融合情報学輪講資料 2014/07/25

Multipath TCP 適用時のデータセンターネットワークでのショートフローに対する改善

Improving the effect of the short flows in the data center network with Multipath TCP

工学系研究科 電気系工学専攻 関谷研究室 修士課程 2 年 37-136482 藤居 翔吾

Abstract-
As increasing the number of servers in a datacenter, the effective network topology for utilization of massive computer clusters has been studied. Recently, using MPTCP for the beneficial use has been tackled this problem. As another solution, distributed processing system enabled to handle big data. Today's the technique of distributed processing is that massive workers process data followed by the query from the master nodes. As the result, a large quantity of short flow is generated in a datacenter network. MPTCP can achieve the effective consumptions of the resources with multipath, but a researcher reported MPTCP causes the delay of flow completion for short flows. In this paper, I verified the MPTCP's effects for short flows in the datacenter network with the traffic of distributed processing service. There are two negative factors, bottleneck at aggregation switch and multiplexed flow patterns in one host, and I clarified short flow delays arise under what circumstances.

Index- **Multipath TCP, Short flow, Flow completion time, FatTree topology, Datacenter network**

1 Introduction

多様な端末がネットワークに接続できるようになり, 様々な端末から大量かつ多種多様なデータの取得が可能となった. 特にトラフィックデータ量の増加傾向は顕著で, 18∼24 ヶ月単位で総データ容量が 2 倍になるという予測がされている [1]. また Facebook では, 300 ペタバイト以上のデータ量を保有しており, 1 日あたりに 1 ペタバイトのデータを解析している [3]. このように近年では, クラウドによるビッグデータの活用が着目され, ウェブ検索エンジンや SNS(Social Networking Service) など, リアルタイムに近いレスポンスを返すような場面において使われ始めている. そのようなクラウドサービスには近年, より良いユーザーエクスペリエンスの要求が高まってきており, Amazon では 100[ms] の遅延により売り上げが 1% 下がる, といった報告 [2] があるように, レスポンス遅延の影響は深刻な問題となっている. そのため, 大規模データをより高速に解析することが求められており, データセンターではサーバの運用台数が増加の一途をたどっている. しかし, 大量の計算機資源から最大限の性能を引き出すためには, 従来の仕組みではデータセンター内トラフィックに対してトラフィックが集中する問題に対応できないため, 計算機資源を有効活用するための研究が盛んに行われている [4–12]. そのようなスケーラビリティ拡大には, ネットワークトポロジー, アプリケーション, プロトコルに対するアプローチがある.

ネットワークトポロジーを改良するアプローチでは, 従来の単純な階層構造では, データセンター内で発生するトラフィックに対して帯域が最大限割り当てられない [6]. そのため, 近年ではそのようなトラフィックに対してスイッチを多段に構成することで帯域を有効利用するトポロジーが提案されている.

大量のデータの処理速度を改良するアプローチでは, 分散処理のために partition-aggregate 計算モデルが提案されている. MapReduce [4], Dryad [5] 等の分散処理フレームワークは, この計算モデルに従っており, 今日の大規模クラウドサービスにおいては, 必要不可欠である.

プロトコルを改良するアプローチでは, 従来の TCP を拡張した Multipath TCP (MPTCP) [14] をデータセンターネットワークに用いる提案がされている [6–8]. MPTCP

を用いることにより, 複数の経路を同時に利用し, スループットを向上させることが期待されている.

しかし, 分散処理フレームワークを用いることで, フローサイズの小さい大量の Query が発生し, MPTCP は, フローサイズの小さいトラフィック (ショートフロー) に対しては, TCP よりも性能が劣化する問題が報告された [10].

このような背景から, 大規模データセンターネットワークへの MPTCP 適用時のショートフロー遅延の問題は並列分散処理アプリケーションの性能の面で深刻な問題である. そこで本論文では, データセンターネットワークにおけるショートフローに対する MPTCP の影響を検証し, その原因を分析する. 特に, ns-3 シミュレーションを用いたトラフィック解析結果を示し, どのような状況でショートフローが遅延するのかその要因を示す.

2 関連研究

本章では, これまでに報告されている複数経路利用によるフロー完結時間短縮化技術について簡潔に述べ, その優位性や問題点を示す.

