P4言語によるネットワークキャッシュに基づく効率的パケット再送 ―Coded-MPMC一対多ファイル転送への実装―

著者	倉田　真之, 柴田 将拡, 鶴 正人
雑誌名	電子情報通信学会技術研究報告. IN, 情報ネットワーク
巻	120
号	414
ページ	244-249
発行年	2021-02-25
その他のタイトル	P4-based efficient packet retransmission using in-network cache ― Introduction to Coded-MPMC one-to-many transfer ―
URL	http://hdl.handle.net/10228/00008382

一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IN2020-95(2021-03)

P4 言語によるネットワークキャッシュに基づく効率的パケット再送

— Coded-MPMC 一対多ファイル転送への実装 —

倉田　真之† 　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: †kurata.masayuki318@mail.kyutech.jp, ††{shibata,tsuru}@cse.kyutech.ac.jp

あらまし　データセンタ内や地理的に分散配置されたデータセンタ間で, 大容量のファイル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に対応するために, 高速かつ高効率な一対多ファイル転送の重要性が高まっている. そこで私たちは, 全二重ネットワーク上で各受信者が送信者からの最大フロー量を完全に利用して, 理論最小時間でファイル取得を完了するために, **Coded-MPMC**(送信者符号化を導入した複数経路マルチキャスト) による一対多転送を提案した. Coded-MPMC 転送は OpenFlow プロトコルを用いて実装されており, 各受信者が理論最小時間に近い時間でファイル取得を行えることが確認されている. しかしパケットロスが発生した場合, 送信者と受信者間で行われる再送処理に時間がかかってしまうことで取得完了時間が遅くなり, Coded-MPMC 転送全体のパフォーマンスが悪化してしまうという問題がある. そこでデータプレーンをプログラム可能な P4 言語を利用し, スイッチ自身がキャッシュしたデータを用いて再送応答を行う再送処理法を実装した. 本報告では, Coded-MPMC 転送にこの開発したスイッチを導入することで, 再送処理に要する時間の短縮や再送に利用されるネットワーク資源の節約を行えることを示す.

キーワード　P4 言語, 再送, マルチパス転送, マルチキャスト転送, 送信者符号化, 最大フロー量

P4-based efficient packet retransmission using in-network cache

— Introduction to Coded-MPMC one-to-many transfer —

Masayuki KURATA†, 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: †kurata.masayuki318@mail.kyutech.jp, ††{shibata,tsuru}@cse.kyutech.ac.jp

Abstract　Fast and efficient one-to-many file transfers are increasingly essential to deal with the rapid increase in traffic due to the sharing, replication, or movement of large-sized files and software within the data center and between geographically distributed data centers. Therefore, we proposed a one-to-many transfer using **Coded-MPMC** (Multipath Multicast with Sender coding) to complete the file retrieval in the theoretical minimum time by fully utilizing the max-flow value from the sender to each recipient over a fully-controlled network with full-duplex links. The Coded-MPMC transfer is implemented using the OpenFlow protocol, and we confirmed that each recipient could retrieve the file in the time close to its theoretical lower-bound time. However, when packet loss occurs, the retransmission process between the sender and the recipient slows down the retrieval completion time, and the performance of the Coded-MPMC transfer deteriorates. To solve this problem, using the P4 language which can program the data plane, we implemented the efficient retransmission method that the switch itself responds to retransmission requests using cached data instead of the sender. In this paper, we introduce the developed switch to the Coded-MPMC transfer, and show that the new retransmission allows to shorten the retrieval completion time of each recipient and save the network resources used for the retransmission when packet loss occurs.

Key words　P4 language, Retransmission, Multipath transfer, Multicast transfer, Sender coding, Max-flow value

1. はじめに

OpenFlow プロトコルは SDN (Software Defined Networking) 技術の実用的な手段であり, 爆発的に増加するデータトラヒック量に対応するために, データフローを集中的に制御し柔軟な経路変更等を行う. SDN は実際にデータセンタ内だけではなく, 地理的に分散配置されたデータセンタ間のトラヒックを制御するために Google [1] や Microsoft [2] で採用されており, 特別に SD-WAN と呼ばれている.

また, 単一の送信者から複数受信者へ大容量のファイルを共有するにあたって, ユニキャスト転送を使用する場合, 特定のリンクに同じデータが重複して転送されてしまうことがあるため, 非効率的なネットワーク資源の利用に繋がる. 一方で, マルチキャスト転送を使用する場合は, 複数の受信者に同時にデータを配信することでネットワーク資源を効率的に利用できる. これらの背景から, SD-WAN 上でマルチキャスト転送を利用した一対多転送の取り組みが数多く報告されている [3].

