データセンター環境におけるショートフロー通信改善手法の一提案

東京大学工学系研究科

修士2年　藤居翔吾

Outline

- データセンター環境におけるショートフロー通信改善手法の一提案

1. 研究背景
2. 関連研究
3. 実トラフィック解析
4. 検証実験
5. 結論
6. (提案手法)

研究背景

ビッグデータ：データの爆発的増加が深刻…

Facebookでは約300ペタバイトのデータを保有

1日に1ペタバイトのデータを処理[3].

データセンターでは..

スケールアウト：サーバ台数の増加, 数万~十万台規模

信頼性：ホスト間通信で複数パスにより冗長化

クラウドサービス：データセンター内での横のトラフィック増加

大量の機器, 大容量のリンクでデータセンターを改善!!

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

研究背景

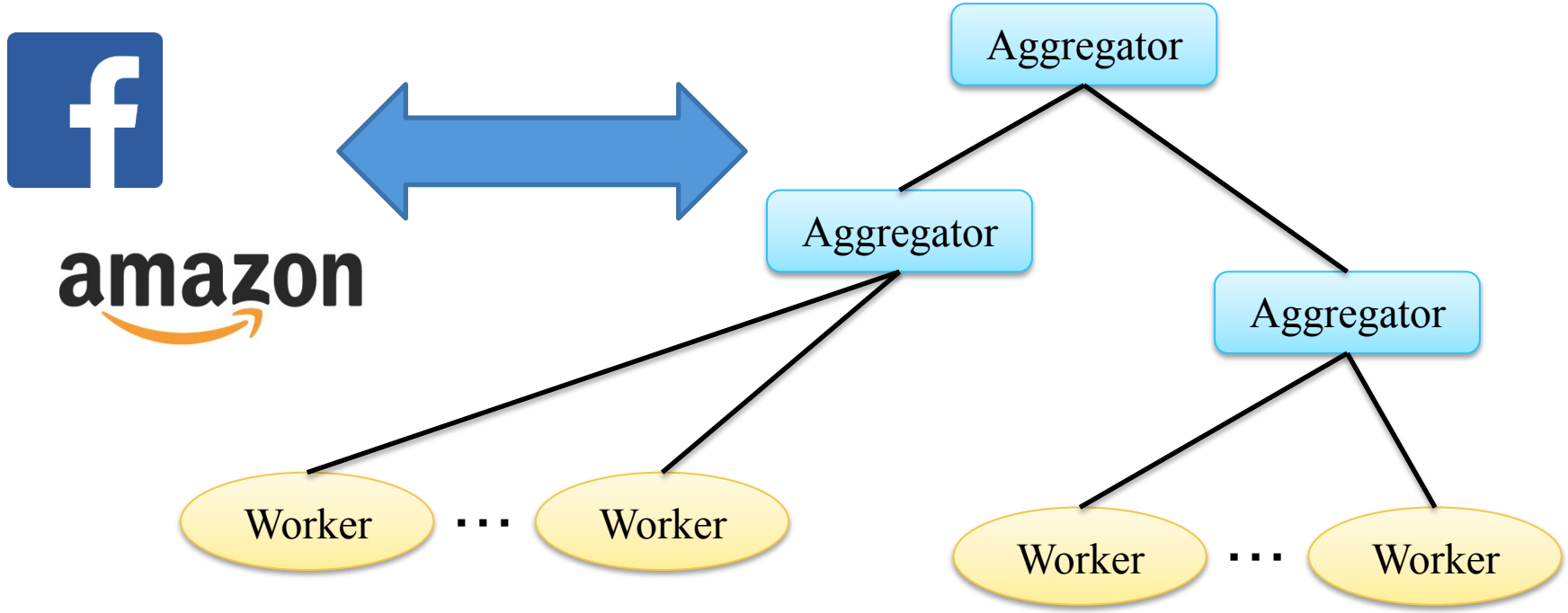
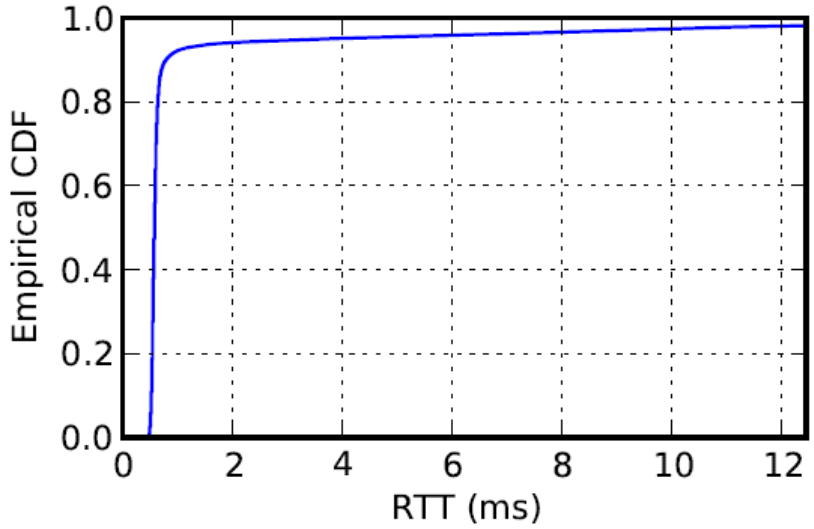