2010 年に Alizadeh らによって, データセンターネットワーク特有のトラフィックパターンに特化して, パラメータを決定するアルゴリズムが提案された [9]. データセンター特有のバースト性のあるトラフィックが引き起こす問題点として, キューの生成によるキューイング遅延, キュー溢れによるパケットロス, スイッチのバッファに掛かる負荷がある. これらの問題に対し, キューの蓄積を制御するためのバッファサイズをアルゴリズムから動的に設定することで, 大部分のキューの伝送時間を短縮することを可能にした. これらの問題に対し, 経由するスイッチにおいて, ECN(Explicit Congestion Notification) によってエンドホストに輻輳を通知し, キューの大部分が占有される前にウィンドウサイズを動的に変化させ, キューサイズを小さく保つアルゴリズムによって, 大部分のキューの伝送時間を短縮することを可能にした. しかし, 大規模計算資源を想定したトポロジーにおける検証がされておらず, また各ネットワークデバイスに細かなチューニングを必要とするため, 大規模データセンターでは運用面での問題がある. さらに, サイズの大きいフローの割合の大きいトラフィックの中では, その影響によりウィンドウサイズを小さくする制御が働くため, サイズの小さいフローの完結時間への効果が小さい, と報告されている [12].

2011 年に Costin らによって, MPTCP を用いたデータセンターネットワークモデルが提案された [10]. 近年の大規模計算資源を有効活用するために提案されたネットワークトポロジーでは, 高性能なデバイスや特殊な機器を必要とせず, 汎用デバイスのみを用いてホスト同士の通信の際に経路が複数用意されている. これまでは通信に使わない経路をセカンダリ経路として利用することで, 耐障害性を持たせていたのに対し, 提案されたデータセンターモデルでは, MPTCP を用い複数経路を同時に利用する事で, 耐障害性を保ちながら, 帯域を最大限利用する事を可能にした. また, 様々なトポロジーに MPTCP を適用することで, 従来の TCP よりも高いスループットが出せることを示した. しかし, サイズの小さいフロー ($\leq 70KB$) のフロー完結時間に着目すると, TCP よりも時間がかかるという問題点があった.

2012 年に Zars らによって, 複数レイヤー間でトラフィックを監視し, しきい値を設定することによるフロー完結時間の短縮化技術を提案した [11]. サイズの異なるフローが混在するネットワークにおいては, サイズが小さいフローがサイズの大きいフローに圧迫され, 伝送遅延が大きくなる問題があったが, この提案では, データリンク層からアプリケーション層までの各層が, 相互にトラフィックを監視する機能をスイッチに実装し, 優先度をつけ, バッファサイズを調整することで, フロー完結時間の悪化を抑えることを可能にした. しかし, 実験では Click [13] を用いて実装を行っており, 現実世界での全てのネットワーク機器の置き換えが必要となるので, 実現は難しい.

以上で述べたことをまとめると, 近年のデータセンターネットワークに対して, 以下のような要求が考えられる.

- 大規模計算機を有効活用するトポロジーの利用
- 分散処理の際に発生する大量のサイズの小さいフローの送信時間の短縮
- 特殊な実装やデバイスを用いず, シームレスな運用の実現

3 データセンターネットワーク

本章では, データセンターネットワークを構成する技術に関して, その概要を述べる.

3.1 Multipath TCP

MPTCP は, 一つの経路でデータ転送する TCP を拡張し, 複数のインタフェース, あるいは複数のポートを用いてデータ転送をするプロトコルである [14]. クライアントが複数の IP アドレスを持っていた場合, 新たにサブフロー[*1]のコネクションが確立される. 追加されたサブフローは, クライアントの持つインターフェースが 1 つの場合, 同じ IP アドレスで異なる送受信ポートを用いる. インターフェースを複数持つ場合には, 異なる IP アドレスの組み合わせで通信を行う. ルーティングに関しては, 複数の宛先 IP アドレス, 送信元アドレスからそれぞれ経路決定される. このように, アプリケーション層より下のレイヤーのみで複数の経路を使ってデータ転送を行うため, アプリケーション側が MPTCP での通信を意識することなくデータ転送ができる.

MPTCP では, サブフローが, それぞれのシーケンス領域を持ち, 経路状態に合わせて輻輳制御をする Coupled Congestion Control Algorithm がある [15]. このアルゴリズムには, TCP と同様に AIMD(additive-increase and multiplicative-decrease) による輻輳制御がサブフロー単位で行われる. 以下に AIMD アルゴリズムを示す.

- サブフロー r において, 1ACK ごとにウィンドウサイズ ω_r を $\min(\frac{\alpha}{\omega_{total}}, \frac{1}{\omega_r})$ 増加させる.
- サブフロー r において, パケットロス時にウィンドウサイズ ω_r を $\frac{\omega_r}{2}$ へ減少させる.

