複数台 iSCSI Initiator を用いた高遅延ネットワーク環境における TCP 輻輳ウィンドウ制御手法の性能評価

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　コンピュータシステムにおけるデータ量の増大に伴い，効率的にストレージを管理したいという要望が高まっている．iSCSI を用いることにより広域な IP-SAN が低コストで構築でき，データセンタなどの遠隔地にデータをバックアップすることが容易となるため，ストレージのアウトソーシングといったサービスへの利用が期待されている．iSCSI を用いたストレージアクセス時には TCP 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が密接に関連しているため，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のばらつきを抑制することが性能の向上につながる．

本稿では，複数台サーバからストレージにアクセスする環境を想定し，各サーバの輻輳ウィンドウを動的にコントロールして，スループットを均一化する手法を提案する．さらに，提案手法に改良を加えてスループットを向上する新たな手法も検討し，想定環境に双方の提案手法を適用し，iSCSI ストレージアクセスの性能評価を行った．
キーワード　iSCSI，輻輳ウィンドウ，遠隔ストレージ，複数台 iSCSI Initiator，動的コントロール

Performance Analysis of TCP Congestion Window Control Method using Multiple iSCSI Initiator in a Long-Latency Environment

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　As the volume of data computer systems process increases, it is important that storage is managed efficiently. Because iSCSI can configure the wide area IP-SAN with low cost and can make easy data backup of remote place, for example data center, iSCSI is expected to be used as Storage Outsoursing service. Throughput has a close relationship with the size of Congestion Window of TCP parameter on storage access using iSCSI. Therefore it is important to control Congestion Window in dynamic and blance throughput unevenness.

In this paper, we suppose an environment in which a storage is accessed from multiple servers and propose the method of controlling each server's Congestion Window in dynamic. Moreover, we have examined a new method that improves the first proposed method, and evaluated the performance of iSCSI Storage Access using two proposed methods in the supposed environment.
Key words　iSCSI, Congestion Window, Remote Storage, Multiple iSCSI Initiator, Dynamic Control

1. ま え が き

光ファイバなどを用いた高速なブロードバンドネットワークが企業や一般家庭で容易に利用可能となり，ストレージ内のデータをネットワークを介した遠隔地へバックアップすることが多くなってきている．ストレージ利用効率の向上や，管理コストの削減を目的としたストレージ統合に SAN(Storage Area Network) が広く用いられ，その実績は高い評価を得ている．

SAN はファイバチャネルを用いて構築する FC-SAN と，Ethernet 及び TCP/IP を用いて構築する IP-SAN に大別される．

FC-SAN は現世代の SAN とも言われ，現在企業などでも広く利用されている．プロトコル処理が軽く，サーバの CPU にかかるデータ転送負荷が小さいという利点を持っているが，接続距離に制限があったり，FC 製品が高価であることにより導入コスト，管理コストが増加するため，気軽に導入することが難しい．そのため，安価なコストで構築可能な IP-SAN が次世代 SAN として注目され，2003 年 2 月に IETF により承認された iSCSI が，IP-SAN の標準的なデータ転送プロトコルとして利用されている [1] [2].

iSCSI は，これまで DAS(Direct Attached Storage) で利用されてきた SCSI コマンドを，TCP/IP パケット内にカプセル化することにより，サーバ (Initiator) とストレージ (Target) 間でデータの転送を行う．しかし，SCSI over iSCSI over TCP/IP over Ethernet という複雑なプロトコルスタック構成のオーバーヘッドなどが影響し，iSCSI による通信は大幅に性能が劣化することがわかっている [3].

我々はこれまで，TCP パラメータである輻輳ウィンドウを動的にコントロールする輻輳ウィンドウコントロール手法を提案し，Initiator と Target を 1 対 1 で接続した環境における iSCSI ストレージアクセスの性能向上について検討してきた [4]．そこで本稿では，これまでの環境を複数台の Initiator から Target にストレージアクセスする環境に発展させ，各 Initiator の輻輳ウィンドウを動的にコントロールする複数台 Initiator 輻輳ウィンドウコントロール手法 I を提案する．iSCSI を用いたストレージアクセスとしては，遠隔バックアップなどで利用されるシーケンシャルリードアクセスを取り上げる．提案手法 I を想定環境に適用し，遅延時間を変化させて性能評価を行う．また，提案手法 I に改良を加えて性能を向上させた複数台 Initiator 輻輳ウィンドウコントロール手法 II も提案し，同様の環境で性能評価を行う．さらに実験結果から，提案手法を用いなかった場合と用いた場合の考察を行い，本手法の有効性を確認する．

本稿は以下のように構成される．まず 2 節で研究背景を述べる．3 節では，複数台の Initiator を対象とした輻輳ウィンドウコントロール手法 I について紹介し，4 節で提案手法 I を用いた場合と用いなかった場合それぞれの環境で iSCSI ストレージアクセスを行い，提案手法の性能を評価する．5 節では提案手法 I を改良した，複数台 Initiator 輻輳ウィンドウコントロール手法 II の概要を述べ，6 節において提案手法 II の性能を評価する．7 節で各実験結果から詳細な考察を行い，8 節で関連研究について触れる．最後に 9 節でまとめを述べる．

