システムソフトウェアと
オペレーティング・システム 90－2
（２００２．６．２７）

HiTactixを応用した Darwinストリームサーバ向けI/Oエンジンの設計と実装

竹内 理　　レ・モアル ダミエン　　野村 賢

（株）日立製作所システム開発研究所

高速アクセス網の普及に伴い、高品質なストリームデータの有料配信サービスが注目されてきている。このサービスの実用化には、高い配信能力、配信品質の保証機能、既存の市販ストリームサーバとの機能互換性のすべてを兼ね備えるストリームサーバが必要である。

本稿では、既存の市販ストリームサーバにわずかな改変を加えるだけで、その機能互換性を維持しつつ、配信性能の改善と、配信品質保証機能の追加を実現する外付けI/Oエンジン方式の提案を行う。さらに、Darwinストリームサーバを外付けI/Oエンジン方式に対応させるために施した改変の概要について述べる。最後に改変したDarwinストリームサーバの性能評価結果について述べ、配信性能が改変前と比して5倍程度向上したこと等を明らかにする。

Design and Implementation of Darwin Streaming Server I/O Engine over HiTactix

Tadashi Takeuchi, Damien Le Moal and Ken Nomura

Systems Development Laboratory, Hitachi, Ltd.

With broadband networks expansions, charged streaming services of high quality stream data are likely to become common. In order to efficiently realize such services, stream servers providing high data throughput performance, mechanisms to assure stream quality and compatibility with existing stream servers are desirable.

In this paper, we propose the *external I/O engine architecture* which can enhance data throughput performance and provide QoS assurance mechanisms by adding small modification to existing stream servers. We present an outline of the modifications we added to the Darwin Streaming Server in order to conform it to this architecture. Finally, we present evaluation results of the modified Darwin Streaming Server, and show that the modified server can deliver 5 times the performance of the original one.

1. はじめに

近年、光ファイバ網、CATV 網等の高速アクセス網が普及しつつある。これら高速アクセス網を提供している各キャリアは、自社が提供するアクセス網への加入者のさらなる増加を促進し、かつ、基本料金以外の追加収入をもたらすサービスの提供を模索している。特に、高品質なストリームデータの有料配信サービスは、上記サービスの有力候補として注目されている[1]。

しかしながら、高品質なストリームデータの有料配信サービスは、現在までのところ実用化に至っていない。その理由の一部として、既存の市販ストリームサーバに以下の問題があることがあげられる。

1） 既存の市販ストリームサーバのストリーム配信性能が低いため、少ないユーザで一つのサーバを共有せねばならない。そのため、サービス提供コストが高くなる[2]。

2） 既存の市販ストリームサーバはストリーム配信の品質保証を行なえない[3]。そのため、有料サービス化が困難である（ユーザは課金されているにもかかわらず、満足いく品質でストリームデータの配信を受けられない可能性がある）。

しかし一方で、サービス普及のためには、既存の市販ストリームサーバとの機能互換性を維持し、多数のユーザが保有するストリームクライアントアプリケーションや、サービス管理者が使いなれている各種管理アプリケーション（ストリーム配信モニタ等）を従来と同様に利用可能にした方が望ましい。

著者らは次世代ストリームサーバ向け専用 OS HiTactix の研究開発を行ってきた。HiTactix は、高速 I/O 機能や I/O レート保証機能等の機能を提供している[4〜7]。

本稿では、既存の市販ストリームサーバにわずかな改変を加えるだけで、その機能互換性を維持しつつ、HiTactix を利用して当該サーバのストリーム配信性能の改変、ストリーム配信品質保証機能の追加を実現する外付け I/O エンジン方式を提案する。さらに、本方式を用いて Darwin ストリームサーバ[1]の改変を行ない、その性能を定量的に評価する。

以下、まず外付け I/O エンジン方式の概要について説明する。次に Darwin ストリームサーバの改変内容について説明する。最後に、改変した Darwin ストリームサーバの性能評価結果について述べる。

2. 外付け I/O エンジン方式の概要

本節では、既存の市販ストリームサーバのストリーム配信性能向上、ストリーム配信の品質保証機能の提供を可能にすべく報告者らが新規に提案した、「外付け I/O エンジン方式」の概要について述べる。

