リード・ソロモン符号化複数経路マルチキャストによる一対多ファイル転送時間の最小化

著者	倉田　真之，平良　憲司，柴田　将拡，鶴　正人
雑誌名	電子情報通信学会技術研究報告．CQ，コミュニケーションクオリティ
巻	118
号	503
ページ	129-134
発行年	2019-03-07
その他のタイトル	Minimization of reception completion times by one-to-many file transfer using MultiPath-MultiCast with Reed-Solomon coding
URL	http://hdl.handle.net/10228/00007410

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

リード・ソロモン符号化複数経路マルチキャストによる一対多ファイル転送時間の最小化

倉田　真之† 　平良　憲司† 　柴田　将拡†† 　鶴　　正人††

† 九州工業大学先端情報工学府　〒 820–8502 福岡県飯塚市川津 680–4
E-mail: †{kuratam,heira}@infonet.cse.kyutech.ac.jp, ††{shibata,tsuru}@cse.kyutech.ac.jp

あらまし　分散配置されたデータセンタやサーバ間での大規模なデータやソフトウェアの共有, 複製, または移動によるトラヒック量の急激な増加が問題となっている. そこで, 単一の送信者から各受信者への max-flow を達成する複数経路（max-flow 経路）を用いて, 各受信者が自身の最短時間で受信を完了する最適スケジュールを実現するために, 複数経路マルチキャストによる一対多ファイル転送（MPMC）が検討されてきた. MPMC では，転送ファイルを均等長のブロックに分割し, 同一受信者へ複数のブロックを max-flow 経路で同時転送すると共に, 複数の受信者へ同一ブロックをマルチキャスト転送する. 本報告の提案手法は, リード・ソロモン符号化を用いて必要な数の符号化ブロックを生成し, フェーズ（ある群が受信完了した後、次の群が受信完了するまでの期間）毎に異なる符号化ブロックを転送することで, 受信者間の前フェーズまでの受信ブロックの違いに影響されずにブロック割当を最適化できる. さらに, ヒューリスティックなブロック割当順序管理方法を導入し, 大規模な実世界のネットワークトポロジに対しても最適スケジュールが容易に生成できることをシミュレーションで検証した. また, 実装した提案手法の動作確認を OpenFlow エミュレータ上で行った.

キーワード　OpenFlow, 複数経路マルチキャスト転送, 一対多ファイル転送, max-flow 問題, リード・ソロモン符号化

Minimization of reception completion times by one-to-many file transfer using MultiPath-MultiCast with Reed-Solomon coding

Masayuki KURATA†, Kenji HEIRA†, Masahiro SHIBATA††, and Masato TSURU††

† Computer Science and Systems Engineering, Kyushu Institute of Technology　680 4 Kawazu, Iizuka-shi, Fukuoka, 820–8502 Japan
E-mail: †{kuratam,heira}@infonet.cse.kyutech.ac.jp, ††{shibata,tsuru}@cse.kyutech.ac.jp

Abstract　A rapid increase in network traffic has caused a problem along with the penetration of sharing, duplicating, or migrating a large-sized data and software among distributed servers or sites. We previously proposed the one-to-many file transfer using MultiPath-MultiCast (MPMC) on OpenFlow to realize an optimal schedule in which each recipient can complete the file reception in its minimal time using the max-flow paths from a single sender. In MPMC, a file is divided into equally-sized blocks; different blocks are concurrently transmitted to the same recipient on multiple paths; while the same block is concurrently transmitted to multiple recipients by multicast. This report newly proposes the coded-MPMC in which a sender proactively generates a necessary number of coded blocks using Reed-Solomon coding and transmits different coded blocks in each phase (a period between when a set of recipients completed and when the next set of recipients complete), allowing an optimal block allocation regardless of a difference among recipients' already-received blocks in the previous phases. A few heuristics in the block allocation order are developed in coded-MPMC and shown to efficiently find optimal schedules on large-scale real-world network topologies through simulation. A preliminary implementation of coded-MPMC is verified on an OpenFlow emulator.

