ファイル転送時間短縮のための蓄積中継形フロースケジューリング手法

佐藤広和[†1]　中山慶彦[†1]
岩見亮太[†1]　鶴　正人[†1]

ネットワーク上でのファイル同士の競合によるファイル転送時間の増加を防ぐために競合を時間的に回避することを目的とした蓄積中継形の 2 つのフロースケジューリングを考案し，各スケジューリングでの蓄積ノードの位置とトポロジのリンク帯域の変化による影響をシミュレーションにより検証した．その結果，提案手法では送信ノード-蓄積ノード間の帯域によりその性能に差が生じるため，トポロジの帯域によってどちらのスケジューリングを採用すべきか，どこに蓄積ノードを設置すべきかについて明らかにした．また，提案手法によるファイル転送では従来の end-to-end 転送よりも平均転送時間の短縮が可能であることを示した．

Flow scheduling by storage relay method for reducing file transmission time

Hirokazu Sato,[†1] Yoshihiko Nakayama,[†1]
Ryota Iwami[†1] and Masato Tsuru[†1]

This thesis is about inspection of file transmission on network by two flow schedulings by storage relay method. Storage relay method makes files avoid the competition between files on transmission route. So, it helps average of the file transmission time reduce. Inspection way is simulation by C programming. As a result, the proposed methods indicate different performance depending on bandwidth between sender node and storage node. So, it's shown that which scheduling is effective and where to place storage node for reducing transmission time. Also, it's shown average of the file transmission time by proposed methods is shorter than one by tradittional methods.

1. はじめに

近年のインターネットサービスの多様化により動画やソフトウェアといった大容量のファイルを送受信するユーザの増加が今後も進むと考えられる．そこで問題となるのが，インターネット上のファイルの転送経路でのファイル同士の競合によるファイル転送時間の増加である．既存のファイル転送では，リンクを通るファイル数に応じてファイルの転送速度が決定するため，あるリンクで多数のファイルが競合を起こすと，その分だけ転送速度が低下し転送時間が増加する．この問題を解決するための手法として空間的にファイル同士の競合を回避する MPLS(Multi-Protocol Label Switching) がある．だが MPLS を導入するためには多大な労力が必要であり，トポロジに依存して有効でない場合がある．

そこで，本稿ではファイル同士の競合を時間的に回避，緩和する手法として蓄積中継形ファイル転送を検討する．蓄積中継形ファイル転送では，ネットワーク上に蓄積ノードという通過するファイルを一旦蓄積し，適切なタイミングで転送再開が可能な装置を配置した環境でのファイル転送を行う．このシステム上では従来起こっていたファイル同士の競合を回避，緩和でき，ファイルの転送時間の短縮が期待される．本稿では，2 つの蓄積中継形のフロースケジューリングを提案し，C 言語による単純なモデルでのシミュレーションによりその性能の分析・評価を行う．

2. 蓄積中継形ファイル転送

本章では，蓄積中継形ファイル転送手法について説明する．従来のファイル転送方式では，ファイルの転送要求があると end-to-end でパケットが連続的に送信される．一方，蓄積中継形ファイル転送では，ファイルの経路上にある蓄積ノードに到達した任意のファイルをそのノードで蓄積することで，ファイル同士の競合を回避，緩和することが可能であり，リンクを効率的に利用して転送時間を短縮できる．

2.0.1 蓄積中継形転送による転送時間短縮の概要

図 1 のようなトポロジでノード 1 からファイル A(100[Mbit]) が，ノード 2 からファイル B(100[Mbit]) がノード 3 を通りノード 4 まで同じ時刻に転送が開始される場合を考える．各リンクの帯域は全て 100[Mbps] とする．従来転送方式ではファイル A とファイル B は

†1 九州工業大学
Kyushu Institute of Technology

end-to-end で転送されるため，ノード 3-ノード 4 間で競合してしまい，可用帯域幅はそれぞれ 50[Mbps] となり，各ファイルの転送時間は

ファイル A の転送時間 $= 100[Mbit] \div 50[Mbps] = 2.0[sec]$

ファイル B の転送時間 $= 100[Mbit] \div 50[Mbps] = 2.0[sec]$

である．一方，図 2 に示す蓄積中継形転送ではファイル A はノード 1 からノード 4 へ end-to-end で転送を行い，ファイル B は蓄積ノード 3 で一旦蓄積する．そして，ファイル A の転送完了後に蓄積ノード 3 からノード 4 へ転送を再開する．ファイル B のフローを蓄積することでファイル A とのノード 3-ノード 4 間での競合を時間的に回避し各ファイルの可用帯域幅を大きくすることができる．ファイル A の可用帯域幅は 100[Mbps] となるため，その転送時間は