外付け I/O エンジン方式の概要を図 1 に示す。

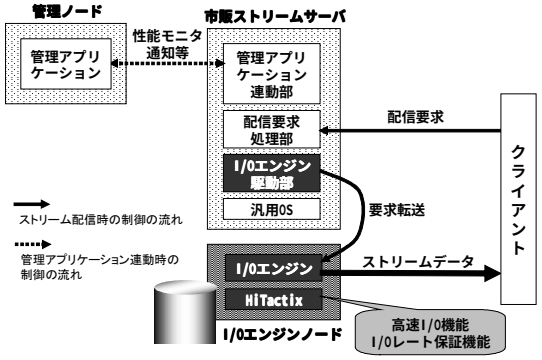

図 1 外付け I/O エンジン方式

本方式は、以下の特徴がある。

1） 市販ストリームサーバ（Linux 等の汎用 OS が動作する）と、I/O エンジンノード（HiTactix が動作する）を並置する。

2） 市販ストリームサーバに「I/O エンジン駆動部」を搭載する。従来は、クライアントから配信要求が到達した場合、市販ストリームサーバ自身が汎用 OS を用いたストリーム配信（ディスク I/O やネットワーク I/O）を実行していた。「I/O エンジン駆動部」はこの代わりに、上記配信要求到達時に、並置してある I/O エンジンノードに配信要求の転送を行う。

3） I/O エンジンノードに「I/O エンジン」を搭載する。「I/O エンジン」は転送された配信要求を受理すると、配信要求で指定されるストリームデータをディスクから読み出し、同じく配信要求で指定されるクライアントに対して当該データをネットワーク送信する。上記 I/O の実行は、HiTactix を用いて行なう。

4） 市販ストリームサーバ-クライアント間、市販ストリームサーバ-管理ノード間のインタフェースは変えない。すなわち、市販ストリームサーバで利用していたストリームクライアントアプリケーションや管理アプリケーション（ストリーム配信の性能モニタ等）は、本方式を適用してもそのまま流用できる。

本方式では、ストリームデータの I/O は、

[1] QuickTime フォーマットのストリームデータの配信が可能なストリームサーバ。Apple 社が実装した。ソースコードが公開されており、誰でも自由に入手、改変できる。なお、QuickTime をはじめとする本稿に記載されている各製品名は各社の登録商標である。

HiTactix の高速 I/O 機能を用いて行なう。そのため、汎用 OS を用いてストリームデータの I/O を行なっていた従来方式と比して、配信性能を向上できる。また、HiTactix は I/O レート保証機能も提供しているため、ストリーム配信の品質保証も可能になる。

また、本方式の実現のために必要となるソフトウェア開発コストは膨大にならない。まず、ストリーム配信に関しては、市販ストリームサーバアプリケーションへの「I/O エンジン駆動部」の搭載、及び HiTactix 上の「I/O エンジン」の新規実装だけで実現できる。また、管理アプリケーションとの連動の実現に関しても、ソフトウェア開発コストは膨大にならない。性能モニタ情報を I/O エンジンノードから取得する、等の機能追加を図１に示す「管理アプリケーション連動部」に施すのみで既存方式と同等の機能を提供できる。

3. Darwin ストリームサーバの改変内容

本節では、外付け I/O エンジン方式を適用すべく著者らが Darwin ストリームサーバに加えた改変内容の詳細について述べる。そして、少ないソフトウェア開発工数で、Darwin ストリームサーバの機能互換性を維持しつつ、外付け I/O エンジン方式が実現可能であることを明らかにする。