Key words　OpenFlow, MultiPath-MultiCast transfer, One-to-many file transfer, Max-flow problem, Reed-Solomon coding

1. はじめに

クラウド, 分散コンピューティングおよびコンテンツ配信ネットワーク (CDN) 技術の普及に伴い，分散配置されたデータセンタやサーバ間でのデータおよびソフトウェアの共有，複製, または移動の必要性が高まっている．新しい IoT 技術向けのエッジクラウドコンピューティングは，地理的に広い場所に分散した多数の異種の受信者への大規模な配信，たとえばソフトウェアの更新，データベースの複製，プロセスの移行などの必要性をさらに加速させている．そのため，データセンタ間だけでなく内でも，単一の送信者がネットワークパスを介して大容量のファイルを複数の受信者に配信するという高速で効率的な一対多のファイル転送が重要となる．

一対多ファイル転送では, 複数の受信者間で共有されるリンクのリンク容量の浪費や送信者の過負荷を防ぐためにマルチキャスト転送が基本的に採用されている．しかし, 単一のマルチキャストツリーを使用する場合 [1], 最も遠い場所にある受信者がボトルネックとなり転送完了時間に悪影響を与えてしまう．この欠点を対処するために, 受信者を適切にグループ化し複数のマルチキャストツリーを使用する様々な方式が検討された [2] [3] [4].

しかしながら，それらの方式は, 送信者から各受信者への複数のネットワーク経路の総容量を十分に利用しておらず, 各受信者のファイルの受信完了時間を最小化しない．そこで, 私たちは全二重有線リンクで完全に制御された OpenFlow ネットワーク上で, 単一の送信者から複数の受信者にファイルを配信する複数経路マルチキャストによる一対多ファイル転送モデル（MPMC:MultiPath-MultiCast）を提案した [5]．その手法では，転送ファイルを均等長のブロックに分割し，最短で受信が完了可能な受信者 (主受信者) に対し，フェーズ (ある群が受信完了した後, 次の群が受信完了するまでの期間) 毎に異なるブロックを max-flow を達成する経路 (max-flow 経路) で同時転送すると共に, 他の受信者 (副受信者) へも一部のブロックをマルチキャストにより転送し, 無駄な重複転送を防ぐことでファイル転送時間の短縮を可能にする．このとき，トポロジや帯域に応じて，複数受信者への転送経路および各経路に対するブロック分配の決定を適切に行う必要がある．既存手法では，副受信者に対してあらかじめ決められた順序に従い，途中でその順序を変更することなく転送ブロックを割当てるが，各フェーズで複数の副受信者に対し転送ブロックを割当てる際，受信者間の所持ブロックの違いが次フェーズ以降にも影響を与え，最終的な受信完了時間を悪化させる場合が存在した．

本報告では, この問題を解決するために, coded-MPMC と呼ばれる手法を提案する．その手法では, 送信者が事前にオリジナルブロックからリード・ソロモン符号化を用いて必要な数の符号化ブロックを生成し, 各フェーズで転送する．受信者はオリジナルブロック個数分の異なるブロックを受け取ることで, 元のファイルが復元可能となる．加えて, 大規模で複雑なネットワークに対して自身の最短の受信完了時間を実現する最適スケジュールを容易に発見するために, ヒューリスティックなブロック割当管理方法を導入する．また, coded-MPMC のための送信者, 受信者, Ryu フレームワーク [7] を利用した OpenFlow Controller(OFC) システムを実装し, Open vSwitch [8] を用いて OpenFlow エミュレータ (Mininet [9]) 上で動作させる．

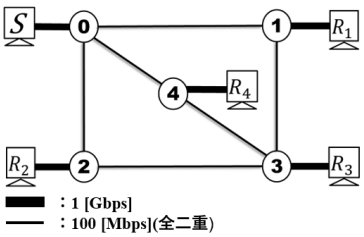

図 1　MPMC ファイル転送ネットワーク例

以下，2 節で MPMC ファイル転送とその問題点を説明する．3 節で提案手法である coded-MPMC について述べ, 4 節で実世界の大規模で複雑なネットワークに対するシミュレーション評価, 5 節で Mininet 上での動作検証を行う．最後に 6 節でまとめを行う．

2. MPMC ファイル転送

2.1 MPMC ファイル転送モデル

ファイルは、スイッチなどの中継ノードとそれらをつなぐ全二重リンクからなるネットワークを介して単一の送信者 S から N_R 人の受信者 R_i に転送される．また, MPMC ファイル転送のモデルを簡潔化するためにリンク間でロスは発生せず, S または R_i のいずれかに直接接続されているリンクは, ボトルネックにならないほど十分に大きいリンク容量を持っているとする．図 1 は, 単一の送信者, 4 人の受信者, および 5 つの中継ノードを有する OpenFlow ネットワークで, 中継ノードとホスト間で 1[Gbps] のリンク容量, 各方向の中継ノード間で 100[Mbps] のリンク容量を有する．