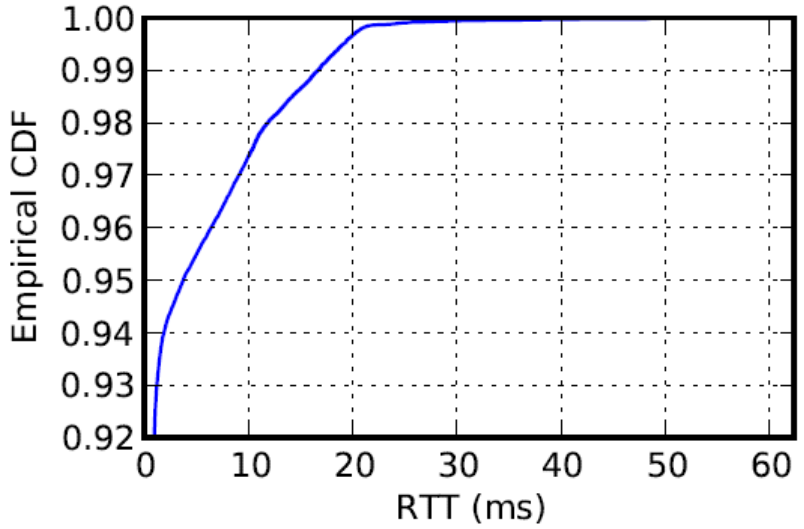
Fig1.データセンター内で構成されるクラスター

- クエリー/レスポンス：サイズの小さいフロー(ショートフロー)
- 低レイテンシ通信が求められている

既存のTCPではサイズの小さいフロー(ショートフロー)が圧迫される

Amazon EC2 us-west-2内で異なるインスタンス間のRTTを計測[*]





平均RTT: 0.5ms

99-thRTT: 17ms

Fig2.　EC2同一リージョン内での異なるインスタンス間のRTT

[*] H. Xu and B. Li. RepFlow: Minimizing flow completion times with replicated flows in data centers. In Proc. IEEE INFOCOM, 2014.

研究背景

なぜ、ショートフローに着目するのか?

分散・並列処理技術 : ビッグデータ、大量の計算資源の活用

分散・並列処理では大量のショートフローを生成してしまう!!

データセンタートラフィックの80%がショートフロー[14].

ショートフローは大規模計算処理の高速化のために極めて重要な要素

既存のデータセンターネットワークを改善する上で
ショートフロー遅延の問題は重要

[14]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.

