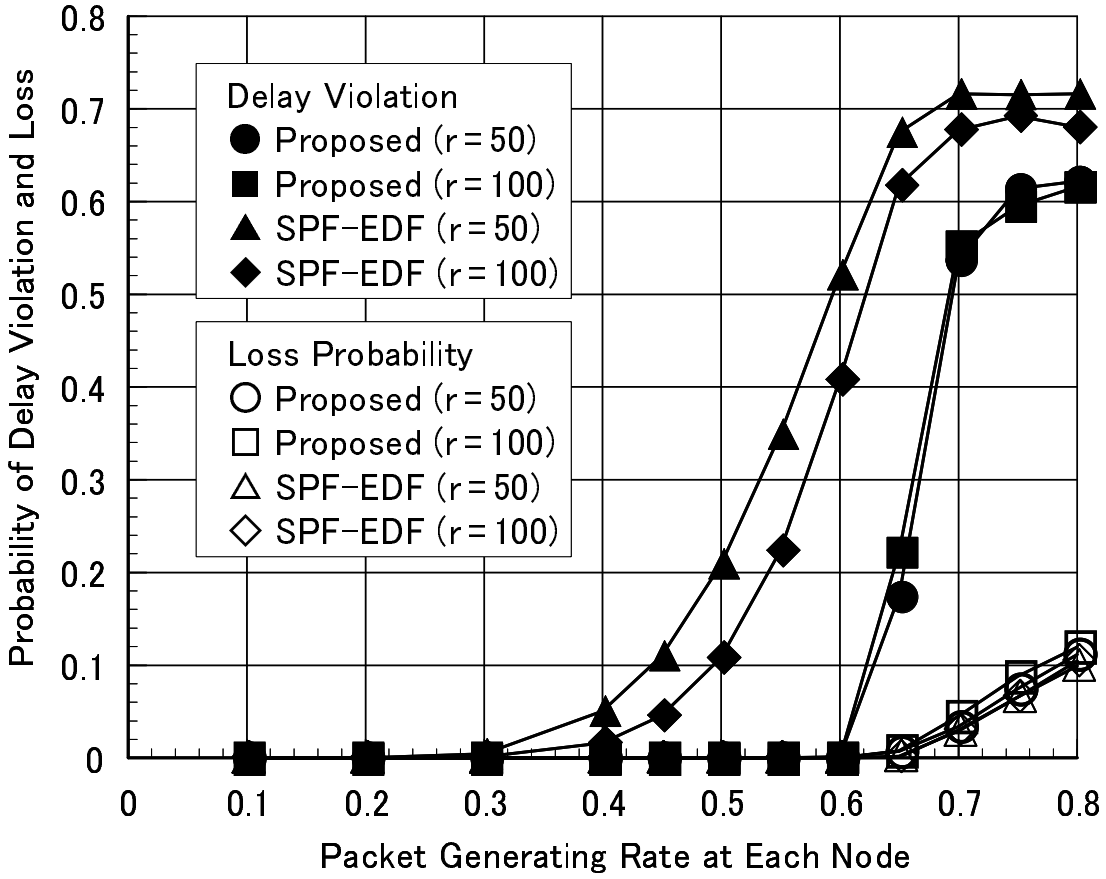


図 4.9　デッドライン違反率とパケットロス率の特性（提案方式，SPF-EDF 方式，t_{d} は指数分布乱数）

次に，ロス率の特性を見ると，両方式とも $\lambda = 0.6$ を超える領域でパケットロスが発生し，提案方式の方が従来の SPF-EDF 方式よりもわずかにロス率が高い．これは，迂回経路の選択によりネットワーク帯域を余分に消費するためである．しかし，パケットロスが発生する領域ではデッドライン違反率が高く，大きな問題にはならないといえる．

最後に，一般的なデッドラインの要求値として，生起時刻からデッドラインまでの時間 t_{d} が指数分布乱数の場合について考える．ただし，デッドラインを守れない確率の高い (最小経路遅延) + 50 よりも小さい t_{d} の要求は拒絶されるとし，t_{d} を (最小経路遅延) + 50 + x (x は平均 r の指数分布乱数）で与える．なお，50 は遅延マージンとして便宜的に定めた値である．デッドライン違反率とロス率の特性を図 4.9 に示す．図 4.8 と図 4.9 の結果を比較すると，指数分布乱数のデッドラインの場合は提案方式と SPF-EDF 方式の性能差が更に大きくなり，逆に，ロス特性の差については小さくなっている．また，SPF-EDF 方式では r の値によって違反率が変化するが，提案方式ではそれほど変化しない．これは，

ネットワークの状況とユーザ要求に応じた動的な遅延余裕の分配が実現されるためと考えられる．

トラヒックが変動する場合については，変動が緩やかでルーチングテーブルの更新がこれに十分追随できる場合は定常状態に近い性能を示し，トラヒック変動のピーク時におけるデッドライン違反率が平均の違反率を支配する．一方，変動が急な場合は，ピーク時の影響がルーチングテーブルと違反率に遅れて反映されることになり，遅延予測の誤差が生じることから性能が劣化することになる．詳細な検討については，実際の IP ネットワークの特性を考慮した遅延予測と併せて，今後の検討課題としたい．

4.4.4 ネットワーク規模の影響

ネットワーク規模が異なる場合についてSPF-EDF方式と提案方式の比較を行う．図4.4はノード数$N = 100$（$= 10 \times 10$）のネットワークであるが，ノード数Nを25，36，49，64，81，100と変化させた場合について，あて先ノードにデッドラインを過ぎてから到着する確率（デッドライン違反率）とロス率を求めた．パケット生起時刻$t_{\rm gen}$からデッドライン$t_{\rm deadline}$までの締切り時間$t_{\rm d} = t_{\rm deadline} - t_{\rm gen}$を，(最小経路遅延)+(平均$r$の指数乱数)+50で与えた．シミュレーション結果を図4.10（a）～（f）に示す．なお，ここでいうデッドライン違反率とロス率は総送信パケット数に対する違反パケット総数とロスパケット総数の割合であり，デッドライン違反率がおよそ0.7で飽和しているのはバッファあふれによるパケットロスが支配的となるためである．

図4.10からわかるように，ノード数$N = 25$ではSPF-EDF方式と提案方式の特性に大きな違いは見られないが，ノード数$N = 36$の場合では，従来方式のSPF-EDF方式はノード生起率$\lambda \geqq 0.7$でデッドライン違反が見られるのに対して，提案方式ではλが0.95になるまで違反がない．この差はノード数Nの増加によって顕著になる．なお，ノード生起率λが一定の場合，ノード数Nの増加によってネットワーク全体の負荷が増大することに注意しなければならない．

ノード数$N = 100$の場合，従来方式では$\lambda \leqq 0.3$の領域でしか違反率が0にならないのに対して，提案方式では$\lambda \leqq 0.6$の領域でデッドライン違反がない．また，$N = 49$，64，81についても同様のことがいえる．以上のことから，評価ネットワークのようなトポロジにおいては，ネットワーク規模にa余り依存せず，提案方式が優れているといえる．なお，前述したように，ここでの議論においてはソフトな遅延保証を考慮しており，厳密にはデッドライン違反率が0であるとは言い切れないことに注意する．

続いて，$r = 50$と$r = 100$の特性を比較すると，提案方式を適用した方がデッドライン違反率の差が小さいことがわかる．これは，制御が飽和している状態にあり，遅延特性に対する要求締切り時間$t_{\rm d}$の感度（Sensitivity）が低いことを示している．このことは，提案方式が裾確率（Tail Probability）の公平性に対して有効に機能していることを示唆している．本検討では，平均でのデッドライン違反率を評価したが，更に遅延分布特性を導出すればこのことが明らかになるであろう．

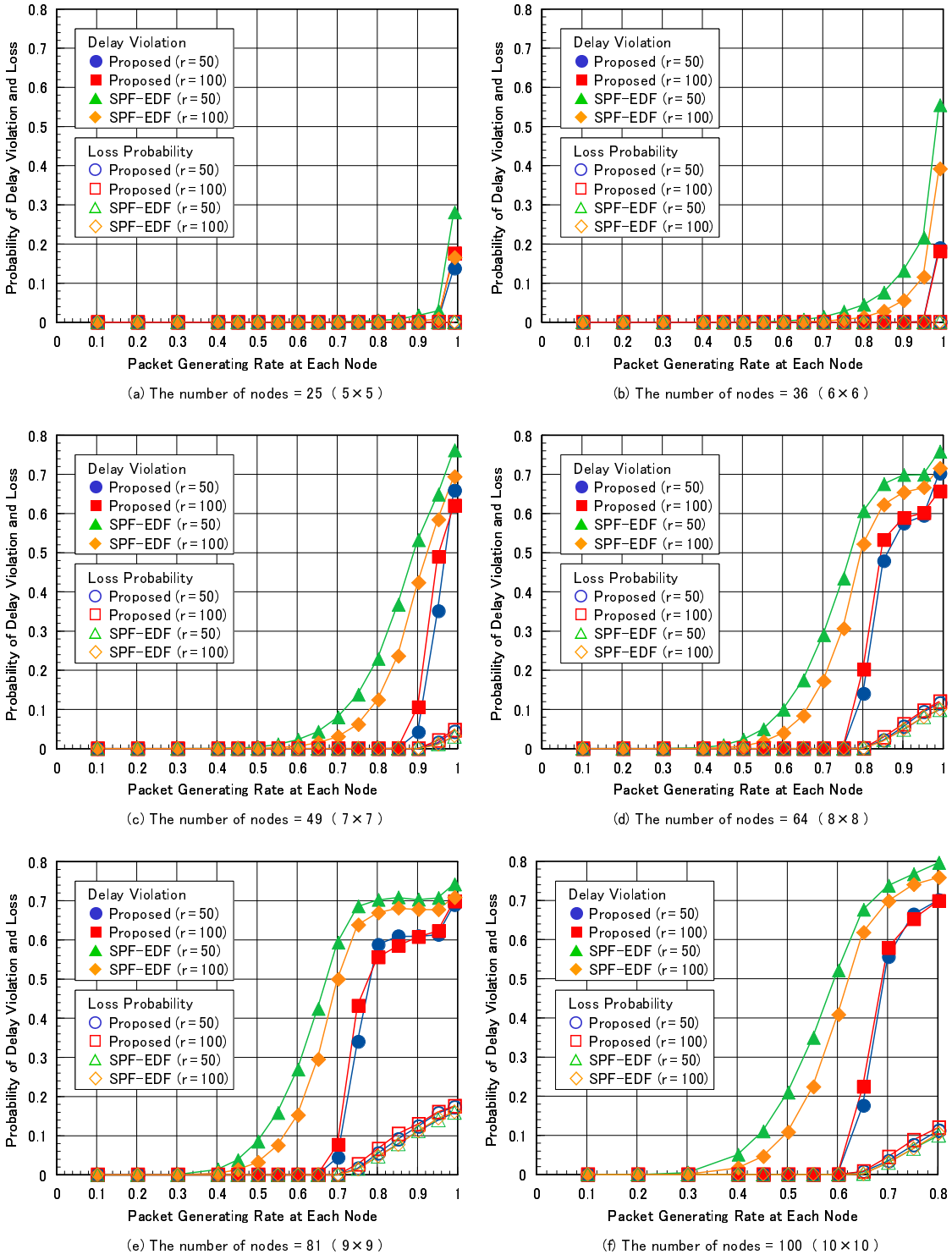


図 4.10　ネットワーク規模とデッドライン違反率の関係
(提案方式，t_{d} =(平均 r の指数乱数)+(最小経路遅延)+50)

表 4.3　提案ノードの配置パターン

Allocation type	Pattern name	#proposed nodes	Node positions of proposed nodes
All	P0	100	All nodes
Central allocation	P1	4	44, 45, 54, 55
	P2	14	33, 34, 35, 43, 44, 45, 46, 53, 54, 55, 56, 64, 65, 66
	P3	30	23, 24, 25, 26, 32, 33, 34, 35, 36, 42, 43, 44, 45, 46, 47, 52, 53, 54, 55, 56, 57, 63, 64, 65, 66, 67, 73, 74, 75, 76
	P4	52	12, 13, 14, 15, 16, 22, 23, 24, 25, 26, 27, 31, 32, 33, 34, 35, 36, 37, 41, 42, 43, 44, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 62, 63, 64, 65, 66, 67, 68, 72, 73, 74, 75, 76, 77, 83, 84, 85, 86, 87
Uniform allocation	P5	25	1, 3, 5, 7, 9, 20, 22, 24, 26, 28, 41, 43, 45, 47, 49, 60, 62, 64, 66, 68, 81, 83, 85, 87, 89
	P6	40	2, 5, 8, 10, 13, 16, 19, 22, 25, 28, 30, 33, 36, 39, 42, 45, 48, 50, 53, 56, 59, 62, 65, 68, 70, 73, 76, 79, 82, 85, 88, 90, 93, 96, 99
	P7	60	0, 1, 3, 4, 6, 7, 9, 11, 12, 14, 15, 17, 18, 20, 21, 23, 24, 26, 27, 29, 31, 32, 34, 35, 37, 38, 40, 41, 43, 44, 46, 47, 49, 51, 52, 54, 55, 57, 58, 60, 61, 63, 64, 66, 67, 69, 71, 72, 74, 75, 77, 78, 80, 81, 83, 84, 86, 87, 89, 91, 92, 94, 95, 97, 98
	P8	75	0, 2, 4, 6, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 25, 27, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 42, 44, 46, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61, 63, 65, 67, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 82, 84, 86, 88, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99

Note: #nodes in the network is 100.

