DEWS2005 3B-o1

iSCSIアクセス時のTCP輻輳ウィンドウ制御のための アルゴリズムの検討

豊田　真智子† 　山口　実靖†† 　小口　正人†

† お茶の水女子大学　〒 112–8610 東京都文京区大塚 2–1–1
†† 東京大学 生産技術研究所　〒 153–8505 東京都目黒区駒場 4–6–1
E-mail: †machiko@ogl.is.ocha.ac.jp, ††sane@tkl.iis.u-tokyo.ac.jp, †oguchi@compter.org

あらまし　ストレージ管理コストを低下させるために登場した IP-SAN の技術の中で，iSCSI が注目を集めている．iSCSI を利用したストレージアクセス時には TCP 層のパラメータの１つである輻輳ウィンドウとシステム性能に密接な関連のあることが確認されている．本稿では，iSCSI ストレージアクセス時に確認されるスループットのばらつきを抑制するために提案した輻輳ウィンドウコントロール手法において，より早く輻輳ウィンドウを一定値に収束させるためのアルゴリズムの検討を行う．線形探索法に加え，新たに 2 分探索法と輻輳ウィンドウ値通知法を提案し，これらの実装を行い，アルゴリズムの評価を行った．

キーワード　iSCSI，輻輳ウィンドウ，制御アルゴリズム，遠隔ストレージ

A Proposal of Algorithm for Controlling TCP Congestion Window on iSCSI Access

Machiko TOYODA†, Saneyasu YAMAGUCHI††, and Masato OGUCHI†

† Ochanomizu University　Otsuka 2–1–1, Bunkyo-ku, Tokyo, 112–8610 Japan
†† Institute of Industrial Science, The University of Tokyo
Komaba 4–6–1, Meguro-ku, Tokyo, 153–8505 Japan
E-mail: †machiko@ogl.is.ocha.ac.jp, ††sane@tkl.iis.u-tokyo.ac.jp, †oguchi@compter.org

Abstract　iSCSI is one of the most promising technologies in IP-SAN, which is to decrease storage manegement cost. Congestion Window of TCP parameter and system performance have a close relationship in accessing remote storage with the iSCSI. In this paper, we examine the algorithms to make the Congestion Window settled to the definite value as soon as possible in dynamic Congestion Window control method, which is proposed to balance the throughput unevenness. In addition to linear search method, we proposed binary search method and Congestion Window notification method, implemented them on the experimental system, and evaluated the algorithms.

Key words　iSCSI，Congestion Window，Control Algorithm，Remote Storage

1. はじめに

ブロードバンドネットワーク技術の発展により，多くの企業がネットワーク経由で遠隔地にある様々なデータをまとめて管理しようとする要望を持つようになった．動画や音声などのデータをデジタル化してコンピュータで処理し，ネットワーク経由で扱う機会も多くなっている．それに伴い，マルチメディアコンテンツなどのアプリケーションが蓄積するデータ量も増大している．計算機システムが処理するデータ量の飛躍的な増加に対し，これを保持するストレージ管理コストの増大を抑制するため，SAN（Storage Area Network）が登場した．

SAN はサーバとストレージ間を接続する高速のネットワークであり，サーバがストレージに直接接続される DAS（Direct Attached Storage）とは異なり，複数のサーバでストレージを共有することができ，ストレージを集中的に管理することが可能となる．現在普及している SAN は，ファイバチャネルを用いて構築される FC-SAN である．しかし FC-SAN では，導入コスト等が問題となり，Ethernet と TCP/IP を用いて構築される IP-SAN が次世代 SAN として期待されている．

IP-SAN の中で最も注目を集めているのが iSCSI である [1] [2]. しかし iSCSI を用いて遠隔ストレージアクセスを行った場合には，TCP/IP のみで構築されたネットワークと比較してスルー

プット低下が問題となる [3]．またスループットは，TCP パラメータの 1 つである輻輳ウィンドウと相関関係があり，輻輳ウィンドウ低下時にはスループットも低下することがわかっている [4]．iSCSI は常に Read/Write リクエスト後にデータが一斉送信されるため，TCP のセルフクロッキングが効かず，バースト性が高くなり，エラーによる輻輳ウィンドウ低下が起こりやすい．そこで我々は文献 [5] において，輻輳ウィンドウの状態に基づき，ストレージアクセスする際のブロックサイズを変化させ，輻輳ウィンドウを動的にコントロールすることによりスループットを安定させる手法を提案した．本手法により，輻輳ウィンドウを限界値で一定に保ったままストレージアクセスを行うことが可能となる．

iSCSI はその構成の特徴から，遠隔地にストレージを配置した長距離接続の環境においてその効力を発揮するものであると期待されている．このような高遅延環境においては，パケットを送信してから ACK が返信されるまでの時間が長くなり，何もせずに待っているだけという無駄な時間が増えてしまう．この待ち時間は，一度に送信できるパケット数が小さいときほど長くなり，この間のスループットは劇的に低下すると考えられる．そのため，送信パケット数である輻輳ウィンドウが変化を繰り返し，スループットが増減を繰り返すよりも，輻輳ウィンドウの値を一定値に保ち，安定したスループットでストレージアクセスを行う方が効率が良い．本研究において，現在の輻輳ウィンドウコントロール手法は線形探索法を用いて実装しており，輻輳ウィンドウが一定値となるまでの時間がアクセスするブロックサイズの初期値に依存し，収束時間にばらつきが生じてしまう．そのため，より早い時間で輻輳ウィンドウを収束させ，最適な通信状態を達成するためのアルゴリズムを検討する必要がある．