以下、ストリーム配信を実現するために施した改変内容と、管理アプリケーションとの連動を実現するために施した改変内容に分けて説明する。

3.1. ストリーム配信の実現

改変前の Darwin ストリームサーバにおけるストリーム配信の実現モジュールの構成を図 2 に示す。

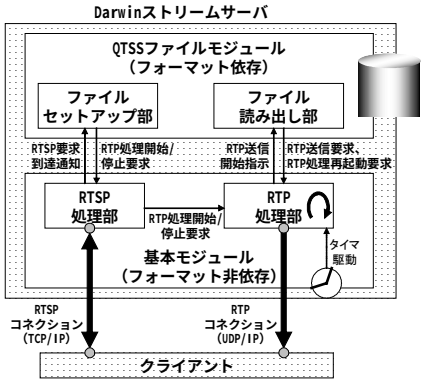

図 2 Darwin ストリームサーバのストリーム配信実現モジュールの構成（改変前）

Darwin ストリームサーバはクライアントとの間に以下の 2 つのコネクションを形成してストリーム配信を実行する。

1） 配信開始／停止等の制御データを送受する RTSP[8]コネクション（TCP/IP プロトコルを使用する）

2） ストリームデータを送受する RTP[9]コネクション（UDP/IP プロトコルを使用する）

改変前の Darwin ストリームサーバは、ストリーム配信処理を、基本モジュールと QTSS ファイルモジュールにて行う。基本モジュールは、配信するストリームデータのフォーマットに依存しない処理を実行する。一方、QTSS ファイルモジュールはフォーマット依存の処理を実行する。

両モジュールは、RTSP コネクションを介して送受される制御データ（以後「RTSP 要求」と呼ぶ）の到達を契機に動作を開始する。RTSP 要求到達時には、基本モジュールの RTSP 処理部が RTSP 要求を受理する。そして QTSS ファイルモジュールのファイルセットアップ部に RTSP 要求到達通知を送る。ファイルセットアップ部はファイルオープン／クローズ等の、当該 RTSP 要求を処理する際に必要となるファイル操作を実行する。また、配信開始要求／停止の RTSP 要求到達が到達した場合には、ファイルセットアップ部は RTSP 処理部を介して RTP 処理部に対して RTP 処理開始／停止要求を発行する。RTP 処理部は、ストリームデータの配信を行うタスクの起動／停止を行う。

また、両モジュールは上記タスクがタイマを契機に駆動した際にも動作する。RTP 処理部が管理する上記タスクは、RTP 送信開始指示を QTSS ファイルモジュールのファイル読み出し部に対して発行する。ファイル読み出し部は、単位時間分のストリームデータをディスクから読み出し、RTP 処理部に対して RTP 送信要求を発行する。RTP 処理部は RTP コネクションを介して、当該ストリームデータをクライアントに配信する。さらに読み出したストリームデータを解析し、次のストリームデータがディスクから読み出すべき時刻を決定する。そして、RTP 処理部に対して RTP 処理再起動要求を発行し、指定した時間が経過した後に、上記タスクを再びタイマ契機に駆動することを要求する。

外付け I/O エンジン方式を用いたストリーム配信を実現するために、図 3 に示すストリーム配信実現モジュールの改変を行った。

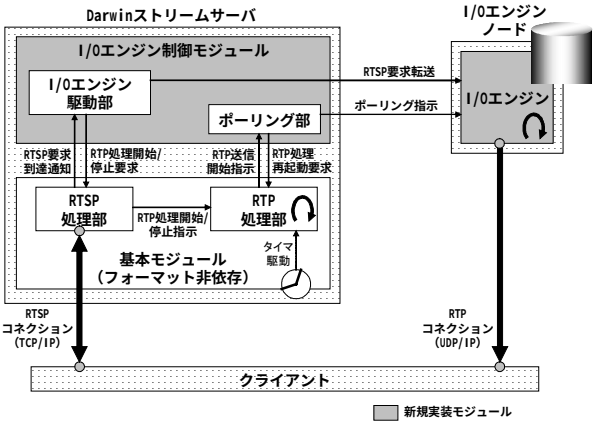

図 3 ストリーム配信実現モジュールの改変内容

本改変では、QTSS ファイルモジュールを I/O エンジン制御モジュールに置き換えた。I/O エンジン制御モジュールは、Darwin ストリームサーバと I/O エンジンノードとの連動を実現するために新規実装したモジュールである。I/O エンジン制御モジュールは、I/O エンジン駆動部とポーリング部を含む。

I/O エンジン駆動部は、RTSP 処理部から RTSP 要求到達通知を受け取った際に動作する。上記通知を受け取ると当該要求を I/O エンジンに転送する。要求を転送された I/O エンジンは、転送された RTSP 要求に応じてストリームデータのディスクからの読み出し、及び RTP コネクションを介した当該データのクライアントへの配信を行う。また、I/O エンジン駆動部は、改変前と同様に RTP 処理部が管理するタスクの起動／停止処理等も実行する。