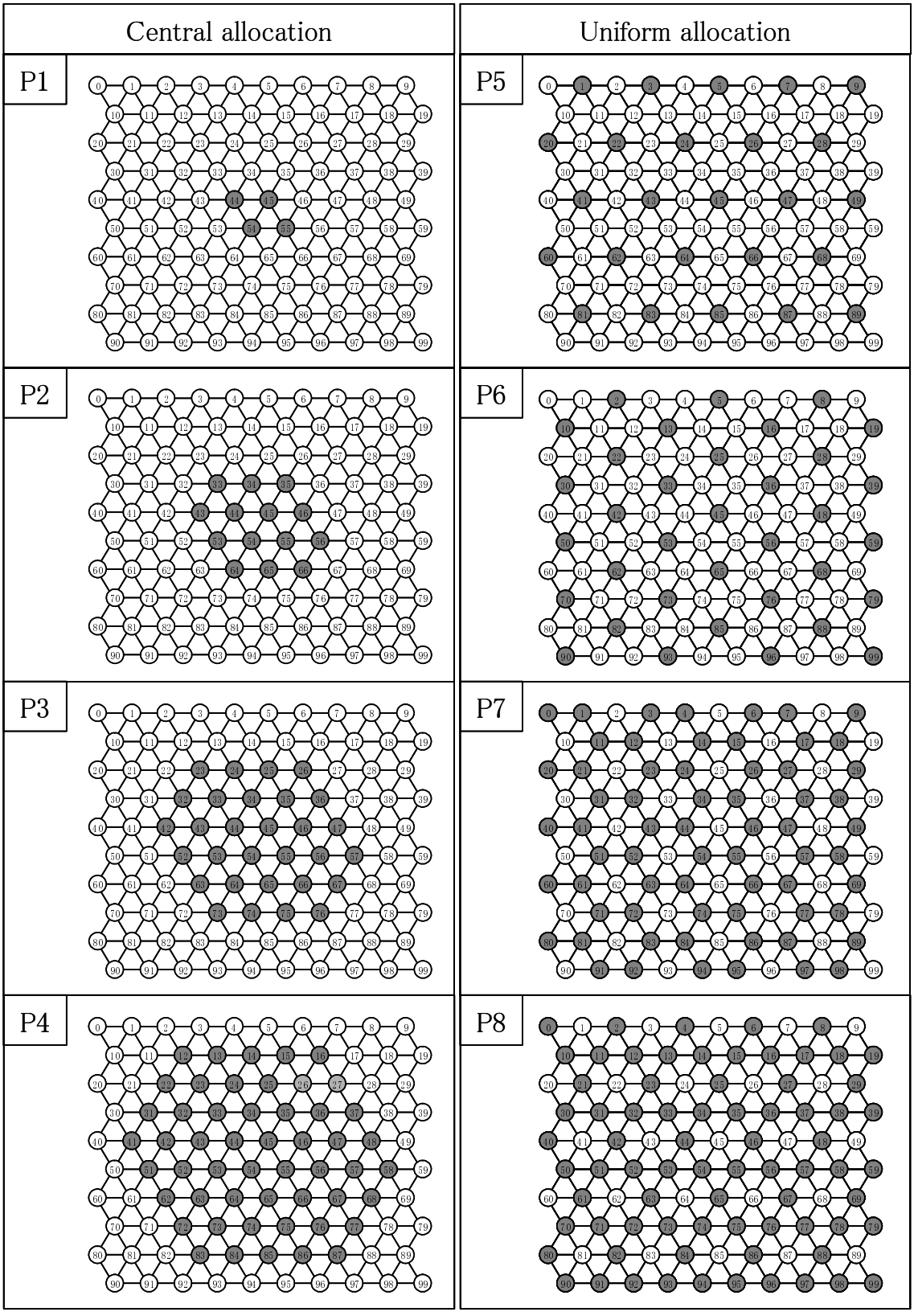


図 4.11　提案ノードの配置パターン

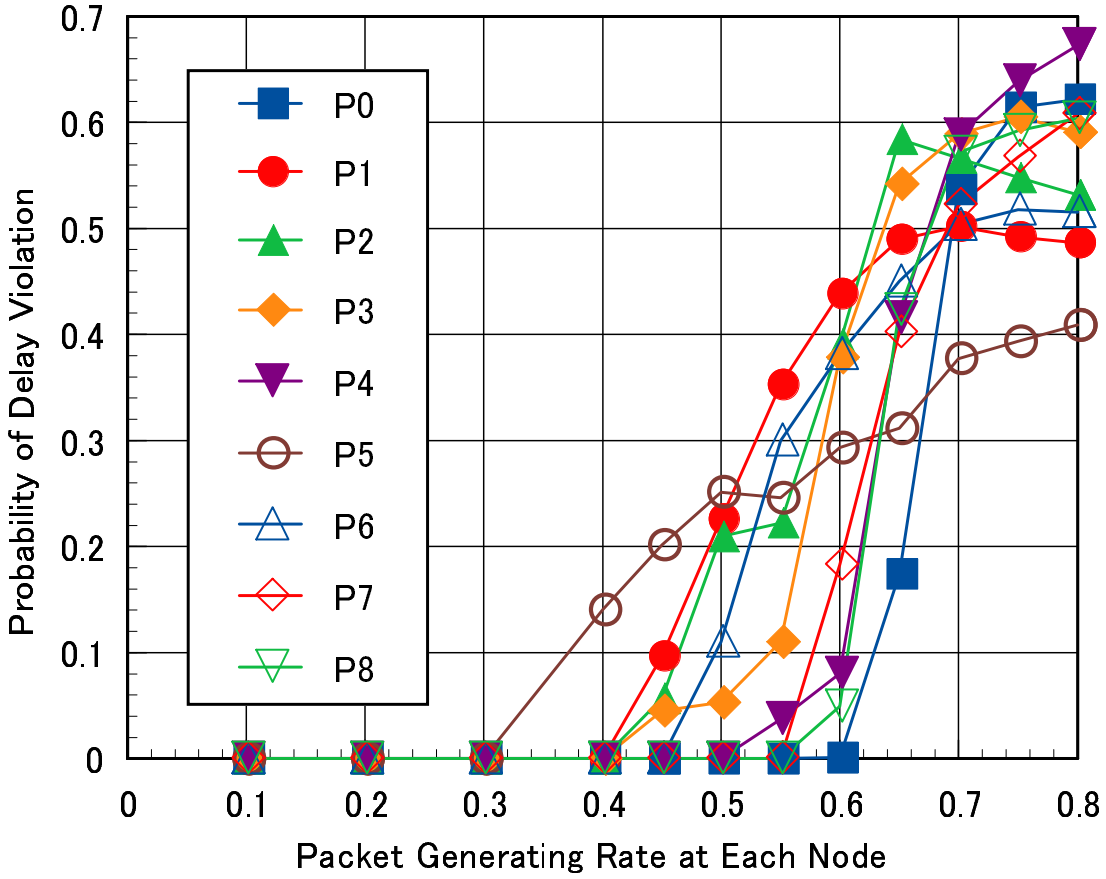


図 4.12　提案ノードの配置パターンとデッドライン違反率の関係（提案方式，t_{d} =(平均 50 の指数乱数)+(最小経路遅延)+50）

4.4.5　提案ノードが段階的に導入される場合の特性

ここまでの議論においては，すべてのノードが提案方式の機能を有するノード（以下，提案ノードと称する）である場合について検討を行ってきた．本節では，提案ノードがネットワーク内に段階的に導入される場合について考える．このとき，ネットワーク内に，提案ノードと従来ノードが混在することになる．従来ノードでは，IP ヘッダ内にデッドラインがラベル付けされていても，それを検出せずに，すべてのパケットを SPF-FCFS 方式で中継する．

評価ネットワークのノード数 N は，図 4.4 と同じく $N = 100$ とした．表 4.3 に示した P0〜P8 の 9 種類の提案ノードの配置パターンについて性能を評価する．ここで，P0 はすべてのノードが提案ノードの場合である．P1〜P4 はノード負荷が高い中央部のノードを提案ノードとする配置であり，それぞれの提案ノード数は 4，14，30，52 である．一方，P5〜P8 は均等に分布するような配置であり，提案ノード数はそれぞれ 25，40，60，75 で

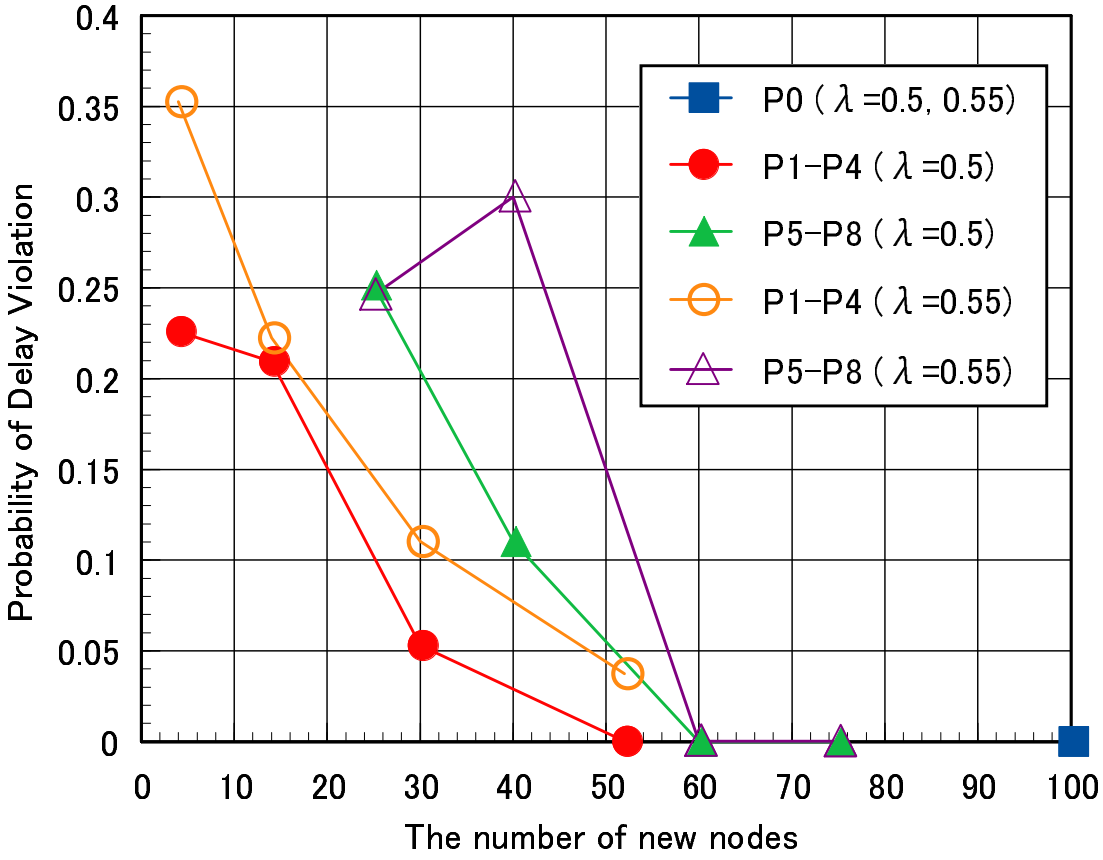


図 4.13　提案ノードの配置パターンとデッドライン違反率の関係（提案方式，t_{d} =(平均 50 の指数乱数)+(最小経路遅延)+50，$\lambda = 0.5$，及び，$\lambda = 0.55$）

ある．具体的な配置パターンを図 4.11 に図示する．

これらの提案ノードの配置パターンについて，デッドライン違反率の特性を求めた．t_{d} を 4.4.4 節における図 4.10 の検討の場合と同様に指数乱数で与えた．シミュレーション結果を図 4.12 に示す．すべてのノードが提案ノードの場合（P0）では，$\lambda \leqq 0.6$ の範囲でデッドラインを守れるが，提案ノード数の減少とともに λ の範囲が狭くなり，特性が劣化している．提案ノードを中央部に配した場合（P1〜P4）と均一に配した場合（P5〜P8）を比較すると，後者の方が劣化が大きい．この結果から，提案ノードは負荷の大きい位置に対して隣接するように配した方が効果的であることがわかる．

ノード生起率 $\lambda = 0.5$ 及び $\lambda = 0.55$ の場合について，提案ノード数とデッドライン違反率の関係について図 4.13 に整理し直して示す．このグラフからも，中央部の負荷が大きいノードを提案ノードとすることが効果的であることがわかる．評価に用いたシミュレーション条件では，違反率を 5%以下とするには，中央部のおよそ半数のノードが提案ノー

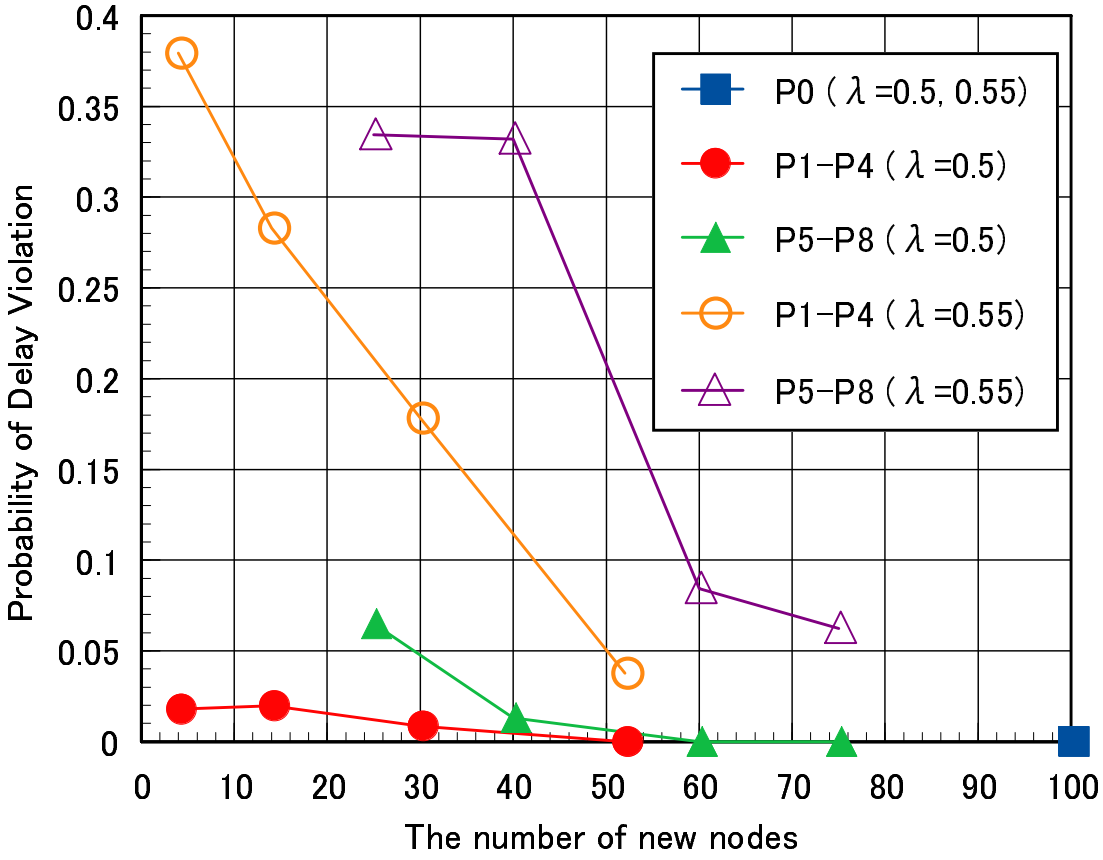


図 4.14　提案ノードの配置パターンとデッドライン違反率の関係（提案方式，$t_{d1} = 100$ （50%），$t_{d2} = 500$ （50%），$\lambda = 0.5$ 及び $\lambda = 0.55$）