そこで本稿では，文献 [5] で提案した輻輳ウィンドウコントロール手法において，輻輳ウィンドウが一定値となるまでのコントロール時間を短縮するアルゴリズムの検討を行う．新たに提案するコントロール手法では，2 分探索法のアルゴリズムに基づき，またはデータ送信側から通知された輻輳ウィンドウの値に基づき，再アクセスするブロックサイズ値を決定する．これらのアルゴリズムを実装した実験システムを用いて，iSCSI プロトコルで遠隔ストレージアクセスを行った．その結果，線形探索法に基づいた手法よりも新たに提案する手法の方が効率的であることが確認された．また輻輳ウィンドウが収束した際のパラメータ間の関係をモデル化することにより，本制御手法を解析的に評価する．

本稿は以下のように構成される．まず 2 章で研究背景を述べる．3 章では NIC のバッファサイズを変更した場合の輻輳ウィンドウの振る舞いについて紹介する．4 章で提案する輻輳ウィンドウコントロール手法のアルゴリズムについて概説する．5 章でアルゴリズムを実装して実験を行い，その結果について議論する．6 章で実験から得られた結果より，NIC ディスクリプタ，輻輳ウィンドウ限界値，最適ブロックサイズの関係について述べる．7 章で関連研究について触れ，最後に 8 章でまとめを述べる．

2. 研 究 背 景

2.1 輻輳ウィンドウ，ブロックサイズ，スループットの関係

本節ではまず，本稿におけるキーワードとなる輻輳ウィンドウ，ブロックサイズ，スループットについて述べる．

輻輳ウィンドウとは，ネットワークの輻輳制御を目的としてデータ送信側が送信データ量を制限するためのパラメータであり，受信側からの確認応答なしで連続送信を行う最大パケット数である．通信中にエラーが起こらない場合には，TCP 実装が規定されたアルゴリズムに基づき輻輳ウィンドウをより大きな値に設定するが，異常を検出すると輻輳ウィンドウを低下させ，低い値から再度通信を開始する．この場合 TCP 実装においては，データ受信側がどこまでデータを受け取ったか送信側に知らせるための確認応答パケットである ACK を 1 つ受信するごとに 1 ずつ増加するという実装になっている．ブロックサイズは一度に送信するデータ長であり，この値を増加させることで送信するデータ量が増加し，スループットは向上する．しかし，あまりブロックサイズを大きくしすぎると，データ送信側が保持している送信バッファが溢れてしまい，TCP 実装がエラーを検出して輻輳ウィンドウが減少してしまう．すなわちバッファ溢れのエラーを起こさない範囲で最大となる輻輳ウィンドウの限界値が存在する．通信における性能評価基準であるスループットを向上させることは非常に重要であるが，そのためにはブロックサイズと輻輳ウィンドウ双方の適切なバランスを保つことが必要であると言える．

2.2 iSCSI ストレージアクセスにおける問題点

iSCSI は 2003 年 2 月に IETF により承認された新しい技術であり，ストレージ機能提供側を Target，ストレージ要求側を Initiator と呼ぶ．SCSI コマンドをカプセル化して TCP/IP ネットワーク上に転送するため，ブローバンドインターネットが普及した今日においては，遠隔地にストレージを配置した長距離接続の環境において，その効力を発揮するものと期待されている [1] [2].

iSCSI は複雑な階層構造を持つプロトコルである．ストレージアクセス時には，SCSI プロトコル，iSCSI プロトコル，TCP/IP プロトコル，ネットワークシステムなどが関連して動作するため，各層における処理が全体の通信性能を低下させてしまう [3]．そのため，内部動作を把握することが性能向上にとって重要である．

そこで我々は，iSCSI を用いてストレージアクセスを行った際の性能低下の原因を調べるため，通信において重要な階層となる TCP 層の振る舞いの把握を行った．Iniaitor から Target にシーケンシャルリードアクセスを行った場合には，アクセスするブロックサイズによって，輻輳ウィンドウが一定値となったり，増減を繰り返す鋸型の変化を行ったりという，異なる振る舞いを示すことが確認された [4]．またスループットは，輻輳ウィンドウの低下時には一時低下し，通信性能が不安定になっていることが確認された．

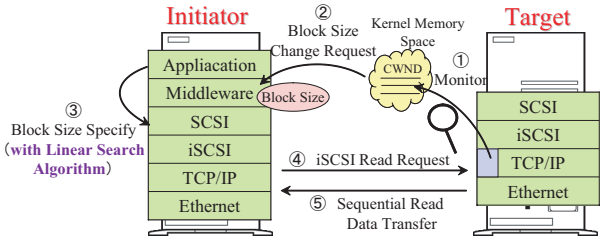

図 1　輻輳ウィンドウコントロール手法の概念図

2.3　輻輳ウィンドウコントロール手法

スループットを安定させるために提案した輻輳ウィンドウのコントロール手法の概要を図 1 に示す．本実験では，通信時に TCP 実装が行っているフロー制御で管理されている TCP パラメータをモニタし，可視化することでその状態を把握している．Linux の TCP のソースコード内のモニタしたい箇所にモニタ関数を挿入し，TCP パラメータを外部からアクセス可能なカーネルメモリ空間に記録するように実装を行い，カーネルを再構築する．これにより，Linux カーネル内部の TCP 実装で管理されている TCP パラメータを，カーネルメモリ空間にアクセスするための特殊ファイルを読み出すことにより確認が可能となる [6].

