融合情報学輪講資料 2013/12/6

Multipath TCP によるデータセンターネットワークの改善
Improving the datacenter 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 1 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, we verified the utility of the datacenter network with MPTCP, and consider the effects of MPTCP by flow size.**

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, SNS 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, 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 [4–11]. 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) [13] 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 適用時の問題点を把握するために, フローサイズの小さいトラフィックに対する性能を検証し, 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, TCP 適用時との比較を行う.

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]. データセントラフィックの引き起こす問題点として, キューの生成による遅延, 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. 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.

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 [12] 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, 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 [13]. 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 [14]. 輻輳制御には, 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 [14].

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

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.

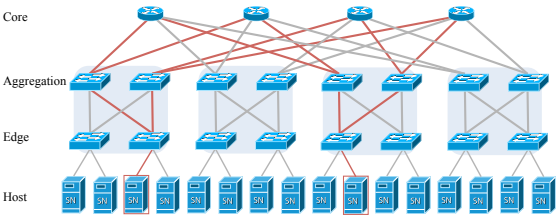

Fig.1　Fattree topology

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, 分散処理フレームワークを用い, 一般的に図 2 に示すように partition-aggregate 構造をとる. 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 [16].

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.

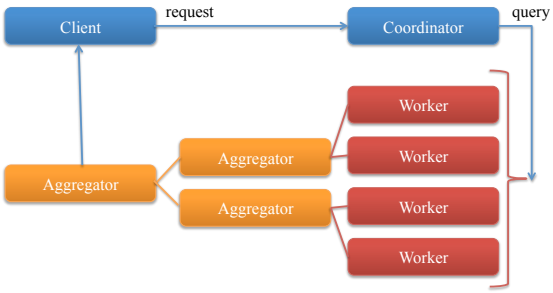

Fig.2　Partition-aggregate model on distribution system

4　再現シミュレーション実験

この章では, Raiciu らによって示した FatTree-MPTCP ネットワークモデルでのフローサイズの小さいトラフィックに対する性能評価シミュレーションを再現し, 解析を行った結果を示す.

4.1　再現シミュレーション実験環境

Raiciu らは [10] において, 各プロトコルがフローサイズの小さなトラフィックに対して及ぼす影響の評価を行い, フローサイズの小さいトラフィックに関しては, MPTCP によりフロー完結時間を遅延させることを示した. そのときのシミュレーション環境は, 以下の通りである. ネットワークトポロ

ジーには, 4:1 にオーバーサブスクリプションされた FatTree を用いている. ベンチマークトラフィックについては, ホスト同士の 1 対 1 通信を用いている. 全てのホストのうち, 33% を TCP または MPTCP により継続してデータ転送 (Back-ground traffic) を行う. 残りの host を使って, TCP による 70Kbyte のデータ転送をを毎 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を計測している.

今回の再現実験には ns-3 Direct Code Execution [15] を用い, MPTCP は, Linux カーネルソースを用いた [17]. 図 3 に, シミュレーションで用いた FatTree(k=2) トポロジーを示す. このトロ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.

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	16
MPTCP	v0.86
Link core-aggr	400Mbps
Link aggr-edge	200Mbps
Link edge-host	100Mbps
RTT	0.5ms
Window size	100KB

Table.1 Testbed on network 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 RTT が 1[ms] 以下であるためである [18].

ウィンドウサイズについては, 以下の帯域幅遅延積 (BDP) の式から, 400Mbps を最大限利用できるだけの値を設定した.

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

各帯域については, 16 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 再現結果

図 4, 表 2 に, 上記の実験環境で再現した結果を示す. 再現結果から, フローの様子を完結時間別に 4 パターンに分類することができることが分かった. 表 3 にそのフローパターンの定義を示す.

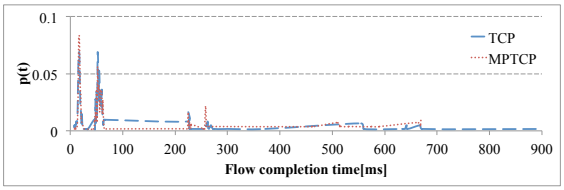

Fig.4 The result of the reproduction experiment

Protocol	AVG comp time[ms]	Stdev[ms]	95^{th} percentile[ms]
TCP	78.4	122.5	266.7
MPTCP	91	140.6	510.5

Table.2 Average flow completion time and stdev on reproduction experiment

4.3 考察

表 3 に示した各フローパターンについて, それぞれの特性を分析する.

Flow pattern	Comp time[ms]	Packet loss
Full window	∼30	
Intensive flow	∼60	
Delay with loss	200∼300	√
Extreme delay	300∼	√

Table.3　Flow pattern classified by completion time

4.3.1　パケットロスが発生しないフローパターン

図 5 に Full window と Intensive flow のデータ転送の様子を示す.