ドであればよいことになる．

次に，50%の生起パケットに対して $t_{d1} = 100$，残りの 50%に対して $t_{d2} = 500$ のデッドラインを与えた場合について同様に特性を求めた．図 4.14 に結果を示す．P5〜P8 の場合は図 4.13 の結果と大きな変化はないが，P1〜P4 については著しい特性の改善が見られ，提案ノードが全ノード数の 2 割以下であっても十分であるといえる．これは，t_d の差が大きい場合については改善効率が高まり，先の結果と同様に，負荷の高いノードを提案ノードとすればよいことを示している．

以上のように，締切り時間 t_d の分布は特性に大きく影響を与えると考えられ，今後，更に検討を進めたい．

4.5 むすび

本章では，IP ネットワークにおけるデッドラインに基づく経路制御・優先制御法を提案した．本方式は，コネクション設定を必要とせず，パケットごとに記述されたデッドラインと遅延予測によって中継ノードごとに柔軟に経路を選択し，動的に優先度を設定する．計算機シミュレーションの結果から，従来方式と比較して，高負荷までデッドラインを守れることを確認し，本方式の有効性を示した．

第5章
パケットごとの締切り時刻に基づくネットワークを用いた高信頼性通信

5.1 まえがき

年々増加傾向にあるコンテンツのダウンロード型通信においては，そのほとんどが高信頼性通信である．すなわち，コンテンツデータごとのソフトな遅延保証を実現するためには，高信頼性通信を考慮した制御が必要となる．

高信頼性通信においては，ARQ 方式（自動再送要求方式）[76], [77] が広く用いられており，データの送信と同時に折返し回線でデータ送達の確認を行い，正しく受信されなかった場合は再送が行われる．このとき，回線速度，送受信間での往復遅延（RTT: Round Trip Time），再送回数などがデータ配送時間に影響を与え，システムの性能を左右することになる．RTT は，伝送路の伝搬遅延と，経路上の各ノードにおけるノード遅延（パケットの処理時間と待ち時間の合計）からなる．伝搬遅延は空間的な要因で生じる遅延であり経路制御により制御可能であるが，伝送路の距離が長い場合は制御が難しくなる．一方，ノード遅延は優先制御によって比較的柔軟に制御することができる．

さて，近年のネットワークの高速化に伴って，RTT がデータ配送時間に対して占める割合が増し，スループットの低下を招く主たる要因となりつつある [10]．例として，2.5MB のデータを 100Mbits/s の回線を用いて往復で 20,000km を伝送する場合を考えると，往復の伝搬遅延はおよそ 0.1 秒となる．伝送路上のノード遅延の合計の平均が 0.1 秒と見積もられるとき，RTT は約 0.2 秒となる．この場合の高信頼性通信におけるデータ配送時間と RTT の関係について，図 5.1 に示す．横軸はネットワークの回線速度 LS bits/s，縦軸はデータサイズ DS bytes である．RTT がデータ配送時間に対して占める割合 R は，$R = \frac{RTT}{\frac{8DS}{LS}+RTT}$ で与えられるが，図中の網掛けの領域では $R > 50$ であり，回線速度の増加に伴って，RTT の影響を無視できない DS の範囲が広くなる．

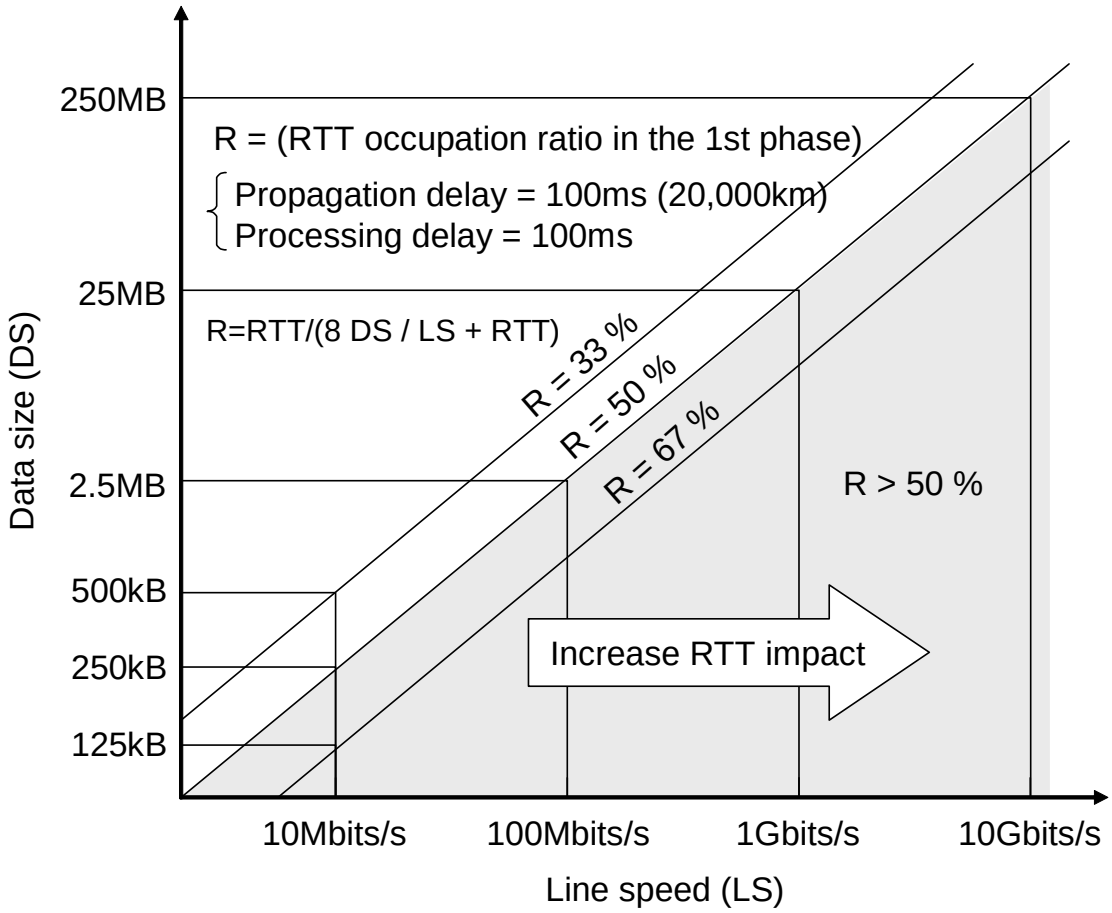


図 5.1　高信頼性通信におけるデータ配送時間と RTT の関係

ARQ 方式に関してはこれまで多くの研究が行われている．送受信バッファが無限大といった理想的な条件下では，Selective Repeat ARQ 方式（選択再送方式）が最大のスループット特性を呈し，最適な方式となるが，送受信バッファサイズが有限の場合はスループットが劣化する．この問題を解決するために再送多重が提案され，更に，送受信バッファサイズが有限である場合にスループットを最大化する多重再送方式などの検討が行われている．このように，スループットの観点からは様々な検討が進められているが，締切り時刻を考慮したデータ配送のためには，スループットではなく時間の観点からの検討が必要となる．これまでの研究では主として RTT が一定のもとで検討が進められてきたが，本章においては，第 4 章で提案したパケットごとの締切り時刻に基づくネットワークを用いて RTT を制御できる場合について検討する．

以上のことから，本章においては，高信頼性通信をパケットごとの締切り時刻に基づくネットワーク上で行う場合において，再送回数に応じて適切な締切り時刻と多重送信数を与えることで，送信完了までの所要時間，すなわち，コンテンツのデータ配送時間を制御

する方式を提案し，その特性を評価する．

5.2 データ配送時間制御方式

図 5.2 に選択再送方式を用いた高信頼性通信について示す．図 5.2（a）は従来の選択再送方式の送信手順を示したものである．この例ではパケット 1 において 2 回のエラーと再送要求が発生している．再送回数の増加に従って大きな送信バッファが必要されるようになる．データ配送時間はデータサイズに加えてエラーによる再送回数と RTT の両方にも依存することがわかる．

提案方式による送信手順を，図 5.2（b）に示す．従来と同様に，選択再送方式を用いて高信頼性通信を実現する．また，第 4 章の図 4.1 のノード構成と同様に，経路上の各ノードに EDF 待ち行列を具備させてエンド・エンド間での伝送遅延時間を制御し，再送回数の増加に応じて早い締切り時刻を割当てるようにする．

また，送信多重方式を用いることで再送回数を削減し，RTT の影響を低減させる．送信多重とは，パケットを複製して送信し，エラーに対する保護を行うものであり，送信パケット数が増加するが再送回数を減少させることができる．したがって，パケット数の増加による最大スループットの低下と再送回数の減少によるデータ配送時間の短縮化の間にトレードオフが存在する．再送回数に応じて適切な多重送信数を設定すれば，スループットの低下を抑制しながらデータ配送時間を短縮化することができると考えられる．

本章では，これらの制御の組合せによって，データ配送時間の短縮化を図る．また，複次的に，所要の送信バッファサイズが小さくなる効果も得られるようになる．

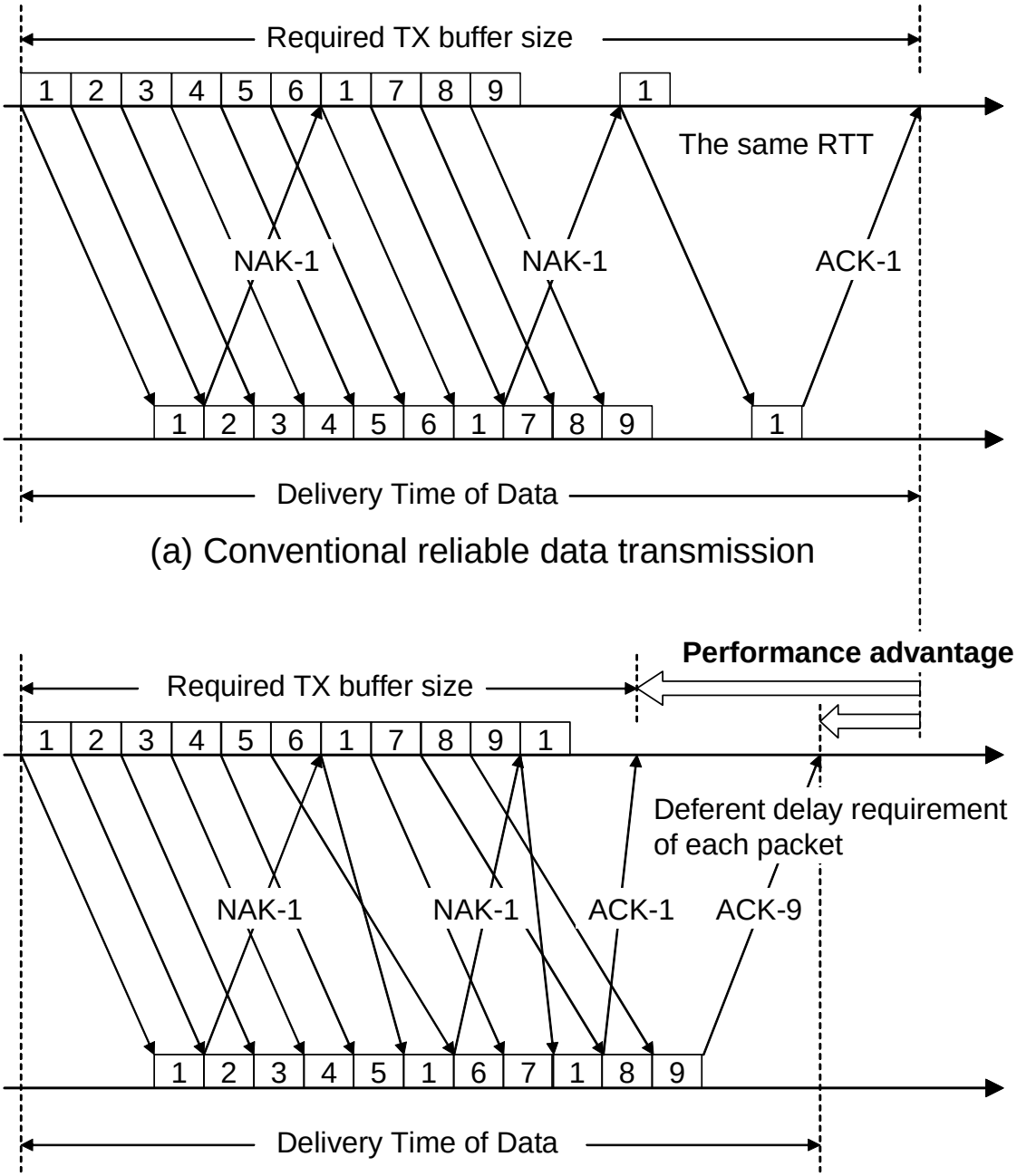


図 5.2　提案方式によるデータ配送時間の改善効果

5.3 性能評価

5.3.1 EDF 待ち行列の特性

データ配送時間を評価するに際して，まず，単一の EDF 待ち行列の特性について議論する．EDF 待ち行列は 1961 年に提案されて以来，長期に渡って持続的に研究が行われているが [33]，厳密な性能解析結果は未だに得られていない．HOL-PJ 方式の平均の待ち時間の導出法については文献 [78] で報告されており，これが EDF 待ち行列の近似解析に相当するが，一般条件における待ち時間分布の解析については，筆者の知る限りにおいてはない．ただし，負荷が高い場合の近似解は経験的に知られており，待ち時間を w，締切り時間を $t_{\rm d}$ とすると，その累積確率密度関数は次式で与えられる [38].

$$\mathrm{Prob}[w \leqq t] = 1 - a\, e^{-b(t-(t_{\rm d}-\overline{t_{\rm d}}))} \tag{5.1}$$

ここで，a と b は裾確率の形状を定めるパラメータである．

本章における検討では負荷が低い場合も考慮されるべきであるので，この結果で待ち時間分布を近似することは好ましくない．また，理論導出も困難であることが予想されることから，計算機シミュレーションによって EDF 待ち行列の特性を導出することにする．

到着パケットに割当てられるノード内のローカルデッドラインが指数分布に従うと仮定する．各ノードで到着パケットに設定されるローカルデッドライン $t_{\rm d}$ の確率密度関数を次のように与える．