ファイル A の転送時間 $= 100[Mbit] \div 100[Mbps] = 1.0[sec]$

である．一方，蓄積中継したファイル B はまず，ノード 2 から蓄積ノード 3 へ転送を行う．そのリンクでの可用帯域幅は 100[Mbps] なので

蓄積ノード 3 までの転送時間 $= 100[Mbit] \div 100[Mbps] = 1.0[sec]$

となる．同時にファイル A の end-to-end での転送も完了するためファイル B は蓄積ノード 3 での待ち時間はない．蓄積ノード 3 からノード 4 までの可用帯域幅は 100[Mbps] なので

受信ノード 4 までの転送時間 $= 100[Mbit] \div 100[Mbps] = 1.0[sec]$

となり，ファイル B の転送時間は

ファイル B の転送時間 $= 1.0[sec] + 1.0[sec] = 2.0[sec]$

となる．ここで従来転送方式と蓄積中継形転送の各ファイルの転送時間を比較すると，ファイル B は変わらないがファイル A は従来転送の半分に短縮できていることが分かる．このように，先行フローの転送は高速化され，中継ノードで待機していた後続フローは従来通りの転送時間で転送が可能となり平均転送時間が短縮される．1 回の転送だけであれば後続フローのユーザにとってメリットはないが，このようなネットワーク上で複数のユーザが複数回のファイル転送を行うならば，個々のユーザにとっても平均転送時間が短縮される．

2.1 転送スケジューリング

本稿では 2 つの転送スケジューリングを考案し，シミュレーションによる転送時間の測定によりその性能を分析した．その内容を以下に示す．なお，前提としてフローモデル (ファイルサイズ，経路)，リンク利用率は把握できるものとする．

(1) 単純スケジューリング

- 全ファイルの内，最小ファイルサイズのファイルのみ end-to-end 転送を行い，その他のファイルは経路上で最初に到達する蓄積ノードに蓄積
- 蓄積したファイルはファイル長が小さい順に優先順位を高くし，蓄積ノード-受信ノード間の転送開始
- 蓄積するファイルは，蓄積ノード以降の経路上に先行転送フローが利用していない空き帯域があれば，その帯域を使い蓄積ノード-受信ノード間の転送開始

(2) グループスケジューリング

- ボトルネックリンク[⋆1]を共有するファイル毎にグループを構成
- 各グループ間で独立して単純スケジューリングによる転送を行う

end-to-end 転送されるファイルは，単純スケジューリングでは全ファイルで最小のもの，グループスケジューリングでは各グループで最小のものであり，最小でないもので end-to-end 経路に空き帯域がある場合，送信ノードは蓄積ノードへ蓄積転送を行い，蓄積ノードが送信ノードから送られてきたデータを受信ノードに転送する．また，蓄積されるファイルは蓄積ノードまでは競合があっても，シミュレーション開始時に転送を始める．

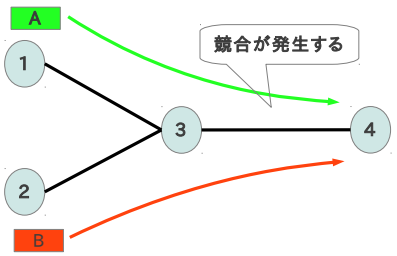

図 1 従来転送方式

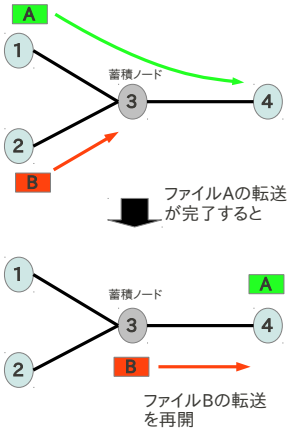

図 2 蓄積中継形転送方式

⋆1 可用帯域を決定するリンク

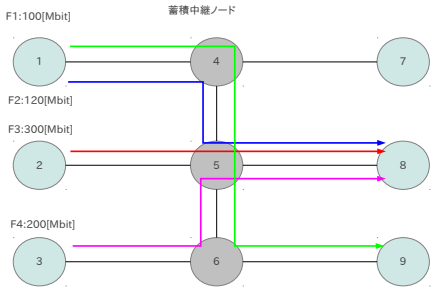

図 3 フローモデル

表 1 図 3 の各リンクの帯域幅

リンク	帯域幅 [Mbps]
ノード 1-ノード 4，ノード 2-ノード 5，ノード 3-ノード 6	200
その他のノード間	100

2.2 帯域割り当てと空き帯域利用

ファイル転送において各ファイルへの帯域割り当てはファイルサイズで決定する優先順位が高いものから行う．但し，end-to-end 転送ファイルの優先順位が最も高い．優先順位が 2 番目以降のファイルは経路上に自分よりも優先順位が高いファイルが使っていない空き帯域があれば検知して転送に利用する．ここで，図 3 のフローがあった場合に提案した 2 つのスケジューリングを適用した場合の概要を示す．