関連研究

- データセンター環境におけるショートフロー遅延の改善
  - キュー長の改善：DCTCP(2010), HULL(2012)
  - 優先度に基づいたスケジューリング：D3(2011), PDQ(2012), DeTail(2012), pFabric(2013)
  - 再送制御の高速化： FastLane(2013), DIBS(2014), CP(2014)

提案された手法のすべてがスイッチに対して既存の実装の修正が必要

⇔RepFlow(2013)アプリケーションの書き換えが必要

実環境への適用が困難

研究について

想定: 既存のネットワークと大量の計算資源でビッグデータを処理する

データセンターネットワークの**要求案件**

1. 大量の計算資源を有効活用するトポロジー
2. シームレス性 : 特殊な実装、デバイスを用いずに性能向上
3. 並列分散処理を想定したアプリケーション性能向上を目的とした改善

アプローチ:

1. **マルチパスネットワークモデル：FatTreeトポロジー**
2. **MPTCPを利用(エンドノードOSのみ改良)**
3. **ショートフローの完結時間を改善**

コンスタントに性能の出せるデータセンターネットワークを目指す

コンスタントな性能を実現するために

データセンターショートフロー遅延問題の解消に向けて

1. **実環境での並列分散処理が生成するトラフィックの解析**

   トラフィックパターンの特徴

   ショートフロー遅延が生じる背景の検討

2. **マルチパス環境における複数のNIC, 複数の経路を利用した改善手法**

   仮定：複数の経路を利用し, スイッチやエンドノードの持つ複数のキューへと負荷を分散させることで, 単一のキューへ負荷による遅延を抑えられる

   期待する結果：余分な遅延なくフローが完結する

実トラフィック解析

実トラフィック解析環境

位置づけ

データセントートラフィックの**一例**として, 広く利用されている並列分散処理システムの生成する通信を観測

並列分散処理アプリケーション

・Presto : hadoopベース分散SQLエンジン

・定常状態：処理ノードの監視, データベースの更新

・ジョブ実行：クエリー, 検索結果の出力

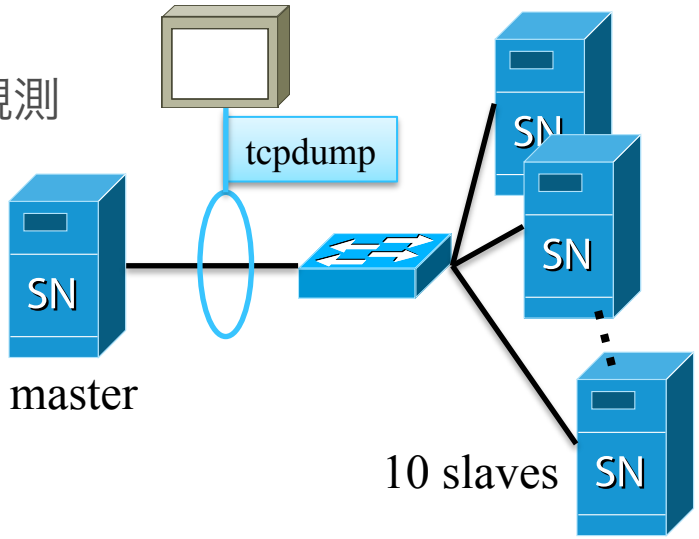


Fig3. 実トラフィック解析環境

測定

・定常状態：ジョブ命令を与えていない中で約10時間程度

・ジョブ実行：処理ノード全体に対し１分程度で完了するジョブ

・ジョブ：" *\select from*$テーブル*where* $条件"