ポーリング部は、RTP 処理部の管理するタスクが RTP 処理開始指示を発行した際に動作する。ポーリング部は、RTP 処理開始指示を受理すると、ストリームデータをディスクから読み出す代わりに、I/O エンジンに対してポーリング要求を発行する。I/O エンジンはポーリング要求を受け取ると、当該タスクにより管理されているストリーム配信が正常に行われているか否かを検査する。正常に行われていなければ、ポーリング部は異常処理を実行し、ストリーム配信の強制終了処理を行う。またポーリング部は、改変前と同様に RTP 処理再起動要求を発行するが、上記発行時に指定する時刻として、次のストリームデータを読み出すべき時刻ではなく、次にポーリングを実行すべき時刻を指定する。

本改変によっても、Darwin ストリームサーバの機能互換性を維持し、Darwin ストリームサーバ-クライアント間のインタフェースは変わらない。まず、RTSP、RTP コネクションを介して送受されるデータの内容に変更はない。また、クライアントにとっては、RTP コネクションの接続先が Darwin ストリームサーバから I/O エンジンノードに変更されているが、この接続先は RTSP コネクションで送受される制御データ内容に基づき動的に決定する仕様になっているので[8]、インタフェースを変更する必要はない。

本改変に伴うソースコードの変更量は表 1 に示す通り 8K ステップ弱に抑えられた。

表 1 ソースコード変更量（ストリーム配信分）

モジュール名	変更量
I/O エン制御モジュール（I/O エンジン駆動部、ポーリング部）	2.8K ステップ
I/O エンジン	5.0K ステップ
総計	7.8K ステップ

3.2. 管理アプリケーションとの連動の実現

改変前の Darwin ストリームサーバにおける管理アプリケーション連動の実現モジュールの構成を図 4 に示す。

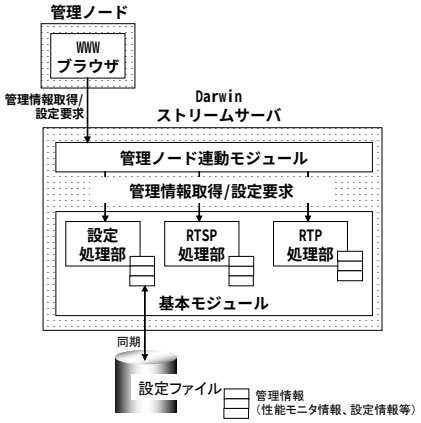

図 4 管理アプリケーション連動実現モジュールの構成（改変前）

Darwin ストリームサーバの管理アプリケーションは、管理ノード上の WWW ブラウザ上で動作す

る。

改変前の Darwin ストリームサーバは、この管理アプリケーションとの連動を実現するため、管理ノード連動モジュールを搭載している。本モジュールは、WWW ブラウザからの管理情報取得／設定要求の到達を契機に動作する。性能モニタ情報（ネットワーク I/O レート等）や設定情報（ストリームデータの格納ディレクトリ等）は基本モジュールが管理しているため、上記要求を受けた管理ノード連動モジュールは基本モジュールに対して管理情報取得／設定要求を発行する。なお、基本モジュールは必要に応じて Darwin ストリームサーバの設定ファイルを更新し、基本モジュールが管理している管理情報との同期をとる。