この TCP パラメータモニタの仕組みを Target に実装し，Target から輻輳ウィンドウの通知を受けて，アプリケーションによるストレージアクセスのブロックサイズを調節する機能をミドルウェアとして実装した．輻輳ウィンドウは，送信側のデバイスドライバのバッファが溢れることによる Local Congestion エラーを検出した場合（CWR），重複 ACK，SACK を受信した場合（Recovery)，タイムアウトを検出した場合（Loss）の 3 つのエラーのいずれかが起こった場合に低下する．本実験時には CWR が頻度の高いエラーとして検出されているため，CWR エラーにより輻輳ウィンドウが低下した場合に本手法によるコントロールを行うものとした．また，Linux の TCP 実装は，通信中に一度輻輳ウィンドウが設定されると，そのウィンドウの値を使い切らない限り輻輳ウィンドウが変化しないという特徴を持ち，この場合は現在アクセスしているブロックサイズよりも大きなブロックサイズで通信を行うことができる可能性がある．そのため，輻輳ウィンドウが一定値であると判断した場合には，現在アクセスしているブロックサイズよりさらに大きなブロックサイズで再アクセスを行うようにした．これらの方針に基づいた輻輳ウィンドウコントロール手法を，以下に述べる．

（1）　Target で輻輳ウィンドウをモニタし，変化を観察する

（2）　観察中に CWR が検出され，輻輳ウィンドウが低下した場合にはブロックサイズの低下要求を，輻輳ウィンドウが一定値であると判断した場合には，ブロックサイズの増加要求を Initiator に行う

（3）　通知を受けた Initiator では，ミドルウェアが線形探索アルゴリズムに基づいてブロックサイズを決定し，アプリケーションがブロックサイズを再指定する

（4）　Initaitor から Target にシーケンシャルリードコマンドを送信し，ストレージアクセスを行う

表 1　使用計算機

CPU	Intel Xeon 2.4GHz
Main Memory	512MB DDR SDRAM
OS	Initiator, Target ： Linux2.4.18-3 Dummynet ： FreeBSD 4.9 - RELEASE
NIC	Initiaotr, Target : Intel PRO/1000XT Server Adapter Dummynet : Intel PRO/1000MT Server Adapter

（5）　Target が Initiator に向けて要求されたブロックサイズのデータ転送を実行する

これらの処理を CWR を検出するか，輻輳ウィンドウが一定値であると判断する度に繰り返す．ストレージアクセス後に最初に CWR を検出した場合は初めて輻輳ウィンドウが一定値となった時を，最初に一定値であると判断した場合は CWR を検出し輻輳ウィンドウが低下した後初めて一定値となった時を，それぞれ輻輳ウィンドウが収束したと判断する．この時，本手法によって計算される最適なブロックサイズとなる．

3.　NIC バッファサイズを変更した性能測定実験

本実験で用いた NIC（Network Interface Card）は，通信時に TCP 層から受け取ったデータを保持するためのバッファサイズを，ディスクリプタ値を設定することにより変更することが可能となっている．本章では，NIC ディスクリプタを変更した場合の輻輳ウィンドウとスループットの振る舞いについて述べる．

3.1　実 験 環 境

本実験は以下の環境で行った．Initiator と Target 間は Gigabit Ethernet で接続し，TCP/IP 接続を確立した．サーバとストレージ間が離れている場合のストレージアクセスを想定した実験を行うため，Ethernet の接続途中に人工的な遅延装置として FreeBSD Dummynet [7] を挟み，片道遅延時間を“ 2ms ”，往復“ 4ms ”に設定し，クロスケーブルで接続した．Initiator，Target，Dummynet はすべて PC 上に構築し，Initiator，Target には Linux を，Dummynet には FreeBSD をインストールした．iSCSI ネットワークにおける性能を調べるため，Target はメモリモードで動作させ，ディスクアクセスを伴わないように設定した．実験で使用した計算機の環境を表 1 に示す．

また，本実験で用いた iSCSI 実装において，Target にはニューハンプシャー大学 InterOperability Lab が提供する UNH IOL reference implementation ver.3 on iSCSI Draft 18 [8] を用いた．この UNH 実装では，大きなブロックサイズで read コマンドを発行しても，SCSI 層において要求したブロックサイズより小さなブロックサイズに分割してデータを送信する [3]．本実験ではこの実装による性能測定への影響を避けるために，Initiator には UNH 実装を用いず，UNH 実装の Initiator と同等の機能を持ち，かつ大きなブロックサイズのデータ転送を行う自作 Initiator を用いて実験を行った．この自作 Initiator は通常のユーザ空間のアプリケーションとして動作し，iSCSI Target と TCP/IP コネクションを確立して iSCSI プロトコルで通信を行うものである．

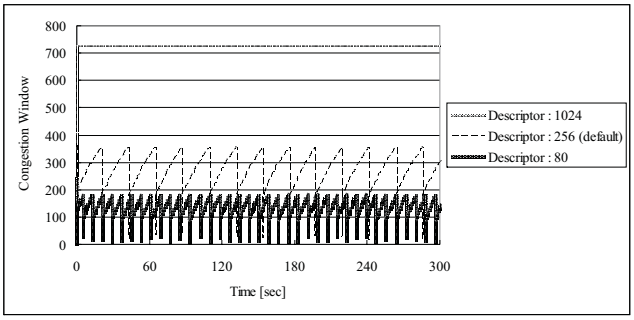

図 2　NIC ディスクリプタを変更した時の輻輳ウィンドウの時間変化

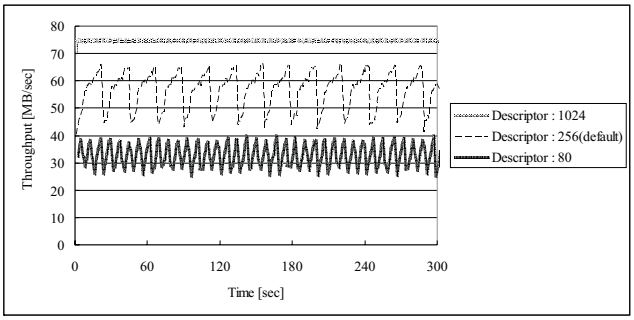

図 3　NIC ディスクリプタを変更した時のスループットの時間変化

3.2　実 験 概 要

Initiator から Target にシーケンシャルリードアクセスを行い，データ送信側である Target の NIC ディスクリプタを変更することで送信バッファサイズを変更し，その時の輻輳ウィンドウとスループットの変化を観察した．NIC ディスクリプタは NIC ドライバにおけるバッファの識別子数であり，これを変化させることにより，ドライバが利用できるバッファの量が変化する．この値は 80 から 4096 までの間で変更することができ，デフォルトは 256 に設定されている．本章の実験では，デフォルト値である 256 と，それよりバッファサイズを小さくした場合として 80，バッファサイズを大きくした場合として 1024 を設定して，各値における測定を行った．また，Initiator から Target に要求するブロックサイズは，すべて 1024KB に設定した．