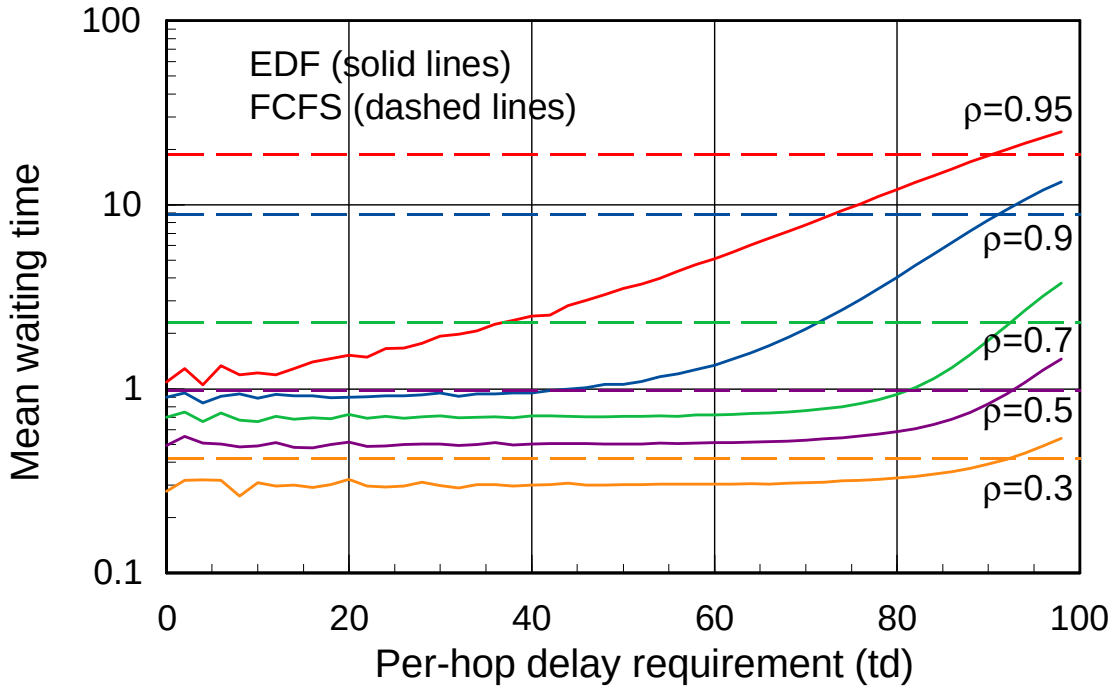
$$f_{t_{\rm d}}(t) = \alpha e^{-\alpha(\beta-t)} \tag{5.2}$$

ここで，α と β はローカルデッドラインの分布形状を定めるパラメータである．

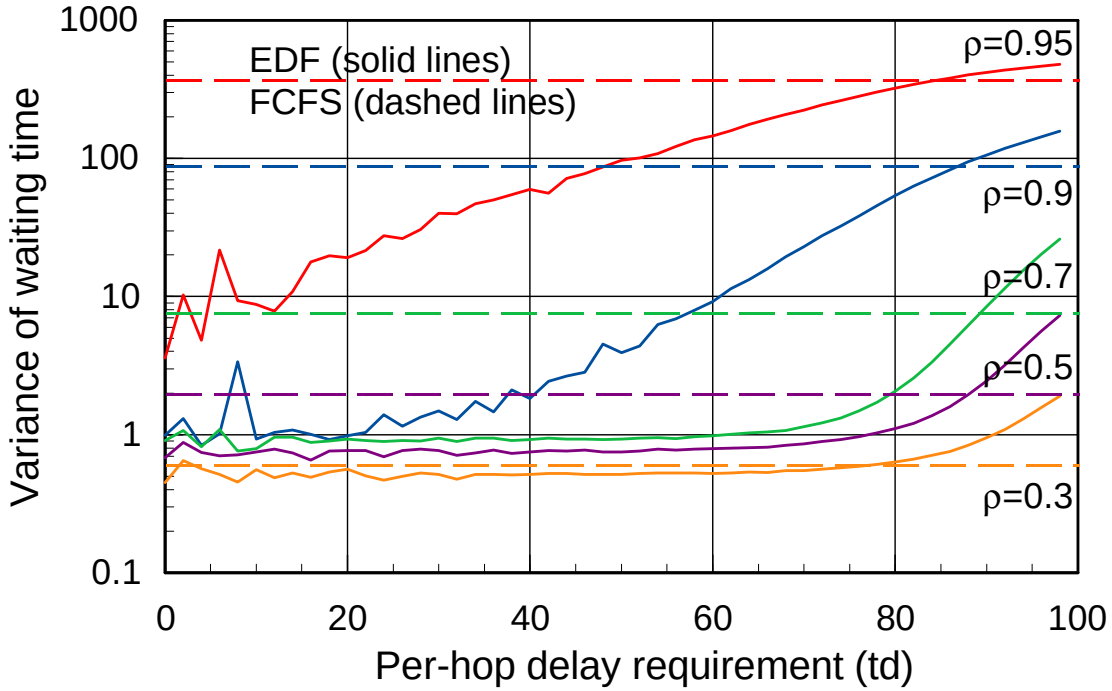
M/M/1 モデルを仮定し，リンク利用率を ρ ($= \lambda/\mu$) とする．ここで，λ と μ は入力リンクにおけるパケット到着率と出力リンクにおけるパケットサービス率である．$\alpha = 1/4.34$, $\beta = 100$，$\mu = 1$ の場合について計算機シミュレーションを行い，平均待ち時間 $\overline{w(t_{\rm d})}$ と待ち時間の分散 $\sigma_w^2(t_{\rm d})$ を求めた．結果を図 5.3（a）及び図 5.3（b）にそれぞれ示す．また，比較のため，FCFS 待ち行列の特性を重ねて示す．FCFS 待ち行列においては，平均待ち時間 $\overline{w} = \frac{\rho/\mu}{1-\rho}$，待ち時間の分散 $\sigma_w^2 = \frac{\rho(\mu-(1-\mu)\rho)/\mu^2}{(1-\rho)^2}$ であり，図中において破線で示した．平均のローカルデッドラインは $\overline{t_{\rm d}} = \beta - 1/\alpha = 95.7$ である．図 5.3 の結果から，およそ $t_{\rm d} < \overline{t_{\rm d}}$ の領域において FCFS 待ち行列よりも優れた特性が得られ，平均の待ち時間が大きく減少していることがわかる．例えば，$\rho = 0.95$ においては，$t_{\rm d} = 40$ のローカルデッドラインが与えられたパケットと $t_{\rm d} = 100$ のローカルデッドラインが与えられたパ

ケットの平均の待ち時間を比較すると，$\overline{w(40)}/\overline{w(100)}$ はおよそ $1/10$ である．

なお，この計算機シミュレーションで用いたパラメータは，初回の送信時を $t_{\mathrm{d}} = 100$ のローカルデッドラインで行われることを想定しており，平均待ち時間の値は，FCFS 待ち行列と比較して若干劣化することなるが，上記のように再送時の改善の度合の方が大きいことが期待される．



(a) Mean waiting time in an EDF queue



(b) Variance of waiting time in an EDF queue

図 5.3　EDF 待ち行列の遅延特性

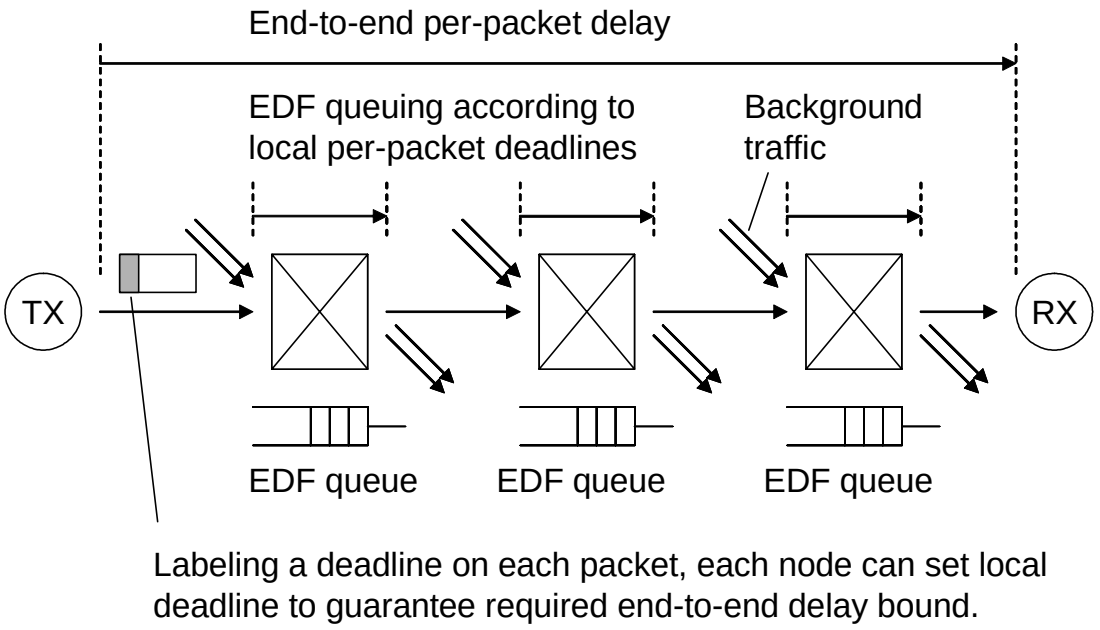


図 5.4　パケットごとの締切り時刻に基づくパケット中継

5.3.2　縦列接続された EDF 待ち行列の特性

RTT を評価するため，図 5.4 に示すような h 個のノードが縦列接続されたネットワークモデル（タンデム型ネットワークモデル）について検討する．各ノードには EDF 待ち行列が実装され，背景トラヒックが重畳されている．簡単のため，各ノードにおける処理として EDF 待ち行列による優先制御だけを考慮する．また，各ノード内の EDF 待ち行列におけるローカルデッドライン t_{d} はすべて等しいとする．

計算機シミュレーションによって，ホップ数 $h = 10$ のネットワークにおける $\rho = 0.95$ の場合の遅延特性を求めた．結果を図 5.5 に示す．t_{d} が大きい場合は，図 5.5 の片対数グラフにおいて遅延分布特性が上に凸の二次関数に近い形状となっているが，これは中央極限定理によってホップ数 h の増加に従って経路遅延が正規分布に漸近するためである．$t_{\mathrm{d}} = 60$，$t_{\mathrm{d}} = 80$，$t_{\mathrm{d}} = 100$ の特性について，ピークの右半分の特性を比較すると，その間隔はおよそ 200 となっている．これは，式 (5.1) を参照すれば，$\Delta t_{\mathrm{d}} \cdot h$ の関係となっていることから容易にわかる．

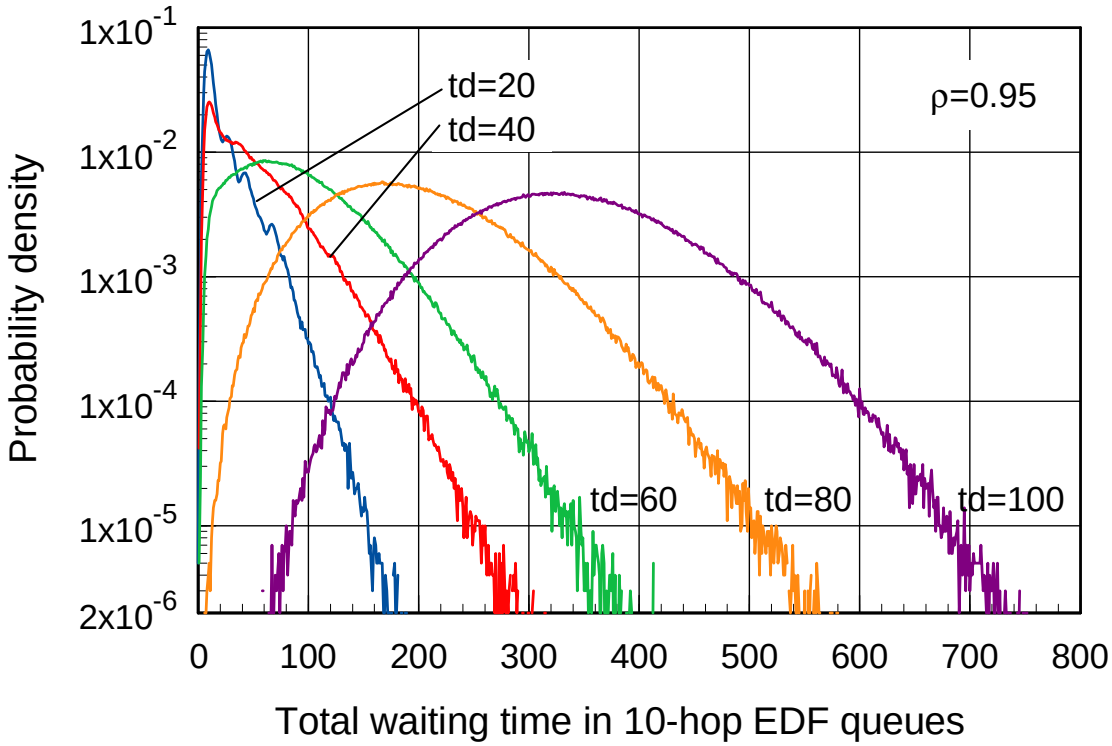


図 5.5　多段に接続された EDF 待ち行列の遅延分布特性 （$h = 10$）

5.3.3 データ配送時間

高信頼性通信におけるデータ配送時間を定義するためには，まず，配送終了時間を定義する必要がある．エラーが確率的に生じる条件下では，再送が有限回数で必ず終了することを保証できず，無限時間までの再送を考慮する必要がある．したがって，ある一定時間が経過すると再送によってエラーパケットの個数が 0 になることを保証することはできないが，平均である個数以下になるように収めることはできる．

そこで，データ配送の終了条件を，コンテンツ当りのエラーパケットの個数がある値以下になることで定義することにする．データ配送の終了条件を次式で与える．

$$p^K M < 10^{-5} \tag{5.3}$$

ここで，p はパケットエラー率，M はデータサイズ（データを構成するパケット数），K は所要送信回数である．

同じパケットエラー率の条件のもとで，データ配送時間が最大となる最悪のエラーパターンは，データを構成するパケット中で一番最後のパケットが毎回エラーになる場合に相当する．この場合のデータ配送時間 T は次式のように計算することができる．

$$\begin{aligned} T &= N_1 M + 2h\, w(t_{\mathrm{d}1}) + 2(h+1)\, l \\ &\quad + N_2 M p^{N_1} + 2h\, w(t_{\mathrm{d}2}) + 2(h+1)\, l \\ &\quad \cdots \\ &= \sum_{i=1}^{F} N_i M p^{\sum_{j=0}^{i-1} N_j} + \sum_{i=1}^{F} 2h\, w(t_{\mathrm{d}i}) + 2F(h+1)\, l \end{aligned} \tag{5.4}$$