外付け I/O エンジン方式を適用した Darwin ストリームサーバを管理アプリケーションと連動させるため、図 5 に示す改変を行った。

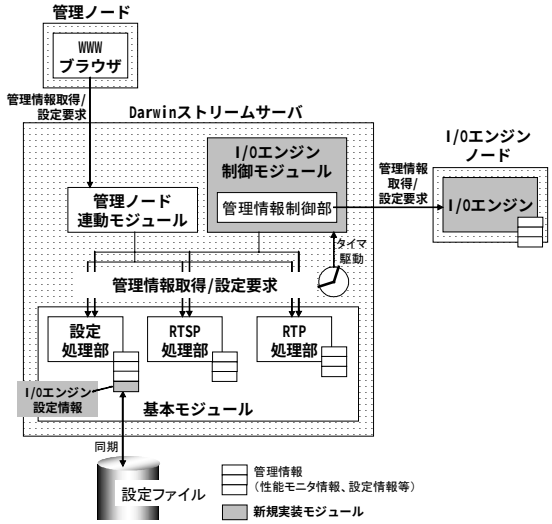

図 5 管理アプリケーション連動実現モジュールの改変内容

まず、基本モジュールが管理する設定情報に、I/O エンジン設定情報（I/O エンジンノードの IP アドレス等）を追加した。

加えて、3.1 節で述べた I/O エンジン制御モジュールに管理情報制御部を追加した。

管理情報制御部はその初期化時に基本モジュールに対して管理情報取得要求を発行し、I/O エンジンに通知すべき管理情報を取得する。そして取得した情報の設定要求を I/O エンジンに対して発行する。また、管理情報制御部は周期的にタイマを契機に駆動し、性能モニタ情報等の動的に更新される管理情報の取得要求を I/O エンジンに発行する。そして、取得した情報の設定を基本モジュールに要求する。この設定により、基本モジュールは改変前と同様の管理情報を自モジュール内に保持可能になる。

I/O エンジンにも、上記管理情報制御部の動作に対応するため、性能モニタ等の管理情報を管理する機能や、I/O エンジン制御モジュールからの管理情報取得／設定要求に応える機能を追加した。

本改変を行っても、Darwin ストリームサーバの機能互換性を維持し、Darwin ストリームサーバ-管理ノード間のインタフェースは変わらない（管理ノード連動モジュールには改変を加えていない）。

また表 2 に示す通り、本改変により変更したソースコードの量は 1K ステップ強に抑えられた。

表 2 ソースコード変更量
（管理アプリケーション連動分）

モジュール名	変更量
基本モジュール	0.1K ステップ
I/O エンジン 制御モジュール	0.2K ステップ
I/O エンジン	1.0K ステップ
総計	1.3K ステップ

4．性能評価実験

外付け I/O エンジン方式を適用した Darwin ストリームサーバと、適用していない既存 Darwin ストリームサーバとの間で、ストリーム配信性能、及びストリーム配信品質保証性能の比較実験を行った。以下、上記各実験の内容と、実験結果の概要について述べる。

4.1．ストリーム配信性能の比較実験

ストリーム配信性能の比較実験で用いた実験システムの構成を図 6 に示す。

本実験システムは、Darwin ストリームサーバ（外付け I/O エンジン方式適用版）、I/O エンジンノード、Darwin ストリームサーバ（外付け I/O エンジン方式非適用版）、負荷ジェネレータからなる。本実験で用いたハードウェアの仕様を表 3 に示す。

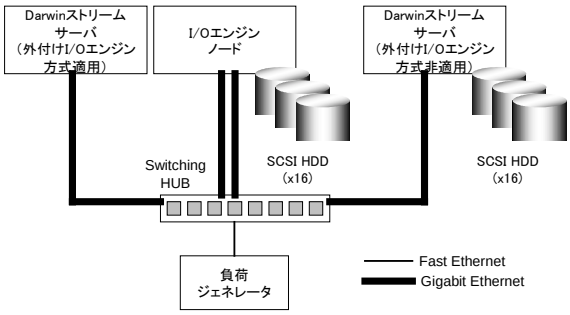

図 6 ストリーム配信性能比較実験システム

表 3 ハードウェア仕様
（ストリーム配信性能比較実験）

ノード名	ハードウェア仕様
Darwin ストリーム サーバ （外付け I/O エンジン 方式適用版）	PC/AT 互換機 （PentiumIII 1.26GHz 搭載） Gigabit Ethernet NIC（x1） （Alteon AceNIC チップ搭載）
I/O エンジンノード	PC/AT 互換機 （PentiumIII 1.26GHz 搭載） Gigabit Ethernet NIC（x2） （Alteon AceNIC チップ搭載） Ultra160 SCSI カード（x4） （LSILogic 53C1010 チップ搭載） Ultra160 HDD（x16）
Darwin ストリーム サーバ （外付け I/O エンジン 方式非適用版）	PC/AT 互換機 （PentiumIII 1.26GHz 搭載） Gigabit Ethernet NIC（x1） （Alteon AceNIC チップ搭載） Ultra160 SCSI カード（x4） （LSILogic 53C1010 チップ搭載） Ultra160 HDD x 16
負荷ジェネレータ	PC/AT 互換機 （PentiumIII 600MHz 搭載） Fast Ethernet NIC（x1） （Intel DC21143 チップ搭載）