3.3　実験結果と考察

前節の実験結果として図 2，図 3 を得た．本実験時には，頻繁に輻輳ウィンドウが低下する原因として，主に CWR が確認されている．CWR は送信側の NIC バッファが溢れることにより検出されるため，NIC ディスクリプタを大きく設定することでこのエラーは回避される．

これらの様子は図 2 から確認される．ディスクリプタ値が 80, 256 の時は，設定された NIC バッファサイズより送信ブロックサイズが大きいため，輻輳ウィンドウは定期的に大きく低下して鋸型の変化を示し，スループットも輻輳ウィンドウの変化に伴い増加減少を繰り返す．この時，NIC バッファサイズがより大きい 256 の方が，輻輳ウィンドウの限界値とスループットは大きくなる．

一方ディスクリプタ値を 1024 まで大きくすると，NIC バッファサイズが十分大きくなるため，送信時にバッファが溢れる事がなくなり，輻輳ウィンドウは一定値となる．また，この時スループットは一定値となり，安定した性能を示す．この場合は，送信バッファにまだ余裕があるため，現在アクセスしているブロックサイズよりも大きなブロックサイズでアクセスすることが可能であると考えられる．

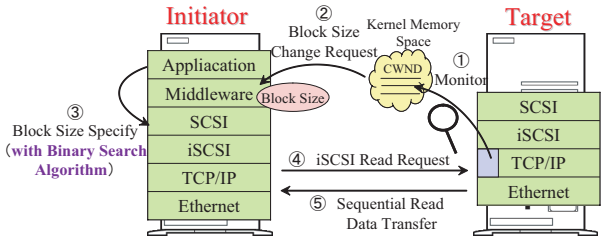

図 4　2 分探索アルゴリズムに基づく輻輳ウィンドウコントロール手法の概念図

4.　輻輳ウィンドウコントロール手法における制御アルゴリズム

本章では，新たに検討を行った制御アルゴリズムに基づき，輻輳ウィンドウの制御を行う手法の概要について述べる．

2 章において述べた手法は，Target からの通知を受けて，再指定するブロックサイズを単純に一定値ずつ増加あるいは低下させる線形探索アルゴリズムを用いた実装である．そこで，一般に線形探索アルゴリズムより効率の良いアルゴリズムとされている 2 分探索アルゴリズムと，Target が保持している輻輳ウィンドウ値から再指定するブロックサイズを決定する輻輳ウィンドウ値通知アルゴリズムの 2 種類を検討し，実装を行った．

4.1　2 分探索アルゴリズムに基づく輻輳ウィンドウコントロール手法

2 分探索アルゴリズムを用いる手法は以下の手順で行う．また，概念図を図 4 に示す．

（1）　Target で輻輳ウィンドウをモニタし，変化を観察する

（2）　観察中に CWR が検出され，輻輳ウィンドウが低下した場合にはブロックサイズの低下要求を，輻輳ウィンドウが一定値であると判断した場合には，ブロックサイズの増加要求を Initiator に行う

（3）　通知を受けた Initiator では，ミドルウェアが 2 分探索アルゴリズムに基づいてブロックサイズを決定し，アプリケーションがブロックサイズを再指定する

（4）　Initaitor から Target にシーケンシャルリードコマンドを送信し，ストレージアクセスを行う

（5）　Target が Initiator に向けて要求されたブロックサイズのデータ転送を実行する

これらの処理を CWR を検出するか，輻輳ウィンドウが一定値であると判断する度に繰り返す．この手順は，図 4 の③において Initiator で再指定するブロックサイズを，Target の指示により探索範囲を 2 分しながら目的とする値に近づけていく，2 分探索アルゴリズムにより決定するという以外は線形探索アルゴリズムに基づく手法とほぼ同様である．Target では Initiator に送信した要求内容と，CWR エラーにより輻輳ウィンドウが

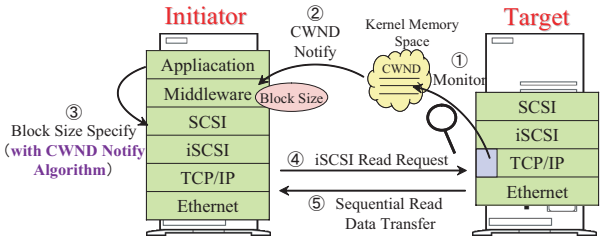

図 5　輻輳ウィンドウ値通知アルゴリズムに基づく輻輳ウィンドウコントロール手法の概念図

低下した場合の限界値を収束判断のために記録しておく．輻輳ウィンドウの収束判断は，輻輳ウィンドウが一定値となった時に，記録した輻輳ウィンドウと現在の輻輳ウィンドウを比較し，その差が十分小さい時に収束したと判断し，この時本手法から計算される最適ブロックサイズとなる．

4.2　輻輳ウィンドウ値通知アルゴリズムに基づく輻輳ウィンドウコントロール手法

輻輳ウィンドウ値通知アルゴリズムを用いる手法は以下の手順で行う．また，概念図を図 5 に示す．

（1） Target で輻輳ウィンドウをモニタし，変化を観察する

（2） 観察中に CWR が検出され，輻輳ウィンドウが低下した場合にはその時の輻輳ウィンドウ値を，輻輳ウィンドウが一定値であると判断した場合には輻輳ウィンドウの限界値を Initiator に通知し，Target でも通知した輻輳ウィンドウ値を記録する