MPMC ファイル転送では, ネットワーク上の全リンク容量を利用するために, 送信者から受信者への max-flow 経路を使用してファイルが配信される．受信者が一人の場合は, 標準的な **max-flow 問題**を採用することで容易にその経路が導出されるが, 受信者が複数人の場合は, 同様に導出するのが困難である．そこで, 私たちは次小節で紹介する max-flow 問題に基づいてスケジュールを生成する貪欲なブロック割当方式を提案した．

この論文で考慮される重要な性能評価指標は各受信者の受信完了時間であり, それは送信者がファイルを少なくとも 1 人の受信者に送信し始めてから各受信者がファイルの内容全体を受信するまでの期間として定義される．b_{Si} を送信者 S から R_i への max-flow 経路の集約帯域, M_{Si} を単位帯域 B に対する集約帯域の比である **max-flow 量**とする．このとき, 各受信者の受信完了時間の下限値は, ファイルサイズを L とすると L/b_{Si} となる．ファイルは M_{Si} の最小公倍数である N 個のブロックに分割され, 各ブロックサイズは L/N となる．

図 2 は, 図 1 における S から R_3 への max-flow 経路を示しており, B は 100[Mbps], b_{S3} は 300[Mbps], M_{S3} は 3 となる．他の受信者も同様に導出され, M_{S1}, M_{S2}, M_{S4} は 2 となる．ゆえに, ファイルは 6 個のブロックに分割される．

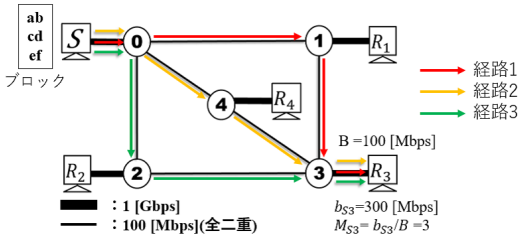

図 2　R_3 への max-flow 経路

2.2　MPMC ファイル転送スケジューリング

MPMC では, 1 フェーズもしくは複数フェーズを使用して単一の送信者が各受信者へブロックを配布する. 各フェーズで, 送信者は受信者の max-flow 経路を利用して複数のブロックを同時に転送すると共に, リンク容量を効率的に利用するためにブロックはマルチキャストを用いて転送される. OFC はネットワークトポロジ, リンク容量, 送信者と受信者の位置をあらかじめ把握しており, フェーズ毎にどのブロックをどの経路を利用して転送するかのスケジュールを決定する. このプロセスは**ブロック割当**と呼ばれ, 各受信者の受信完了時間を最小化することを目的とする. MPMC ファイル転送のスケジューリングアルゴリズムは以下のようになっている.

- 最初に, OFC は未受信者 R_i の最良転送完了時間 T_i を導出する. T_i は, R_i の未受信ブロックを max-flow 経路を用いて転送するのに必要な時間で, K_i を R_i の未受信ブロック数とすると, $\left(\lceil \frac{K_i}{M_{Si}} \rceil \times \frac{L}{BN}\right)$ で導出される. そして, 0 ではない T_i を持つ受信者の中で最小の値を持つのものが**主受信者** R_P として選択され, この期間 (**フェーズ**) で R_P の受信が完了するようにブロック割当が設計される. 同値の T_i を持つ受信者が複数いる場合は, **事前受信者優先度**と呼ばれるスケジューリングの入力として与える受信者順序を基に決定される.
- R_P 以外の受信者を**副受信者**と呼ぶ. 全ての副受信者は, ブロック割当の優先順序を決定するために, T_i に基づいて昇順にソートされる. 同値の T_i が複数存在する場合は, 事前受信者優先度を基に決定される. 決められた順序に従って, 未使用のリンク容量と未割当の副受信者がそのフェーズに存在する限り, 副受信者は max-flow 経路の 1 つに（部分的に）重なる可能性がある割当済み経路上の未受信ブロックが割当てられる. これにより, 主受信者の max-flow 経路は副受信者らに向かって分岐し, 複数のマルチキャストツリーが形成される.
- そのフェーズにおけるブロック割当が終了すると, 未受信者から次の R_P が選択され次のフェーズが開始される. 各受信者に対するこの貪欲なブロック割当方式は, 全受信者の受信が完了するまで繰り返される.

全受信者の受信完了時間が下限値となるスケジュールは, **最適スケジュール**と呼ばれる. 最適スケジュールもしくはそれに近いスケジュールを見つけるために, 事前受信者優先度をランダムに与えて様々なスケジュールを生成する.