ここで, ω_{total} は全てのサブフローのウィンドウサイズの総和, α は送信速度の増加量を示すパラメータで, 以下のように定義される [15].

$$\alpha = \omega_{total} \times \frac{\max_r \frac{w_r}{RTT_r^2}}{(\sum_r \frac{w_r}{RTT_r})^2} \quad (1)$$

ここで, RTT_r はサブフロー r でのラウンドトリップ時間を示している. MPTCP での輻輳制御には二つの性質ある. 一つは, サブフローのウィンドウサイズは, 全てのウィンドウサイズの大きさに依存するということである. これにより, 混雑したサブリンクにおいては, ウィンドウサイズが抑えられ, ロードバランスができる. 二つ目は, MPTCP のアルゴリズムによって, TCP での輻輳制御よりも悪化する事を回避している事である. しかし, もし複数のサブフローがそれぞれ混雑のないサブリンクを利用する場合, いずれかのコネクションが帯域を占有する可能性がある.

[*1] 複数の TCP コネクションの内, ある一つのコネクションにおけるフロー

3.2 FatTree トポロジーと MPTCP によるデータセンターモデル

この節では, データセンターを構成する要素について述べる.

3.2.1 トポロジー

従来のデータセンターモデルでは, Host が Edge スイッチにつながり, これらのスイッチが Aggregation スイッチに集約され, core スイッチに接続するといったように, 階層的にトポロジーを形成していた [6]. このような単純な階層構造を持つトポロジーは, トラフィックの大部分がデータセンター外の通信には有効であった. しかし, 今日のようなデータセンター内で生じるトラフィックが大半を占める場合, 帯域の割当が適切でなくなる. このような, データセンター内のトラフィックが主であれば, 階層型のトポロジーはボトルネックを引き起こす可能性がある. 近年の研究 [6–8] では, トラフィックがデータセンター内に集中した時の問題を, 物理的なアプローチとして, トポロジーを工夫する事で解消を試みている.

図 1 のように, FatTree [6] では, Core スイッチを複数用いる事で, 物理パスの最大帯域を供給する. また, 比較的狭い帯域の経路と汎用的な性能のスイッチを多数用いる.

このようなトポロジーを用いる事で, データセンター内のトラフィックに対し, 帯域を十分に使う事ができる. しかしこのような密な配置により, 複数の経路が形成され, ルーティングをどのように決定すべきかという問題も生じる事となる [10]. 例えば図 1 のような FatTree トポロジーでは, 4 通りの経路が考えられる. これら複数の経路をリンクエラー時の冗長性を持たせる目的だけでなく, 性能向上に活用することが求められている.

3.3 データセンターにおけるトラフィックシナリオ

大量の計算機資源を有効活用するためには, 並列分散処理フレームワークを用いられ, 一般的に図 2 に示すように partition-aggregate 構造をとる. 並列分散処理フレームワークでは, 多数の処理ノードと分散処理の制御をする

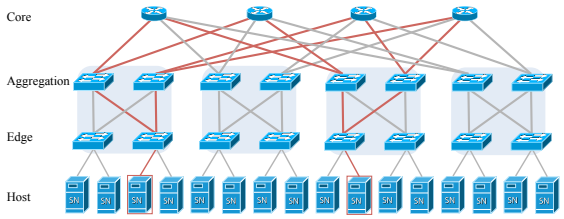

Fig.1　Fattree topology

管理ノードから構成されており, 管理ノードから Query が発行され, 分散処理がそれを受け取り, レスポンスを返す. このとき, トラフィックパターンが (1)*Query traffic*, (2)*Short message traffic*, (3)*Backgroung traffic* の 3 つに分類される [9].

Query traffic. Query traffic とは, 大規模計算処理を分割して並列処理する際に, aggregator ノードから処理ノードへ具体的な処理を割り当てるためのトラフィックである. Query traffic の特徴は, 非常に小さいフローサイズ (2KB∼20KB) で、処理全体の遅延に非常に強く影響を及ぼす事である. また並列分散処理システムの構成上, Query traffic は ms∼ μs 単位で Query が生成され, バースト性を持つと言える [9].

Short message traffic. Short message traffic とは, 処理ノードの動作を制御するためのトラフィックである. Short message traffic の特徴は, フローサイズは 50KB∼1MB で, Query traffic と同様に処理全体の遅延に影響を及ぼすという事である. しかし, Querry traffic ほどのフロー数は生成されず, 生成時間間隔も秒単位である.