（3） 通知を受けた Initiator では，ミドルウェアが輻輳ウィンドウからブロックサイズを決定し，アプリケーションがブロックサイズを再指定する（ブロックサイズの計算方法は後述する）

（4） Initaitor から Target にシーケンシャルリードコマンドを送信し，ストレージアクセスを行う

（5） Target が Initiator に向けて要求されたブロックサイズのデータ転送を実行する

これらの処理を CWR を検出するか，輻輳ウィンドウが一定値であると判断する度に繰り返す．この手法では線形探索アルゴリズム，2 分探索アルゴリズムを用いた手法と異なり，Target において輻輳ウィンドウを記録することで，Initiator のみが保持するブロックサイズを Target でも確認できる．そのため，この情報を基に Initiator の状態を把握し，ブロックサイズ変更の指示を出すことが可能となる．ブロックサイズは以下の式から計算することができる．

転送ブロックサイズ $[byte]$ = 輻輳ウィンドウ値×最大転送単位（MTU）

本実験時の MTU（Maximum Transmission Unit）は Ethernet の最大セグメント長（1500Byte）から TCP/IP ヘッダ（オプションを含む）を除いた 1448Byte で計算を行った．

Target では，輻輳ウィンドウが低下した時の輻輳ウィンドウ値から計算したブロックサイズの最大値（MaxBlockSize と呼ぶ）と，輻輳ウィンドウが一定値となったときの輻輳ウィンドウ値から計算したブロックサイズの最小値（MinBlockSize と呼ぶ）を保持する．この時輻輳ウィンドウが一定値となるための最適ブロックサイズは，MaxBlockSize と MinBlockSize の間に存在することになる．そのため，輻輳ウィンドウが一定値であると判断する度に，MaxBlockSize，MinBlockSize と輻輳ウィンドウから計算される現在の転送ブロックサイズを比較し，それらの差をとることによって輻輳ウィンドウが収束しているか否かを判断する．Target で収束と判断した場合には Initiator に通知を行わず，この時，本手法から計算される最適ブロックサイズと判断する．

5.　提案アルゴリズムを用いた輻輳ウィンドウコントロール手法の実験

本章では，2 章，4 章で紹介した線形探索法，2 分探索法，輻輳ウィンドウ通知法の各アルゴリズムを実装したマシンにおいて，輻輳ウィンドウをコントロールした実験を行い，その性能について評価する．

5.1　実 験 概 要

各アルゴリズムによる輻輳ウィンドウコントロール手法を実装した場合のストレージアクセスの性能を測定するため，Initiator から Target にシーケンシャルリードアクセスを行い，輻輳ウィンドウ，ブロックサイズ，スループットの測定を行った．実験は，3.1 節で述べた環境を用い，設定が異なる環境での実験を行うため，ブロックサイズの初期値，NIC ディスクリプタを変更し，この時の振る舞いを考察した．ブロックサイズの初期値は 128KB と 1024KB とし，NIC ディスクリプタはデフォルト値である 256 と，より大きな値である 1024 を設定し，これらの組み合わせで測定した．

5.2　NIC ディスクリプタ 256 における実験結果

前節の実験結果として，NIC ディスクリプタを 256 に設定した場合の各アルゴリズムにおける測定結果を示す．図 6，図 7，図 8 は，ブロックサイズの初期値を 128KB とした場合の各コントロール手法における実験結果である．

この場合，ブロックサイズの初期値が小さいため，送信側バッファが溢れることはない．従って，各手法において最初のストレージアクセス時には輻輳ウィンドウが一定値となるため，Target からブロックサイズの増加命令が発行され，各アルゴリズムに従って制御される．その後輻輳ウィンドウは，限界値で一定となる．

次にブロックサイズの初期値を 1024KB とした場合の各コントロール手法における実験結果を，図 9，図 10，図 11 に示す．

この場合は，ブロックサイズが大きすぎてバッファが溢れてしまうため，Target からブロックサイズの減少命令が発行される．その後，各アルゴリズムにより制御され，最適ブロックサイズに収束する．

5.3　NIC ディスクリプタ 1024 における実験結果

本節では，NIC ディスクリプタを 1024 に設定した結果を示す．ブロックサイズの初期値を 128KB にした場合の結果が，図 12，図 13，図 14，ブロックサイズの初期値を 1024KB にした