2. 研 究 背 景

2.1 輻輳ウィンドウ

TCP では，輻輳制御において輻輳ウィンドウという概念を用いている．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自主的に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パケット (= ACK) なしに連続送信を行う最大パケット数である．通常，輻輳ウィンドウは ACK を 1 つ受信するごとに増加する．通信時の状態が正常であれば ACK 受信ごとに輻輳ウィンドウは増加するが，エラーが検出

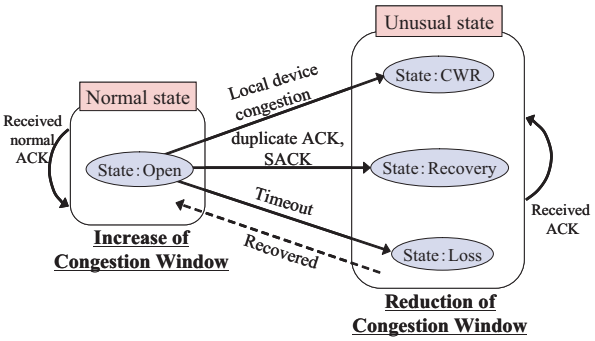

図 1　Linux TCP 実装の状態遷移図

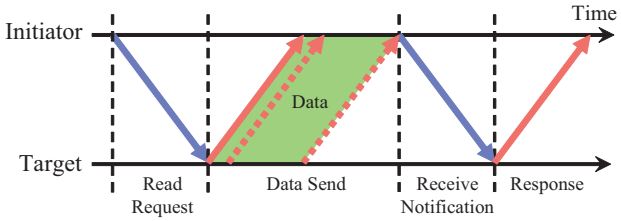

図 2　iSCSI シーケンシャルリードアクセスのシーケンス

されると異常と判断され，輻輳ウィンドウは減少する（図 1）．輻輳ウィンドウが低下する原因としては，送信側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 つが挙げられる．また，Linux の TCP 実装では，通信中に一度設定された輻輳ウィンドウは，そのウィンドウの値を使い切らない限りは変化しないという特徴を持ち，この時スループットはほぼ一定の値で安定することが確認されている [5].

2.2 複数台 Initiator を用いた iSCSI ストレージアクセスにおける課題

TCP/IP のみで構成されたネットワークにおいてソケット通信を行っている場合（以降，ソケット通信と呼ぶ），TCP のセルフクロッキング機能が時間の経過と共に動作し，ACK の受信タイミングに合わせてデータが少しずつ送信されるため，パケットの送出速度が適切に調節される．一方 iSCSI を用いた通信では，常に Read リクエストを受信した後にデータが一斉送信されるという特徴があるため，いつまでたってもバーストが消えることがない (図 2)．そのため，ソケット通信に比べ CWR エラーが起こりやすくなり，輻輳ウィンドウの成長も小さい．図 3 は，サーバとストレージが 1 対 1 で接続された環境において，片道遅延時間が 16ms の場合のソケット通信と iSCSI 通信の輻輳ウィンドウの時間変化を示したものである．この図において，輻輳ウィンドウの低下はすべて CWR エラーによるものである．ソケット通信の輻輳ウィンドウは，変動はあるものの比較的大きな値まで成長するのに対し，iSCSI 通信の輻輳ウィンドウはソケット通信よりも成長が小さいことが確認できる．また，我々は文献 [5] において，輻輳ウィンドウとスループットが密接に関連しており，輻輳ウィンドウが増加減少の鋸型の変化を繰り返す場合には，輻輳ウィンドウの振舞に伴いスループットも不安定になることを述べた．iSCSI を利用したストレージアクセ