実トラフィック解析：トラフィック量

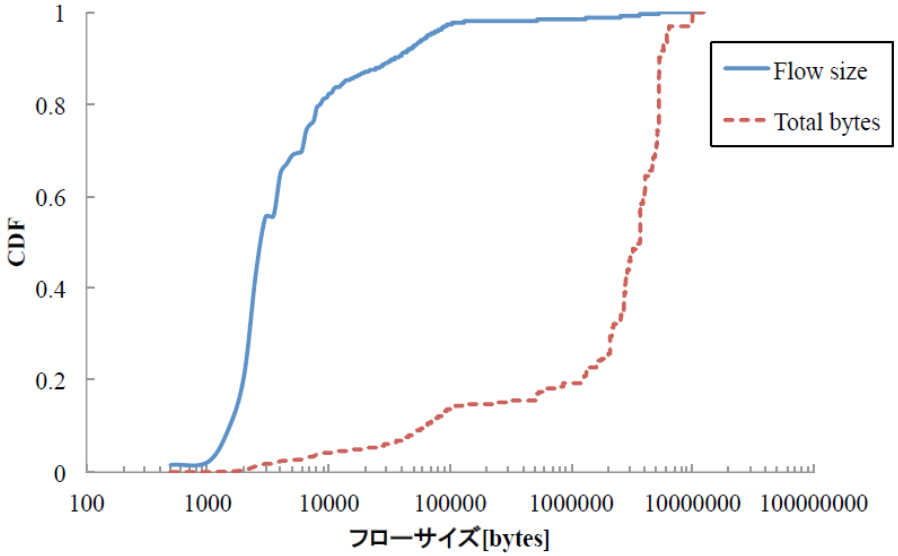


Fig4. Presto クラスタの定常時のトラフィック分布

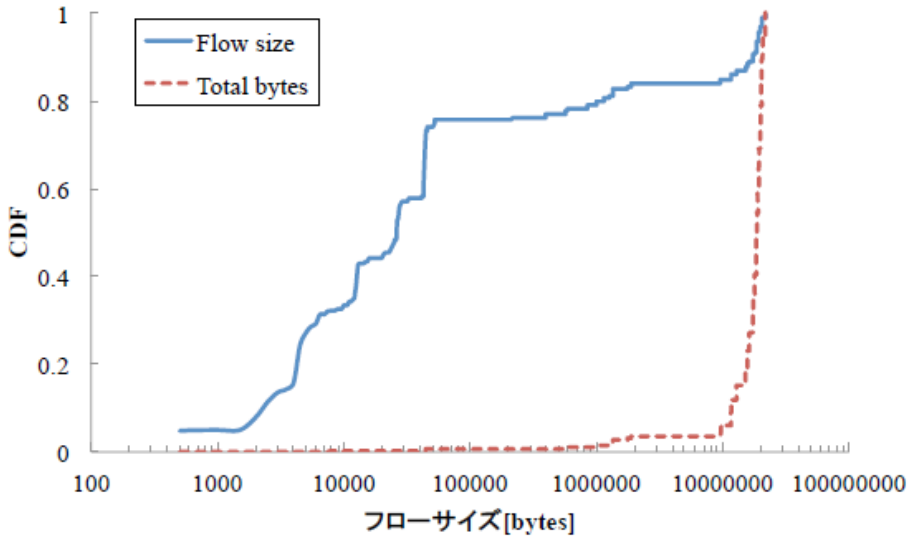


Fig5. Presto クラスタのジョブ実行時のトラフィック分布

- ショートフローの割合が大きい
- フロー数は少ないがサイズの大きいフローが大部分の通信量を占める

実トラフィック解析：定常状態

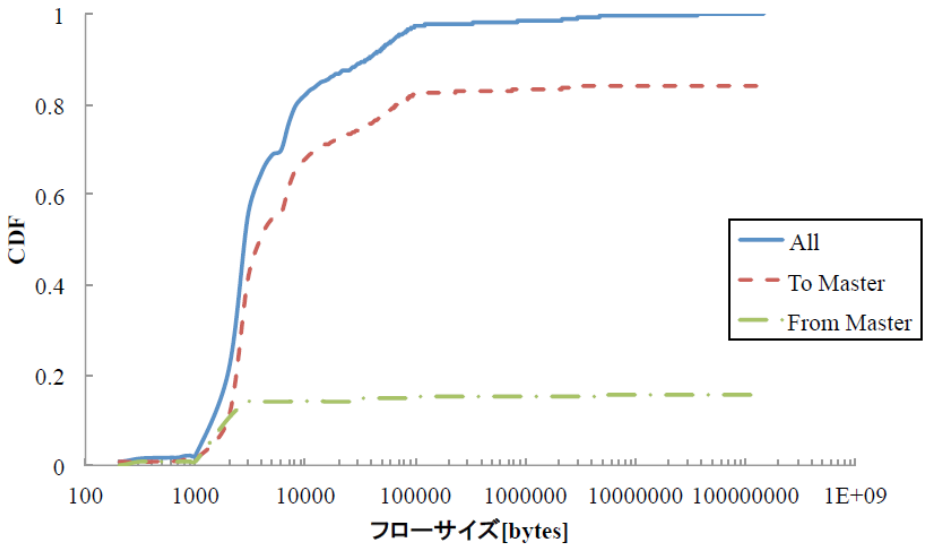


Fig5.管理ノードから見た定常時のトラフィック累積分布

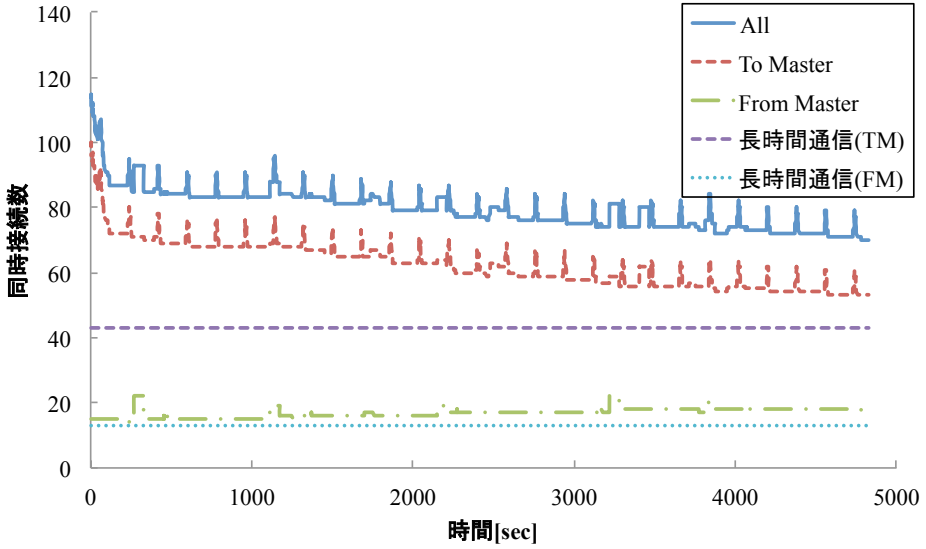


Fig6.定常時トラフィック:同時接続数の分布

- 管理ノードへのトラフィック量が多く、比較的通信時間が短いフローが存在している
- 長時間通信を行うフローが固定的に存在している

実トラフィック解析：並列分散処理実行時

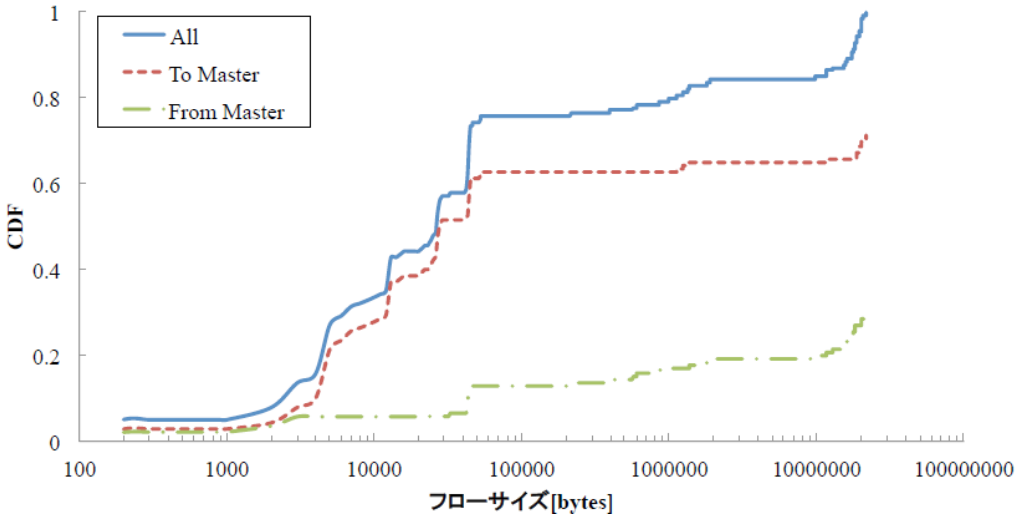