Backgroung traffic. Backgroung traffic は, 各処理ノードへ更新データを送信するトラフィックである. Backgroung traffic の特徴は, フローサイズが 1MB∼50MB と大きいことにある. さらに, その生成時間間隔は大きい. また, Backgroung traffic での更新データは, 処理精度の向上に寄与するが, 処理に必須ではないので, 処理全体の遅延にはつながらない. また, Alizadeh らは, 実際のデータセンターのトラフィックでは, Background traffic が発生する数は少ないが, 全体の通信量の大部分が Backgroung traffic によって占められていると報告しており, 経路全体へ影響を及ぼす可能性があることを示している [17].

つまり, データセンターネットワークのトラフィックのうち, 分散処理開始時に生成される Query traffic が遅延すると, 処理全体に対し遅延を引き起こすので, Query traffic のフロー完結時間は極めて重要なメトリックである.

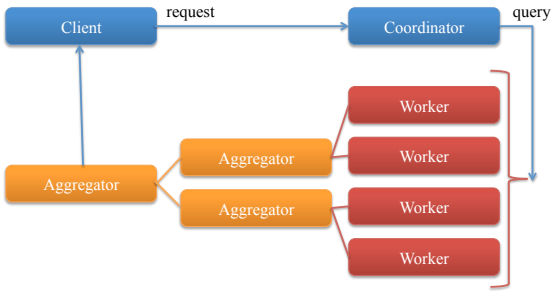

Fig.2　Partition-aggregate model on distribution system

3.4　スイッチ

現在用いられている汎用的なスイッチ機器では複数のフローを多重に扱うための共有メモリを持つ. そして共有メモリプールから MMU(Memory Management Unit) によって各インターフェースが利用できるメモリ量を動的に割り当てる事で, 複数の通信を公平に処理する事を目指す. しかし, 比較的安価なスイッチでは制御できるメモリ量が制限されているため, 様々な性能障害を引き起こす [21].

3.4.1　Incast

図 3(a) に示すように, 短期間に一つのインターフェースへとフローが集中した場合, 用意されているバッファを使い果たし, 最悪の場合パケットロスを引き起こす. これは, 3.3 小節で示した partition-aggregate 構造によるもので, リクエストを受けた処理ノードが同期して一斉にレスポンスを返すことにより, そのレスポンスを集約して受け取るアグリゲートホストが接続しているスイッチでのポートのキューサイズが大きくなる. こうした問題に対して, アプリケーションレベルにおいては二つのアプローチがある. 一つは, レスポンスのサイズを意図的に小さくし, スイッチバッファの圧迫を抑えることである. もう一つは, それぞれのリクエストにジッタを混ぜる事で, レスポンスを同期させないことである [22]. さらに, パケットロスを生じた際へのアプローチとしては, RTO_{min} を小さくする事でパケットロスの影響を抑える事ができる.

3.4.2 Queue buildup

3.3 小節で示したように, 並列分散処理のレスポンスには直接影響しない Backgroung traffic は, スイッチバッファにパケットロスを引き起こすほどの影響を及ぼし, そのポートがボトルネックとなる可能性がある. 図 3(b) に示すように, Backgroung traffic と Query traffic が同じポートを利用する場合, 大きく二つの性能障害を引き起こす. 一つは, 上記に示した Incast 問題によるパケットロスである. もう一つは, サイズの大きいフローによるショートフローのキューイング遅延が生じる, Queue buildup 問題である. この障害は, Incast 問題とは無関係であり, パケットロスは生じないため RTO_{min} による改善はできない. さらに, 多数の同期された Query traffic も必要としない. そのため, キューイング遅延の問題に対する唯一の解決策は, キューサイズをなるべく小さく保つことにより, キューに溜まったパケットを素早く排出する事である.

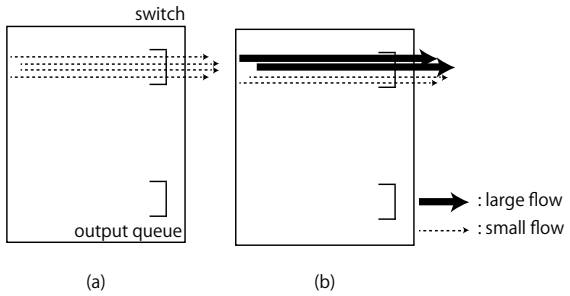

Fig.3 Two ways in which flows interact on a multi-ported switch resulting in performance problems.