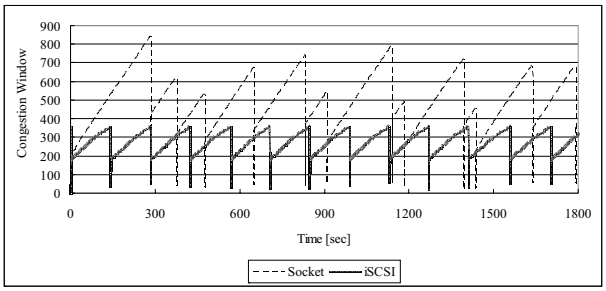

図 3　ソケット通信と iSCSI 通信における輻輳ウィンドウの時間変化

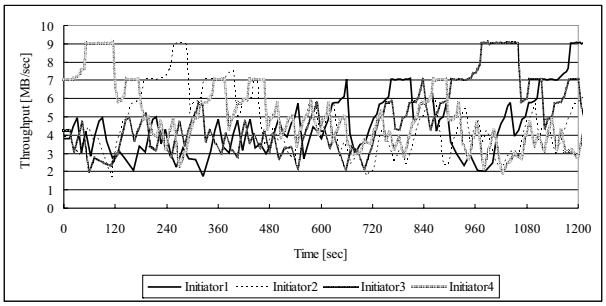

図 4　複数台 Initiator を用いた場合のスループットの時間変化

スを行う場合は，CWR エラーの頻度を減らし，輻輳ウィンドウを大きな値で保つことが性能向上につながる．

図 4 は，片道遅延時間 16ms において，4 台の Initiator を用いて iSCSI ストレージアクセスを行った場合の各 Initiator におけるスループットの時間変化を示している．どの Initiator のスループットも上下に大きくばらついており，ある 1 台が高い性能を示す場合，残りの 3 台の性能が低下する様子が確認される．また，ある時間においてどの Initiator の性能が高くなるかは不規則なため，各 Initiator の性能差が大きくなる場合がある．効率的なストレージアクセスを実現するためには，各 Initiator から Target へのアクセスを均一化し，性能を安定させることが重要であると言える．

3. 複数台 Initiator 輻輳ウィンドウコントロール手法 I：スループット均一化手法

本節では，iSCSI ストレージアクセスにおいて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複数台 Initiator 輻輳ウィンドウコントロール手法 I について述べる．

前節で述べたように，iSCSI を用いたネットワークにおけるパケットの振舞は，TCP/IP のみで構成されたネットワークの振舞と異なるため，性能を向上させるためには iSCSI の振舞を考慮した特別な処理が必要である．また，Initiator を複数台用いることによって，各 Initiator の性能差にばらつきができてしまうため，Initiator と Target を 1 対 1 に接続した環境とは異なった処理が必要となる．そこで我々は，Initiator と Target を 1 対 1 に接続した環境を想定して，文献 [4] において提案した輻輳ウィンドウコントロール手法を改良し，複数台の Initiator からのアクセスを想定した多対 1 接続環境に対応させて，輻輳ウィンドウをコントロールする手法を提案する．本手法の概要を図 5 に示す．

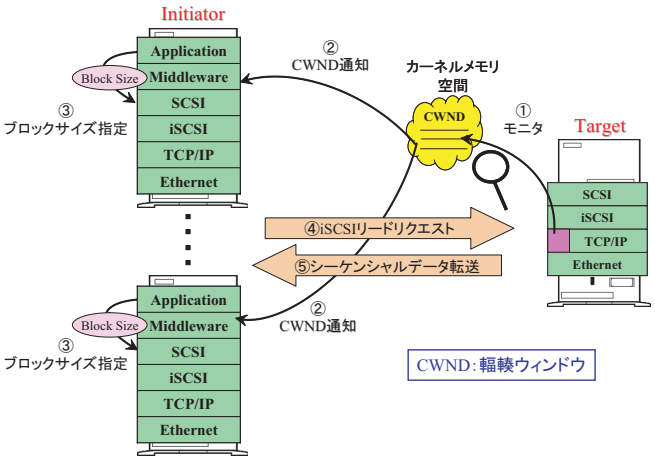

図 5　提案手法 I の概念図

輻輳ウィンドウはカーネル空間において管理される TCP パラメータであるため，通常のユーザプログラムでその値を知ることはできない．そこで，TCP ソースコードにモニタ用の関数を挿入し，ユーザ空間からもアクセス可能なカーネルメモリ空間に記録する仕組みを作成した．これにより，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って，輻輳ウィンドウなどの TCP パラメータを確認することが可能となる．提案手法は，この仕組みを Target に実装し，Target からの輻輳ウィンドウ通知を受けて Initiator のアプリケーションがストレージアクセスのブロックサイズを調節する仕組みをミドルウェアとして提供する機能を持つ．輻輳ウィンドウの収束は iSCSI アクセスにおけるブロックサイズを調節することで行うものとする．

Initiator を複数台用いた場合，送信バッファはコネクション数に応じ分割して割り当てられ，輻輳ウィンドウはコネクションごとに設定されることが確認されている [6]．本手法では，モニタしているその他の TCP パラメータを用いることにより，各 Initiator の輻輳ウィンドウを識別して取り出している．これにより，Target は各 Initiator の振舞を一括して管理できるため，より効率的な制御を行うことができる．本手法による輻輳ウィンドウの制御手順を以下に示す．

（1） Target で各 Initiator の輻輳ウィンドウをモニタし，変化を観察する．

（2） 輻輳ウィンドウの振舞によって以下の異なる処理を行う．

（a） 観察している Initiator のいずれかで CWR が検出され，輻輳ウィンドウが低下した場合

- エラーが検出された Initiator の輻輳ウィンドウの最大値 (低下する直前の最大の輻輳ウィンドウ) を Initiator ごとに記録し，各 Initiator の輻輳ウィンドウ最大値の中で最も小さな値をすべての Initiator に通知する．
- 通知した輻輳ウィンドウ値を Target においても記録する．

（b） すべての Initiator の輻輳ウィンドウが同じ値で一定値であると判断した場合

表 1　使用計算機

CPU	Initiator: Intel PentiumIII 800MHz Target, Dummynet: Intel Xeon 2.4GHz
Main Memory	Initiator: 640MB Target, Dummynet: 512MB
OS	Initiator, Target: Linux2.4.18-3 Dummynet: FreeBSD 4.9 - RELEASE
NIC	Initiator, Dummynet: Intel PRO/1000MT Server Adapter Target: Intel PRO/1000XT Server Adapter