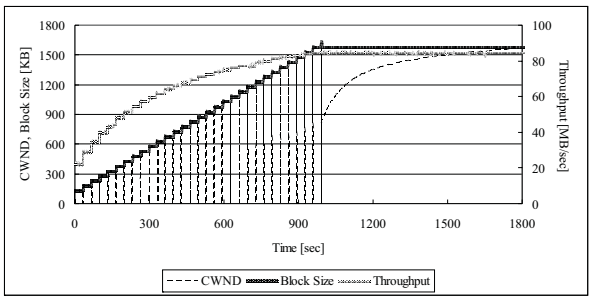

図 6　線形探索アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 128KB）

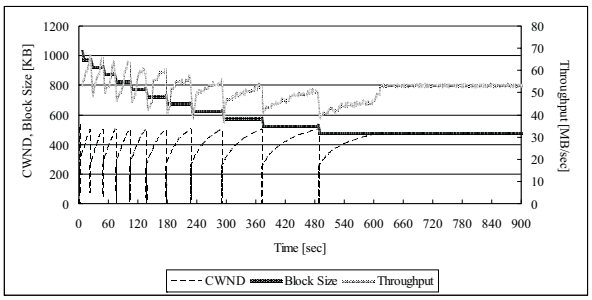

図 9　線形探索アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 1024KB）

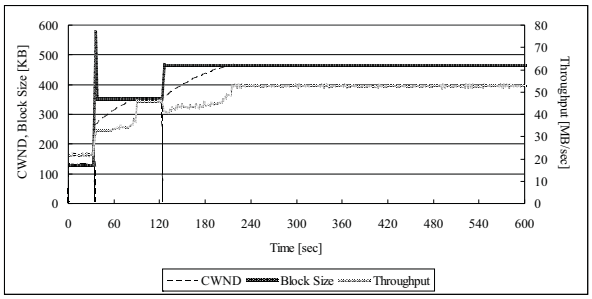

図 7　2 分探索アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 128KB）

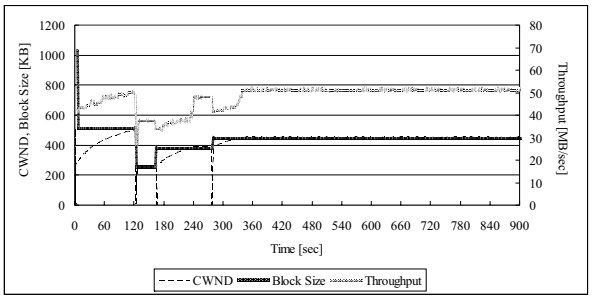

図 10　2 分探索アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 1024KB）

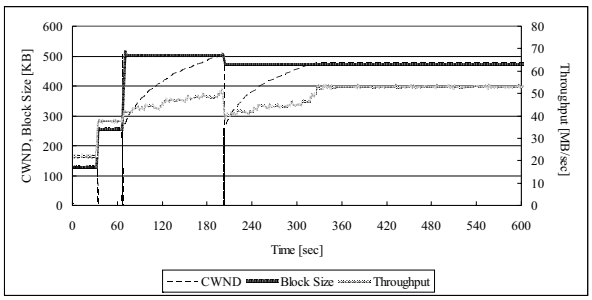

図 8　輻輳ウィンドウ通知アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 128KB）

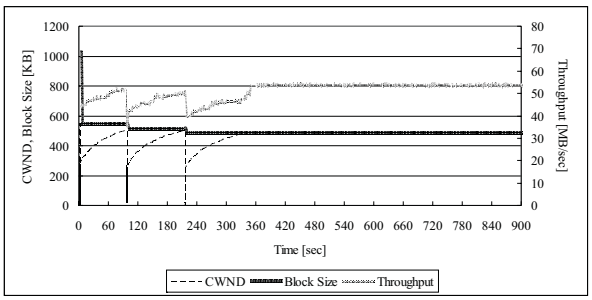

図 11　輻輳ウィンドウ通知アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 1024KB）

場合の結果が，図 15，図 16，図 17 である．

どの場合においても，NIC バッファが十分大きいため，実験を行ったブロックサイズの初期値では溢れる事はなく，ブロックサイズを増加させていく制御となる．そのため，Target から最初にブロックサイズの増加命令が発行され，各アルゴリズムによって制御される．なお，図 17 において，輻輳ウィンドウが一定値となった後，急激な低下を繰り返しているが，これは遅延装置の限界によるものである．

5.4 考　　察

線形探索アルゴリズムに基づく手法では，Target からの命令に従い，単純にブロックサイズの増加または減少を繰り返すため，前節および前々節の殆どの実験結果において，輻輳ウィンドウが収束するまでに最も時間がかかっている．それに対し，本稿で新たに提案した 2 分探索アルゴリズムに基づく手法と，輻輳ウィンドウ通知アルゴリズムに基づく手法は，比較的早期に収束可能な手法であると言える．輻輳ウィンドウが低下した場合，その回復速度は TCP 実装に依存する．Linux の TCP 実装においては，収束が近づくにつれより回復速度が低下する傾向が見られるため，2 分探索アルゴリズムでは，設定するパラメータによって図 16 のように収束時間が遅くなってしまうという場合がある．また，輻輳ウィンドウ通知アルゴリズムに基づく手法の場合にも，Target から通知された輻輳ウィンドウを基に計算したブロックサイズが必ずしも最適なブロックサイズであるということではないため，収束までに何度か輻輳ウィンドウが低下することがある．これらのアルゴリズムを比較すると，2 分探索アルゴリズムに基づく手法の場合，アルゴリズムの性質から図 10 のように輻輳ウィンドウが収束するまでの間に，収束値よりも小さなブロックサイズでストレージアクセスを行う場合が存在する．それに対し輻輳ウィンドウ通知アルゴリズムに基づく手法では，一度 CWR エラーを検出するとその時のブロックサイズから比較的近いブロックサイズで収束すると考えられるため，大幅なブロックサイズの低下は起こらない．これらの理由から総合的に判断した場合，我々が提案する輻輳