- 単純スケジューリング

4 つのファイルで最小ファイルサイズであるファイル 1 は end-to-end 転送を行う．他の 3 つのファイルはファイル 1 はノード 4 に，ファイル 3 はノード 5 に，ファイル 4 はノード 6 に蓄積する．優先度が 2 位であるファイル 2 はノード 4-ノード 5 間で帯域を確保できないため，蓄積のみが行われる．優先度が 3 位であるファイル 4 は経路に空き帯域があるため蓄積をしつつ，蓄積ノード 6 のバッファにたまったデータをノード 8 に転送する．優先度が 4 位であるファイル 3 はノード 5-ノード 8 間で帯域が確保できないため蓄積のみを行う．ファイル 1 の転送完了後，ファイル 2 は優先順位が 1 位になるがまだ蓄積中である．そのため蓄積をしつつ，蓄積ノード 4 のバッファにたまったデータをノード 8 に転送する．また，ファイル 4 の蓄積ノード 6-ノード 8 間の空き帯域利用は中断される．そしてファイル 2 の蓄積ノード 4 への蓄積が完了後はそのまま蓄積ノード 4 のバッファにたまったデータのノード 8 への転送を続ける．蓄積ノード 4 のバッファにたまったデータがなくなり次第，ファイル 4 の蓄積ノード 6 からノード 8 への転送を再開する．ファイル 4 の蓄積ノード 6 からノード 8 への転送完了後，ファイル 3 の蓄積ノード 5 からノード 8 への転送を始める．

- グループスケジューリング

スケジューリングにより，ファイル 1 をグループ 1 に，ファイル 2，ファイル 3，ファイル 4 をグループ 2 にグループ分けされる．グループ 1 ではファイル 1 が end-to-end 転送され，グループ 2 では最小ファイルサイズのファイル 2 が end-to-end 転送される．ファイル 3 はノード 5 に，ファイル 4 はノード 6 に蓄積する．グループ 2 の優先順位 2 位であるファイル 4 は経路に空き帯域があるため蓄積ノード 6 への蓄積をしつつ，蓄積ノード 6-ノード 8 間の転送を行う．ファイル 1 の転送完了後，ファイル 2 の可用帯域幅は 50[Mbps] から 100[Mbps] になるためファイル 4 の空き帯域利用は中断する．そしてファイル 2 の end-to-end 転送完了後，ファイル 4 の蓄積ノード 6-ノード 8 間の転送が再開され，その転送が完了したらファイル 3 の蓄積ノード 5-ノード 8 間の転送が始まる．

3. シミュレーションモデル

本稿では C 言語によるファイル転送時間を測定するシミュレータを作成し，蓄積ノードの位置とリンク帯域を変えた 9 通りのシミュレーションパターンを用意し，フローモデル (ファイル長・経路) をパラメータとして変更し，ファイルの転送時間の測定を行った．

3.1 ネットワークモデル

本稿で検証したトポロジは図 4 に示す 4 × 5 のメッシュトポロジである．各送信ノードから 2 つの可変長のファイルが，シミュレーション開始時に同時転送される．なお，このシミュレーションでは各ファイルの転送経路とファイルサイズは既知であり，蓄積ノードのキューバッファは無限，パケットロス，キュー遅延，伝搬遅延はないと仮定する．また，蓄積ノードは各ファイルの転送完了タイミングとリンク利用率を把握できるものとする．

3.2 フローモデル

本稿では 100 種類のフローモデル (8 つのファイル) に対するシミュレーションを行った．各ファイルのファイルサイズは 50[Mbit]~500[Mbit] のランダムな値である．但し，一回のシミュレーション中で同じサイズのファイルはないものとする．また，各シミュレーションでの転送ファイルの経路はランダムに決定する．但し，各ファイルが送信ノード-受信ノード間を最短経路で到達するような経路となる．

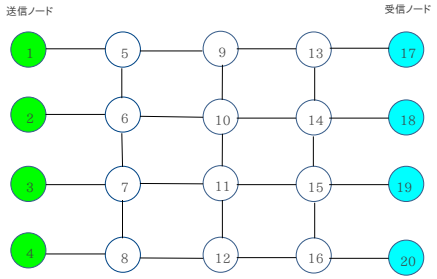

図 4 4 × 5 メッシュトポロジ

3.3 転送時間決定アルゴリズム

本シミュレーションでの各ファイルの転送時間決定アルゴリズムを示す.