- その時の輻輳ウィンドウの限界値 (CWR エラーが起こらなかった場合の最大値) をすべての Initiator に通知し，通知した輻輳ウィンドウ値を Target においても記録する．

（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．

（4） Initiator 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．

（6） この処理を，すべての Initiator の輻輳ウィンドウが同じ値で一定値と判断され，最大値と限界値の差が十分小さくなるまで繰り返す．

本手法適用後，すべての Initiator の輻輳ウィンドウは CWR エラーが起こらない限界値で一定に保たれ，その時のブロックサイズが本手法から計算される最適値となる．なお，本手法においてミドルウェアが指定するブロックサイズは以下の式を用いて計算した．

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（1500Bytes）から TCP/IP ヘッダ（オプションを含む）を除いた 1448Bytes である．

4. 提案手法 I の性能測定実験

本節では，提案手法 I を実装した環境において，遅延時間が異なる場合の性能を測定するため，iSCSI ストレージアクセスにおいて提案手法 I を用いた場合と用いない場合の実験を行う．

4.1 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．遅延がない場合の実験には 1000Base-T スイッチングハブを，遅延がある場合の実験には人工的な遅延装置である FreeBSD Dummynet [7] を Ethernet の接続途中に挿入した．iSCSI を利用したストレージアクセスにおけるネットワークの性能を調べるため，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されてしまい，これにより iSCSI ストレージアクセスの性能が大きく低下してしまうことが確認されている [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，Initiator に 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も行える自作 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．また，ブロックサイズが同じであれば UNH 実装の Initiator とほぼ同じスループットを示すため，自作 Initiator 使用による違いは十分小さいものであり，性能に影響を及ぼさないものであると言える．

4.2 実験概要

4 台の Initiator から Target へシーケンシャルリードアクセスを行い，遅延時間を変化させた場合の性能測定を行う．この時 raw デバイスを用いることによりキャッシュの影響を排除した．Initiator が指定するブロックサイズは，提案手法を用いない場合は 1024KB に，提案手法を用いた場合はその初期値を 1024KB に設定した．また，Target から読み込むデータサイズの合計は 10GB として実験を行った．

4.3 実験結果

前節の実験を行った結果として，片道遅延時間 16ms(Round Trip Time: 32ms) において，提案手法 I を用いずに iSCSI シーケンシャルリードアクセスを行った場合の輻輳ウィンドウ，スループットの時間変化を図 6，7 に，4 台中 1 台の Initiator において輻輳ウィンドウとスループットを比較した時間変化グラフを図 8 に示す．また，提案手法 I を用いた場合の輻輳ウィンドウ，スループットの時間変化を図 9，10 に，4 台中 1 台の Initiator において輻輳ウィンドウとスループットを比較した時間変化グラフを図 11 に示す．これらの図における輻輳ウィンドウ低下の原因はすべて CWR エラーによるものである．

提案手法 I を用いない場合，各 Initiator のスループットが不安定であり，輻輳ウィンドウも不規則な増加減少を繰り返していることが確認される (図 6，7)．また，輻輳ウィンドウが大きく成長した場合にはスループットは高くなるが，あまり成長せず小さな値である場合にはスループットは低いことがわかる (図 8)．1 回のシーケンシャルリードアクセスにおけるアクセスブロックサイズは大きいが，各 Initiator へのデータ転送は不均一であり，その性能差は大きい．

一方提案手法 I を用いたストレージアクセスの場合，Target において CWR エラー検出の度に Initiator のミドルウェアが機能し，アクセスブロックサイズを調節することにより輻輳ウィンドウをコントロールする．ブロックサイズがやや低い値に設定されるため性能は少し低下するが，最終的に CWR エラーは回避され，輻輳ウィンドウが一定値となる (図 9)．その結果ス