実験は、負荷ジェネレータから各 Darwin ストリームサーバに対して可変のストリーム数の配信要求を送信し、I/O エンジンノードまたは Darwin ストリームサーバ（外付け I/O エンジン方式非適用版）のストリーム配信レート及び CPU 負荷がどのように変動するかを測定した。負荷ジェネレータは、I/O エンジンノードまたは Darwin ストリームサーバ（外付け I/O エンジン方式非適用版）が保持する 16 台のハードディスクから均等に配信するように要求を発行し、各ハードディスクの I/O 負荷を均一にした。また、ディスクキャッシュは使用しないようにした。

測定結果を図 7 及び図 8 に示す。各グラフの横軸は負荷ジェネレータが配信を要求したストリーム数を、縦軸は各ノードのストリーム配信レート及び CPU 負荷を表している。

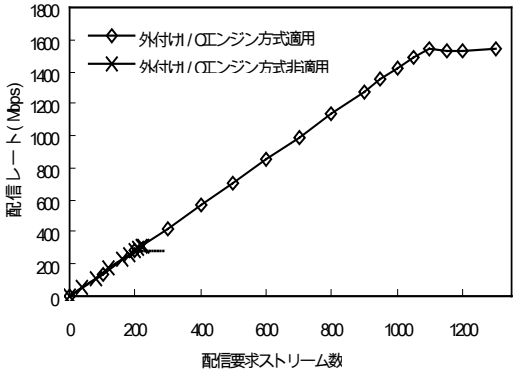

図 7 測定結果（配信レートの変動）

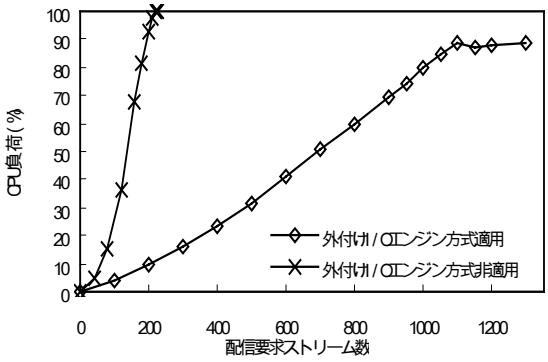

図 8 測定結果（CPU 負荷の変動）

測定の結果、以下のことが明らかになった。

1） 外付け I/O エンジン方式非適用の Darwin ストリームサーバのストリーム配信性能は 300Mbps 程度である。外付け I/O エンジン方式を適用すると、ストリーム配信性能が約 5 倍向上し、1.5Gbps 程度のストリーム配信が可能になる。これは、高速 I/O 機能を提供する HiTactix を用いてディスク I/O やネットワーク I/O を実行する外付け I/O エンジン方式の効果が表れていと考えられる。

2） 外付け I/O エンジン方式非適用の場合、CPU 負荷が 100%に到達するため、配信要求ストリーム数を増やしてもストリー

ム配信レートが増大しない。一方、外付け I/O エンジン方式を適用した場合、CPU 負荷が 100%に満たないにもかかわらず、ストリーム配信レートが増えなくなる。これは、I/O エンジンノードの PCI バスの帯域が飽和しているためだと考えられる[2]。