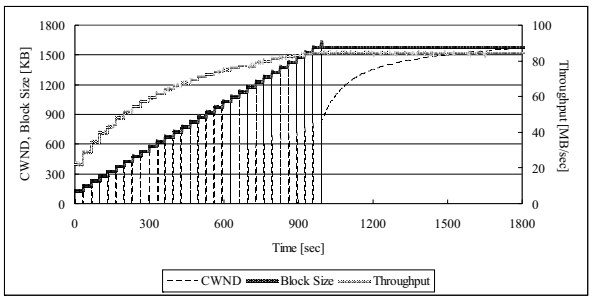

図 12　線形探索アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 128KB）

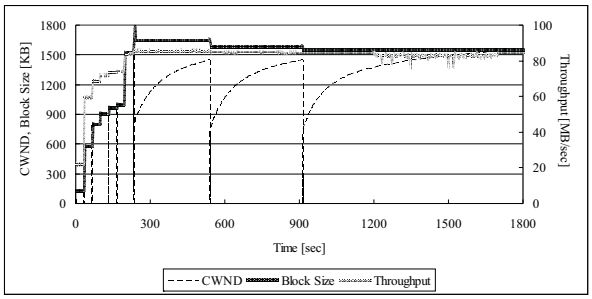

図 13　2 分探索アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 128KB）

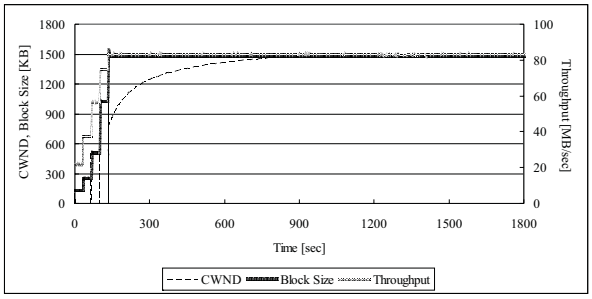

図 14　輻輳ウィンドウ通知アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 128KB）

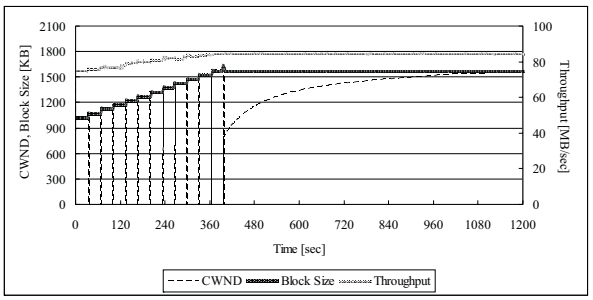

図 15　線形探索アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 1024KB）

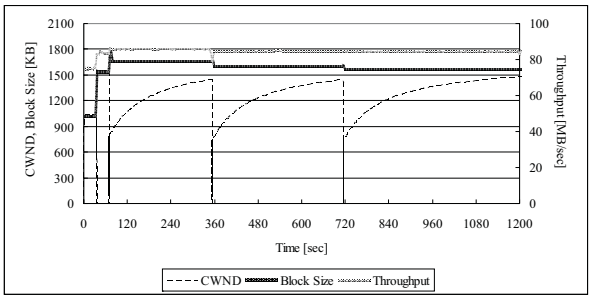

図 16　2 分探索アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 1024KB）

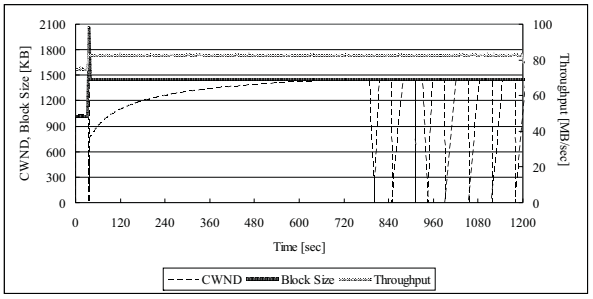

図 17　輻輳ウィンドウ通知アルゴリズムを用いた場合の輻輳ウィンドウ，ブロックサイズ，スループット時間変化（ブロックサイズ 1024KB）

ウィンドウコントロール手法においては，輻輳ウィンドウ通知アルゴリズムが最適なアルゴリズムであると言える．

なお，各アルゴリズムにおいて，増加命令発行時に輻輳ウィンドウが急激に低下しているが，これは一時的な低下であり，時間にすると数百マイクロ秒程度であるため，性能にはほとんど影響を及ぼさないと考えられる．

6. NIC ディスクリプタ，輻輳ウィンドウ，ブロックサイズの関係

本章では，実験から得られた結果を基に，NIC ディスクリプタ，輻輳ウィドウ限界値，最適ブロックサイズについてモデル化を行い，これらの関係についての詳細を述べる．

本実験のように，Initiator と Target 間に遅延が存在する通信環境を想定する．この環境においては，Initiator からの ACK が返信されるまでに時間がかかるため，ブロックサイズを増加させると輻輳ウィンドウは使い切られ，ACK を受信する度に輻輳ウィンドウの値が大きくなるが，やがて CWR による限界値に達する．この時ブロックサイズは最適値である．すなわち，輻輳ウィンドウの限界値は最適ブロックサイズに比例する．一方，ブロックサイズが大きくなると一度に送信するデータが増えるため，ディスクリプタにより指定された値を越えると NIC バッファが溢れてしまう．従って，最適ブロックサイズは NIC バッファにより決定される．これらの事から，NIC ディスクリプタ，輻輳ウィンドウ，ブロックサイズには以下のような関係があると考えることができる．

最適転送ブロックサイズ $= m \times NIC$ ディスクリプタ
（m：正数）

輻輳ウィンドウ限界値 $= n \times$転送ブロックサイズ
（n：正数）

上記モデリング結果を検証するため，3.1 節の実験環境を用い，NIC ディスクリプタを変更した場合に輻輳ウィンドウが鋸型の変化を繰り返す時の最大ブロックサイズと，その時の輻輳

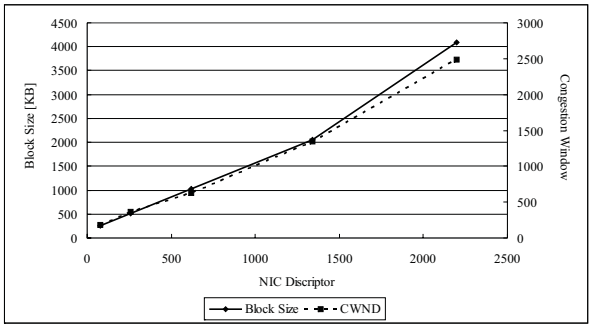

図 18　NIC ディスクリプタ，輻輳ウィンドウ限界値，最適ブロックサイズの関係

ウィンドウ限界値の測定を行った．結果を図 18 に示す．

図 18 から，NIC ディスクリプタとブロックサイズ，NIC ディスクリプタと輻輳ウィンドウには線型の関係があることが確認できる．このことから，遅延が存在する通信環境においては，NIC ディスクリプタ，輻輳ウィンドウ，ブロックサイズには上記関係式が成り立つと言える．