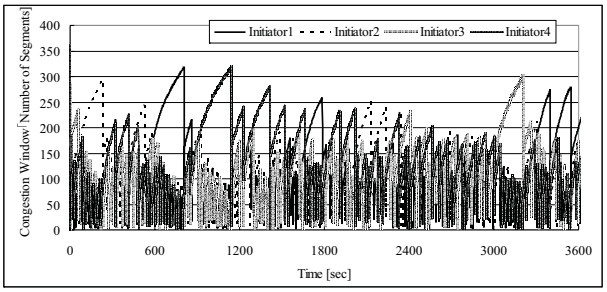

図 6　提案手法を用いない場合の輻輳ウィンドウの時間変化

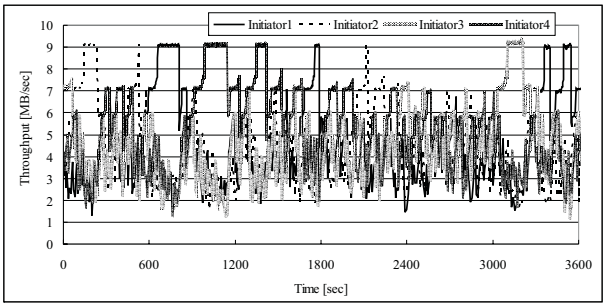

図 7　提案手法を用いない場合のスループットの時間変化

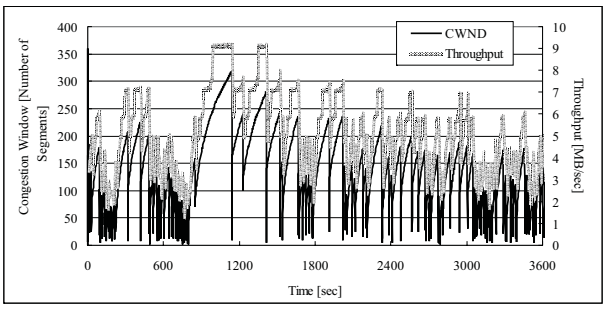

図 8　提案手法を用いない場合の輻輳ウィンドウ，スループットの時間変化

ループットは安定し，各 Initiator の性能差はほとんどみられない (図 10)．そのため，各 Initiator へのデータ転送は均一化され，安定した通信を行うことが可能となる．

5.　複数台 Initiator 輻輳ウィンドウコントロール手法 II：スループット均一化 + 性能向上手法

本節では，前々節で提案した複数台 Initiator 輻輳ウィンドウコントロール手法 I に改良を加えることにより，スループットのばらつき抑制と性能向上を目的とした複数台 Initiator 輻輳ウィンドウコントロール手法 II を提案し，その概要を述べる．

前々節で提案した複数台 Initiator 輻輳ウィンドウコントロール手法 I を用いることにより，Target から各 Initiator へのデータ転送を均一化し，信頼性の高いストレージアクセスを行うことができる．しかし，Target における CWR エラー検出の度に，単純にアクセスブロックサイズを低下させるという実装であるため，安定した通信を行うことができる一方，エラー頻度が高い場合にはアクセスブロックサイズが小さくなり，結果としてスループットが低下してしまう．

ブロックサイズは一度に送信するデータ長であるため，この

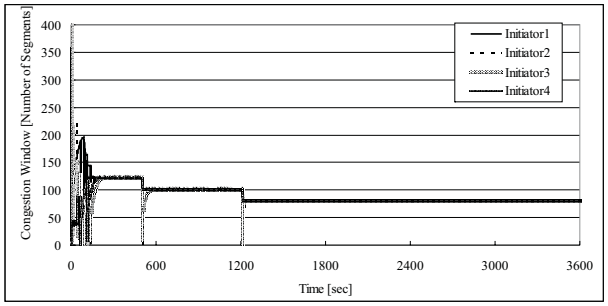

図 9　提案手法 I を用いた場合の輻輳ウィンドウの時間変化

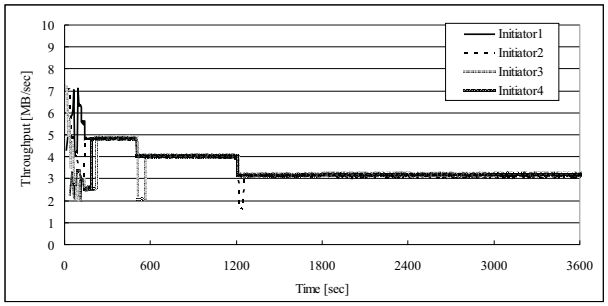

図 10　提案手法 I を用いた場合のスループットの時間変化

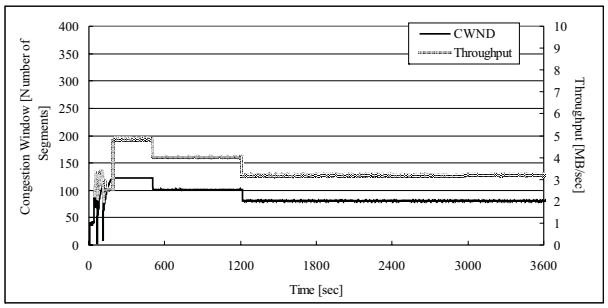

図 11　提案手法 I を用いた場合の輻輳ウィンドウ，スループットの時間変化

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，ブロックサイズをあまり大きくしすぎると送信バッファが溢れてしまい，CWR エラーを検出して輻輳ウィンドウが低下する．そのため，ブロックサイズと輻輳ウィンドウ双方のバランスを保ちつつ，スループットを向上することが重要となる．また，輻輳ウィンドウが一定値となる時，スループットも安定することが確認されている [5]．そこで，前節の実験結果を考慮し，CWR エラーは検出されるが，輻輳ウィンドウが一定値となった場合には最適であると判断し，それ以降エラーが検出されてもブロックサイズの減少命令を発行しない，複数台 Initiator 輻輳ウィンドウコントロール手法 II を検討した．新たに検討した提案手法 II は以下のように動作する．