2.3　MPMC ファイル転送例

ここでは, 図 1 のネットワークで MPMC ファイル転送を実行した際の例を紹介する. 事前受信者優先度 (R_1, R_2, R_3, R_4)

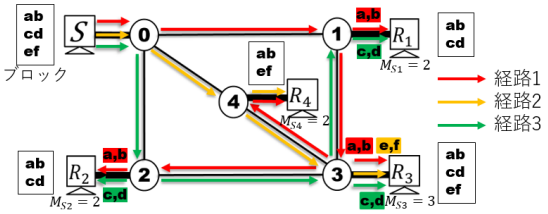

図 3　MPMC によるブロック割当 (1 フェーズ目)

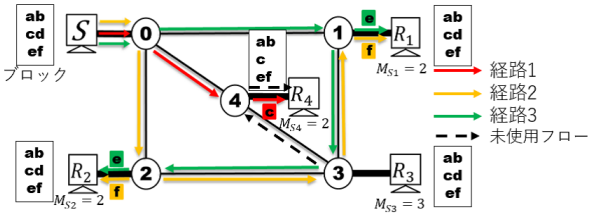

図 4　MPMC によるブロック割当 (2 フェーズ目)

が与えられているとする. まず, 各受信者の未受信ブロック数と max-flow 量から最良転送完了時間 T_i を導出すると $T_3 < T_1 = T_2 = T_4$ となるので, 事前受信者優先度に基づいてブロック割当順序 (R_3, R_1, R_2, R_4) が決定される. 送信者 S は, 主受信者 R_3 の max-flow 経路を用いてブロックを複数経路転送すると共に, 副受信者にマルチキャストでブロックを転送する. このとき, 各経路に乗せるブロックの数は $\lceil \frac{K_3}{M_{S3}} \rceil = 6/3 = 2$ 個となる. 図 3 は, 1 フェーズ目のブロック割当後の様子と各受信者の所持ブロックを示している.

R_1, R_2, R_3 の受信が完了していないので, 2 フェーズ目へと移行する. 1 フェーズ目と同様に, 最良転送完了時間と事前受信者優先度に基づいてブロック割当順序 (R_1, R_2, R_4) が決定される. S は主受信者 R_1 の max-flow 経路を用いて, 各経路にブロックを $\lceil \frac{K_1}{M_{S1}} \rceil = 2/2 = 1$ 個ずつ乗せて複数経路転送すると共に, 副受信者にブロックをマルチキャスト転送する. 図 4 は, 2 フェーズ目のブロック割当後の様子と各受信者の所持ブロックを示している. このとき R_4 は 6 つのブロックを受け取ることができていないため, 3 フェーズ目が開始される.

2.4　帯域効率的なブロック割当

R_1, R_2, R_4 の最良転送完了時間が同値であるにも関わらず, 2 フェーズ目で R_1, R_2 は受信を完了するが R_4 は完了しない. これは, 図 4 の (--→) で示される未使用フローが存在し, S から R_4 への max-flow 経路を完全に使用できていないことによる. この原因として, 前フェーズにおける各受信者の所持ブロックの違いが挙げられる. 図 3 の 1 フェーズ目における所持ブロックを確認すると, R_1, R_2, R_4 は 4 つのブロックを受信しているが, R_1, R_2 は (a, b, c, d), R_4 は (a, b, e, f) となっており所持ブロックが異なる. 2 フェーズ目の割当順序は (R_1, R_2, R_4) であるので, 主受信者 R_1 にブロック (e, f) を割当てた後, R_2 にもマルチキャスト転送を用いて同様のブロックを割当てる. 最後に, R_4 へのブロック割当を行う. R_4 の max-flow 経路を遡った際, 0 番のノードからブロック (c) を受け取ることができるが, 3 番ノードを経由する max-flow 経路に残余容量が存在しないのでブロック (d) を受け取ることができない.

あるフェーズで受信者 R_i が S から R_i への max-flow 経路

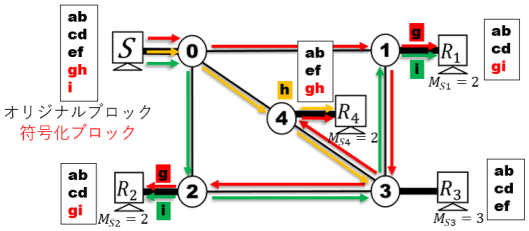

図 5　coded-MPMC によるブロック割当 (2 フェーズ目)

を完全に利用できた際, 受信者 R_i は**帯域効率的なブロック割当**を行えたといえる. つまり, 図 4 の R_4 に対して, 帯域効率的なブロック割当を行えていないといえる. 全フェーズで全受信者が帯域効率的なブロック割当を行うことができた場合, 送信者から受信者への max-flow 経路を全て同時に利用できたことを意味し, 全受信者のファイル受信完了時間は下限値に達する. MPMC ファイル転送では, 例として取り上げた図 1 のように, 帯域効率的なブロック割当が部分的に行われずに, 最適スケジュールが発見できないネットワークトポロジが存在する.

