ストリーミングサーバ向き通信機構の設計と実装

清水 雅純 † 勝部 弘嗣 † 御田村 晃 † 瀧本 栄二 † 毛利 公一 †† 大久保 英嗣 ††
† 立命館大学大学院理工学研究科 †† 立命館大学理工学部情報学科

1 はじめに

近年, ギガビット・イーサネットなどの登場により, LAN において大容量のマルチメディアコンテンツを配信することが可能となってきている. このようなサービスでは, サーバが多数のクライアントに対してサービスを提供する. そのため, サーバは, クライアントに対して効率的にデータ転送を行う必要がある. また, クライアントが動画や音声を円滑に再生するためには, サーバが安定したデータ転送を行う必要がある.

従来のデータ転送手法では, メモリコピーやコンテキストスイッチなどのオーバヘッドが発生し, サーバの性能低下を招いている. また, データ転送に関する時間的制御がアプリケーションレベルで行われているため, 厳密なリアルタイム性の保証は困難であり, サーバのデータ配信処理にジッタが発生する.

本稿で提案する機構は, 冗長なメモリコピーやコンテキストスイッチによるオーバヘッドを削減することにより, 効率的なデータ転送を実現する. さらに, フロー毎の帯域制御を行い, 安定したデータ転送を提供する. これにより, ストリーミングサーバは, 本機構を利用して効率的かつ安定したデータ転送を行うことが可能となる.

以下, 本稿では, 2 章で従来の OS におけるデータ転送手法, 3 章で通信機構の設計, 4 章で評価について述べ, 最後に 5 章でまとめと今後の予定について述べる.

2 従来の OS におけるデータ転送手法

Linux や FreeBSD といった OS では, 2 次記憶装置上のデータを効率良く転送するために, sendfile システムコールを提供している. sendfile は, カーネル内で処理を完結することにより, メモリコピーやコンテキストスイッチのオーバヘッドを削減している. しかし, 2 次記憶装置からのデータ読出し速度とネットワークの帯域を制御する機能を持っていない. そのため, 安定したデータ転送が必要となるストリーミングサービスに sendfile を適用することは適切ではない.

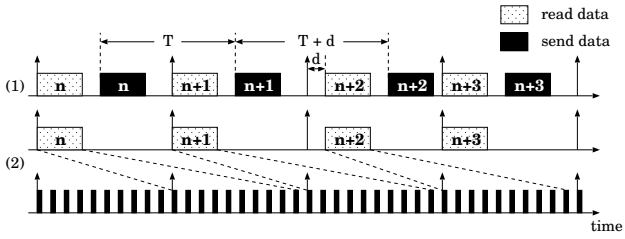

図 1 I/O の周期実行

また, sendfile では, 2 次記憶装置からのデータ読出し処理とネットワークインタフェースへの送信処理は逐次的に行われる. そのため, どちらか一方の処理がブロックされた場合, 他方の処理の周期が揺らぐことになる. 図 1-(1) では, n+1 回目までパケットの送信処理を周期 T で実行しているが, n+2 回目のデータ読出し処理において時間 d の遅延が発生した場合, n+1 回目と n+2 回目のパケット送信処理の時間間隔が T+d となり, データ転送にジッタが発生する.

3 通信機構の設計

3.1 配信処理の並列化

本機構では, 2 章で述べた問題点を解決するために, 2 次記憶装置からのデータ読出し処理とネットワークへのパケット送信処理を 2 つに分離し, それぞれの処理を異なるカーネルスレッドによって実行する. これにより, それぞれの処理を異なる周期で実行することが可能となるため, 2 次記憶装置から読み出すデータサイズを大きく保ちつつ, 小さなパケットを連続して送信することができる. そのため, 2 次記憶装置の利用効率を高く維持しながら, 帯域を細かく制御することが可能である. さらに, データの読出し処理とパケットの送信処理を並列に行うことが可能となるため, I/O のスループットが向上する.

また, 図 1-(2) のようにデータの読出し処理が遅延した場合においても, データの読出し処理が次の周期までに完了可能であれば, パケットの送信処理には影響を与えない. このように, それぞれの処理が, 他方の処理に影響を与えないため, サーバは安定したデータ転送を行うことが可能となる.