3.3.1 end-to-end 転送ファイル

各ファイルのファイルサイズの値が入っている送信ノード用配列から各ファイルに割り当てられている可用帯域× 0.01（本タイムドリブン形シミュレーションでは 0.01[sec] 毎に，計算するため）を減算していき，送信ノード用配列が 0 以下になれば，転送完了として転送時間が決定する.

3.3.2 蓄積中継されるファイル

各蓄積ファイルは送信ノード-蓄積ノード間の可用帯域と蓄積ノード-受信ノード間の可用帯域が計算される．そして，以下の 3 つの計算を同時に行う.

- 各蓄積ファイルの送信ノード用配列から各蓄積ファイルの送信ノード-蓄積ノード間の可用帯域× 0.01 を減算
- 各蓄積ファイルの蓄積ノード用配列に各蓄積ファイルの送信ノード-蓄積ノード間の可用帯域× 0.01 を加算
- 各蓄積ファイルの蓄積ノード用配列から各蓄積ファイルの蓄積ノード-受信ノード間の可用帯域× 0.01 を減算

上から説明すると，1 つ目は送信ノードからのデータの転送，2 つ目は蓄積ノードへのデータの蓄積，3 つ目は蓄積ノードからのデータの転送となっている.
そのため，各蓄積ファイルの送信ノード用配列が 0 以下になれば，蓄積完了とし，蓄積完了かつ蓄積ノード用配列が 0 以下になれば転送完了として転送時間が決定する.

なお，可用帯域幅は 0 以上であり，Max-min fair によって以下のように決定される.

(1) 各ファイルの許容可用帯域を 1[Mbps]，2[Mbps] … と上げていく.
(2) 全ファイルで最小となる，1 つ目のファイルの可用帯域が決まる.
(3) 決定済みの可用帯域を除いた帯域に対して，可用帯域が決定されていない全てのファイルの可用帯域が決まるまで繰り返す.

3.4 性能評価指標

性能評価指標として，従来転送による平均転送時間と提案した 2 つのフロースケジューリングを適用した蓄積転送の平均転送時間の比較を行う．結果は，蓄積転送の平均転送時間から従来転送による平均転送時間を引いた値を基にヒストグラムとし，その値が負であれば平均転送時間の短縮効果があり，0 であれば効果なし，正であれば悪影響となり，負でありかつその絶対値が大きい程性能が高いといえる.

4. シミュレーション結果

本章で蓄積ノードの位置とトポロジのリンク帯域を変化させた場合の各スケジューリングによる平均転送時間（各ファイルの転送完了までの時間の 8 個のファイルに対する平均）の従来転送（全ファイルの同時並行転送）からの短縮時間をヒストグラムとして示す.

4.1 トポロジの全帯域が 100[Mbps] のとき

全帯域が 100[Mbps] のときの結果を図 5 に示す．図 5 より，送信ノードで蓄積した場合が最も性能が良い．これは蓄積ノードが送信ノードでない場合，送信ノード-蓄積ノード間で競合が発生するが，送信ノードで蓄積することでこの競合を回避し，リンクの利用率の低下による転送時間の増加を抑制しているためである．また，単純スケジューリングとグループスケジューリングを比較すると，単純スケジューリングの方が性能が良いことが分かる．これは，グループスケジューリングにおいて，他のグループに属するファイルとの競合が発生することによって，リンク利用率が低下し，蓄積ファイルの蓄積ノードでの待ち時間が増加することによる影響だと考えられる．例えば，図 6 のようなフローモデルにおいて，グループスケジューリングを行う場合を考える．全帯域は 100[Mbps] である．提案スケジューリングを適用し，グループ分けすると，グループ 1 がファイル 2，3，4 でありグループ 2 がファイル 1，グループ 3 がファイル 5，グループ 4 がファイル 6 となる．グループ 1 ではファイル 3 が end-to-end 転送され，他のファイルは蓄積される．また，他の