そこで私たちは, 全二重有線リンクを持つ OpenFlow ネットワーク上で, 各受信者が理論最小時間でファイル取得を完了するために提案された **MPMC** (Multipath Multicast) 転送 [4] に基づき, 送信者符号化を導入した **Coded-MPMC** 転送 [5] を提案した. Coded-MPMC 転送は MPMC 転送とは異なり, 数多くの大規模ネットワークトポロジ上でのシミュレーション評価を通して, 各受信者の取得完了時間を最小化できることが示されている. また, 日米間に渡る OpenFlow テストベッド上での検証を通して, 各受信者が理論最小時間の 1.02 倍以内の時間でファイル取得を完了できることが示されている [6].

しかしパケットロスが発生した際に, 従来手法では送信者と各受信者間で行われるユニキャスト転送を用いた再送処理を行っているため, 各受信者の取得完了時間が悪化してしまうという問題があった. そこで本報告では Coded-MPMC 転送全体のパフォーマンスを向上させるために, 各中継スイッチ自身がキャッシュされたデータを用いて, 送信者の代わりに再送応答を行うスイッチを P4 言語 [7] を用いて開発した. 本報告では, これらのスイッチを導入した Coded-MPMC 転送が, 再送処理を伴う取得完了時間の短縮と再送処理で利用されるネットワーク資源の節約が行えることを, 伝搬遅延とパケットロス率が各リンクに設定された Mininet [8] 上でのエミュレーションを通して示す.

2. Coded-MPMC 転送

Coded-MPMC 転送 [5] はマルチキャスト転送, マルチパス転送, 送信者符号化を統合した一対多転送法であり, 全二重リンクを持つネットワーク上で, 受信者 R_X が送信者 S からの最大フロー量 M_X を完全に利用したファイル取得により, 各受信者の取得完了時間を最小化することを目的としている. この手法では, 送信者はリード・ソロモン (RS) 符号を用いて, ファイル分割によって作られた N 個のブロック (オリジナルブロック) から K 個の符号化ブロックを生成する. その後, 送信者は各フェーズで適切なマルチキャストフロー集合を用いて, ブロックを集約されたリンク容量が M_X である複数経路 (最大フロー経路) を用いて R_X に対してブロックをマルチパス転送すると同時に, マルチキャスト転送を用いて中継ノードで複製したブロックを他の受信者に転送する. 最後に, 任意の異なる N 個のブロックを受信した受信者は, RS 符号を用いた復元処理によりファイル全体を取得する. 最適なブロック転送スケジュールでは, 各フェーズにおける未受信者は, 自身の最大フロー量を用いて異なるブロックを受信し, その結果として最小の取得完了時間でファイル全体を取得する.

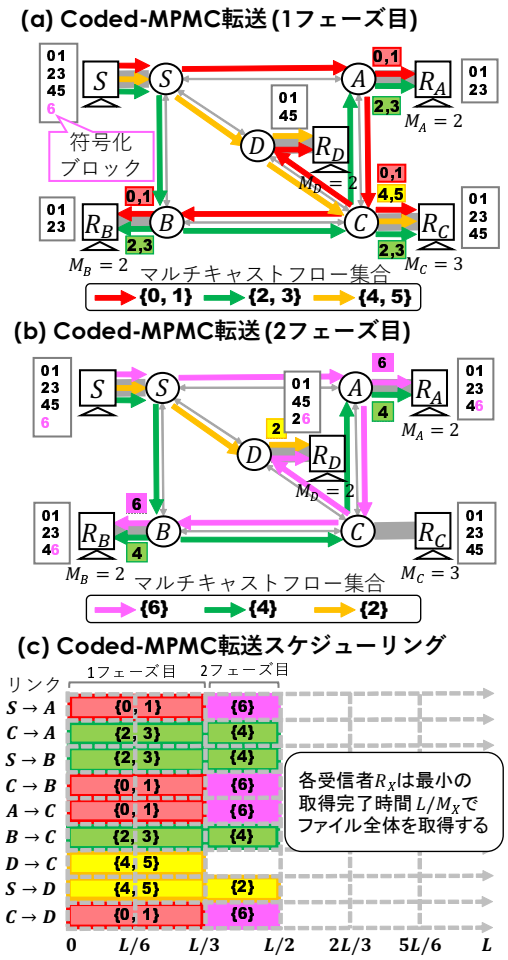

図 1　送信者 S から受信者 R_A, R_B, R_C, R_D への Coded-MPMC による一対多ファイル転送の例

図 1 に中継ノード間のリンク容量が 1, 中継ノードと送信者/受信者間のリンク容量が十分に大きい値に設定された全二重ネットワーク上での, 送信者 S から最大フロー量が 3 である受信者 R_C, 最大フロー量が 2 である受信者 R_A, R_B, R_D への Coded-MPMC による一対多ファイル転送の例を示す.