3.　coded-MPMC ファイル転送

3.1　coded-MPMC 概要

以前私たちは, coded-MPMC を提案した [6]. まずは, 今回紹介する新 coded-MPMC と旧 coded-MPMC の違いを述べる. 旧 coded-MPMC では, 送信者は XOR 符号化を用いて, 前フェーズでの各受信者の受信ブロックの違いを無くすために, オリジナルブロックを 2 つもしくはそれ以上組み合わせて符号化ブロックを生成する. しかし, 旧 coded-MPMC では図 1 のような簡易トポロジでは最適スケジュールが生成されるが, 大規模で複雑なネットワークトポロジでは生成されない, もしくは容易に生成されなかった. 旧 coded-MPMC とは対照的に, 新 coded-MPMC では旧手法で最適化されなかったネットワークトポロジで最適スケジュールが生成可能になる.

今回紹介する coded-MPMC では, 送信者は事前に N 個のオリジナルブロックから GF(2^8) 上のリード・ソロモン符号化を用いて必要な数の符号化ブロックを事前に生成する. N 個のオリジナルブロックは 1 フェーズ目に転送され, それ以降のフェーズでは全受信者が受信を完了するまで符号化ブロックが転送される. 各受信者は N 個の異なるブロックを揃えることで, 元のファイルを復元可能となる. この性質を利用することで, 前フェーズまでの各受信者の所持ブロックの違いを考慮せずにブロックを配布することが可能となり, 最適スケジュールの生成が容易になる.

1 フェーズ目を除いた他のフェーズでは, 送信者は事前生成されたブロックプールから必要数の符号化ブロックを選択し転送する. 基本的に各フェーズで転送されるブロックは今までのフェーズで配布されていないブロックであるが, そのフェーズまでで配布したブロックの中で未受信者が所持していないブロックがある場合はそれを選択する. このブロックの再利用は, 事前に生成する符号化ブロック数を削減し, エンコード時間の短縮に繋がる. さらに, あるフェーズで全受信者に対して帯域効率的なブロック割当を行えた際, 配布するブロックを変更し次フェーズでも同じブロック割当を使用する. これにより, 次フェーズ以降も帯域効率的なブロック割当が行なえるため, 最適スケジュールもしくはそれに近いスケジュールの生成がより容易となる.

図 1 のトポロジに対する coded-MPMC のブロック割当を示す. 1 フェーズ目のブロック割当は, 前節の MPMC と同様に行われる (図 3). 2 フェーズ目では, 1 フェーズ目で全受信者が max-flow 経路を完全に利用した帯域効率的なブロック割当を行えているため,　符号化ブロック (g, h, i) を用いて 1 フェーズ目と同じブロック割当を使用する (図 5). 送信者は全受信者が未所持である符号化ブロックを転送することで, 前節の MPMC では使用できなかった図 4 のフロー (⇢) が使用可能となり, 2 フェーズ目にファイル転送が完了する.

3.2　ブロック割当順序の変更

図 1 のような簡易トポロジの場合, coded-MPMC で最適化スケジュールを探索するために全てのスケジュールを確認するのは容易である. 一方で, 大規模で複雑なネットワークトポロジの場合, 同値の最良転送完了時間を持つ受信者が複数存在するので事前受信者優先度次第で様々なブロック割当順序が導出される. 私たちの考案したスケジューリング方法では, ある受信者へのブロック割当は後に割当が行われる受信者に影響を与えるマルチキャストツリーを構築してしまうことがある. これにより, ブロック割当順序次第で多様なスケジュールが生成されるので, 全てのスケジュールを確認するのは不可能である. そのため, 大規模なネットワークトポロジではブロック割当順序を管理することが重要となる.

ブロック割当順序が具体的にどのような影響を与えるかを述べる. あるスケジューリングにて, 同じ最良転送完了時間を持つ受信者集合を $\{R_1, R_2, ..., R_i, ..., R_{i+K}\}$ としたときのブロック割当順序が $(R_1, R_2, ..., R_{i+K})$ で, R_i のブロック割当に対して帯域効率的なブロック割当が実行されなかったときを考える. R_i がこのようなブロック割当となったのは, $R_j (j < i)$ のブロック割当が行われた際, R_j の max-flow 経路が R_i の max-flow 経路の一部と重なることによる影響が大きい. この事実は, ブロックが割当てられる受信者は後続の受信者のブロック割当に影響を与えるべきではないことを示唆する. それゆえに, coded-MPMC が各フェーズで帯域効率的なブロック割当を行えるようにする 3 つのヒューリスティックなブロック割当順序管理方法を導入する.