グループはグループ構成ファイルが 1 つしかないため，全て end-to-end 転送される．ここでグループ 1 のファイル 3 とグループ 2 のファイル 1 が競合してしまい，ノード 6-ノード 10 間に空き帯域が生じるが，グループ 1 の他のファイルの経路上のノード 6-ノード 10 間の先のリンクを他グループのファイルが占有しているため，蓄積ノード-受信ノード間の転送を開始できない．そのためリンク利用率が低下し、ファイル 2 とファイル 4 の蓄積ノードでの待ち時間が増加してしまうという問題が生じる．

また，図 5 の（c），(d）と（e），(f）から送信ノード以外で蓄積した場合に単純スケジューリングによる転送よりもグループスケジューリングによる転送の性能が良いことが分かる．これは，単純スケジューリングを行った場合のリンク利用率の低下によるものであると考えられる．例えば，図 7 のフローモデルにおいて単純スケジューリングによる転送を行う場合を考える．なお，図 7 のトポロジの全帯域は 100[Mbps] である．まず，ファイルサイズが最小であるファイル 1 が end-to-end 転送され，他のファイルは最初に到達する蓄積ノードに蓄積される．ここで蓄積ファイルで最も優先順位が高いファイル 3 は蓄積ノード-受信ノード間に空き帯域があるため，送信ノード-蓄積ノード間転送と同時に蓄積ノード-受信ノード間の可用帯域を 100[Mbps] として転送を始める．しかし，ファイル 3 は送信ノード-蓄積ノード間を 50[Mbps] の可用帯域で転送を行っているため，蓄積ノード-受信ノード間のリンクの半分しか利用できない．また，ファイル 3 がノード 6-ノード 9 間を占有しているため，ファイル 5 の蓄積ノード-受信ノード間の転送ができないという問題も起こる．このようなリンク利用率の低下がファイル転送時間の増加を引き起こしている．この問題は送信ノード-蓄積ノード間の可用帯域が蓄積ノード-受信ノード間の可用帯域よりも小さく，蓄積ノードのバッファにあまりデータが蓄積されていないときに起こる．一方，同じフローモデルに対してグループスケジューリングによる転送を行う場合を考える．グループ構成としては，ファイル 1，ファイル 2，ファイル 4 がノード 4-ノード 7 間でボトルネックリンクを共有しているためグループ 1 となり，ファイル 3，ファイル 5 がノード 6-ノード 9 間でボトルネックリンクを共有しているためグループ 2 となる．各グループ内で単純スケジューリングを適用すると，グループ 1 はファイル 1 が end-to-end 転送，ファイル 2 は蓄積ノード-受信ノード間に空き帯域があるためその空き帯域を使い，送信ノード-蓄積ノード間転送と同時で蓄積ノード-受信ノード間転送を始める．グループ 2 ではファイル 3 が可用帯域を 50[Mbps] として end-to-end 転送を行い，ファイル 5 は蓄積ノード-受信ノード間に空き帯域があるため，送信ノード-蓄積ノード間転送と同時で蓄積ノード-受信ノード間転送を始める．このようにグループスケジューリングによって利用率が低いリンクがあっても，他のグループのファイルと分割して利用することでリンク利用率が上がるため，単純スケジューリングよりもグループスケジューリングの性能が良いと考えられる．