ここで，F は全送信フェーズ数，N_i は i 回目のフェーズでの多重送信数，$w(t_{\mathrm{d}})$ は往復でのホップ数 $2h$ の縦列接続された EDF 待ち行列の待ち時間の確率変数，l はノード間の伝搬遅延である．フェーズ数は全送信回数と同じ意味であるが，送信多重を行っていることを強調する意味で，以下において送信フェーズ数と称する．明らかに，$F \leqq K$，$N_0 = 0$ のとき，所要送信回数 K は $K = \sum_{j=0}^{F} N_j$ を満たす．

データ配送時間をある締切り時間以下に抑えるために，全送信フェーズ数 F の決定と，所要送信回数 K をどのようにして N_i に分配するのかが重要な問題となる．パケットエラー率 p が比較的小さい場合の最適な分配法が次式で与えられることは，直感的に容易に推測できる [45].

$$N_i = \begin{cases} 1 & (1 \leqq i \leqq F-1) \\ K - F + 1 & (i = F) \end{cases} \tag{5.5}$$

5.3.4 デッドライン違反率

ここでは，要求されたデータ配送時間以下に実際の配送時間を抑えることについて考える．式（5.4）を確率変数の変換式と捉えると，確率変数 T は，確率変数 $w(t_{\mathrm{d}})$ を変換して得られることになる．以下において，T が要求されたデータ配送時間 $T_{\mathrm{delivery-deadline}}$，すなわちコンテンツの締切り時間を超える確率を導出する．この確率をコンテンツのデッドライン違反率と呼ぶことにする．デッドライン違反率 P_{vio} は次式で与えられる．

$$P_{\mathrm{vio}} = \mathrm{Prob}[T > T_{\mathrm{delivery-deadline}}] \tag{5.6}$$

中央極限定理により RTT が正規分布に従うとすると，以下の近似式が利用できる．

$$P_{\mathrm{vio}} \approx \frac{1}{2}\mathrm{erfc}\left(\frac{T_{\mathrm{m}}}{\sqrt{2\sum_{i=1}^{F} 2h\,\sigma_w^2(t_{\mathrm{d}\,i})}}\right) \tag{5.7}$$

ここで T_{m} は次式で定義される遅延余裕時間である．

$$T_{\mathrm{m}} = T - \overline{T} \tag{5.8}$$

5.3.5 シミュレーション

基本特性を検討するため，$M = 100$，$p = 10^{-2}$，$K = 4$，$h = 10$，$l = 2$ の評価条件について，i 回目の送信フェーズにおけるローカルデッドライン $t_{\mathrm{d}\,i}$ を式（5.9）で与えた場合のデータ配送時間を求めた結果を図 5.6 に示す．

$$t_{\mathrm{d}\,i} = 100 - 20(i-1) \tag{5.9}$$

太線で示された特性は平均のデータ配送時間である．コンテンツのデータサイズが小さいとき，この時点においては，およそ半分のパケットが遅延要求を満たす．一方，細線で示された特性は $P_{\mathrm{vio}} = 10^{-5}$ を満足する境界である．図から，FCFS 待ち行列を用いた従来方式と比較して，提案方式は $F = 1$ の場合を除いてデータ配送時間が短縮化され，良い性能が得られていることがわかる．また，評価に用いた条件においては，$F = 2$ が最適な分配となっている．実際の利用においては，要求されたデータ配送時間に応じて F，N_i のパラメータを選択することになる．データ配送時間が短縮化されることでパラメータの組合せ数が増加し，ユーザやネットワーク管理者が柔軟にネットワークを利用できるようになる．

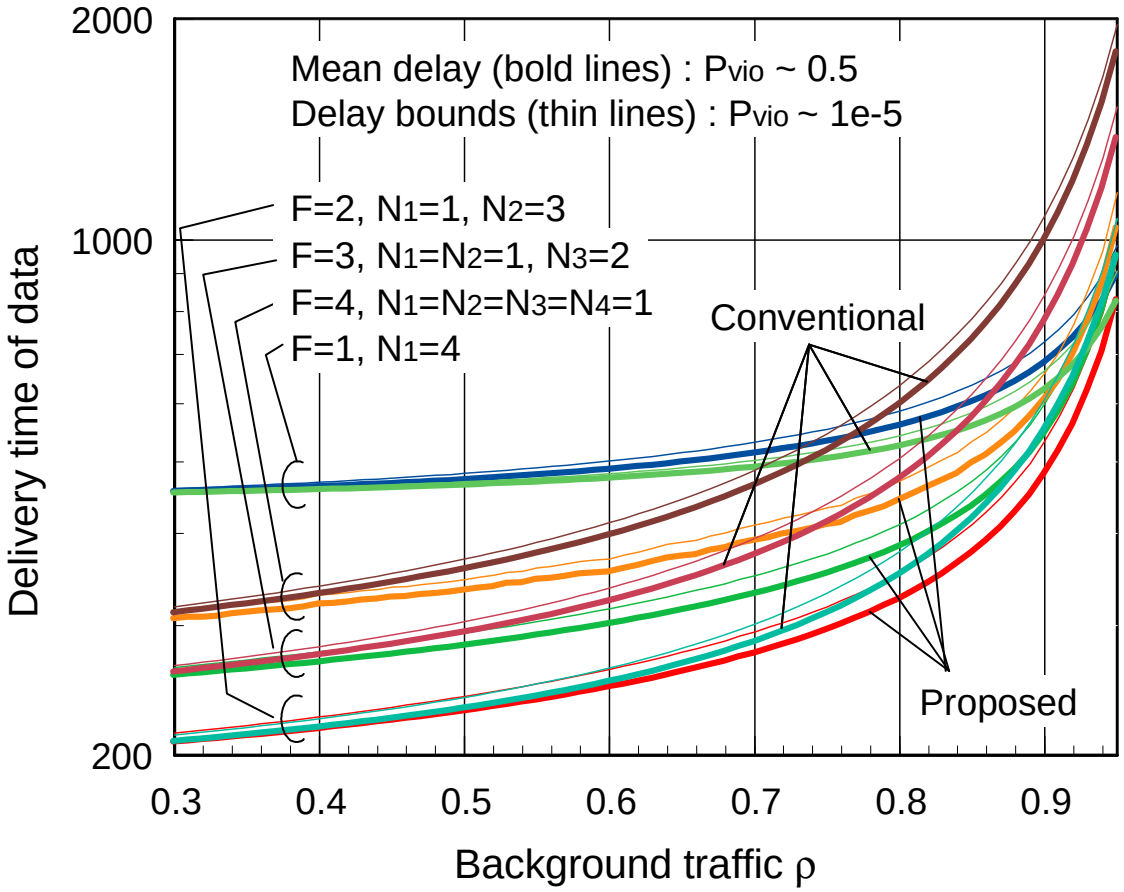
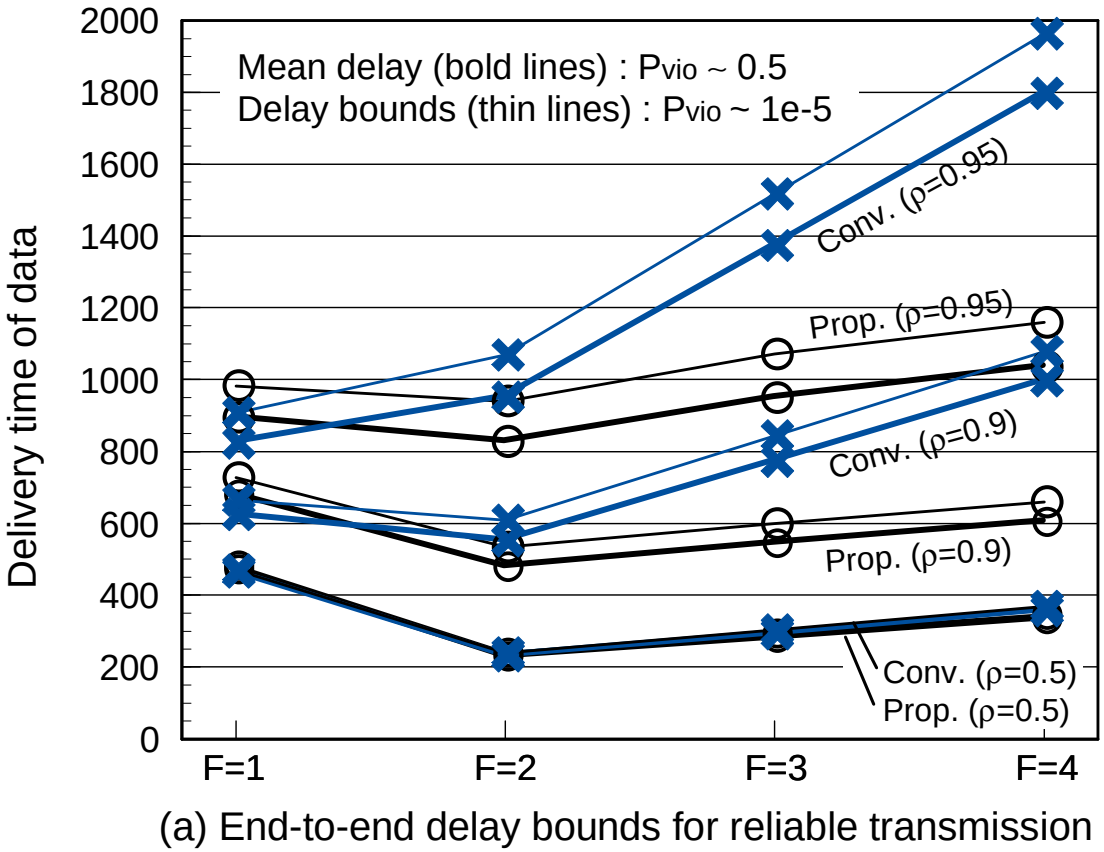


図 5.6　高信頼性通信における遅延特性

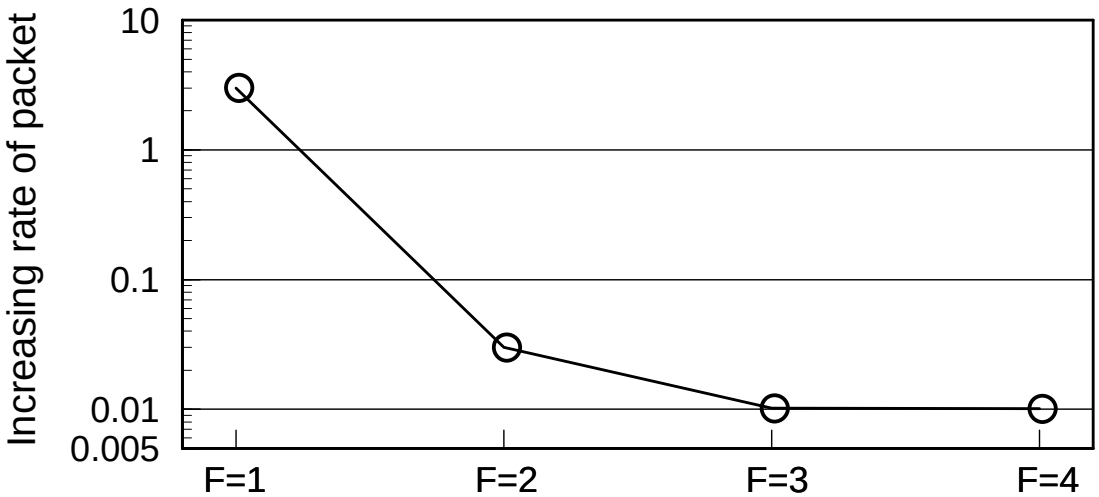
続いて，全送信フェーズ数 F に関する遅延特性の比較を行うために，図 5.6 の結果について横軸を F として再描画した特性を図 5.7（a）に示す．背景トラヒックを含めたネットワーク負荷が $\rho = 0.5$ であるときは，FCFS 待ち行列を用いる従来方式と提案方式の特性はほとんど同一となる．しかし，$\rho = 0.9$，$\rho = 0.95$ と高負荷になるに従って，従来方式の場合のデータ配送時間は RTT の増加によって急激に増加するが，提案方式では配送時間の増加が小さく抑えられていることがわかる．

最後に，データ配送時間の短縮化と最大スループットの低下のトレードオフについて考える．図 5.7（b）は，全送信フェーズ数 F の違いによる全送信パケット数（スループットの逆数）の増加率 R を示している．増加率 R は，次式で与えられる．

$$R = \sum_{i=1}^{F} N_i \, p^{\sum_{j=0}^{i-1} N_j} \tag{5.10}$$



(a) End-to-end delay bounds for reliable transmission



(b) Increasing rate of packet

図 5.7　提案方式と FCFS 待ち行列を用いた場合との比較

図 5.6 の結果では，$F = 2$ のときデータ配送時間が最小となったが，F の値が大きいほどパケット増加率が小さくなり高スループットとなる．つまり，トレードオフの関係が存在することを確認できる．しかし，$F = 2$ の場合のパケット増加率 R は 0.033 とわずかであり，評価条件においてはこれ以上の R の低減化は不要であると考えられる．

以上の検討結果から，提案方式はネットワークの負荷への影響が小さく，再送パケットに高い優先度を与えたとしてもインパクトが小さいといえる．

本章における検討では，データ配送時間の短縮化の観点で各送信パラメータについて遅延特性の検討を行った．しかし，実際の利用においては逆の視点による考察が求められる．具体的には，コンテンツごとに所要のデッドラインを与え，その値から最適な送信パラメータを導出し，データを配送する必要がある．これについては，今後，さらに検討を進めたいと考えている．

5.4 むすび

本章では，高信頼性通信をパケットごとの締切り時刻に基づくネットワーク上で行う場合において，再送回数に応じて締切り時刻と多重送信数を適切に与えることで，コンテンツのデータ配送時間を制御する方法を提案し，その特性を評価した．この結果，従来方式と比較してデータ配送時間を短縮化できることを確認した．これによって送信パラメータの組合せ数が増加することから柔軟性が高まり，提案方式の有効性を確認することができた．