4 検証実験

これまでの研究において報告された MPTCP によるショートフロー性能劣化の問題 [10] を受け, その再現実験を行うことにより, 原因を解析し, 二つの要因を明らかにした [20]. 一つ目は, MPTCP は TCP よりも多くのトラフィックを排出する事で, パケットロスを伴う遅延が生じるということ. 二つ目は, MPTCP の実装上の問題から, 最大二つの経路しかデータ転送に利用できないということである. これらの結果を受けて, MPTCP によるバックグラウンドトラフィックが利用している経路を回避し, 比較的輻輳が起こっていない経路を適切に選ぶ事で, ショートフローのフロー完結時間 (FCT) が改善できるのではないかと, 仮説を立てた. この章では, シミュレーション実験を用いて, その仮説の検証, また MPTCP によるショートフローの遅延がどういった状況で発生するのかを明らかにする.

4.1 再現シミュレーション実験環境

ネットワークトポロジーには, 図 4 のような FatTree を部分的に抽出した 2:1 にオーバーサブスクリプションされたトポロジーを用いる. このトポロジーでの物理パスでは, 一つのサブフローが 1 本の物理パスを占有するように, 設計している. すなわち, 4 つのサブフローを使う場合, ホストの 4 インターフェースに対しそれぞれ 4 つ IP アドレスが割り当てられる. また, Host-Edge 部分には, IP アドレスの数だけインターフェースを用意し, Aggregation-Edge 部分も, それに従いインターフェースを追加する. さらにルーティングに関しては, Core1∼Core4 に分散するようにルーティングテーブルを設定した. ベンチマークトラフィックについては, 二つのペアに対してホスト同士の 1 対 1 通信を用いている. 一方のペアに対しては, MPTCP により継続してデータ転送 (バックグラウンドトラフィック) を行う. 他方のペアに対しては, TCP による 2∼70Kbyte のデータ転送 (ショートフロー) を毎 200[ms] のポアソン生起させ, 転送完了までにがかかった時間 FCT(Flow Completion Time) を計測している.

今回の再現実験には ns-3 Direct Code Execution [16] を用い, MPTCP は, Linux カーネルソースを用いた [18]. 図**??**に, シミュレーションで用いた FatTree(k=2) トポロジーを示す.

表 1 に再現シミュレーション環境に対する各パラメータをまとめる.

Parameter	Value
Nodes	4
MPTCP	v0.88
Link aggr-edge	200Mbps
Link edge-host	100Mbps
RTT	0.5ms
Receive buffer size	50KB

Table.1 Testbed on verificating simulation

4.1.1 設定パラメータに対する有効性の検証

伝搬遅延については RTT(Round Trip Time) として, 0.5[ms] に設定した. これは, 一般的なデータセンター内の

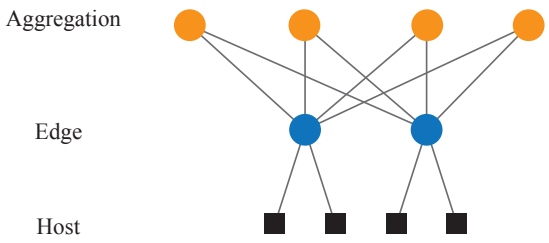

Fig.4　Network topology on verificating simulation

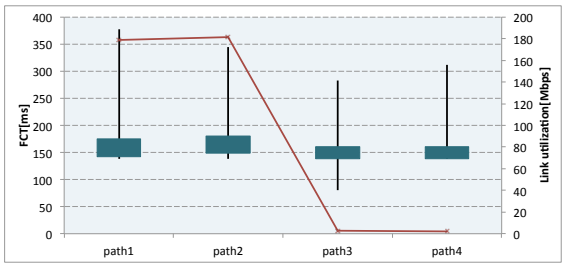

Fig.5　Flow completion time and link utilization for 70kb benchmark traffic

RTT が 1[ms] 以下であるためである [19].

バッファサイズについては, 以下の帯域幅遅延積 (BDP) の式から, 200Mbps×4 を最大限利用できるだけの値を設定した.

$$BDP[\text{byte}] = \text{帯域幅 [bps]} \times RTT \div 8 \qquad (2)$$

各帯域については, 4 のノードを使って輻輳を引き起こす現象を再現するために, 実際のデータセンターのような広帯域のネットワークと比べ, 狭い帯域を設定した.