最適スケジュールもしくはそれに近いスケジュールを調査したところ, 各 max-flow 経路の平均長もしくは最大長の昇順にブロック割当が行われているものが多く存在した. この観測に基づき, “Path-length ascending order allocation ” と呼ばれる 1 つ目の手法が実装された. この手法では, max-flow 経路のリンク総容量が小さい受信者からブロック割当を行うために, ブロック割当順序は各受信者の max-flow 経路の平均長の昇順にソートされる. 大規模なトポロジでは, 平均長が同値の受信者が複数いる場合が考えられる. その場合は, 各受信者の max-flow 経路の最大長の昇順にソートされ, なお同値のものが存在する場合は事前受信者優先度に依存する.

表 1　coded-MPMC 比較方式

方式	調査する組み合わせ
(A)	Random order
(B)	Path-length ascending order allocation
(C)	(A) + Deprioritized reallocation
(D)	(A) + Prioritized reallocation
(E)	(B) + (C) + (D)

表 2　評価シナリオ

トポロジ	Columbus		Switch	
ファイルサイズ	100 [Mbytes]			
リンク容量	100 [Mbps]			
受信者	送信者を除く全ノード			
送信者 ID	31	44	35	69
下限値の平均	4.44[s]	4.44[s]	4.92[s]	4.92[s]

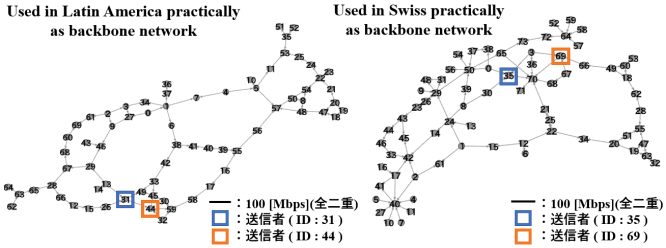

図 6　トポロジ：(左)Columbus (右)Switch

残り 2 つの手法は，ブロック割当順序の部分的な動的変更に着目する．帯域効率的なブロック割当を行えない受信者が存在するとき，ブロック割当順序の動的変更を行うことでスケジュールの探索範囲を拡張する．2 つ目の手法は，“Deprioritized reallocation ”と呼ばれ，帯域効率的なブロック割当を行えない受信者がいた場合は，同じ最良転送完了時間を持つ割当順序リストの末尾に置かれブロック割当が延期される．この手法が適応された R_i は R_{i+K} の後に割当が行われ，R_{i+K} 以前に割当てられた受信者の max-flow 経路からブロックがマルチキャストされることで帯域効率的なブロック割当を行えるようになる可能性がある．3 つ目の手法は，“Prioritized reallocation ”と呼ばれ，2 つ目の手法を用いた後に適用される．帯域効率的なブロック割当が行えなかった受信者を，そのフェーズで同じ最良転送完了時間を持つ割当順序リストの先頭に置いて同一フェーズをもう一度行う．この手法は，現段階の実装では不必要なループを避けるために一度のみ実行される．この手法が適用された R_i は，R_1 の前にブロック割当が行われ，他の受信者からブロック割当の影響を受けることなく帯域効率的なブロック割当を行えるようになる可能性がある．

4.　性 能 評 価

4.1　評 価 方 法

coded-MPMC 一対多ファイル転送とヒューリスティックなブロック割当管理方法の有用性を調査するために，表 1 の 5 つの方式を導入した coded-MPMC を比較する．“Random order ”は，最良転送完了時間と事前受信者優先度からブロック割当順序を決定する従来のスケジュール生成方法である．

私たちの手法で生成されるスケジュールは，事前受信者優先度に大きく影響する．そこで，各方式に同一の 1000 個の事前受信者優先度をランダムに与え，各スケジュールで生成される全受信者の受信完了時間を調査しその平均を導出する．全受信者の平均受信完了時間が下限値の平均値となるスケジュールは，最適スケジュールであることを意味する．