第6章
結論

本論文は，ネットワークの高速化とコンテンツの大容量化に対応可能な「高効率化コンピュータ通信システム」の実現を目的として，新しい通信端末の構成法と，柔軟性の高いネットワーク制御法の提案と，その性能評価について筆者が行った研究成果をとりまとめたものである．本研究では，以下に示す成果を得た．

1. 大容量データの高速入出力に適した通信端末のハードウェア構成法として，ハードディスクの高速化を実現するために汎用パーソナルコンピュータの内部バス上に複数の RAID コントローラを実装し，これらを同時に制御することで二階層の並列化を実現し，高速化を図った．更に，大容量データの一括入出力動作に適したファイルシステムを提案し，ソフトウェア処理のオーバヘッドを削減した．これらコンピュータ内通信の高速化・大容量化を図ったシステムを試作し，実験的評価を行い，105Mbytes/s の連続読出し速度を実現できることなどを確認した．

2. 汎用ワークステーション上で高速入出力通信端末を実現する場合のソフトウェア構成法を提案した．断片化した処理ブロックを再構成する機能と，中間バッファメモリを介在させない新しいインタフェースを実装することで，コンピュータ内通信のボトルネックを解消し，UNIX システム上の高速 RAID ファイルサーバにおいて約 110Mbytes/s の連続読出し速度を実現できることなどを実験により確認した．

3. 従来方式とは異なるコネクション管理を必要としない柔軟性の高い QoS について検討した．具体的には，パケットごとに締切り時刻を与え，これに基づいた優先制御と経路制御を行い，コンピュータ間通信における柔軟なエンド・エンド間での遅延保証を実現した．従来方式と比較して高負荷まで適用可能であることをシミュレーションにより確認した．

4. 高信頼性通信において，再送回数に応じて適切な締切り時刻をパケットに割当て，

更に多重再送を用いることで，データ配送時間を制御する方式を提案した．この提案方式はデータ配送時間の短縮化に関して有効に機能し，遅延制御の柔軟性が高まることを確認した．最大スループットの低下がわずかに見られたが，ネットワークの負荷への影響が小さく抑えられることを確認した．

本研究に関する今後の課題を，以下に挙げる．

1. 階層数の増加や，より高速な内部バスの活用などの検討
2. 高速ファイルシステムを用いて長時間待ち合わせが可能なネットワークの検討
3. UNIX システムにおける通信バッファの利用に適した高効率なメモリ管理方式の検討
4. 実際の IP ネットワークの特性を考慮した遅延予測技術の検討
5. 他のネットワークモデルにおける性能やトラヒック変動の影響の検討
6. EDF 待ち行列の待ち時間分布の理論導出

謝辞

本論文をまとめるにあたり，一方ならぬ御指導，御教示を賜った大阪市立大学 大学院工学研究科 教授 村田 正 博士，並びに，同助教授 杉山 久佳 博士には，長期にわたって研究内容に関する御討論と論文の綿密な御校閲をいただいた．ここに深甚なる感謝の意を表する．また，常日頃より大変有益な御助言を賜っている 大阪市立大学 大学院工学研究科 教授 藤原 値賀人 博士，教授 辰巳 昭治 博士，元教授 岡本 次郎 博士，並びに，鈴鹿工業高等専門学校 電子情報工学科 講師 吉川 英樹 博士に深く感謝申し上げる．更に，筆者が電気通信大学 電気通信学部に在学中の頃から長きにわたり，公私ともに御指導をいただき，また，研究の推進に種々の面で御指導，御助言を賜った大阪市立大学 大学院工学研究科 助教授 岡 育生 博士に深謝の意を表する．

筆者の研究活動は，学生時代において御指導を賜った先生方の影響によるところが大きい．電気通信大学に在学中，及び，その後の各学会会場において，貴重な御教示をいただいた 群馬大学 教授 小野里 好邦 博士，早稲田大学 教授 嶋本 薫 博士，電気通信大学 講師 鈴木 洋一 博士，近畿大学 理工学部 助教授 笹野 博 博士に深く感謝申し上げる．また，大阪市立大学大学院 在学中における恩師であり，貴重なご教示をいただいた 大阪市立大学 名誉教授 奥本 隆昭 博士に心から感謝申し上げる．

本論文の執筆にあたっては，大阪市立大学 大学院工学研究科 電子情報系専攻 通信システム工学研究室の皆様に貴重な御指摘，御意見をいただいた．ここに謝意を表する．また，研究遂行にあたっては，大阪市立大学 大学院工学研究科の多くの先生から数々の御援助と御激励をいただいた．ここに心から感謝の意を表する次第である．

本研究は，筆者が日本電信電話株式会社（NTT） 未来ねっと研究所 メディアネットワーキング研究部において，1997 年頃より 1999 年にかけて従事した超高速瞬時ファイル一括転送サービスに関する研究業務をきっかけとして取り組んできた研究テーマである．以下，筆者が NTT に在職中に御世話になった当時の所属名，役職名で，以下の方々に御礼を申し上げる．

本研究を行う機会を与えていただいた NTT 未来ねっと研究所 所長 三木 哲也 博士（現

在 電気通信大学 教授），所長 山下 一郎 氏（現在 NTT エレクトロニクス株式会社 取締役），所長 青山 友紀 博士（現在 東京大学 教授），所長 河内 正夫 博士（現在 NTT 先端技術総合研究所 所長），同研究所 メディアネットワーキング研究部 部長 佐野 浩一 博士，主幹研究員（グループリーダ） 金田 哲也 博士（現在 株式会社エヌ・ティ・ティ エムイー），主幹研究員（グループリーダ） 前田 洋一 氏（現在 NTT アクセスサービスシステム研究所），主幹研究員（グループリーダ） 森崎 正人 氏，主任研究員 橋本 仁 博士に心から感謝申し上げる．

また，当時の職場において同じ研究グループのメンバとして課題に取組み，有益な議論と励ましを頂いた NTT 未来ねっと研究所 メディアネットワーキング研究部 主任研究員 小野田 哲也 博士，研究主任 深田 陽一 氏，研究主任 吉川 太郎 氏，研究主任 小林 正啓 氏，小田部 悟士 氏，尾花 和昭 氏に心から感謝申し上げる．

更に，NTT 入社当初から退職するまでの間，御指導・御鞭撻を賜り，大変御世話になった NTT 未来ねっと研究所 研究企画部 育成研推担当課長 山林 由明 博士，NTT 光ネットワークシステム研究所 研究企画部 育成研推担当課長 小口 喜美男 博士（現在 NTT 未来ねっと研究所 メディアネットワーキング研究部 部長），同研究所 光加入者システム研究部 主幹研究員（グループリーダ） 渡辺 隆市 博士，主任研究員 山野 誠一 氏（現在 NTT アクセスサービスシステム研究所 主幹研究員），NTT 第一部門 柿沼 隆馬 氏（現在 NTT 未来ねっと研究所）に深く感謝申し上げる．また，当時，各方面における学会発表や展示会などにおいて，一緒に業務に従事していただいた NTT 未来ねっと研究所の社員の皆様にも，深く感謝の意を表する．

なお，本研究の一部は，平成 13 年度日本学術振興会科学研究費（奨励研究（A） 課題番号 13750364），並びに，平成 14 年度文部科学省科学研究費（若手研究（B） 課題番号 13750364）の助成を受けて行われた．関係者の皆様に深く感謝申し上げる．

参考文献

[1] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: An Overview," *RFC 1633*, June 1994.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," *RFC 2475*, Dec. 1998.

[3] 間瀬憲一, "インターネットにおけるスケーラブルなアドミッションコントロール方式," 信学誌, vol. 85, no. 9, pp. 655–661, 2002 年 9 月.

[4] 辻岡哲夫, 岡 育生, 藤原値賀人, 奥本隆昭, "衛星系と地上系の 2 通信路を用いた多次元符号化変調方式," 信学論 (B-II), vol. J76-B-II, no. 5, pp. 407–414, 1993 年 5 月.

[5] 辻岡哲夫, 岡 育生, 藤原値賀人, 奥本隆昭, 村田 正, 小牧省三, 森永規彦, "VSAT 衛星通信における多次元符号化の復号誤り特性," 信学技報 SAT93-75/CS93-144, pp. 85–90, 1993 年 11 月.

[6] T. Onoda, T. Tsujioka, R. Kakinuma, and S. Yamano, "Service-uniform ONU based on low cost audio AD/DA converters and CDM with novel code word sets," *IEICE Trans. Commun.*, vol. E82-B, no. 9, pp. 1446–1458, Sept. 1999.

[7] 辻岡哲夫, 小野田哲也, "パーシャルレスポンスを用いたキャリアレス AM/PM," 1995 信学ソ大（通信）, B-531, 1995 年 9 月.

[8] 辻岡哲夫, 小野田哲也, "UTP-3 を用いた 311Mbps 伝送方式の実験的検討," 1996 信学総大, B-935, 1996 年 3 月.

[9] 辻岡哲夫, 吉川太郎, 小野田哲也, "ギガビットスループットのための高速転送技術 〜 DMA (Direct Memory Access) beyond ATM Network 〜," 信学技報 CS96-34, pp. 15–20, 1996 年 6 月.

[10] S. Kotabe, T. Tsujioka, and T. Onoda, “Throughput characteristic of TCP with window size expansion and network direct memory access over a large delay-bandwidth link,” *IEICE Trans. Commun.*, vol. E81-B, no. 12, pp. 2357–2363, Dec. 1998.

[11] T. Kanada, T. Onoda, I. Yamashita, and T. Aoyama, “Netwarp: An ultra-high-throughput computer communications Service,” *IEEE Network Magazine*, pp. 44–50, Jan. 1997.

[12] T. Tsujioka, and T. Onoda, “An ultra high-speed file server with 105 Mbytes/s read performance based on a personal computer,” *IEICE Trans. Commun.*, vol. E81-B, no. 12, pp. 2503–2508, Dec. 1998.

[13] T. Tsujioka, K. Obana, and T. Onoda, “Design and implementation of a high-speed file server based on PC-UNIX,” *IEICE Trans. Electron.*, vol. E82-C, no. 12, pp. 2191–2200, Dec. 1999.

[14] T. Tsujioka, and T. Onoda, “105Mbytes/s ultra high-speed network file server,” *Proceedings of Taiwan-Japan joint-workshop on the latest development of telecommunication research (TJCOM ’98)*, pp. 59–64, Hsinchu, Taiwan, Jan. 1998.

[15] 辻岡哲夫，吉川太郎，小野田哲也, “70Mbytes/s を超える高速 PC ファイルシステム,” 1997 信学総大 B-7-188, 1997 年 3 月.

[16] 辻岡哲夫，小野田哲也，“80Mbytes/s を超える PC ファイルシステム,” 信学技報 MR97–3, pp. 13–18, 1997 年 6 月.

[17] 辻岡哲夫, 尾花和昭, 小野田哲也, “超高速 UNIX ファイルサーバ 〜 高速ネットワークと同等の連続入出力速度を目指して 〜,” 信学技報 IN98-57, pp. 25–30, 1998 年 9 月.

[18] 辻岡哲夫, 飯田光一, 杉山久佳, 村田 正, “IP ネットワークにおけるデッドラインに基づく優先制御・経路制御法,” 信学論 (B), vol. J86-B, no. 12, pp. 2501–2510, 2003 年 12 月.

[19] 辻岡哲夫, 尾花和昭, 小野田哲也, “送達時刻指定に基づく経路制御方式の性能評価,” 1998 信学ソ大（通信）, B-7-116, 1998 年 9 月.

[20] 辻岡哲夫, 尾花和昭, 小野田哲也, “時間を優先度とした優先制御待ち行列に関する一考察,” 信学技報 IN98-210, pp. 161–166, 1999 年 3 月.

[21] 辻岡哲夫, 飯田光一, 杉山久佳, 村田 正, “適応型マルチパスルーティングにおけるデッドラインに基づく優先制御・経路制御法の検討,” 信学技報 NS2002-19, pp. 9–16, 2002 年 4 月.

[22] 辻岡哲夫, 飯田光一, 杉山久佳, 村田 正, “デッドラインに基づく適応型マルチパスルーティング網の性能評価,” 信学技報 IN2002-26, pp. 25–30, 2002 年 6 月.

[23] T. Tsujioka, H. Sugiyama, and M. Murata, “A study of reliable transmission scheme for achieving end-to-end delay bounds by EDF queuing based on per-packet deadlines,” *Proceedings of International Symposium on Information Theory and Its Applications (ISITA 2002)*, pp. 451–454, Xi ’an, PRC, Oct. 2002.

[24] 辻岡哲夫, 杉山久佳, 村田 正, “不均一重み付け光直交符号を用いた CDMA の性能評価,” 第 24 回情報理論とその応用シンポジウム (SITA 2001), pp. 387–390, 2001 年 12 月.

[25] 辻岡哲夫, 杉山久佳, 村田 正, “不均一重みを有する光直交符号のランダム探索に関する検討,” 2002 信学ソ大（通信）, B-8-35, p. 275, 2002 年 9 月.

[26] 辻岡哲夫, 杉山久佳, 村田 正, “Ultra Wide Band Impulse Radio において完全な電力制御が行えない場合の性能劣化とその改善について,” 信学技報 CS2002-30, pp. 45–50, 2002 年 6 月.

[27] D. A. Patterson, G. Gibson, and R. H. Katz, “A case for redundant arrays of inexpensive disks (RAID),” *Proceedings of ACM-SIGMOD International Conference on Management of Data*, Chicago, pp. 109–116, Chikago, USA, June 1988

[28] S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman, “The design and implementation of the 4.3BSD UNIX Operating System,” Addison-Wesley Publishing, 1989.

[29] M. K. McKusick, W. N. Joy, S. J. Leffler, and R. S. Fabry, “A fast file system for UNIX,” *ACM Trans. on Computer Systems*, vol. 2, no. 3, pp. 181–197, Aug. 1984.

[30] S. LoVerso, N. Paciorek, A. Langerman, and G. Feinberg, "The OSF/1 unix filesystem (UFS)," *Proceedings of 1991 Winter USENIX Conference*, pp. 207–218, Dallas, Jan. 1991.

[31] R. Comerford, et al., "Technology 1998 analysis & forecast: Computers," *IEEE Spectrum*, pp. 43–47, Jan. 1998.

[32] A. Parekh, and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The multiple node case," *IEEE/ACM Trans. Networking*, vol. 2, pp. 137–150, April 1994.

[33] S. T. Liang, and M. C. Yuang, "Realization of earliest-due-date scheduling discipline for ATM switches," *IEICE Trans. Commun.*, vol. E81-B, no. 2, pp. 363–372, Feb. 1998.