Full window では, TCP コネクション確立後, サーバーがすぐに最大ウィンドウサイズ分だけパケットを送り, クライアントからの ACK が返ってくると, 随時次のパケットを送っていた. これは, 経路に輻輳がなく, 多くのウィンドウを利用できたということであり, 30[ms] 以下でデータ転送を完了した.

一方, Intensive flow では, サーバーが最大ウィンドウサイズに満たない量のパケットを送り, クライアントからまとめて送られてくる ACK を受け取った後, 集約してパケットを送っていた. その結果, コネクションの切断時に遅延が起こり 60[ms] 程度転送時間がかかった.

4.3.2　パケットロスが生じたフローパターン

図 6 に Delay with loss と Extreme delay のデータ転送の様子を示す. いずれのフローパターンもデータ転送中にパケットロスが発生し, 再送処理, 重複 ACK 確認応答を行った. パケットロスが起きた原因は, 短時間にフローサイズの小さいトラフィックが中継ルータを集中したためである. 実際, 200[ms] のポアソン生起のうち, 数 10ms 単位の短い期間でトラフィックが発生したとき, 中継ルータにおいてパケットロスが生じた.

Delay with loss では, TCP コネクション確立後に数パケットのデータ転送を行い, パケットロスによるタイムアウトを生じた. その後, 再送処理を経て, Maximum Segment Size (MSS) である 1460[byte] でパケットを伝送した.

一方, Extreme delay では, TCP コネクション確立直後にパケットロスによるタイムアウトを生じた. その後も, パケットロスは生じないものの, 400[ms] 頃まで伝搬遅延が生じていた. また, 図 6 における二つのグラフの傾きは, セグメントサイズ最小値の 586[byte] に設定され, 転送速度が上がらなかったことを表している. これは, 中継するルータにおいて QoE 制御による帯域制限が発生したことを示している. 実際, 同時刻に流れていた Background traffic のスループットには変化がなく, QoE 制御の rate control により Back-ground traffic のデータ転送が優先され, ベンチマークトラフィックにはウィンドウサイズが制限されたと考えられる.

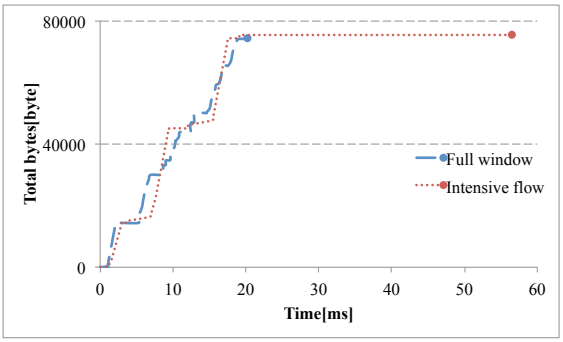

Fig.5　Comparison between Full window and Intensive flow

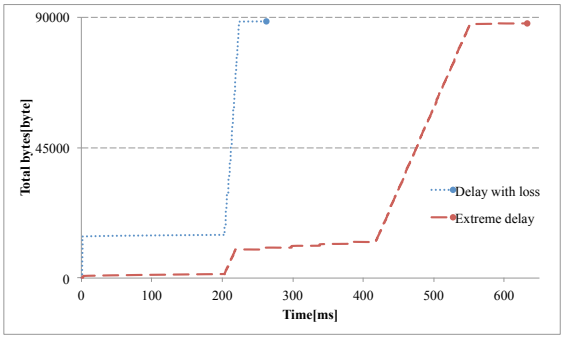

Fig.6　Comparison between Delay with loss and Extreme delay

4.3.3　TCP v.s. MPTCP

今回の再現実験において, TCP と MPTCP でフロー完結時間に差を生じた要因は, パケットロスが発生する割合にある. 図 7 に再現実験でのフロー完

結時間ごとの累積確率分布を示す. パケットロスを生じないフローに関しては, 両者に性能差を感じなかったが, この図から, MPTCP を用いた方が, パケットロスを引き起こし遅延を生じさせる割合が大きいということが分かる.

このように MPTCP が帯域を大きく占有することにより他のトラフィックを圧迫することは, MPTCP の輻輳制御によるものだと考えられる. 混雑のない経路でデータ転送する場合, MPTCP では積極的にウィンドウサイズを増やそうとするため, 他のフローに対し遅延を引き起こしたと推測される.