4.2　検証結果

図 5 に上記の実験環境での結果として, 70KB のショートフローの FCT とそれぞれの経路の利用率を示す. この結果から, background traffic よるリンク利用率の高い path1, path2 において, FCT の分散が大きくなっている事が分かる. これは, 大部分のフローは遅延が生じる事なく完結できるが, 遅延が生じる割合, またその遅延量が大きくなることを示している. 実際に, 図 6 においてはフローサイズを変え, FCT の大きかった下位 5% に着目したグラフであるが, 利用率の大きい path1, path2 と比較し, path3, path4 におけるショートフローの FCT は, 最もサイズの小さい 2[kb] フローにおいても 10[ms] 程度の差があった.

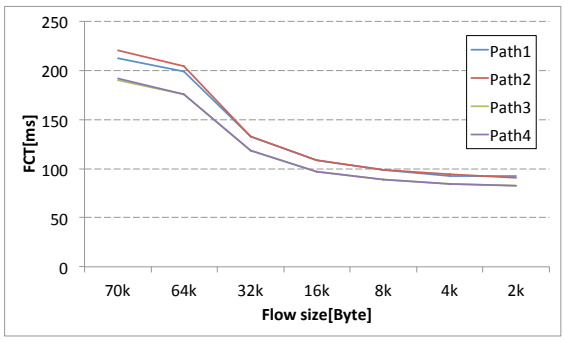

Fig.6　95 percentile FCT for 70kb flow

また, 図 7 では平均 200[ms] でポアソン生起するフローの発生間隔と遅延が生じるフローの関係を示している. この結果から, フローの発生間隔が短く, 複数のフローが同時刻に混在する状況において, 遅延が生じる事が分かる.

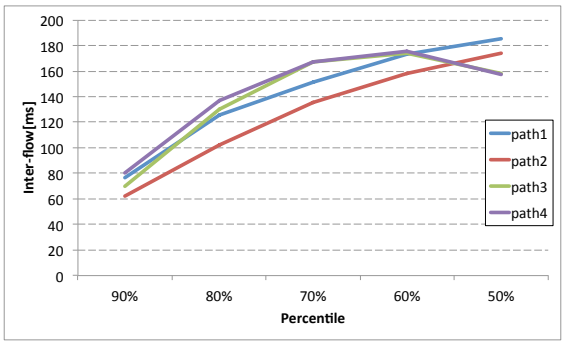

Fig.7　Inter-flow with percentile for 70kb benchmark traffic

これらの事から, 遅延を引き起こす要素としては, 「リンク利用率」, 「フロー発生間隔」があると考えられ, それぞれ, Queue buildup, Incast を引き起こす要因となる.

4.2.1　リンク利用率の影響

リンク利用率による遅延の影響がどのように生じているのかを解析するため, 各スイッチでキャプチャした pcap データを基にして, バックグラウンドトラフィックの有無によるパケット発生間隔の変化の解析を行った. pcap データには path1 における 70kb フローを用いた. すなわち, バックグラウンドトラフィック有の場合には, 一つの経路に二つのフローが混在する事になる. 図 8 にパケット発生間隔の差分の累積を示す. この差分の値が大きければ大きいほど, そのホストにおけるキューイング遅延が大きかった事を示している. バックグラウンドトラフィックが流れ

ている時の Sender ホスト-Edge スイッチ間でのパケット発生間隔の変化から, Edge スイッチのキューイング遅延がボトルネックとなっている事が分かり, Queue buildup での性能障害が起きていることが分かる.