図 1(a) で示すように, S は全最大フロー量の最小公倍数 (この例では 6) 個のオリジナルブロックから, RS 符号を用いて 1 個の符号化ブロックを生成する. そして, ブロック集合 {0, 1}, {2, 3}, {4, 5} をフロー量が 1 の 3 つのマルチキャストフローを用いて配信する. このとき, 各受信者 R_X が M_X 個のマルチキャストフローを用いてブロックを受信している (M_X を完全に利用している) ため, 1 フェーズ目の Coded-MPMC 転送は最適であるといえる. R_A, R_B, R_D は 6 つの異なるブロックの

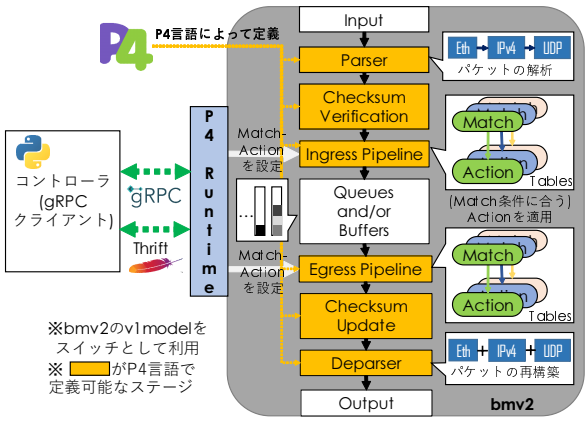

図 2　P4 言語と bmv2 の関係

受信が完了していないため, 2 フェーズ目に移行する.

2 フェーズ目では, 図 1(b) で示すように, ブロック集合 {6}, {4}, {2} をフロー量が 1 の 3 つのマルチキャストフローを用いて配信する. 2 フェーズ目も 1 フェーズ目と同様に, 各未受信者 R_X が M_X 個のマルチキャストフローを用いてブロックを受信しているため, 2 フェーズ目の転送も最適であるといえる. このとき, 全受信者が 6 個の異なるブロックの受信を完了したため Coded-MPMC 転送は終了する.

図 1(c) は Coded-MPMC 転送スケジューリングの一連のタイムチャートを表しており, 各受信者 R_X がサイズ L のファイルを理論最小値 L/M_X で取得できていることが分かる.

3. P4 言 語

P4 言語 [7] はネットワーク機器のデータプレーンをプログラムすることを目的に開発されたターゲットデバイス非依存の言語であり, パケット内のフィールドの書き換えや, ヘッダーの追加や削除, パケットへのメタデータの付加等を可能にする. 本実装では, オープンソースのソフトウェアスイッチである bmv2 [9] の v1model と呼ばれるアーキテクチャを P4 言語のコンパイル先として利用する. v1model は図 2 に示される手順で入力されたパケットの処理を実行し, 入力パケットがどのようなプロトコルから構成されているかの解析を行う Parser, チェックサムの検証を行う Checksum Verification, マッチ条件に合うアクションの適用やパケットの書き換えといった処理を行う Ingress Pipeline, Egress Pipeline, チェックサムの更新を行う Checksum Update, 処理されたパケットの再構築を行う Deparser といったステージから成り, 各ステージの処理機能を P4 言語で定義できる. また各 Pipeline で利用されるマッチ条件とそれに対応するアクションが定義されたテーブルは, P4Runtime と呼ばれる API から設定することが可能であり, P4Runtime に対して gRPC や Apache Thrift を用いて間接的に設定を行える.

近年 P4 言語を用いた研究開発が活発に行われている. 例えば, スイッチを通過するパケットにキューイング遅延といったスイッチ内部の情報を埋め込み, それらを収集することでネットワークの状態を監視する INT (In-band Network Telemetory) はその代表例である [10]. また, 送信された少数のパケットをスイッチで継続的にキャッシュし, リンク障害をスイッチ自身が検知した場合に適切な伝送経路に切り換え, キャッシュされたパケットを用いて迅速にパケットを回復する手法が検討されている [11].