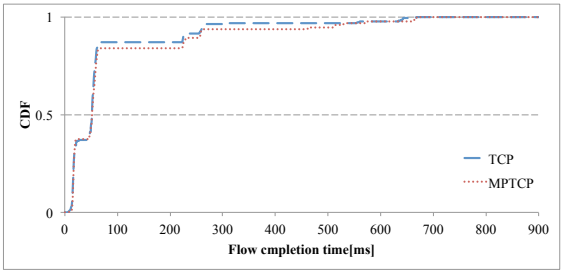

Fig.7 CDF of flow completion time on reproduction experiment

5 評価実験と考察

この章では, MPTCP データセンターネットワークに対して実際のデータセンター環境を想定したシミュレーション実験を行い, その結果について考察する.

5.1 想定環境

今回, データセンター上で partition-aggregate モデルに従う分散処理を想定する. ネットワークトポロジーについては, 前章の FatTree トポロジーを用い, 一つの Pod が管理ノード群として他 Pod の 12 ノードに対し, 処理を指示することを想定する. またベンチマークトラフィックについては, 3.3 節で述べた, 2 つのトラフィックパターンについて評価・考察を行う. メトリックとして表 4 を考える.

Traffic patern	Metric
Query traffic	Comp time[sec]
Short message traffic	Comp time[sec]

Table.4 Metric of each traffic patern

5.2 シミュレーション結果

5.2.1 Query traffic

Query traffic に対する評価として, フローサイズは 1[KB]~16[KB] とした, 12 の処理ノードへ平均 200[ms] のポアソン生起でトラフィックを発生させ, フロー完結時間を測定した. また, Query traffic のみ発生させた場合と, 50% の処理ノードに対し継続的にデータを送信するトラフィックを, Background traffic として同時に発生させた場合の 2 パターンについて評価を行った. その結果を, 図 8, 9 に示す. なお, エラーバーとして 99% 信頼区間を採用した.

この結果から, MPTCP は Query traffic に対し, 直接性能に影響を及ぼさず, Background Traffic による影響が性能差を生じさせたことが分かる. これは, やはり MPTCP が TCP よりも帯域を大きく占有した影響を受けたと考えられる.

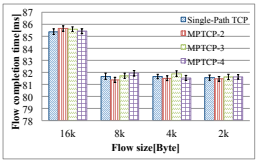

Fig.8 Flow completion time of Query traffic without Background traffic

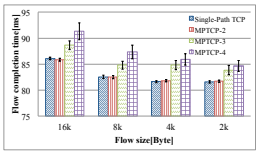

Fig.9 Flow completion time of Query traffic with Background traffic

5.2.2 Short message traffic

Short message traffic に対する評価として, フローサイズは 50[KB]~1[MB] とした. 50% の処理ノードに対し継続的にデータを送信するトラフィックを, Background traffic として同時に発生させた状態で, 同時に 12 の処理ノードへ平均 500[ms] のポアソン生起でトラフィックを発生させ, フロー完結時間を測定した. その結果を, 図 10 に示す.

この結果から, MPTCP は Short message traffic に対し, フロー完結時間を短縮させたことが分かる. これは, 先ほどの Query traffic よりも大きなサイズのフローを流したので, MPTCP により複数経路を利用し, TCP よりも短縮したためである. 実際, フローサイズが小さいと, MPTCP と TCP 間でフロー完結時間の差が小さくなっている.

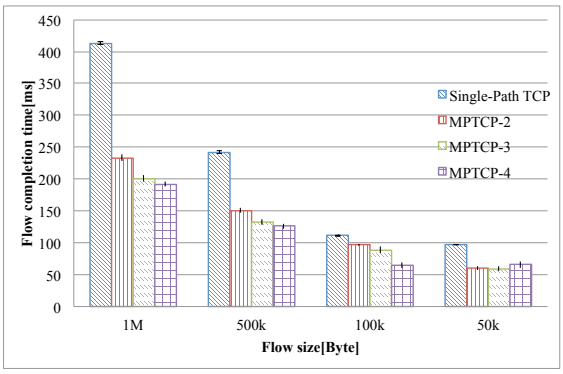

Fig.10 Flow completion time of Short message traffic with Background traffic

6 あとがき

本論文では, MPTCP の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, 4 つのフローパターンに分類することができ, そのうち, パケットロスが生じるフローがデータ転送時間に影響を及ぼすということを示した. 特に, MPTCP ではフローサイズの大きいトラフィックが小さいトラフィックを圧迫し, パケットロスを引き起こす割合が高く, その輻輳制御の公平性に問題が有るということを示した. このことから, MPTCP が帯域を占有する問題を解消できれば, 従来の TCP を用いたデータセンターモデルよりも多様なトラフィックに対し性能の向上を期待される.

今後はトラフィック監視により, パケット衝突を引き起こさないロードバランスにより, フローサイズによらずデータ転送高速化技術について検証していく予定である.

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

[13] 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.

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

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

[16] 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.

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

[18] 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.