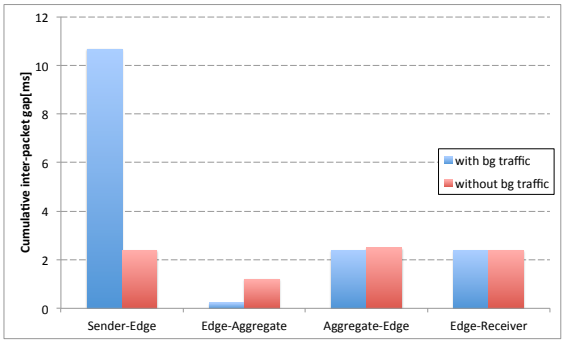

Fig.8　Inter-flow with/without bankground traffic for 70kb benchmark traffic

4.2.2　フロー発生間隔の影響

短いフロー発生間隔におけるフロー多重化による遅延の影響がどのように生じているのかを解析するため, 各スイッチでキャプチャした pcap データを基にして, よるパケット発生間隔の変化の解析を行った. pcap データには 70kb フローのみを用い, フロー発生間隔は 50, 200[ms] でそれぞれの FCT を比較した. 図 9 にフローが完結するまでの通信の様子を示す. 発生間隔 200[ms] のフローでは, 滞りなくデータ転送を 130ms 程度で完了し, 後続のフローとは競合しないことが分かる. 発生間隔 50[ms] のフローでは, 段階的にデータを送り, 待ちが発生している様子が分かるが, これは, 待ちの間に他のフローの通信を行っているからであり, インターフェース単位で見ると常に一定したデータを送信している. そのため, 各フロー毎の完結時間の遅延が生じる.

4.3　考察

これらの解析結果から, 3.4 小節にて示した二つの機能障害が引き起こる要因について述べ, 今後の改善手法の検討を行う.

Queue buildup: フローサイズが大きく, 長時間データを通信し続けるバックグラウンドトラフィックの影響によって中継するスイッチのキューサイズは大きく保たれてしまい, そのポートにおけるキューイング遅延が生じる. その中でフローサイズが小さく, 本質的には短時間で完結できるフローが, 同じポートを使って通信する事で, キューイング遅延やキュー溢れによるパケットロスの影響を受け, フロー完結時間が大きくなる. そうした遅延の影響が引き起こる為には, 同時に多数のショートフローは必要でなく, バックグラウンドトラフィックがキューを圧迫している状況で, 他のフローが流入する事により生じる.

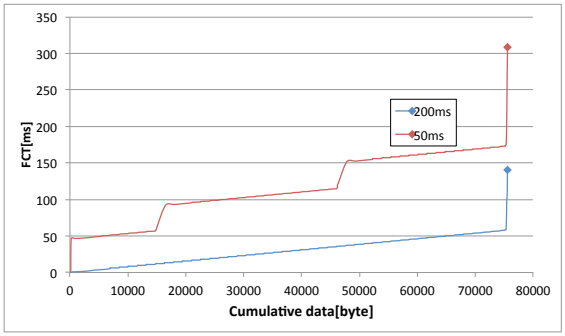

Fig.9　Detail receiving 70kb flow

Incast: サイズの小さいフローが多数生成され, それらのフローが短期間にアグリゲートホストに流入しようし, アグリゲートホストが接続されているスイッチのポートにトラフィックが集中する事で, バッファが圧迫される事によりキューイング遅延, パケットロスが生じる結果, それぞれのフローの完結時間が大きくなる. そうした遅延の影響が引き起こる為には, 各ホストにおけるショートフローの発生間隔が小さい時に, 複数のフローが同時刻に通信を行うことにより生じる.

こうした遅延の影響を軽減する為には, 通信を行うスイッチポートのキューサイズを小さく保ち, 通信を行う事が重要である. 具体的には, 図 5 に示すように, 同じスイッチでも異なる物理ポートを用いる, あるいは Raiciu らが提案している Dual-Homed FatTree のような Edge スイッチを複数設けたトポロジーを用いる事で, 物理的に異なるスイッチを中継する事でキューが圧迫の問題を回避できると考える [10]. こうした経路の切り替えを既存の仕組みを用いて実現する事を考えると, MPTCP の持つ輻輳制御アルゴリズムに対して新たなアルゴリズムを提案する必要があると考える. 実際, MPTCP 輻輳制御に関しては議論が行われており, TCP との親和性と経路状況への対応とのトレードオフの関係性から様々なアルゴリズム提案されてい

る．上記のような遅延の問題を解決する為に, MPTCP の輻輳制御を用いて経路を切り替える事を実現するならば, 従来の制御よりも経路の状況により機敏に対応できる経路切り替えに特化した輻輳制御アルゴリズムが必要であると考えられる．

5 あとがき

本論文では, サイズが大きく長時間通信を行うバックグラウンドトラフィックとサイズが小さくフロー完結時間を短く抑えたいショートフローが混在する状況において, フローサイズの小さいトラフィックに対しては, 従来の TCP よりもデータ転送に時間がかかるという報告 [10] を再現し, その原因を分析した．その結果, ショートフローが通信で用いるリンクの利用率とフローが発生する間隔の二つの要因がショートフローの遅延に影響を及ぼす事を示した．特に, アグリゲートスイッチにおけるキューイングのボトルネックになっており, その問題に対して通信経路を切り替える事によりフロー完結時間を改善した事を示した．このことから, キューサイズが大きくなっているスイッチポートを回避するような経路において通信を行う事を MPTCP の輻輳制御によって実現する事で, フローサイズの小さいトラフィックの完結時間を短縮化し, 並列分散処理のパフォーマンスの改善が期待される．