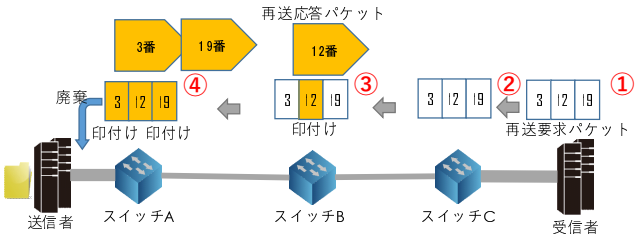

図 3　提案する再送処理法の例

4. 提案手法と実装モデル

4.1 従来の再送法と課題設定

Coded-MPMC 転送では, マルチキャスト転送を行うため L4 のプロトコルとして UDP が採用されている. UDP ではパケットの到達性が保証されていないため, パケットロスが発生した際は各ホストのアプリケーションが対応する必要がある. そこで Coded-MPMC 転送では, 受信者がパケットロスを検知した時, パケットロスによって受け取れていないと考えられる全てのデータの識別子をペイロードに格納した再送要求パケットを送信者へ送信する. これを受信した送信者は, 識別子に対応するデータをペイロードに格納した再送応答パケットを受信者へと送信する. これらの再送要求/応答パケットは（事前に）コントローラーが設定した送受信者間のユニキャスト経路上を転送される.

しかし, この再送処理はエンドツーエンドで行われているため, パケットロスが発生し得る環境で送信者と受信者が地理的に遠い位置に配置されている場合, 以下のような問題が発生することが考えられる.

- 一般的に送信者と受信者間の伝送距離やホップ数が大きい場合は, その間の再送要求/再送応答パケットの往来に時間がかかり取得完了時間が悪化する. また送信者と受信者の間の任意のリンク, もしくはスイッチで再送要求/再送応答パケットがロスすると, 更なる遅延を招く恐れがある.
- パケットをロスした受信者が多いほど, 再送応答/再送要求パケットが送信者付近のリンクに数多く往来することになる. そのため多くのパケットがロスした場合, 送信者付近のネットワークが混雑する可能性がある.

4.2 提 案 手 法

本報告では, 前節で挙げた課題を解決するために, (i)Coded-MPMC 転送のパケット/再送応答パケットに含まれるペイロードデータをキャッシュする, (ii) 再送要求されたデータをキャッシュしている場合は, 送信者の代わりにスイッチ自身が再送応答を実施する, (iii) 再送要求パケットが要求する全てのデータの再送応答が完了した場合は, 再送要求パケットが上流のスイッチ又は送信者へと転送されないようにするために破棄する, といった機能を持ったスイッチを P4 言語で開発した. このスイッ

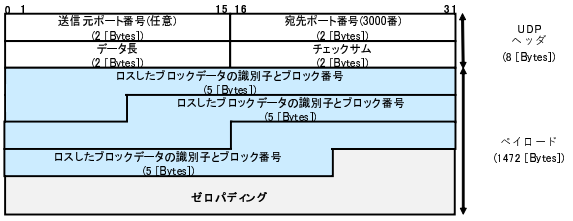

図 4　再送要求パケットのフォーマット

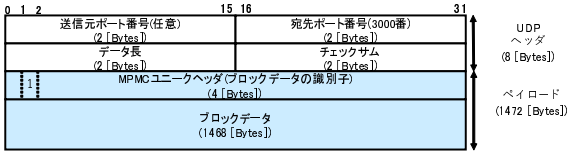

図 5　再送応答パケットのフォーマット

チを導入した際の再送処理手順の例を図 3 で説明する．

⓪　送信者から受信者への Coded-MPMC 転送が行われる．送信者/受信者間に位置するスイッチ A, B, C は中継できた全てのパケットのペイロードデータをキャッシュする．

①　ここでは受信者が 3 番, 12 番, 19 番のパケットのロスを検知したとする．受信者はこれらのパケットの回復を図るために, 1 つの再送要求パケットにロスしたパケットの番号を全て詰め, 送信者宛てに再送要求パケットを送信する．

②　再送要求パケットを受信したスイッチ C は, 該当番号を持つデータがキャッシュされていないかを調査する．このときスイッチ C はデータをキャッシュしていなかったため, 再送要求パケットは何も処理されず上流に転送される．

③　再送要求パケットを受信したスイッチ B は, 12 番のパケットが転送するデータをキャッシュしていたため, 再送要求パケットを 1 つクローンする．そして, クローンされたパケットを 12 番のデータを転送する再送応答パケットに変更し受信者に送信する．まだ全ての番号への再送応答が完了していないため, スイッチ B は 12 番の再送応答を行ったことを上流のスイッチに知らせるために印をつけ, 再送要求パケットを転送する．