[34] 横平徳美, 岡本卓爾, "EDD コネクション受付制御方式における遅延余裕の一割当法 — 最悪リンク遅延に比例した割当て," 信学論（B）, vol. J84-B, no. 8, pp. 1484–1493, 2001 年 8 月.

[35] T. Wu, and E. W. Knightly, "Buffering vs. smoothing for end-to-end QoS: Fundamental issues and comparison," *Proceedings of Performance '99*, Istanbul, Turkey, Aug. 1999.

[36] V. Sivaraman, and F. M. Chiussi, "Statistical analysis of delay bound violations at an earliest deadline first (EDF) scheduler," *Performance Evaluation*, vol. 36, no. 1, pp. 457–470, 1999.

[37] V. Sivaraman, and F. M. Chiussi, "Providing end-to-end statistical delay guarantees with earliest deadline first scheduling and per-hop traffic shaping," *Proceedings of INFOCOM 2000*, Tel Aviv, Israel, March 2000.

[38] R. Schoenen, "An architecture supporting quality-of-service in virtual-output-queued switches," *IEICE Trans. Commun.*, vol. E83-B, no. 2, pp. 171–181, Feb. 2000.

[39] K. Zhu, Y. Zhuang, and Y. Viniotis, "Achieving end-to-end delay bounds by EDF scheduling without traffic shaping," *Proceedings of INFOCOM 2001*, Anchorage, Alaska, April 2001.

[40] J. Liebeherr, and E. Yilmaz, "Workconserving vs. non-workconserving packet scheduling: An issue revisited," *Proceedings of IEEE/IFIP IWQoS '99*, June 1999.

[41] M. Andrews, and L. Zhang, "Minimizing end-to-end delay in high-speed networks with a simple coordinated schedule," *Proceedings of INFOCOM '99*, pp. 380–388, New York, March 1999.

[42] M. Andrews, "Probabilistic end-to-end delay bounds for earliest deadline first scheduling," *Proceedings of INFOCOM 2000*, Tel Aviv, Israel, March 2000.

[43] V. Sivaraman, F. M. Chiussi, and M. Gerla, "End-to-end statistical delay service under GPS and EDF scheduling: A comparison study," *Proceedings of INFOCOM 2001*, vol. 2, pp. 1113–1122, Anchorage, Alaska, April 2001.

[44] M. J. Miller, and S. Lin, "The analysis of some selective-repeat ARQ scheme with finite buffer," *IEEE Trans. Commun.*, vol. 29, no. 9, pp. 1307–1315, Sept. 1981.

[45] 楠堂忠夫, 岡 育生, 藤原値賀人, "連続多重送信選択再送方式に関する考察," 信学論（B-I）, vol. J81-B-I, no. 8, pp. 531–539, 1998 年 8 月.

[46] 木下 誠, 榊原勝己, 弓場芳治, "有限受信バッファ選択再送 ARQ 方式の Gilbert 通信路におけるスループット解析," 信学論（B）, vol. J82-B, no. 4, pp. 569–579, April 1999.

[47] 小田部悟士, 橋本 仁, 萩本和男, "Gbit/s クラスのスループットを実現する大容量データ一括転送方式," 信学論（B）, vol. J85-B, no. 8, pp. 1277–1284, 2002 年 8 月.

[48] PCI ローカルバス仕様書（第 2.0 版）.

[49] Ed. Solari, and G. Wilse, "PCI Hardware and Software, Architecture & Design," Annabooks, 1994.

[50] "Open Design No. 7: PCI バスの詳細と応用へのステップ," CQ 出版社.

[51] A. L. Drapeau, K. Shirri, E. K. Lee, J. H. Hartman, E. L. Miller, S. Seshan, R. H. Katz, K. Lutz, and D. A. Patterson, "RAID-II: A high-bandwidth network file server," *Proceedings of International Symposium on Computer Architecture (ISCA'21)*, pp. 234–244, Chicago, April 1994.

[52] P. Cao, S. B. Lim, S. Venkataraman, and J. Wilkes, "The TickerTAIP parallel RAID architecture," *Proceedings of International Symposium on Computer Architecture (ISCA'20)*, pp. 52–63, San Diego, May 1993.

[53] M. Rosenblum, and J. K. Ousterhout, "The design and implementation on a log-structured file system," *ACM Trans. on Computer Systems*, vol. 10, no. 1, pp. 26–52, Feb. 1992.

[54] J. C. Brustoloni, and P. Steenkiste, "Effects of buffering semantics on I/O performance," *Proceedings of 1996 USENIX Technical Conference*, pp. 277–291, San Diego, CA, Jan. 1996.

[55] K. A. Smith, and M. Seltzer, "A comparison of FFS disk allocation policies," *Proceedings of 1996 USENIX Technical Conference*, pp. 15–25, San Diego, CA, Jan. 1996.

[56] S. R. Kleinman, "Vnodes: An architecture for multiple file system types in Sun UNIX," *Proceedings of 1986 Summer USENIX Technical Conference*, pp. 238–247, El Toro, 1986.

[57] P. M. Chen, and D. A. Patterson, "Maximizing performance in a striped disk array," *Proceedings of International Symposium on Computer Architecture (ISCA '17)*, pp. 322–331, Seattle, Washington, May 1990.

[58] T. Y. Ts'o, and S. Tweedie, "Planned extensions to the Linux ext2/ext3 filesystem," *Proceedings of 2002 USENIX Annual Technical Conference*, pp. 235–244, Monterey, California, USA, June 2002.

[59] A. Sweeney, D. Doucette, W. Hu, C. Anderson, M. Nishimoto, and G. Peck, "Scalability in the XFS file system," *Proceedings of 1996 USENIX Technical Conference*, pp. 1–14, San Diego, CA, Jan. 1996.

[60] R. Bryant, R. Forester, and J. Hawkes, "Filesystem performance and scalability in Linux 2.4.17," *Proceedings of 2002 USENIX Annual Technical Conference*, pp. 259–274, Monterey, California, USA, June 2002.

[61] M. Seltzer, K. A. Smith, H. Balakrishnan, J. Chang, S. McMains, and V. Padmanabhan, "File system logging versus clustering: A performance comparison," *Proceedings of 1995 USENIX Technical Conference*, pp. 249–264, New Orleans, Jan. 1995.

[62] D. Kotz, and D. College, "Disk-directed I/O for MIMD multiprocessors," *ACM Trans. on Computer Systems*, vol. 15, no. 1, pp. 41–74, Feb. 1997.

[63] K. Shirriff, and J. Ousterhout, "Sawmill: A high-bandwidth logging file system," *Proceedings of USENIX Summer 1994 Technical Conference*, pp. 125–136, Boston, June 1994.

[64] A. L. N. Reddy, and P. Banerjee, "An evaluation of multiple-disk I/O systems," *IEEE Trans. on Computers*, vol. 38, no. 12, pp. 1680–1690, Dec. 1989.

[65] M. F. Kaashoek, and W. C. Hsieh, "The logical disk: A new approach to improving file systems," *ACM SIGOPS Conference*, pp. 15–28, 1993.

[66] S. W. Ng, "Improving disk performance via latency reduction," *IEEE Trans. on Computers*, vol. 40, no. 1, pp. 22–30, Jan. 1991.

[67] M. Shimizu, Y. Oue, and K. Ohnishi, "Parallel file access for implementing dynamic load balancing on a massively parallel computer," *IEICE Trans. Inf. & Syst.*, vol. E80-D, no. 4, pp. 466–472, April 1997.

[68] C. Park, and D. Kang, "Synchronous RAID5 with region-based layout and buffer analysis in video storage servers," *IEICE Trans. Inf. & Syst.*, vol. E81-D, no. 8, pp. 813–821, Aug. 1998.

[69] 林 孝典, 山崎真一郎, 森田直人, 相田 仁, 武市正人, 土居範久, "インターネットを用いた複数経路データ伝送方式の性能評価," 信学論（B）, vol. J84-B, no. 3, pp. 523–533, 2001年3月.

[70] D. Thaler, and C. Hopps, "Multipath issues in unicast and multicast next-hop selection," *RFC 2991*, Nov. 2000.

[71] C. Hopps, "Analysis of an equal-cost multi-path algorithm," *RFC 2992*, Nov. 2000.

[72] K. Farkas, "OMP technique in IP traffic engineering," *Periodica Polytechnica, Electrical Engineering*, vol. 44, no. 1, pp. 5–12, 2000.

[73] K. Shinjo, S. Shimogawa, J. Yamada, and K. Oida, "A strategy of designing routing algorithms based on ideal routings," *International Journal of Modern Physics C*, vol. 10, no. 1, pp. 63–94, 1999.

[74] 新上和正, 北川美宏, "AMR の経路選択の検討," 1999 信学総大, B-7-153, p. 262, 1999 年 3 月.

[75] 種田和正, 新上和正, 下川信祐, 山田順一, "適応型マルチパスルーチングの動的振舞い," 1997 信学総大, B-7-193, p. 322, 1997 年 3 月.

[76] S. Lin, D. J. Costello,Jr., and M. J. Miller, "Automatic-repeat-request error-control scheme," *IEEE Commun. Mag.*, vol. 22, no. 12, pp. 5-16, Dec. 1984.

[77] S. Lin, and P. S. Yu, "An effective error control scheme for satellite communications," *IEEE Trans. Commun.*, vol. 28, no. 3, p.395–401, March 1980.

[78] Y. Lim, and J. E. Kobza, "Analysis of a delay-dependent priority discipline in an integrated multiclass traffic fast packet switch," *IEEE Trans. Commun.*, vol. 38, no. 5, pp. 659–665, May 1990.

[79] T. Shiroshita, T. Sano, O. Takahashi, and N. Yamanouchi, "Performance evaluation of bulk-data reliable multicast Transport Protocol," *IEICE Trans. Inf. & Syst.*, vol. E82-D, no. 4, pp. 804–814, April 1999.

著者の発表論文

A．学術論文

[A. 1] 辻岡哲夫, 岡 育生, 藤原値賀人, 奥本隆昭, “衛星系と地上系の 2 通信路を用いた多次元符号化変調方式,” 信学論 (B-II), vol. J76-B-II, no. 5, pp. 407–414, 1993 年 5 月.

[A. 2] Tetsuo Tsujioka, and Tetsuya Onoda, “An ultra high-speed file server with 105 Mbytes/s read performance based on a personal computer,” IEICE Trans. Commun., vol. E81-B, no. 12, pp. 2503–2508, Dec. 1998.

[A. 3] Tetsuo Tsujioka, Kazuaki Obana, and Tetsuya Onoda, “Design and implementation of a high-speed file server based on PC-UNIX,” IEICE Trans. Electron., vol. E82-C, no. 12, pp. 2191–2200, Dec. 1999.

[A. 4] 辻岡哲夫, 飯田光一, 杉山久佳, 村田 正, “IP ネットワークにおけるデッドラインに基づく優先制御・経路制御法,” 信学論 (B), vol. J86-B, no. 12, pp. 2501–2510, 2003 年 12 月.

B．国際会議論文

[B. 1] Tetsuo Tsujioka, “105 Mbytes/s ultra high-speed network file server,” Proceedings of Taiwan-Japan joint-workshop on the latest development of telecommunication research (TJCOM ’98), pp. 59–64, Hsinchu, Taiwan, Jan. 1998.

[B. 2] Tetsuo Tsujioka, Hisayoshi Sugiyama, and Masashi Murata, “A study of reliable transmission scheme for achieving end-to-end delay bounds by EDF queuing based on per-packet deadlines,” Proceedings of International Symposium on Information Theory and Its Applications (ISITA 2002), pp. 451–454, Xi ’an, PRC, Oct. 2002.

Ｃ．口頭発表論文

[C. 1] 辻岡哲夫, 竹井 淳, カルロスバルデス (Valdez Carlos), 嶋本 薫, 小野里好邦, 岡 育生, 藤原値賀人, “VSAT を用いた衛星通信実験計画,” 信学技報 SAT91-93/RCS91-43, pp. 13–18, 1991 年 10 月–11 月.

[C. 2] 辻岡哲夫, 嶋本 薫, 小野里好邦, 岡 育生, 藤原値賀人, “VSAT を用いた衛星通信実験: ～ 多次元符号化の実験的検討 ～,” 第 14 回情報理論とその応用シンポジウム (SITA ’91), pp. 113–116, 1991 年 12 月.

[C. 3] 辻岡哲夫, 嶋本 薫, 小野里好邦, 岡 育生, 藤原値賀人, “VSAT を用いた多次元符号化実験,” 信学 ’92 春大, B-246, 1992 年 3 月.

[C. 4] 辻岡哲夫, 岡 育生, 藤原値賀人, 奥本隆昭, “誤り率の異なる２通信路を用いた多次元符号化変調方式に関する一考察,” 第 15 回情報理論とその応用シンポジウム (SITA ’92), pp. 41–44, 1992 年 9 月.

[C. 5] 辻岡哲夫, 岡 育生, 藤原値賀人, 奥本隆昭, “２通信路を用いた多次元符号化方式,” 信学 ’92 秋大, B-144, 1992 年 9 月.

[C. 6] 辻岡哲夫, 岡 育生, 藤原値賀人, 奥本隆昭, “衛星系と地上系の２通信路を用いた多次元符号化の ARQ に関する一考察,” 信学 ’93 春大, B-192, 1993 年 3 月.

[C. 7] 辻岡哲夫, 川池純一, 岡 育生, 藤原値賀人, 奥本隆昭, 村田 正, 小牧省三, 森永規彦, “衛星系と地上系の２通信路を用いた多次元符号化の実験的検討,” 第 16 回情報理論とその応用シンポジウム (SITA ’93), pp. 291–294, 1993 年 10 月.

[C. 8] 辻岡哲夫, 岡 育生, 藤原値賀人, 奥本隆昭, 村田 正, 小牧省三, 森永規彦, “VSAT 衛星通信における多次元符号化の復号誤り特性,” 信学技報 SAT93-75/CS93-144, pp. 85–90, 1993 年 11 月.

[C. 9] 辻岡哲夫, 小野田哲也, “パーシャルレスポンスを用いたキャリアレス AM/PM,” 1995 信学ソ大（通信）, B-531, 1995 年 9 月.

[C. 10] 辻岡哲夫, 小野田哲也, “ユニバーサル CPN インタフェースの検討,” 信学技報 CS95-142, pp. 29–34, 1995 年 11 月.

[C. 11] 辻岡哲夫, 小野田哲也, “UTP-3 を用いた 311Mbps 伝送方式の実験的検討,” 1996 信学総大, B-935, 1996 年 3 月.