4.2. ストリーム配信品質保証実験

ストリーム配信品質保証性能の比較実験も、図 6 に示した実験システムを用いて行った。

本実験でも、負荷ジェネレータから各 Darwin ストリームサーバに対して可変のストリーム数の配信要求を送信し、I/O エンジンノードまたは Darwin ストリームサーバ（外付け I/O エンジン方式非適用版）のパケットジッタがどのように変動するかを測定した。ここで言うパケットジッタとは、ストリームデータを格納する各パケットを送信すべき時刻と、実際の送信時刻との差を言う。送信すべき時刻は、QuickTime フォーマットのストリームデータに格納されているヒント情報[10]から算出できる。また、I/O エンジンノード及び Darwin ストリームサーバ（外付け I/O エンジン方式非適用版）は 7 分間のストリームデータを配信する。パケットジッタは、この 7 分間の最大と平均を測定する。この値が小さい程、期待された時刻にストリーム配信を実行しており、ストリーム配信品質をより厳密に保証していると言える。

測定結果を図 9 に示す。グラフの横軸は、I/O エンジン、または Darwin ストリームサーバ（外付け I/O エンジン非連動版）の CPU 負荷（配信要求ストリーム数を可変にした結果変動する）を示す。縦軸は、パケットジッタの大きさを示す。

測定の結果、以下のことが明らかになった。外付け I/O エンジンを適用しなければ、平均で 23ms～55ms 程度のパケットジッタが発生する。特に CPU 負荷が 80%を超えると平均のパケットジッタが大きく増大する。また、最大のパケットジッタは CPU 負荷にかかわらず常に 80ms 以上になる。これに対し、外付け I/O エンジン方式を適用することにより、CPU 負荷にかかわらず平均のパケットジッタは 0 になる。最大のパケットジッタは 7 ms～13ms で、外付け I/O エンジン方式非適用の場合の 1/11 以下である。

外付け I/O エンジン方式では、I/O 実行タスクが厳密に一定周期でスケジューリングされる（HiTactix の提供するアイソクロナススケジューラ[4]を利用する）。上記はこの効果が表れた結果だと考えられる。

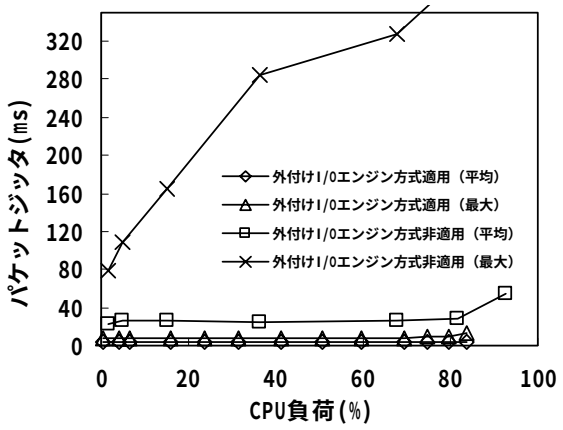

図 9 測定結果（パケットジッタの変動）

また、外付け I/O エンジン方式を適用した場合、HiTactix の持つディスク I/O スケジューリング方式[6]を用いて、ストリームごとに優先度を付けてストリーム配信を行える。すなわち、ディスクが過負荷状態になった場合においても、優先度の低いストリームのディスク読み出しレートを落とすことにより、優先度の高いストリームのデータを期待されるレートでディスクから読み出し、クライアントに配信できる。

外付け I/O エンジン方式を適用した Darwin ストリームサーバが、上記に示すストリーム配信制御を行えることを確認するため、以下に示す実験を図 6 の実験システムを用いて行った。

まず負荷ジェネレータから、1 ストリームの配信要求を Darwin ストリームサーバ（外付け I/O エンジン適用版）に対して送る。この 1 ストリームは、低い優先度での配信を要求する。次に、可変ストリーム数の配信要求を負荷ジェネレータから Darwin ストリームサーバ（外付け I/O エンジン適用版）に対して送る。これらのストリームは、高い優先度で配信するように要求する。なおこれらのストリームは、優先度の低いストリームと同じハードディスクからデータを読み出すことを要求した。優先度の高い配信を要求するストリーム数を変動させた場合、各ストリームの配信

[2] I/O エンジンノードが保持する PCI バスの理論性能は 532MB/s である。1.5Gbps のストリーム配信を行った場合、この約 70%を使用する。