Fig7.管理ノードから見たジョブ実行時のトラフィック累積分布

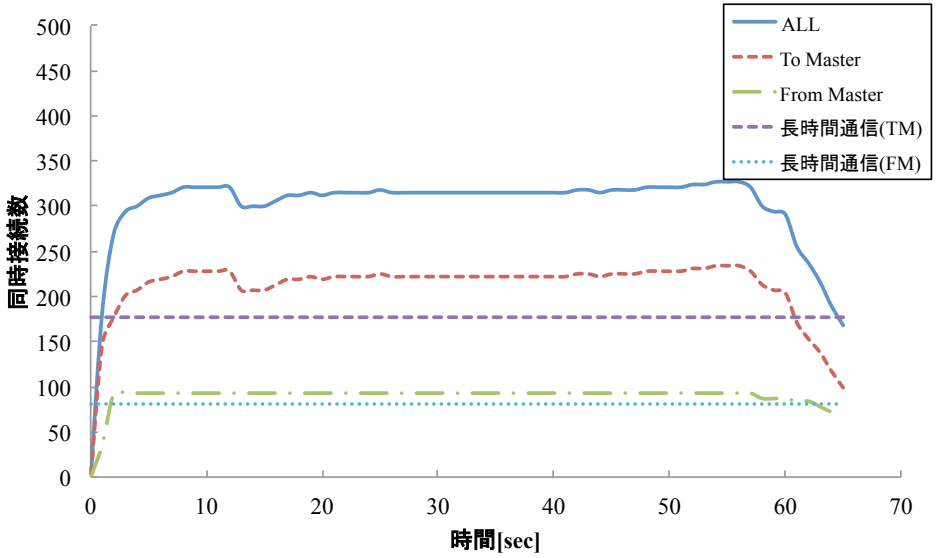


Fig8.定常時トラフィック:同時接続数の分布

- 管理ノードへのトラフィック量が多く、長時間通信を行うフローが固定的に存在している
- ジョブ開始直後にバースト性のあるトラフィック

実トラフィック解析：フロー完結時間

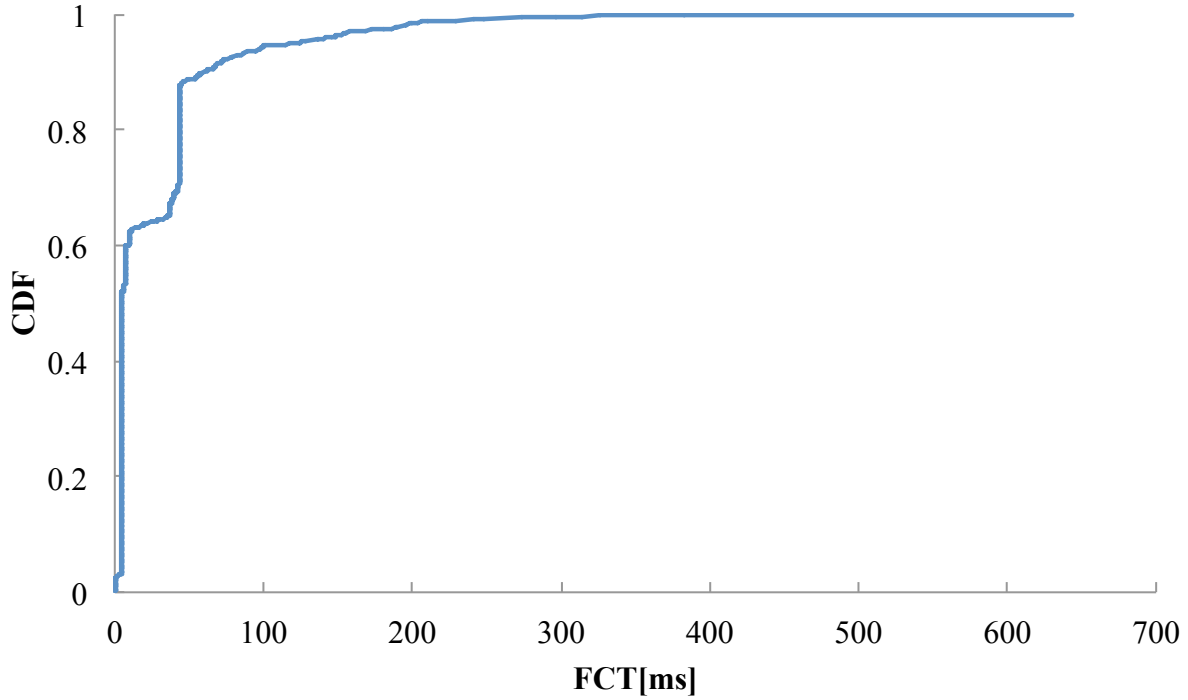


Fig9. Presto クラスタの2KB以下のフローのFCT

- サイズが小さいフローであっても遅延が生じている
- ショートフローの遅延問題

実トラフィック解析：考察

データセンター内ネットワークを想定したトラフィックの特徴

1. 管理ノードへ送信されるトラフィック量は大きい
2. 長い時間通信を行うフローが固定的に存在している
3. ジョブ実行時の管理ノードへのトラフィックには, フローサイズも小さく, バースト性がある

検討すべきトラフィックパターン[8]

- ジョブ開始時のバースト性のあるショートフロートラフィック