[C. 12] 辻岡哲夫, 吉川太郎, 小野田哲也, “ギガビットスループットのための高速転送技術 〜 DMA (Direct Memory Access) beyond ATM Network 〜,” 信学技報 CS96-34, pp. 15–20, 1996 年 6 月.

[C. 13] 辻岡哲夫, 吉川太郎, 小野田哲也, “ギガスループットを実現する ATM-NIC 〜 Netwarp (3) 〜,” 1996 信学ソ大（通信）, B-715, 1996 年 9 月.

[C. 14] 辻岡哲夫, 吉川太郎, 小野田哲也, “70Mbytes/s を超える高速 PC ファイルシステム 〜 Netwarp III 〜,” 1997 信学総大, B-7-188, 1997 年 3 月.

[C. 15] 辻岡哲夫, 小野田哲也, “80Mbytes/s を超える PC ファイルシステム,” 信学技報 MR97-3, pp. 13–18, 1997 年 6 月.

[C. 16] 辻岡哲夫, 吉川太郎, 小野田哲也, “IP ネットワーク上で実現する Network DMA,” 1997 信学ソ大（通信）, B-7-126, 1997 年 9 月.

[C. 17] 辻岡哲夫, 小野田哲也, “100Mbytes/s 超高速 PC-UNIX ファイルサーバ,” 1998 信学総大, B-7-141, 1998 年 3 月.

[C. 18] 辻岡哲夫, 尾花和昭, 小野田哲也, “超高速 UNIX ファイルサーバ 〜 高速ネットワークと同等の連続入出力速度を目指して 〜,” 信学技報 IN98-57, pp. 25–30, 1998 年 9 月.

[C. 19] 辻岡哲夫, 尾花和昭, 小野田哲也, “送達時刻指定に基づく経路制御方式の性能評価,” 1998 信学ソ大（通信）, B-7-116, 1998 年 9 月.

[C. 20] 辻岡哲夫, 尾花和昭, 小野田哲也, “時間を優先度とした優先制御待ち行列に関する一考察,” 信学技報 IN98-210, pp. 161–166, 1999 年 3 月.

[C. 21] 辻岡哲夫, 杉山久佳, 村田 正, “捕捉効果を有する光直交符号の検討,” 信学技報 CS2001-95, pp. 7–14, 2001 年 10 月.

[C. 22] 辻岡哲夫, 杉山久佳, 村田 正, “不均一重み付け光直交符号を用いた CDMA の性能評価,” 第 24 回情報理論とその応用シンポジウム (SITA 2001), pp. 387–390, 2001 年 12 月.

[C. 23] 辻岡哲夫, 飯田光一, 杉山久佳, 村田 正, “適応型マルチパスルーティングにおけるデッドラインに基づく優先制御・経路制御法の検討,” 信学技報 NS2002-19, pp. 9–16, 2002 年 4 月.

[C. 24] 辻岡哲夫, 飯田光一, 杉山久佳, 村田 正, “デッドラインに基づく適応型マルチパスルーティング網の性能評価,” 信学技報 IN2002-26, pp. 25–30, 2002 年 6 月.

[C. 25] 辻岡哲夫, 杉山久佳, 村田 正, “Ultra Wide Band Impulse Radio において完全な電力制御が行えない場合の性能劣化とその改善について,” 信学技報 CS2002-30, pp. 45–50, 2002 年 6 月.

[C. 26] 辻岡哲夫, 杉山久佳, 村田 正, “不均一重みを有する光直交符号のランダム探索に関する検討,” 2002 信学ソ大（通信）, B-8-35, p. 275, 2002 年 9 月.

D．その他

[D. 1] 袴田直人, 境 忠勝, 辻岡哲夫, 嶋本 薫, 小野里好邦, “VSAT を用いた衛星通信実験のための実験支援システムの検討,” 信学 ’92 春大, B-247, 1992 年 3 月.

[D. 2] 山本衛児, 嶋本 薫, 小野里好邦, 辻岡哲夫, 岡 育生, “VSAT を用いた捕捉効果実験,” 信学 ’93 秋大, B-194, 1993 年 9 月.

[D. 3] 小野田哲也, 辻岡哲夫, 柿沼隆馬, 山野誠一, “マルチサービスのためのトランスペアレント ONU による光加入者システム,” 信学技報 CS95-4, pp. 21–26, 1995 年 4 月.

[D. 4] 赤木節子, 吉川英機, 辻岡哲夫, 岡 育生, 藤原値賀人, “シンボル消失を考慮したビタビ復号法,” 信学技報 IT95-8, pp. 43–47, 1995 年 5 月.

[D. 5] T.Onoda, T.Tsujioka, R.Kakinuma, and S.Yamano, “A novel fiber-optic local access system adopting the “Service-uniform” ONU,” Proceedings of ICC ’95, pp. 606–610, Seattle, Washington, June 1995.

[D. 6] 小野田哲也, 辻岡哲夫, “オーディオ用 A/D コンバ-タによるユニバーサルラインカード,” 1995 信学ソ大（通信）, B-584, 1995 年 9 月.

[D. 7] 小野田哲也, 辻岡哲夫, “変曲点抽出によるパルス検出法について,” 信学技報 CS95-175, pp. 13–18, 1996 年 1 月.

[D. 8] 吉川太郎, 辻岡哲夫, 小野田哲也, “アクセス系における大容量コンテンツの瞬時一括転送 〜 Netwarp 〜,” 信学技報 CS96-33, pp. 9–14, 1996 年 6 月.

[D. 9] 小野田哲也, 吉川太郎, 辻岡哲夫, “Netwarp: ギガスループットを実現するファイル転送のためのデータ転送,” 1996 信学ソ大（通信）, B-713, 1996 年 9 月.

[D. 10] 吉川太郎, 辻岡哲夫, 小野田哲也, “ギガスループットのためのデータ転送技術 〜 Netwarp (2) 〜,” 1996 信学ソ大（通信）, B-714, 1996 年 9 月.

[D. 11] 小野田哲也, 辻岡哲夫, 小田部悟士, “Network DMA の長距離転送特性 〜 Netwarp I 〜,” 1997 信学総大, B-7-186, 1997 年 3 月.

[D. 12] T. Onoda, T. Tsujioka, T. Yoshikawa, and T. Kanada, “Netwarp: A gigabit throughput file transfer scheme,” Proceedings of 8th International Workshop on Optical/Hybrid Access Networks (OAN), 9.3, Atlanta, March 1997.

[D. 13] T. Onoda, T. Tsujioka, and T. Kanada, “Network direct memory access for gigabit/s throughput,” Proceedings of the European Conference on Networks and Optical Communications (NOC ’97), pp. 205–211, Antwerpen, Belgium, June 1997.

[D. 14] 小田部悟士, 小野田哲也, 辻岡哲夫, “Network DMA over UDP のデータ再送方式に関する検討,” 1997 信学ソ大（通信）, B-7-127, 1997 年 9 月.

[D. 15] 小田部悟士, 辻岡哲夫, 小野田哲也, “Network DMA の長距離転送特性評価,” 信学技報 CS97-110, pp. 1–6, 1997 年 10 月.

[D. 16] 小笹史郎, 小田部悟士, 辻岡哲夫, 小野田哲也, “Netwarp を実現する複数ルータ間転送アルゴリズムの検討,” 信学技報 IN97-137, pp. 45–52, 1997 年 11 月.

[D. 17] S. Kotabe, T. Onoda, and T. Tsujioka, “Network direct memory access for Gbit/s throughput over a long fat pipe,” Proceedings of Taiwan-Japan joint-workshop on the latest development of telecommunication research (TJCOM ’98), pp. 53–58, Hsinchu, Taiwan, Jan. 1998.

[D. 18] 尾花和昭, 辻岡哲夫, 小野田哲也, “パケットの送達時刻指定を可能とする蓄積型経路制御の検討,” 信学技報 IN98-58, pp. 31–36, 1998 年 9 月.

[D. 19] 尾花和昭, 辻岡哲夫, 小野田哲也, “パケットの送達希望時刻指定を可能とする蓄積型経路制御,” 1998 信学ソ大（通信）, B-7-115, 1998 年 9 月.

[D. 20] S. Kotabe, T. Tsujioka, and T. Onoda, “Throughput characteristic of TCP with window size expansion and network direct memory access over a large delay-bandwidth link,” IEICE Trans. Commun., vol. E81-B, no. 12, pp. 2357–2363, Dec. 1998.

[D. 21] 尾花和昭, 辻岡哲夫, 小野田哲也, “パケットの送達時刻指定を可能とする蓄積型経路制御の実装,” 信学技報 IN98-211, pp. 167–172, 1999 年 3 月.

[D. 22] 尾花和昭, 辻岡哲夫, 小野田哲也, “蓄積型ネットワークにおける優先度制御の検討,” 1999 信学総大, B-7-42, 1999 年 3 月.

[D. 23] T. Onoda, T. Tsujioka, R. Kakinuma, and S. Yamano, “Service-uniform ONU based on low cost audio AD/DA converters and CDM with novel code word sets,” IEICE Trans. Commun., vol. E82-B, no. 9, pp. 1446–1458, Sept. 1999.

[D. 24] T. Yoshikawa, T. Onoda, and T. Tsujioka, “The Netwarp service on UNIX system and its application to a high speed distributed file system,” Proceedings of ICCC ’99, Sept. 1999.

[D. 25] 尾花和昭, 辻岡哲夫, 小野田哲也, “送達希望時間に基づく優先度制御,” 1999 信学ソ大（通信）, B-7-54, 1999 年 9 月.

[D. 26] 入江智和, 辻岡哲夫, 杉山久佳, 村田 正, “論理演算三角法に基づくロスレス符号化〜2 値画像データ圧縮への適用〜,” 信学技報 IE2000-101, pp. 25–30, 2000 年 12 月.

[D. 27] K. W. Jang, T. Tsujioka, H. Sugiyama, and M. Masashi, “Code acquisition scheme using the estimated channel gain in a Rayleigh fading channel,” Proceedings of 5th International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies (KES ’2001), vol. 2, pp. 849–853, Sept. 2001.

[D. 28] 入江智和, 辻岡哲夫, 杉山久佳, 村田 正, “高速データ通信に適したリアルタイム圧縮アルゴリズム: 論理演算三角法における符号辞書の拡張,” 信学技報 CS2001-87, pp. 1–6, 2001 年 10 月.

[D. 29] 宮下智和, 辻岡哲夫, 杉山久佳, 村田 正, “CDMA 車々間通信における電力制御法の検討,” 信学技報 CS2001-99, pp. 37–42, 2001 年 10 月.

[D. 30] 入江智和, 辻岡哲夫, 杉山久佳, 村田 正, “高速データ通信に適したリアルタイム圧縮アルゴリズム － 2 値画像データ圧縮への適用 －,” 映像情報メディア学会誌, vol. 56, no. 5, pp. 849–852, 2002 年 5 月.

[D. 31] 入江智和, 辻岡哲夫, 杉山久佳, 村田 正, “論理演算三角法を実現する符号器の設計と実装,” 2002 信学ソ大（基礎・境界）, A-6-4, p. 118, 2002 年 9 月.

[D. 32] K. W. Jang, T. Tsujioka, H. Sugiyama, and M. Murata, “Serial acquisition using the estimated channel gain in CDMA mobile communication,” Proceedings of International Symposium on Nonlinear Theory and Its Applications (NOLTA2002), pp. 659–662, Xi’an, PRC, Oct. 2002.

[D. 33] K. W. Jang, T. Tsujioka, H. Sugiyama, and M. Murata, “Code acquisition using the estimated channel gain for CDMA mobile communication,” Proceedings of International Conference on Information Technology & Applications (ICITA 2002), 159-1, Bathurst, Australia, Nov. 2002.

[D. 34] H. Sugiyama, T. Tsujioka, and M. Murata, “Ad Hoc network simulator based on DSDV routing method,” Memoirs of the Faculty of Engineering Osaka City Univ., pp. 43–58, Dec. 2002.

[D. 35] 杉山久佳, 辻岡哲夫, 村田 正, “災害現場におけるネットワーク形成を目的とした移動ロボットの行動アルゴリズム,” 第 3 回 SICE システムインテグレーション部門講演会 SI2002, 1P32-01, pp. 255–256, 2002 年 12 月.

[D. 36] 入江智和, 辻岡哲夫，杉山久佳，村田 正， “論理演算三角法を用いたデータ圧縮における新動的演算子選択手法,” 2003 信学総大, A-6-4, p. 172, 2003 年 3 月.

[D. 37] 小寺 宏, 辻岡哲夫, 杉山久佳, 村田 正, “Walsh 符号と多値 PPM を用いた Ultra Wideband Impulse Radio に関する一検討,” 信学技報 CS2003-23, pp. 17–21, 2003 年 6 月.

出願特許

[E. 1] 平成 6 年 12 月 8 日, 特願平 6-305240, 特開平 08-163081,
符号分割多重通信装置,
発明者: 辻岡哲夫, 小野田哲也,　出願人: 日本電信電話株式会社

[E. 2] 平成 7 年 8 月 4 日, 特願平 7-199941, 特開平 09-051320,
通信装置,
発明者: 辻岡哲夫, 小野田哲也,　出願人: 日本電信電話株式会社

[E. 3] 平成 9 年 2 月 28 日, 特願平 09-046356, 特開平 10-240583,
ファイルシステム,
発明者: 辻岡哲夫, 小野田哲也,　出願人: 日本電信電話株式会社

[E. 4] 平成 9 年 3 月 11 日, 特願平 09-056547, 特開平 11-085413,
記録装置,
発明者: 辻岡哲夫, 小野田哲也,　出願人: 日本電信電話株式会社

[E. 5] 平成 10 年 2 月 19 日, 特願平 10-03771, 特開平 11-237958,
ファイルシステムの書き込みと読み出し方法およびそれを使用した装置,
発明者: 辻岡哲夫,　出願人: 日本電信電話株式会社

[E. 6] 平成 10 年 2 月 26 日, 特願平 10-045941, 特開平 11-249937,
コンピュータシステム,
発明者: 辻岡哲夫,　出願人: 日本電信電話株式会社

[E. 7] 平成 10 年 6 月 19 日, 特願平 10-172476,
（追加出願: 平成 10 年 12 月 3 日, 特願平 10-343946）, 特開 2000-078188,
優先経路制御方法及びルータ装置,
発明者: 小野田哲也, 辻岡哲夫, 尾花和昭,　出願人: 日本電信電話株式会社

[E. 8] 平成 11 年 4 月 5 日, 特願平 11-97194, 特開 2000-293316,
通信ディスク装置,
発明者: 辻岡哲夫,　出願人: 日本電信電話株式会社