レートがどのように変動するかを測定した。

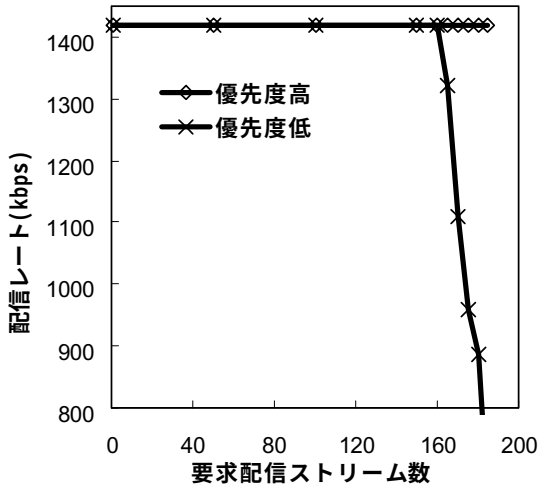

図 10 測定結果（配信レートの変動）

測定結果を図 10 に示す。図 10 のグラフの横軸は優先度の高い配信を要求したストリーム数を、縦軸はストリーム配信レートの平均を表す。ストリーム配信レートは、優先度の高いストリームと優先度の低いストリームのそれぞれの平均値を測定した。

測定の結果、要求配信ストリーム数が 160 を超えてから、期待通りに優先度の低いストリームの配信レートのみが落ち込むこと、及び優先度の高いストリームの配信レートは要求配信ストリーム数が 160 を超えても厳密に一定に保てることが確認できた。

5 まとめ

本稿では、既存の市販ストリームサーバの機能互換性を維持しつつ、HiTactix を利用して当該ストリームサーバのストリーム配信性能向上や、ストリーム配信品質保証機能の追加を実現する外付け I/O エンジン方式を提案した。さらに、Darwin ストリームサーバを外付け I/O エンジン方式に対応させるべく施した改変内容の詳細についても述べた。さらに、改変した Darwin ストリームサーバの性能評価を行い、ストリーム配信性能を 5 倍程度向上できることや、データ送信遅れを 1/11 以下に低減し、より厳密なストリーム配信品質保証が可能になることを確認した。

参考文献

[1] 阿蘇和人，「ブロードバンドに命を吹き込め」，日経コミュニケーション No.343, Jun. 2001.

[2] nCube, “n4 Streaming Media System”, nCube white paper PN109786, Nov. 2000.

[3] N. Venkatasubramanian et. al., “An Integrated Metric for Video QoS”, Proceedings of 5th ACM international conference on Multimedia, Nov. 1997.

[4] 竹内理他，「連続メディア処理向き OS の周期駆動保証機能の設計と実装」，情報処理学会論文誌 Vol. 40, No. 3, Mar. 1999.

[5] 中野隆裕他，「Ethernet 上で QoS を保証する通信方法の設計と実装」，情報処理学会論文誌 Vol. 41, No. 2, Feb. 2000.

[6] 竹内理他，「HiTactix-BSD 連動システムを応用した大規模双方向ストリームサーバの設計と実装」，情報処理学会論文誌 Vol. 43, No. 1, Jan. 2002.

[7] M. Iwasaki et.al., “A Micro-Kernel for Isochronous Video-Data Transfer”, WWCA 97, Mar. 1997.

[8] H. Schulzrinne et. al., “Real Time Streaming Protocol (RTSP)”, RFC-2326, Apr. 1998.

[9] M. Schulzrinne et. al., “RTP: A Transport Protocol for Real-Time Applications”, RFC-1889, Jan. 1996.

[10] “Inside QuickTime: QuickTime File Format”,http://developer.apple.com/techpubs/quicktime/qtdevdocs/PDF/QTFileFormat.pdf.

[11] H. Schulzrinne et. al., “RTP: A Transport Protocol for Real-Time Applications”, RFC-1889, Jan. 1996.

[12] M.B. Jones et. al., “CPU Reservations and Time Constraints: Efficient, Predictable Scheduling of Independent Activities”, Symposium on Operating Systems Principles, pp198-211, 1997.

[13] F.W. Miller et. al,, “General Data Streaming”, 19th IEEE Real-Time System Symposium (RTSS), pp232-241, Nov. 1998.

[14] P. R. Barham, “A Fresh Approach to File System Quality of Service”, 7th International Workshop on Network and Operating System Support for Digital Audio and Video, May. 1997.