④　再送要求パケットを受信したスイッチ C は, 印が付加されていない 3 番と 19 番のパケットが転送するデータをキャッシュしていたため, 再送要求パケットを 2 つクローンする．そして, クローンされた 2 つのパケットをそれぞれ 3 番と 19 番のデータを転送する再送応答パケットに変更し受信者に送信する．その後, 再送応答を行った番号に印を付加する．このとき, スイッチ C は全ての番号に印が付加されていることを確認できたので, これ以上の再送処理は不要であると判断し再送要求パケットを廃棄する．

4.3　パケットフォーマット

再送処理に利用する UDP パケットのフォーマットを説明する．図 4 は再送要求パケットのフォーマットを表しており, 宛先ポート番号が 3000 番に設定されている．ペイロード 1472[Bytes] にはロスしたブロックデータの識別子 4[Bytes] とブロック番号 1[Byte] が 5[Bytes] 単位で, ロスしたパケットの数個分詰め込まれる．ここで, オリジナルブロック, 生成した符号化ブロックを含む全 $N+K$ 個のブロックを順に並べて 1 つの仮想ファイルとみなすとき, ブロックデータの識別子は仮想ファイル内の当該パケットの位置, ブロックは当該パケットが含まれるブロック番号を意味している．

また, 図 5 は再送応答パケットのフォーマットを表しており, 宛先ポート番号は 3100 番に設定されている．ペイロード 1472[Bytes] の先頭 4[Bytes] にブロックデータの識別子が格納されており, 以降の 1468[Bytes] にはブロックデータが格納されている．Coded-MPMC 転送パケットも宛先ポート番号以外は同様のフォーマットであり, 各スイッチがブロックデータ 1468[Bytes] をキャッシュする際はこの識別子に紐づけられる．

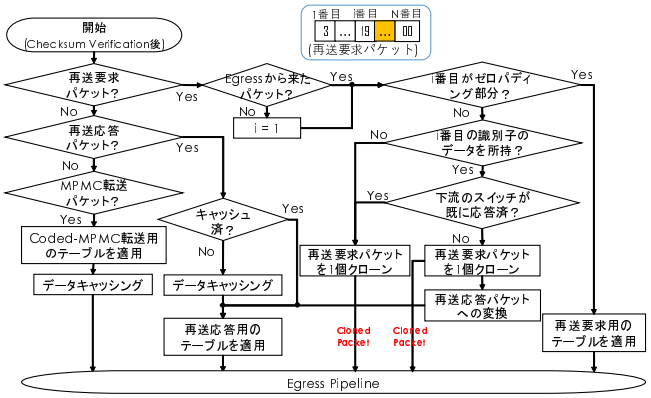

図 6　Ingress Pipeline での論理的処理

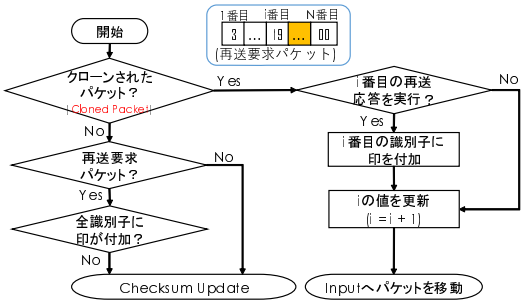

図 7　Egress Pipeline での論理的処理

表 1　Ingress Pipeline に設置されているテーブル

Coded-MPMC 転送用のテーブル		再送要求/再送応答用のテーブル	
マッチ条件	アクション	マッチ条件	アクション
· 入力ポート · 宛先 IPv4 アドレス (マルチキャストアドレス) · UDP 宛先ポート番号	出力ポート の決定	· 送信元 MAC アドレス · 宛先 MAC アドレス	出力ポート の決定

また, 図 5 は再送応答パケットのフォーマットを表しており, 宛先ポート番号は 3100 番に設定されている．ペイロード 1472[Bytes] の先頭 4[Bytes] にブロックデータの識別子が格納されており, 以降の 1468[Bytes] にはブロックデータが格納されている．Coded-MPMC 転送パケットも宛先ポート番号以外は同様のフォーマットであり, 各スイッチがブロックデータ 1468[Bytes] をキャッシュする際はこの識別子に紐づけられる．

4.4　各ステージでの処理

4.2 節で述べた再送処理法を実装するために, P4 言語で各ステージの処理をどのように定義したかを説明する．なお, Checksum Verification, Checksum Update, Deparser ステージでは, 3 節で述べた標準的な処理しか行っていないため説明を省略する．

Parser ステージでは, 入力パケットが下位レイヤー (L2) から順々に解析され, Ethernet, IPv4, UDP の各フィールドの情報が抽出される．その後, Coded-MPMC 転送パケット, 再送要求パケット, 再送応答パケットの宛先ポート番号は, それぞれ 5000 番台, 3100 番, 3000 番であるため, その情報を基にどのパケットが入力されたかを識別しペイロード部分の解析を行う．