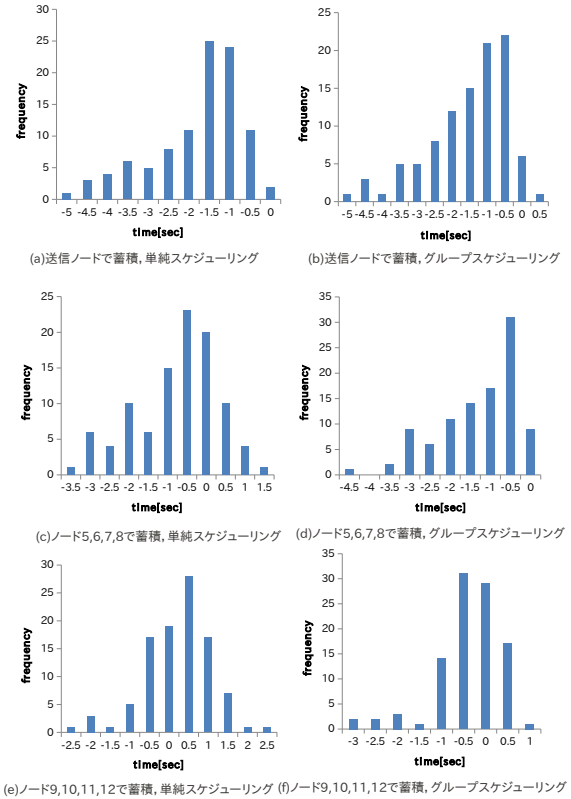

図 5　全帯域が 100[Mbps] のときの平均転送時間の短縮時間

4.2　トポロジの帯域が不均一のとき（ケース 1）

ケース 1 ではノード 1-5，ノード 2-6，ノード 3-7，ノード 4-8 間の帯域が 200[Mbps] であり，その他は 100[Mbps] である．結果を図 8 に示す．結果より，送信ノードで蓄積した

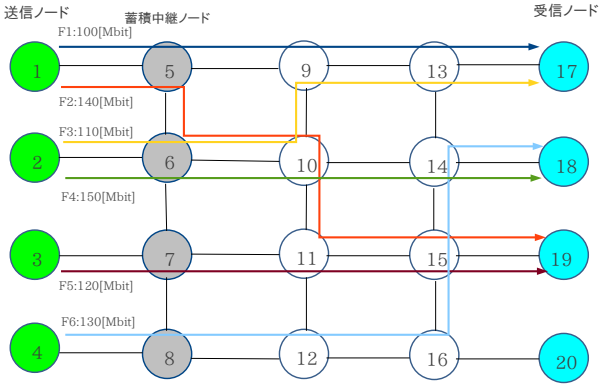

図 6 グループスケジューリングによるリンク利用率低下

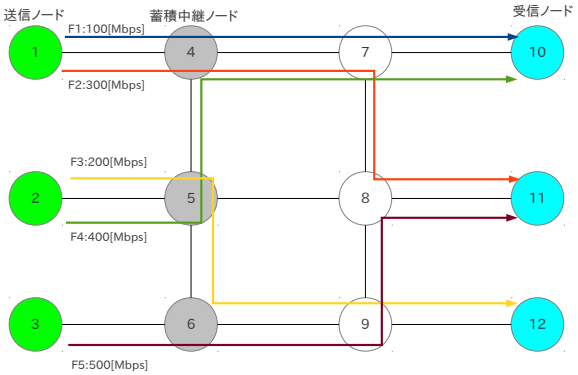

図 7 単純スケジューリングによるリンク利用率低下

場合とノード 5，6，7，8 で蓄積した場合は同じ結果を示していることが分かる．これは各送信ノードから 2 つのファイルが同時転送されるときに，ケース 1 の帯域では，ノード 5，6，7，8 で蓄積しても送信ノード-蓄積ノード間がボトルネックリンクとならない帯域を確保できるためである．つまり，このように送信ノード-蓄積ノード間がボトルネックリンクとならない状況では，送信ノードで蓄積した場合と同様の転送時間短縮効果が送信ノード以外で蓄積しても得られると考えられる．また，送信ノードで蓄積した場合とノード 5，6，7，8 で蓄積した場合にグループスケジューリングよりも，単純スケジューリングの性能が良いことが分かる．これは，グループスケジューリングでの転送で起こる他のグループに属するファイルとの競合によるリンク利用率の低下によって蓄積ファイルの蓄積ノードでの待ち時間が増加するためだと考えられる．

ノード 9，10，11，12 で蓄積した場合に性能が悪いのは，ノード 1-5，ノード 2-6，ノード 3-7，ノード 4-8 間での競合によってそのリンクがボトルネックリンクとなってしまい，リンク利用率が低下し，蓄積ファイルの蓄積ノードでの待ち時間の増加のためだと考えられる．また，単純スケジューリングよりもグループスケジューリングの性能が良いのは，先述したように利用率の低いリンクを他のグループのファイルと分割して利用できているからだと考えられる．

4.3 トポロジの帯域が不均一のとき（ケース 2）

ケース 2 ではノード 9-13，ノード 10-14，ノード 11-15，ノード 12-16 間の帯域が 60[Mbps] であり，その他は 100[Mbps] である．結果を図 9 に示す．結果より，送信ノードで蓄積した場合の性能が良いことが分かる．これは送信ノード-蓄積ノード間の競合を回避できるためだと考えられる．単純スケジューリングとグループスケジューリングを比べると全帯域が 100[Mbps] のときと比べてあまり差がない．全帯域が 100[Mbps] のときは送信ノード-受信ノード間の転送を 100[Mbps] の可用帯域で行うことができ，その経路上のリンク利用率が高いのに比べ，ケース 2 の帯域では，送信ノード-受信ノード間の転送で 60[Mbps] のリンクがボトルネックリンクとなってしまい，リンク利用率が低下する．そのため，グループスケジューリングを行うことで，他のグループのファイルと利用率が低いリンクを分割して利用してリンク利用率が上がり平均転送時間が短縮されているためだと考えられる．