（1）　Target で各 Initiator の輻輳ウィンドウをモニタし，変化を観察する．

（2）　輻輳ウィンドウの振舞によって以下の異なる処理を行う．

（a）　観察している Initiator のいずれかで CWR が検出され，輻輳ウィンドウが低下した場合

- それまでにすべての Initiator の輻輳ウィンドウが

同じ値で一定値となったと判断していれば，最適であるとして Initiator への通知は行わない

- すべての Initiator が一定値であることを判断していない場合には，エラーが検出された Initiator の輻輳ウィンドウの最大値 (低下する直前の最大の輻輳ウィンドウ) を Initiator ごとに記録し，各 Initiator の輻輳ウィンドウ最大値の中で最も小さな値をすべての Initiator に通知し，通知した輻輳ウィンドウ値を Target においても記録する

（b） 一度も CWR エラーを検出せずに，すべての Initiator の輻輳ウィンドウが同じ値で一定値であると判断した場合

- その時の輻輳ウィンドウの限界値 (CWR エラーが起こらなかった場合の最大値) をすべての Initiator に通知し，通知した輻輳ウィンドウ値を Target においても記録する．

（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．

（4） Initiator 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．

（6） この処理を，すべての Initiator の輻輳ウィンドウが同じ値で一定値と判断され，最大値と限界値の差が十分小さくなるまで繰り返す．

本手法適用後，CWR エラーは検出されるが輻輳ウィンドウはほぼ一定の値に保たれ，その時のブロックサイズが本手法から計算される最適値となる．また，Initiator のミドルウェアで再指定するブロックサイズは，提案手法 I と同様の方法で計算する．

6. 提案手法 II の性能測定実験

本節では提案手法 II の評価を行うため，遅延時間を変更して iSCSI ストレージアクセスの性能測定を行う．

6.1 実験概要

4 台の Initiator から Target へシーケンシャルリードアクセスを行い，遅延時間を変化させた場合の性能測定を行う．実験の環境や設定はすべて 4 節と同様である．

6.2 実験結果

前節の実験結果として，片道遅延時間 16ms(Round Trip Time: 32ms) において測定した，輻輳ウィンドウ，スループットの時間変化を図 12，13 に，Initiator1 において輻輳ウィンドウとスループットを比較した時間変化グラフを図 14 に示す．これらの図における輻輳ウィンドウの低下原因は，すべて CWR エラーによるものである．

提案手法 I の場合と異なり，Target において一度輻輳ウィンドウが一定値であると判断されると Initiator に通知を行わない．そのため，一定値となってからも CWR エラーは検出され，輻輳ウィンドウが低い値に設定されるが，すぐにほぼ同じ値にまで回復し，再度一定値となる (図 12)．また，CWR 検出時にスループットも一時低下するが，輻輳ウィンドウの回復と共に増加し，ほぼ安定した通信が行われている様子が図 13 より確認される．

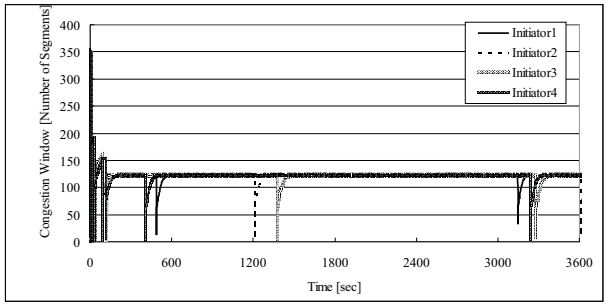

図 12　提案手法 II を用いた場合の輻輳ウィンドウの時間変化

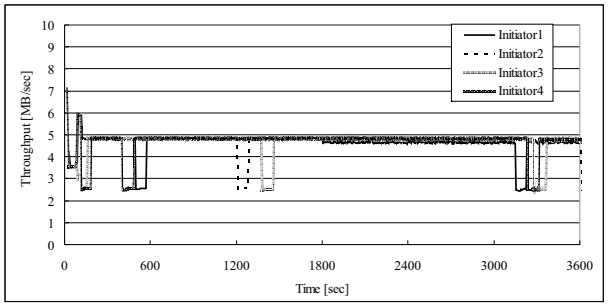

図 13　提案手法 II を用いた場合のスループットの時間変化

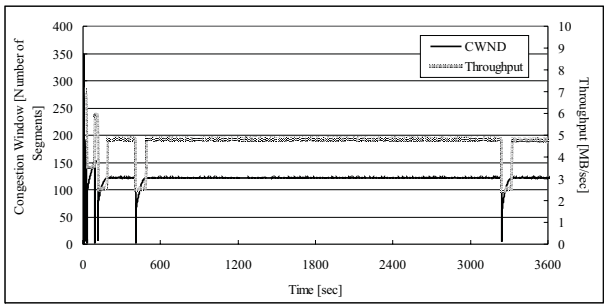

図 14　提案手法 II を用いた場合の輻輳ウィンドウ，スループットの時間変化

7. 考　　察

本節では，これまでの実験結果から，提案手法を用いない iSCSI シーケンシャルリードアクセスの性能と，複数台 Initiator 輻輳ウィンドウコントロール手法 I，II を用いた iSCSI シーケンシャルリードアクセスの性能を，より詳細に評価する．