Ingress/Egress Pipeline では図 6, 7 で表す手順で, 表 1 に示

すテーブルを用いてそれぞれのパケットが処理される．Coded-MPMC 転送パケットや再送応答パケットは Ingress Pipeline でマッチ条件に合うアクションが適用され，スイッチ内にブロックデータが識別子と紐付けてキャッシュされる．これらのパケットは Egress Pipeline で特に処理は行われず，次のステージへと移動する．

一方で，初めてスイッチに入力された再送要求パケットは，パケットのメタデータとして何番目の識別子を参照するかを示す i に 1 がセットされ，ペイロードの先頭 5[Bytes] 中に格納されている識別子のブロックデータがキャッシュされているか，下流のスイッチがその識別子への再送応答を既に実施しているかを確認する．そして再送応答を実施する/しないに関わらず，先頭以降 ($i > 1$) の識別子を確認するために，再送要求パケットをクローンし Egress Pipeline へと移動させスイッチ内部に留めておく．再送応答を実施する場合，入力パケットは Ethernet，IPv4 の送信元アドレス，宛先アドレスのスワップといった処理を経て再送応答パケットへと変換された後，再送応答パケット用のテーブルが適用される．

Egress Pipeline へと移動したクローンされた再送要求パケットは，先頭 5[Bytes] に格納されている識別子の再送応答を行った場合，5 バイト目に 0xff を印として付加する．そして，次の 5[Bytes] に格納されているであろう識別子に対して再送応答が可能であるかを調査するために，$i(= 1)$ の値を 1 つプラスし Input へ移動させる．これにより，今度は再送要求パケットの 2 番目の識別子が確認され，2 番目の識別子に対する再送応答が可能であるかが調査される．この Egress Pipeline から Ingress Pipeline への再送要求パケットの移動は，再送要求パケットに格納されている識別子を全て確認するまで行われる．確認した i 番目の 5[Bytes] に識別子が格納されておらずゼロパディング部分であった場合は，再送要求パケット用のテーブルを適用し Egress Pipeline へと移動させる．最後に再送要求パケットの各識別子に印が付加されているかどうかを調査し，全ての識別子に印が付加されていることが確認されたらスイッチ内で廃棄する．

なお，本検証で P4 言語のコンパイル先として利用している bmv2 [9] は，Egress pipeline から Ingress Pipeline への移動 (recirculate と呼ばれる) の際に，付加したメタデータをパケットと共に移動させることができないため，パケット内部で i の値を保持させる必要がある．そこで本検証では，i を保持する代替手段として IPv4 フィールドの TTL を利用している．

4.5 Coded-MPMC 転送モデル

Coded-MPMC 転送モデルは OpenFlow プロトコルを用いて実装されているが [5]，bmv2 では標準で OpenFlow プロトコルをサポートしていないため，Packet-In や Packet-Out といったスイッチ，コントローラ間のパケットの移動を，gRPC と Python 言語で作成したコントローラを用いて実現し，図 8 で表す手順で Coded-MPMC 転送を実行する．また [5] とは異なり，送信者は全受信者からのファイル取得完了通知を受信した後，コントローラに対して転送完了通知を送信し (⑦)，コントローラは各スイッチにキャッシュされているデータを全て消去