今後は実機を用いてキューイングの問題を再現し, それを回避するネットワークトポロジーと MPTCP の輻輳制御の提案を行う．

参考文献

[1] 日本アイ・ビー・エム株式会社. IBM 第 1 章大容量データのバックアップ, http://www-06.ibm.com/systems/jp/storage/column/backup/01.html

[2] Jim Liddle. Amazon found every 100ms of latency cost them 1% in sales, August 2008. http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/

[3] Facebook. Presto: Interacting with petabytes of data at Facebook, https://www.facebook.com/notes/facebook-engineering/presto-interacting-with-petabytes-of-data-at-facebook/10151786197628920

[4] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[5] Isard, Michael, et al. "Dryad: distributed data-parallel programs from sequential building blocks." ACM SIGOPS Operating Systems Review 41.3 (2007): 59-72.

[6] Al-Fares, Mohammad, Alexander Loukissas, and Amin Vahdat. "A scalable, commodity data center network architecture." ACM SIGCOMM Computer Communication Review. Vol. 38. No. 4. ACM, 2008.

[7] Guo, Chuanxiong, et al. "BCube: a high performance, server-centric network architecture for modular data centers." ACM SIGCOMM Computer Communication Review 39.4 (2009): 63-74.

[8] Greenberg, Albert, et al. "VL2: a scalable and flexible data center network." ACM SIGCOMM Computer Communication Review. Vol. 39. No. 4. ACM, 2009.

[9] Alizadeh, Mohammad, et al. "Data center tcp (dctcp)." ACM SIGCOMM Computer Communication Review 40.4 (2010): 63-74.

[10] Raiciu, Costin, et al. "Improving datacenter performance and robustness with multipath TCP." ACM SIGCOMM Computer Communication Review. Vol. 41. No. 4. ACM, 2011.

[11] Zats, David, et al. "DeTail: Reducing the flow completion time tail in datacenter networks." ACM SIGCOMM Computer Communication Review 42.4 (2012): 139-150.

[12] Alizadeh, Mohammad, et al. "pfabric: Minimal near-optimal datacenter transport." Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. ACM, 2013.

[13] Kohler, Eddie, et al. "The Click modular router." ACM Transactions on Computer Systems (TOCS) 18.3 (2000): 263-297.

[14] Ford, Alan, et al. TCP Extensions for Multipath Operation with Multiple Addresses: draft-ietf-mptcp-multiaddressed-03. No. Internet draft (draft-ietf-mptcp-multiaddressed-07). Roke Manor, 2011.

[15] Raiciu, C., M. Handley, and D. Wischik. "Coupled congestion control for multipath transport protocols." draft-ietf-mptcp-congestion-01 (work in progress) (2011).

[16] Inria「DCE - GETTING STARTED Direct Code Execution」http://www.nsnam.org/~thehajime/ns-3-dce-doc/getting-started.html

[17] Benson, Theophilus, Aditya Akella, and David A. Maltz. "Network traffic characteristics of data centers in the wild." Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010.

[18] ip networking lab「MultiPath TCP - Linux Kernel implementation」http://mptcp.info.ucl.ac.be/

[19] Vasudevan, Vijay, et al. "Safe and effective fine-grained TCP retransmissions for datacenter communication." ACM SIGCOMM Computer Communication Review. Vol. 39. No. 4. ACM, 2009.

[20] 藤居 翔吾, 田崎 創, 関谷 勇司, "MultiPath TCP 適用時のデータセンターネットワークでのフローサイズが与える影響に関する一考察", 電子情報通信学会, 信学技法, vol. 113, no. 364, IA2013-65, pp. 47-52, 2013.P. Agarwal, B. Kwan, and L. Ashvin. Flexible buffer allocation entities for traffic aggregate containment. US Patent 20090207848, August 2009.

[21] P. Agarwal, B. Kwan, and L. Ashvin. Flexible buffer allocation entities for traffic aggregate containment. US Patent 20090207848, August 2009.

[22] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. IEEE/ACM ToN, 1994.

[23] A. Walid, et al. Balanced Linked Adaptation Congestion Control Algorithm for MPTCP draft-walid-mptcp-congestion-control-00, 2014.