The Design and Implementation of Communication Mechanism for Streaming Server

Masayoshi Shimizu†, Hirotsugu Katsube†, Akira Mitamura†, Eiji Takimoto†, Koichi Mouri††, and Eiji Okubo††
†Graduate School of Science and Engineering, Ritsumeikan University
††Department of Computer Science, Faculty of Science and Engineering, Ritsumeikan University

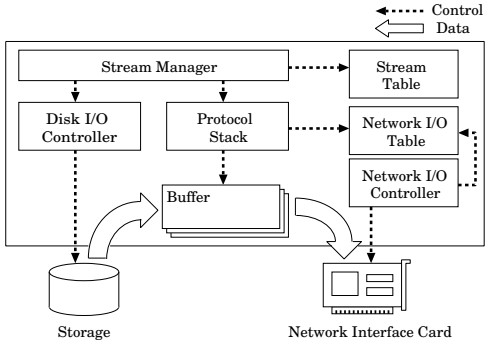

図 2 本機構の構成

3.2 処理の流れ

本機構の構成を図 2 に示す. 本機構は, データの読出し処理とパケットの送信処理を異なるスレッドによって実行する. また, すべての処理をカーネル内で完結することにより, メモリコピーやアドレス空間の切替えによるオーバヘッドを削減している. 以下, 本機構の処理の流れを示す.

(1) ストリーム管理部は, ユーザからのデータ転送要求を受け付け, ストリーム管理テーブルにエントリを追加する. エントリには, 2 次記憶装置から読み出すデータの位置やサイズ, データ転送先のアドレス情報, 要求の優先度と転送速度等を格納する. 以後, 周期的に起動し, ディスク I/O 制御部に対して 1 周期分のデータ読出しを要求する.

(2) ディスク I/O 制御部は, ストリーミング管理部から受け付けた要求の優先度とデッドラインに基づいて, ディスク I/O のスケジューリングを行う. 2 次記憶装置からデータを読み出し, バッファに格納する.

(3) ストリーム管理部は, 読み出されたデータに対するプロトコルスタックの処理を要求する.

(4) プロトコルスタックは, 最初にデータを分割し, それぞれに対してプロトコルスタックの処理を行う. 処理を完了後, ネットワーク I/O 要求テーブルにエントリを追加する. エントリには, ネットワーク I/O 制御部が 1 周期ごとに送信するべきパケット数, 残存パケット数, 送信するべきパケットのアドレス等が格納される.

(5) ネットワーク I/O 制御部は, 一定周期ごとに起動し, ネットワーク I/O 要求テーブルを走査して送信するべきパケットを調べる. それぞれのエントリに対して, 指定された数のパケットをバッファから取り出して送信する.

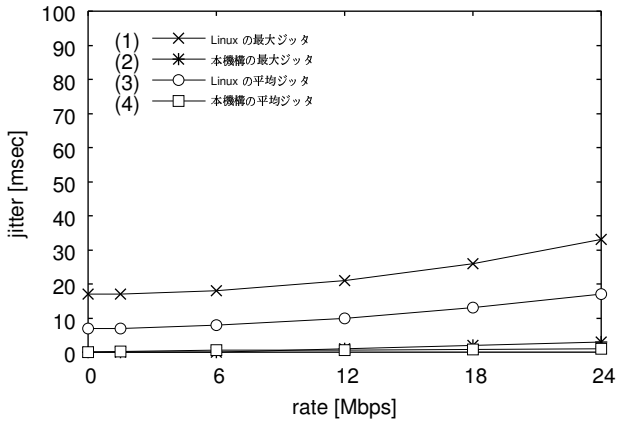

図 3 ジッタの測定結果

4 評価

現在, 我々が開発しているリアルタイム OS Easel [1] に本機構を実装し, 帯域制御機能の評価を行った. 実験は, Intel Celeron 1GHz の CPU と Intel PRO/100 S のネットワークインタフェースカードを搭載した PC/AT 互換機で行った. 送信側の PC からクロスケーブルで接続された受信側の PC に対してデータを転送してジッタを測定し, 本機構と Linux を比較した. 今回の実験では, 帯域制御機能のみを評価するために, メモリ上のデータを転送した.

測定結果を図 3 に示す. 横軸は, データの転送レートであり, 縦軸はジッタの大きさを示す. (1) は, Linux のパケット送信における最大ジッタである. Linux では, データの転送レートが 6Mbps の場合, 最大ジッタは 21msec となる. (2) は, 本機構における最大ジッタである. 本機構では, データの転送レートが 6Mbps の場合, 最大ジッタは 250usec 以下となり, Linux のデータ転送における最大ジッタの 1.2% 以下となった. (3) と (4) は, それぞれ Linux と本機構における平均ジッタである. 24Mbps において, Linux の平均ジッタは 11msec, 本機構では 2msec となった.

5 おわりに

本稿では, ストリーミングサーバ向き通信機構について述べた. 本機構は, 冗長なメモリコピーやコンテキストスイッチによるオーバヘッドを削減し, フロー毎の帯域制御を行う. これにより, ストリーミングサーバは, 効率的かつ安定したデータ転送を行うことが可能となる.

参考文献

[1] 谷出 新: “リアルタイムオペレーティングシステム Easel におけるメモリ管理機構,” 情報処理学会研究報告 2001-OS-86, Vol. 2001, No.21, pp. 91-98 (2001).