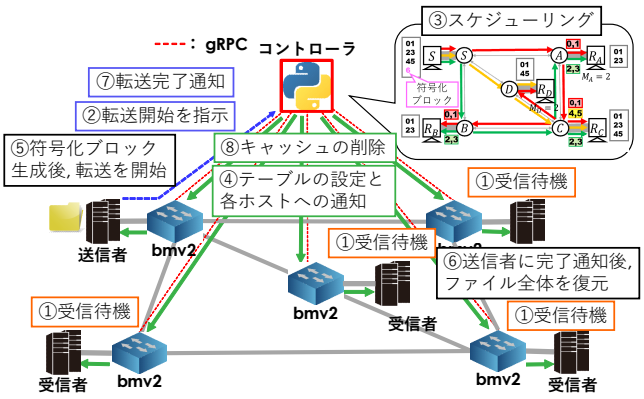

図 8　gRPC を用いた Coded-MPMC 転送モデル

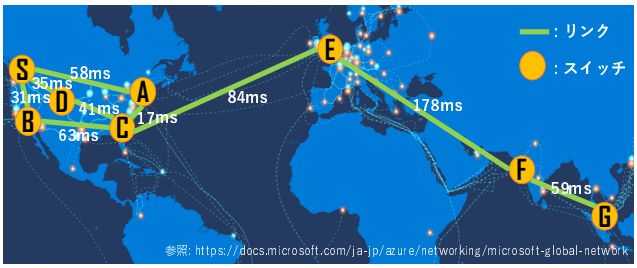

図 9　Mininet 上にビルドするネットワークトポロジと伝搬遅延

する (⑧)．これにより，転送完了後にネットワーク上にキャッシュされたデータが残り続けることを防ぐ．

5. エミュレーション評価

5.1 評価シナリオ

図 9 は本検証で利用する Mininet [8] 上のネットワークトポロジを表しており，各リンクには文献 [12] を基に設定した伝搬遅延とパケットロス率 0.3%が設定されている．スイッチⓈには送信者ホストが接続されており，サイズ 1[MB] のファイルを CBR が 100[kbps] であるマルチキャストフローを用いて，スイッチⒸに接続された最大フロー量が 300[kbps] の受信者 R_C，スイッチⒶに接続された最大フロー量が 200[kbps] の受信者 R_A，スイッチⒻ，Ⓖに接続された最大フロー量が 100[kbps] の受信者 R_F，R_G に配信する．各受信者は初めて Coded-MPMC 転送パケットを取得し始めてから，全パケットを取得するまでの時間を取得完了時間として計測する．そのため，符号化ブロック生成時間やファイルの復元時間は取得完了時間に含まれていない．

また本検証では，各受信者は再送処理の経路として送信者から各受信者への最短経路を利用する．そのため R_A は経路Ⓢ⇔Ⓐ，R_C は経路Ⓢ⇔Ⓐ⇔Ⓒ，R_F は経路Ⓢ⇔Ⓐ⇔Ⓒ⇔Ⓔ⇔Ⓕ，R_G は経路Ⓢ⇔Ⓐ⇔Ⓒ⇔Ⓔ⇔Ⓕ⇔Ⓖを用いて再送処理を行う．各受信者の再送経路は S から R_G への再送経路に重なっているため，スイッチは再送応答に他のスイッチからの再送応答パケットのキャッシュも利用できる．つまり，後から再送処理を行う受信者 R_F, R_G にとって特に，キャッシュによる恩恵を受けやすい経路設定となっている．

5.2 エミュレーション結果

本検証ではエミュレーションを実行するたびに，各リンクでロスするパケット数が異なるため，結果として 10 回試行した際の

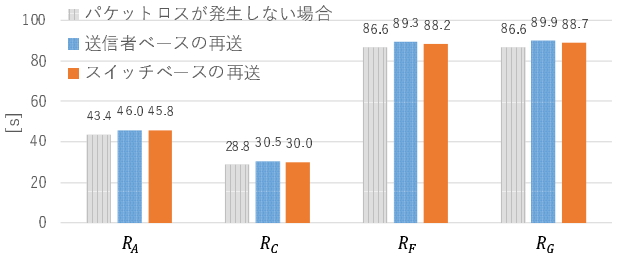

図 10　各受信者のファイル取得完了時間

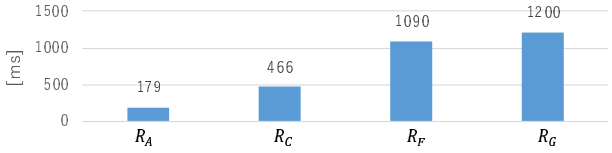

図 11　送信者ベースとスイッチベースの再送の取得完了時間の差

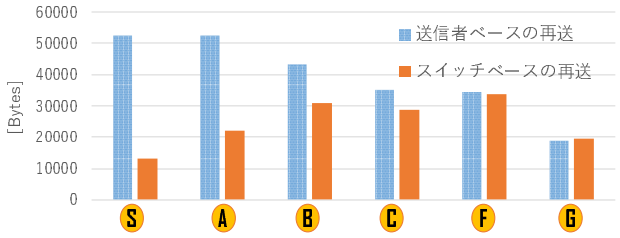

図 12　スイッチから出力された再送要求/応答パケットの総バイト数

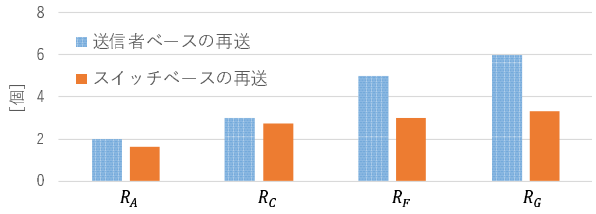

図 13　各受信者が受信した再送応答パケットの平均ホップ数

平均値を採用している．図 10 は各受信者の取得完了時間を表しており，従来の送信者ベースの再送法，提案するスイッチベースの再送法共に，パケットロスが発生しなかった際の取得完了時間に近い値でファイル取得が完了していることがわかる．このことは，送受信者ホストのアプリケーションで実装されている再送プログラムに問題ないことを示唆している．しかし，図 11 で示すように，送信者ベースの再送を行った際とスイッチベースの再送を行った際の各受信者の取得完了時間を直接比較すると，送信者から地理的に遠い (伝搬遅延が大きい) 位置にある受信者ほど，取得完了時間の差が大きく表れていることが分かる．