ノード 5，6，7，8 に蓄積した場合も全帯域が 100[Mbps] のときと比べて，送信ノードで蓄積した場合との差が小さい．これは，全帯域が 100[Mbps] のときにノード 5，6，7，8 で蓄積した場合の end-to-end 転送ファイルの可用帯域は 50[Mbps]，送信ノードで蓄積した場合は 100[Mbps] と倍の差があるのに対して，ケース 2 の帯域の場合は，ノード 5，6，7，8 で蓄積した場合の end-to-end 転送ファイルの可用帯域は 50[Mbps]，送信ノードで蓄積した場合は 60[Mbps] となりその差が小さいため，送信ノード-蓄積ノード間の競合を避

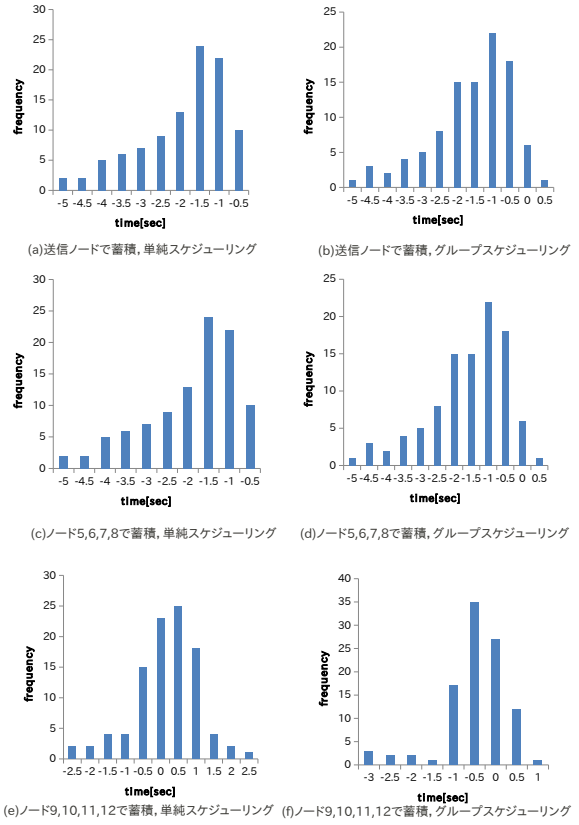

図 8 ケース 1 での平均転送時間の短縮時間

けるメリットが小さくなるためだと考えられる．また，全帯域が 100[Mbps] のときと比べて単純スケジューリングとグループスケジューリングの差が小さいのは，単純スケジューリングで送信ノード-蓄積ノード間の可用帯域が蓄積ノード-受信ノード間の可用帯域よりも小さく，蓄積ノードのバッファにあまりデータが蓄積されていないために起こっていたリンク利用率の低下の影響が抑制されているからだと考えられる．抑制される理由としては，蓄積ノード-受信ノード間の可用帯域が 60[Mbps] までしか上がらないためである．