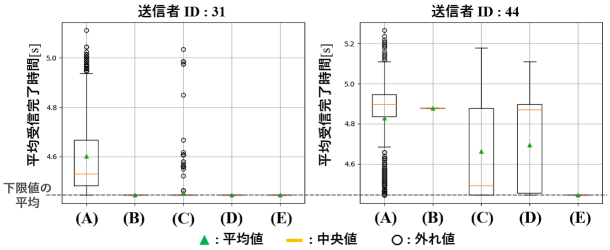

図 7　Columbus でのシミュレーション結果

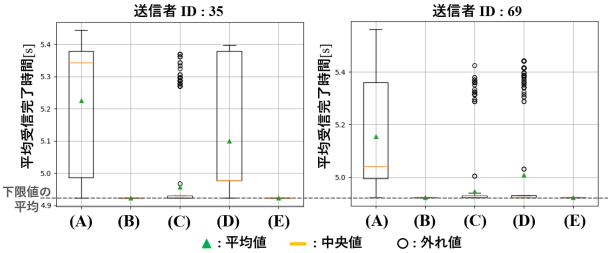

図 8　Switch でのシミュレーション結果

表 2 の評価シナリオを使用し，送受信ホストは各ノードにボトルネックにならないほど十分に大きいリンク容量で直接繋がっているとする．今回使用するトポロジは，従来手法では最適スケジュールが発見されなかった実世界のネットワークトポロジである図 6 の “Columbus ”と “Switch ”[10] を用いる．

4.2　評 価 結 果

図 7, 8 は，各方式を導入した coded-MPMC から生成されるスケジュールの平均転送完了時間の分布を箱髭図で表したものである．(B) を見ると，3 つのシナリオでは全スケジュールが最適化されているが，1 つのシナリオ (Columbus/送信者 ID:44) で最適スケジュールが全く生成されていない．これは，“Path-length ascending order allocation ”のみでは必ずしも最適スケジュールが生成されないことを意味し，ブロック割当順序の動的変更を行う必要性が示唆される．実際，ブロック割当順序の動的変更を行う (C) と (D) は (A) に比べて，各スケジュールの平均転送完了時間が大幅に改善されている．

(E) は，事前受信者優先度に依らず最適スケジュールが生成されており，3 つのヒューリスティックなブロック割当順序管理方法の有用性が伺える．また，副次的な効果として “Path-length ascending order allocation ”を用いることで割当順序の動的変更回数が減少し，最適スケジュールをより早く導出することが可能となる．なお，次数が 2 以上の全てのノードに対して，その位置を送信者とした場合の他の全受信者への一対多転送の性能を評価したところ，(E) を使用することで事前受信者優先度に依らず最適スケジュールが得られることを確認できた．

5.　coded-MPMC ファイル転送の実装

5.1　システム概要

ファイル転送を開始する前に，OFC は，トポロジ情報を取得するために LLDP パケットを全ての OpenFlow Switch(OFS)

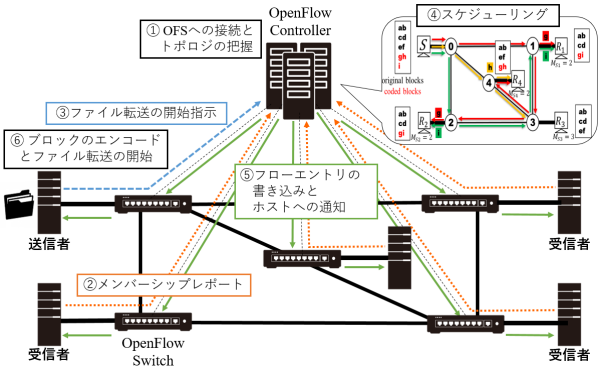

図 9　coded-MPMC ファイル転送の流れ

表 3　Mininet での評価結果

ファイルサイズ	100[Mbytes]		1000[Mbytes]	
平均受信完了時間 [s]	理論値	実験値	理論値	実験値
R_1	4.21	4.29	42.10	42.91
R_2	4.21	4.27	42.10	42.87
R_3	2.80	2.86	28.07	28.60
R_4	4.21	4.28	42.10	42.92

にフラッディングする．このとき, OFC はポートリンクアップ時のリンク容量を取得する．各受信者は, IGMP のメンバーシップレポートを利用して OFC に各受信者の位置を通知する．

続いて, 送信者は OFC にファイルサイズを通知する．これをトリガーにして, OFC は coded-MPMC のスケジュールを生成し, OFS に経路制御を行うためのフローエントリを書き込み, ファイルサイズとスケジュール情報を送信者と受信者に通知する．通知された情報に基づき, リード・ソロモン符号化を用いて必要な数の符号化ブロックを生成しパケット単位でファイル転送を開始する．ここまでの流れを図 9 に示す．