7. 関 連 研 究

iSCSI の関連研究としては，文献 [9], [10], [11], [12] などが挙げられる．文献 [9] は，Sarkar らが iSCSI のソフトウェア実装とハードウェア実装を比較し，文献 [10] では，Radkov が iSCSI と NFS の性能を比較している．しかし，これらは iSCSI 利用時のシステム内部の振る舞いについて把握して性能評価を行っているものではない．また，文献 [11] は，iSCSI の性能が，複数の階層構造を持つことやプロセスの重複により低下すると指摘し，それらを解決する独自の提案を行っている．iSCSI 性能低下の原因の着眼点は同じであるが，性能向上のアプローチが異なる．文献 [12] は，iSCSI Target の内部実装を考慮し iSCSI の性能評価を行うものである．既存のソフトウェアには基本的に変更を加えず，ミドルウェアで容易に対応できる我々とは手法が異なるものである．

文献 [13] や文献 [14] は，既存の TCP に変わる新しい TCP として提案されている技術に注目し，性能評価や通信時の輻輳ウィンドウの振る舞いについて述べたものである．輻輳ウィンドウがシステム性能に関連していることについて議論されている点では共通しているが，これらの文献で述べられている技術は既存の TCP を改良することで性能向上を行っているものであり，TCP に手を加えればその分だけ汎用性は低くなる．従って，広く一般的に用いられている既存の TCP をそのまま利用し，その上での性能向上を目的とする本研究とは異なったものである．

8. ま と め

本稿では，iSCSI ストレージアクセス時に確認されるスループットのばらつきを抑制するために提案した輻輳ウィンドウコントロール手法において，より早く輻輳ウィンドウを一定値に収束させるためのアルゴリズムの検討を行った．新たに 2 分探索アルゴリズムに基づく手法と，輻輳ウィンドウ値通知アルゴリズムに基づく手法を提案して実装し，既存の線型探索アルゴリズムに基づく手法との比較を行った．3 つのアルゴリズムを実装したマシンにおいて，iSCSI シーケンシャルリードアクセスを行い，パラメータを変化させた各環境における輻輳ウィンドウ，ブロックサイズ，スループットの時間変化を考察した．また，NIC ディスクリプタを変更することによる輻輳ウィンドウの振る舞いについて述べ，実験結果から NIC ディスクリプタ，輻輳ウィンドウ限界値，最適ブロックサイズの関係式を示した．今後は本手法を高遅延環境に適用し，その性能評価を行いたい．

謝　　辞

本研究は，一部，文部科学省科学研究費特定領域研究課題番号 13224014 によるものである．

文　　献

[1] iSCSI Specification, http://www.ietf.org/rfc/rfc3720.txt?number=3270/
[2] SCSI Specification, http://www.danbbs.dk/~dino/SCSI/
[3] 山口実靖，小口正人，喜連川優：“ 高遅延広帯域ネットワーク環境下における iSCSI プロトコルを用いたシーケンシャルストレージアクセスの性能評価ならびにその性能向上手法に関する考察 ”，電子情報通信学会論文誌 Vol.J87-D-I, No.2, pp.216-231, February 2004.
[4] 豊田真智子，山口実靖，小口正人：“ iSCSI ストレージアクセス時における TCP 輻輳ウィンドウとシステム性能の関連性評価 ”，FIT2004 第 3 回情報科学技術フォーラム，B-004，pp.107-109, September 2004.
[5] 豊田真智子，山口実靖，小口正人：“ iSCSI アクセス時の TCP 輻輳ウィンドウ制御を用いたシステム性能向上手法の一検討 ”，電子情報通信学会技術研究報告，CPSY2004-50，pp.1〜6，December 2004.
[6] 豊田真智子，山口実靖，小口正人：“ iSCSI ストレージアクセス時の TCP フロー制御のリアルタイム可視化 ”，電子情報通信学会総合大会，B-16-9，pp.618，March 2004.
[7] L.Rizzo: “ dummynet ”, http://info.iet.unipi.it/~luigi/ip_dummynet/
[8] InterOperability Lab: Univ, of New Hampshire, http://www.iol.unh.edu/consortiums/iscsi/
[9] P.Sarkar，S.Uttamchandani，and K.Voruganti：“ Storage over IP: When Does Hardware Support help?, ”*Proc. FAST 2003*，*USENIX Conference on File and Storage Technologies*，pp.231-244，January 2003.
[10] P.Radkov，L.Yin，P.Goyal，P.Sarkar and P.Shenoy：“ Performance Comparison of NFS and iSCSI for IP-Networked Storage, ”*Proc. FAST 2002*，*USENIX Conference on File and Storage Technologies*，pp.101-114，March 2004.
[11] P.Gurumohan，S.Narasimhamurthy，J.Hui：“ Quanta Data Storage: A New Storage Paradigm, ”*Proc. 12th NASA Goddard Conference on Mass Storage Systems and Technologies*，pp.101-107，April 2004.
[12] 藤田智成，小河原成哲：“ iSCSI ターゲットソフトウェアの解析 ”，先進的計算基盤システムシンポジウム SACSIS2004，pp.335-342，May 2004.
[13] 高野了成，石川裕，工藤知宏，松田元彦，児玉祐悦，手塚宏史：“ 並列アプリケーション実行における TCP/IP 通信挙動の解析 ”，IC2003，October 2003.
[14] 熊副和美，堀良彰，鶴正人，尾家祐二：“ JGN を利用した高速トランスポートプロトコルの評価 ”，電子情報通信学会技術研究報告，NS2003-354，IN2003-309，pp.303-308，March 2004.