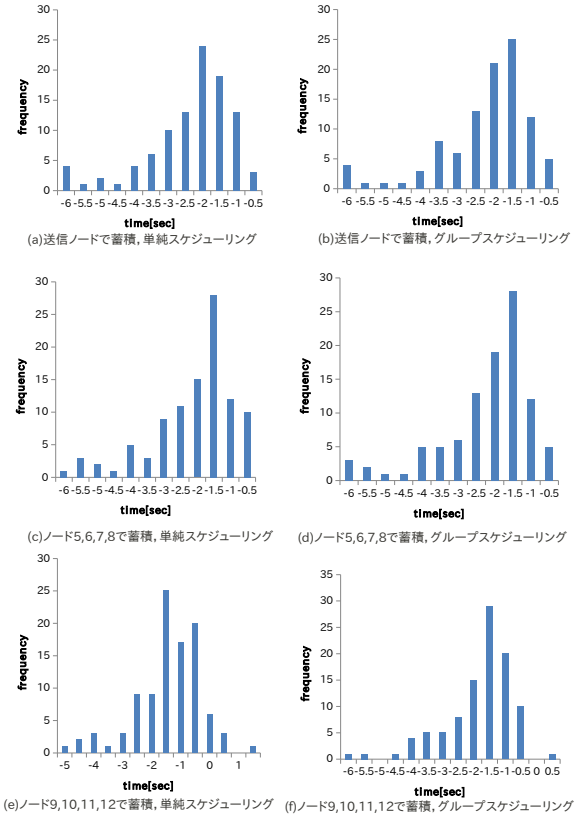

図 9 ケース 2 での平均転送時間の短縮時間

5. 考　　察

本章でシミュレーションによって明らかになった提案フロースケジューリングの今後の課題を検討する．

5.1 単純スケジューリングの問題点

単純スケジューリングでは，

- 蓄積ノードが送信ノード以外である
- 送信ノード-蓄積ノード間にボトルネックリンクが存在する
- 送信ノード-蓄積ノード間の可用帯域より蓄積ノード-受信ノード間の可用帯域が大きい
- 蓄積ノードにあまりデータが蓄積されていない

の全ての条件を満たす状況で蓄積されるファイルの蓄積ノード-受信ノード間の空き帯域を使った転送が開始されると，蓄積ノード-受信ノード間のリンクの利用率が低下し，平均転送時間が増加してしまう．そのため，このような場合蓄積ノード-受信ノード間の可用帯域に送信ノード-蓄積ノード間の可用帯域以上の値をとらせないように制御するか，蓄積ノードにある程度データが蓄積されるまで蓄積ノード-受信ノード間の転送を開始せずに，そのリンクを他のファイルに開放するような改良をすれば，さらに平均転送時間を短縮できると考えられる．

5.2 グループスケジューリングの問題点

グループスケジューリングでは他のグループのファイルとの競合によるリンク利用率の低下という問題がある．一方で，競合する他のグループのファイルのリンク利用率が低い場合にはこの競合が平均転送時間の短縮につながっていた．そのため，競合ファイルのリンク利用率が高ければこの競合を回避し，リンク利用率が低ければ競合を許すようなグループ間の連携が可能になればさらに平均転送時間を短縮できると考えられる．

また，今回はボトルネックリンク共有ファイル毎にグループ分けを行ったが，別の基準によるグループ分けを行った場合の検証も必要である．

5.3 共通の課題

提案した 2 つのフロースケジューリングによる転送を行った場合に平均転送時間は短縮できても，ファイル毎の転送時間では従来転送よりも増加してしまうファイルが存在する．この原因としては，可用帯域の低下による増加と蓄積ノードでの待ち時間による増加が考えられる．

可用帯域の低下による転送時間の増加は，主にグループスケジューリングの際に，end-to-end 転送される各グループで最小ファイルサイズのファイルの可用帯域が従来転送より小さくなる場合に発生する．具体的には，競合する他のグループの end-to-end 転送ファイルの可用帯域が蓄積転送による競合回避によって，従来転送による可用帯域よりも大きくなっている場合，そのファイルと競合するファイルの可用帯域が従来転送による可用帯域よりも小さくなってしまい，転送時間が従来転送より増加してしまう．そのため，end-to-end 転送ファイルにおいては最低でも従来転送による可用帯域を確保するとこの問題は抑制できると考えられる．

蓄積ノードでの待ち時間による転送時間の増加は，優先順位が高いファイルのリンク利用率の低下による，優先順位の低いファイルの蓄積ノードでの待ち時間の増加によるものだと考えられる．そのため，5.1，5.2 で述べたようなリンク利用率を上げるための改良や，フロースケジューリングでファイルサイズにより決定されている各ファイルの優先順位をリンク利用率によって変更するような改良が求められる．

また，今回は同時にファイル転送が開始される場合の検証しか行っていないため，より複雑な異なるタイミングで発生するファイル転送における性能の調査も必要である．

6. ま と め

今回のシミュレーション環境では，条件によって，２つのスケジューリングの優劣が異なったが，ほとんどの場合に従来転送からの平均転送時間の短縮が見られた．また，リンク帯域によって変わる各ファイルのボトルネックリンクが送信ノード-蓄積ノード間に存在するかどうかが重要であった．しかし，提案スケジューリングでは蓄積ノードまでの競合は避けることができないため，蓄積ノードが送信ノードでない場合は性能を悪化させていると考えられる．つまり，この競合を避けることができれば，リンク帯域の変化により柔軟に対応可能なスケジューリングが考えられる．

今後，今回の結果を踏まえ，より性能を向上させるために，送信ノードと経路上の中継ノードの最大２箇所で蓄積（待機）ができるような前提の下で，より高性能なスケジューリングを検討したい．