表 2 から表 6 は，遅延時間を変更し，提案手法を用いた場合と用いなかった場合において，10GB のデータをリードした場合の各 Initiator の平均スループットの一例を比較したものである．提案手法を用いないシーケンシャルリードアクセスの場合，アクセスブロックサイズを大きな値で保つことはできるが，各 Initiator ごとに CWR エラーが頻発し，その度に輻輳ウィンドウが急激に低下する．この振舞を不定期に繰り返すためにスルー

表 2　片道遅延時間 0ms における各 Initiator 平均スループットの比較

Applied Method	Throught [MB/sec]			
	Initiator1	Initiator2	Initiator3	Initiator4
Not Using Proposed Method	28.59	26.50	25.81	27.66
Proposed Method I	28.39	28.44	28.88	28.48
Proposed Method II	28.53	28.52	28.99	28.52

表 3　片道遅延時間 2ms における各 Initiator 平均スループットの比較

Applied Method	Throught [MB/sec]			
	Initiator1	Initiator2	Initiator3	Initiator4
Not Using Proposed Method	17.81	21.32	19.99	21.57
Proposed Method I	17.85	17.97	17.98	17.76
Proposed Method II	17.96	17.97	17.99	17.94

表 4　片道遅延時間 4ms における各 Initiator 平均スループットの比較

Applied Method	Throught [MB/sec]			
	Initiator1	Initiator2	Initiator3	Initiator4
Not Using Proposed Method	13.17	15.13	13.91	14.37
Proposed Method I	9.69	9.68	9.70	9.56
Proposed Method II	13.16	13.24	13.08	13.14

表 5　片道遅延時間 8ms における各 Initiator 平均スループットの比較

Applied Method	Throught [MB/sec]			
	Initiator1	Initiator2	Initiator3	Initiator4
Not Using Proposed Method	6.86	7.12	8.03	12.09
Proposed Method I	6.84	6.88	6.81	6.71
Proposed Method II	7.56	7.64	7.5	7.57

表 6　片道遅延時間 16ms における各 Initiator 平均スループットの比較

Applied Method	Throught [MB/sec]			
	Initiator1	Initiator2	Initiator3	Initiator4
Not Using Proposed Method	3.73	6.71	4.29	4.57
Proposed Method I	3.56	3.60	3.57	3.51
Proposed Method II	4.69	4.72	4.67	4.66

プットも不安定となり，各 Initiator のスループットが大きく異なる場合がある．また，常に特定の Initiator のアクセス性能が高かったり低かったりするわけではないため，Initiator の公平性は失われてしまう．しかし，提案手法を適用することにより各 Initiator の性能を均一にするため，どの Initiator からも公平で効率的にストレージを利用することが可能となる．

提案手法 I は，スループットのばらつきを抑制することを目的としている．CWR エラーが起こらない範囲までブロックサイズを低下させるという実装であるため全体の性能は低下してしまうが，Target から各 Initiator に対するデータ転送を均一化することができ，CWR エラーを生じさせない公平なストレージアクセスを保証する．次に我々は提案手法 I を改良し，各 Initiator のスループットのばらつきを抑制することに加えて性能向上も行う提案手法 II を実装した．提案手法 II では CWR は検出されるが，ブロックサイズを比較的大きな値で保ったままストレージアクセスを行うことができる．

表 7 は，遅延時間を変更した時の，提案手法 I，II を用いた

表 7　各遅延時間における Initiator1 台あたりの平均スループット

1 Way Delay Time	Throught [MB/sec]	
	Proposed Method I	Proposed Method II
0ms	28.55	28.91
2ms	18.00	18.00
4ms	10.31	13.15
8ms	6.61	7.30
16ms	3.53	4.39

表 8　各遅延時間における平均ブロックサイズ

1 Way Delay Time	Block Size [KB]	
	Proposed Method I	Proposed Method II
0ms	69.5	70.3
2ms	109.5	109.0
4ms	103.5	140.5
8ms	106.0	136.0
16ms	109.5	163.0

iSCSI シーケンシャルリードアクセスにおける各 Initiator あたりの平均スループットを，表 8 は，提案手法 I，II それぞれの最終的なアクセスブロックサイズの平均を示している．

遅延が小さな環境の場合，Target が送信したデータに対する確認応答パケット (ACK) は比較的早い間隔で返信されるため，Target における通信の待ち時間は短くなる．また，大量のデータが連続して送信されるため，頻繁に CWR を検出してブロックサイズが変更される．そのため，制御方法に依存することがなく，提案手法 I，II のシステム性能，アクセスブロックサイズにほとんど変化は見られない．

一方遅延が大きな環境の場合，応答時間が長くなることにより送信したデータに対する ACK の到着が遅れ，Target における通信の待ち時間は大幅に増加する．これらの環境において本手法を適用した場合，輻輳ウィンドウが一定値となることにより Target から Initiator に送信されるデータパケット数が一定値に保たれる．これにより，ACK が返信されるまでの間の無駄な通信の待ち時間に連続してパケットを送信し続けることができる．そのため，輻輳ウィンドウをより大きな値で一定に保ち，かつアクセスブロックサイズが大きい提案手法 II の方が大きく性能を向上させることができる．