続いて，再送処理に利用されたネットワーク資源に着目する．図 12 はスイッチ内で新たに生成された再送応答パケットも含む，送信者ベースの再送処理とスイッチベースの再送処理でスイッチから出力されたパケットの総バイト数を示している．送信者ベースの再送では全受信者の全ての再送要求パケットと再送応答パケットがスイッチⓈとⒶを通過するため，スイッチⓈとⒶから出力されるパケットの総バイト数が最も大きい値となっている．一方でスイッチベースの再送では，図 13 で示すように各中継スイッチが再送要求されたデータをキャッシュしていれば，送信者まで再送要求パケットを転送せずにスイッチ自身が再送応答を行うため，再送応答パケットの平均ホップ数が少なくなっている．そのため，再送処理による送信者付近のリンクの利用量を大幅に減らすことができている．

これらの実行結果から提案するスイッチベースの再送法は，再送処理を伴う各受信者の取得完了時間の短縮，送信者付近のネットワーク資源の節約を実現できていることが分かり，提案する再送処理法の有用性が伺える．

6. ま と め

本報告では，Coded-MPMC による一対多転送の再送処理法に着目し，パケットロスが発生した際の取得完了時間の短縮と再送処理によって利用されるネットワーク資源の節約を目的として，P4 言語を用いてスイッチ自身が送信者の代わりにキャッシュしたデータを用いて再送応答を行う再送処理法を開発した．そして，実ネットワークでのファイル転送を想定した Mininet 上でのエミュレーションを通して，実装したスイッチの有用性と実現可能性を示した．今後の方針として，高速にパケットの処理が可能なハードウェアスイッチでの検証を考えている．

なお本研究は，JSPS 科研費 (JP20K11770) 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「高信頼設計エッジ・クラウド・ネットワーク」により得られたものである．

文　献

[1] S. Jain, *et al.*, “B4: experience with a globally-deployed software defined wan,” *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3—14, Aug. 2013.

[2] S. Kandula, I. Menache, *et al.*, “Calendaring for wide area networks,” *Proceedings of the 2014 ACM conference on SIGCOMM*, vol. 44, no. 4, pp. 515–526, Aug. 2014.

[3] S. Islam, *et al.*, “A Survey on Multicasting in Software-Defined Networking,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, p. 355–387, Nov. 2017.

[4] A. Nagata, Y. Tsukiji, and M. Tsuru, “Delivering a File by Multipath-Multicast on OpenFlow Networks,” *2013 5th International Conference on Intelligent Networking and Collaborative Systems*, pp. 835–840, Sep. 2013.

[5] M. Kurata, *et al.*, “Minimizing One-to-Many File Transfer Times using Multipath-Multicast with Reed-Solomon Coding,” *2019 31st International Teletraffic Congress (ITC 31)*, pp. 115–116, Aug. 2019.

[6] M. Kurata, M. Shibata, and M. Tsuru, “Experiments of Multipath Multicast One-to-many Transfer with RS coding over Wide-Area OpenFlow Testbed Network,” *Proceedings of the 1st International Conference on Emerging Technologies for Communications (ICETC)*, 4 pages, 2020.

[7] P. Bosshart, *et al.*, “P4: Programming protocol independent packet processors,” *ACM SIGCOMM Computer Communication Review*, vol.44, no. 3, pp. 87–95, 2014.

[8] “Mininet: An Instant Virtual Network on your Laptop (or other PC),” [Online]. Available: http://mininet.org/, Accessed on: Jan. 27, 2021

[9] “BEHAVIORAL MODEL (bmv2),” [Online]. Available: https://github.com/p4lang/behavioral-model, Accessed on: Jan. 27, 2021

[10] Kim, Changhoon *et al.*, “In-band network telemetry via programmable dataplanes,” *ACM SIGCOMM*, vol. 15. 2015.

[11] T. Qu, R. Joshi *et al.*, “SQR: in-network packet loss recovery from link failures for highly reliable datacenter networks,” *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pp. 1–12, Oct. 2019.

[12] “Global Ping Statistics,” [Online]. Available: https://wondernetwork.com/pings, Accessed on: Jan. 27, 2021