受信者は必要な数のブロックを受け取ったことを送信者に通知することで, 送信者はどの受信者が受信を完了したか把握することができる．最後に, 受信者は所持ブロックをデコードすることで元のファイルを復元する．

5.2　Mininet での評価

OpenFlow エミュレータである Mininet 上で, 図 1 のトポロジを用いて coded-MPMC 一対多ファイル転送の動作確認を行う．送信者は 95[Mbps] の CBR で 100 もしくは 1000[Mbytes] のファイルを送信し, 各受信者がファイルを受信する時間 (ブロックのエンコーディング・デコーディングや補助記憶装置へのファイル書き込み時間を除く) を測定する．

表 3 は, 各受信者の受信完了時間の理論値と平均値を比較したものである．理論値は, 2 節で紹介した受信完了時間 L/b_{Si} に CBR を考慮したものとなっている．比較すると, 理論値に近い実験値が計測されており実装に問題ないことがわかる．実験値が理論値に比べてわずかに遅い原因としては, 各パケットのデータがファイルのどの位置かを示すために追加される独自ヘッダによるオーバヘッドと, ゼロパディングによる送信するデータ総量の増加が考えられる．

6.　ま　と　め

本報告では, リード・ソロモン符号化を用いて必要な数生成された符号化ブロックを, 単一の送信者から MPMC フレームワークを利用して受信者に配布する coded-MPMC を紹介した．符号化ブロックを使用することで, 前フェーズの各受信者の未受信ブロックの違いを考慮せずブロック割当を行うことが可能となる．加えて, 全受信者に対して帯域効率的なブロック割当を行うために, ヒューリスティックなブロック割当順序管理方法を導入した．これらを組み合わせることで, 既存手法では最適化されなかった大規模で複雑なネットワークトポロジにおいて, coded-MPMC は最適スケジュールを容易に生成可能であるというシミュレーション結果を示した．最後に, OpenFlow エミュレータ上で coded-MPMC の動作検証を行った．今後は, 大規模な広域 SDN テストベッドである “RISE ” [11] で検証を行っていく．

今後の方針として, 私たちは同一ネットワーク上での複数の独立した一対多のファイル転送など, 動的な状況に適応するために MPMC フレームワークを拡張する予定である．また, 現在の予備的な coded-MPMC の実装の課題としてパケットロスに対するより柔軟な対応と, エンコーディング・デコーディングのオーバヘッドがあり, これらの改善を行っていく．

なお本研究は, JSPS 科研費 (JP16K00130) の助成および国立研究開発法人情報通信研究機構の委託研究により得られたものである．

文　　献

[1] K. Miller, K. Robertson, et al., “StarBurst Multicast File Transfer Protocol (MFTP) Specification,” Internet Draft, draft-miller-mftp-spec-03.txt , April 1998.
[2] S. Bhattacharyya, J. Kurose, et al., “Efficient rate-controlled bulk data transfer using multiple multicast groups,” IEEE/ACM Trans. Networking 11(6):895–907, 2003.
[3] Y. Zhao, J. Wu, and C. Liu, “On Peer-Assisted Data Dissemination in Data Center Networks: Analysis and Implementation,” Tsinghua Science and Technology, 19(1):51–64, 2014.
[4] M. Noormohammadpour, C. S. Raghavendra, et al., “QuickCast: Fast and Efficient Inter-Datacenter Transfers using Forwarding Tree Cohorts,” Proc. IEEE infocom’18, pp.225–233, 2018.
[5] A. Nagata, Y. Tsukiji, M. Tsuru, “Delivering A File by Multipath-Multicast on OpenFlow networks,” Proc. IEEE INCoS13, pp.835—840, 2013.
[6] K. Ogawa, T. Iwamoto, M. Tsuru, “One-to-many File Transfers Using Multipath-Multicast With Coding at Source,” Proc. IEEE HPCC’16, pp.687–694, 2016.
[7] Ryu SDN Framework, https://osrg.github.io/ryu/ (accessed on Feb 4, 2019.)
[8] Open vSwitch, https://www.openvswitch.org/ (accessed on Feb 4, 2019.)
[9] Mininet, http://mininet.org/ (accessed on Feb 4, 2018.)
[10] The Internet Topology Zoo, http://www.topology-zoo.org/ (accessed on Feb 4, 2018.)
[11] RISE ポータルサイト, https://www.jgn.nict.go.jp/rise/index.html (accessed on Feb 4, 2018.)