本実験では，提案手法を用いなかった場合のアクセスブロックサイズと提案手法を用いた場合のアクセスブロックサイズの初期値は共に 1024KB として測定を行った．しかし提案手法を用いた場合，表 8 によりブロックサイズは 100KB 前後に収束することがわかる．一般的に，ブロックサイズが大きい方が性能は向上すると言われているが，本実験結果から，ブロックサイズが小さくても効率的な制御を行うことにより iSCSI ストレージアクセスの性能は向上することが示された．

8. 関 連 研 究

iSCSI の関連研究としては文献 [9]〜[12]，TCP 輻輳ウィンドウに関連した研究としては文献 [13]〜[15] などが挙げられる．

文献 [9] は iSCSI のソフトウェア実装とハードウェア実装を

比較し，CPU 利用率という点を除いてはソフトウェア実装の方が性能が良いという結果を示している．文献 [10] では，iSCSI と NFS の比較を行い，ファイル操作などの一般的な操作におけるパフォーマンスや，ベンチマークを用いた総合的なパフォーマンスを測定している．これらの研究は，iSCSI の性能を知る上では有用な研究であるが，システム内部の振舞について把握し，性能を評価しているものではない．

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ものであると指摘し，“ quanta ” と呼ばれる固定のデータ単位でデータハンドリングを行うこと提案している．iSCSI 性能低下の原因について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た iSCSI の性能評価を行っている．性能向上のため，カーネルや iSCSI ソフトウェアの設計に変更を加えるというアプローチであり，既存ソフトウェアには基本的に変更を加えず，ミドルウェアで対応している我々とは手法が異なるものである．また，高遅延環境における評価は行われていない．

文献 [13] は，高速かつ高遅延なネットワークにおける既存 TCP の問題について触れ，パケット送信間隔を調節し，ネットワークへの負荷を抑制するものである．しかし，複数の TCP ストリームを用いることで帯域を有効に利用するというアプローチであるため，iSCSI を対象としている我々とは異なるものである．

文献 [14] では，高遅延環境においては CWR エラーを輻輳とみなし輻輳ウィンドウを減少させると通信性能を低下させるが，並列アプリケーションを実行した場合には CWR エラーを輻輳とみなす方が性能が向上することについて述べ，その解決法を紹介している．また，文献 [15] では，長距離，大容量ネットワークにおける TCP の問題について触れ，新しいトランスポートプロトコルとして提案されている，HSTCP（High-SpeedTCP），Scalable TCP，FAST，SABUL（Simple Avaliable Bandwidth Utilization Library）の輻輳ウィンドウの振舞を紹介し，評価を行っている．これらの文献で紹介されている技術は，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，既存の TCP を改変することで性能向上を目指すものである．従って，広く一般に普及している既存の TCP をそのまま利用した性能向上を目的とする本研究とは異なったものである．

9. ま と め

本稿では，複数台の Initiator から Target にアクセスする環境を想定し，iSCSI ストレージアクセスを行った場合に確認されるスループットのばらつきを抑制するため，各 Initiator の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トレージアクセスを行う，複数台 Initiator 輻輳ウィンドウコントロール手法 I，II を提案した．提案手法 I，II を iSCSI シーケンシャルリードアクセスに適用し，遅延時間が異なる場合の性能評価を行った．その結果，提案手法を用いることにより，各 Initiator へのデータ転送を均一化することができ，公平で効率的なストレージアクセスを実現することができた．特に提案手法 II においては，提案手法を用いないシーケンシャルリードアクセスを行った場合とほぼ同等の性能を保ちながら，各 Initiator のスループットを均一化することができるため，本手法の有効性が確認された．

今後は提案手法をさらに改良し，より高性能なストレージアクセスを行うことができるようにしたい．

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：“ 高遅延ネットワーク環境における iSCSI リードアクセス時の TCP 輻輳ウィンドウ制御手法の性能評価 ”，先進的計算基盤システムシンポジウム (SACSIS2005), pp.443-450，2005 年 5 月．

[5] 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.

[6] 豊田真智子，山口実靖，小口正人：“ 複数台の iSCSIInitiator を用いたストレージアクセス時における TCP フローの解析 ”，夏のデータベースワークショップ (DBWS2005)，pp.577-583 (情報処理学会研究報告，2005-DBS-137(II)), pp239-244 (電子情報通信学会技術研究報告，DE2005-106), 2005 年 7 月．

[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.335-342，2004 年 3 月．

[13] 菅原豊，稲葉真理，平木敬：“ インテリジェント NIC を用いた広帯域ネットワーク向け TCP 通信方式 ”，情報処理学会研究報告 2004-OS-97，SWoPP2004，pp.57-64，2004 年 8 月．

[14] 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 ”，IC2003，2003 年 10 月．

[15] 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.303-308，2004 年 3 月．