- バックグラウンドトラフィックが通信している中で, 低レイテンシ通信が求められているショートフローの通信

[8] Alizadeh, Mohammad, et al. "Data center tcp (dctcp)." ACM SIGCOMM computer communication review 41.4 (2011): 63-74

性能障害：スイッチ

並列分散処理のアーキテクチャ：マスタースレーブ

データセンターで用いられる機器：汎用的なもの

- メモリ量が限られている[20]
- 基本的に1NIC, 1CPUコアで処理が行われる[16]

スイッチでの性能障害

- Incast：短期間に一つのNICに多数のフローが集中する
- Queue buildup：レスポンスに影響しないバックグラウンドトラフィックがNICキューを圧迫

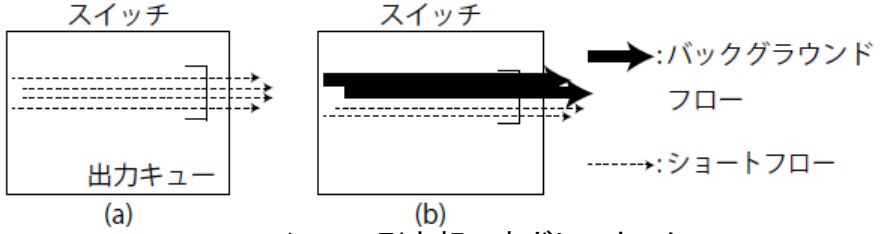


Fig10.スイッチで引き起こすボトルネック

[16] Microsoft corporation. scalable networking with rss, 2005.

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

性能障害：エンドノード

エンドノードでのボトルネック：受信処理での割り込み

- i.e. 1GbEの64バイトフレームの最大受信可能数：毎秒150万

割り込み回数を抑え、CPUリソースを効率的に利用する

- 割り込み処理：interrupt coalescing, ポーリング
  - 一定期間待ってからまとめて割り込ませる
  - 高負荷時に即座に処理することができない
- プロトコル処理：Receive Side Scaling(RSS), オフロード
  - 複数の受信キューを持ったNICを用いてキューを分散させる
  - CPUにかかるネットワーク処理の負荷を軽減
  - 仕組みの複雑化, 即座に処理できない

提案手法：検証項目

提案手法：想定環境

性能障害ボトルネック：単一NICへの負荷集中

- 想定環境：マルチパス環境(i.e. Multipath TCP)
  - 複数のNICにより複数の通信経路がある
  - 既存の実装ではサイズの小さいフローの場合複数経路を使えない：通信完了までコネクションを確立した経路を使い続ける

仮定：

・複数の経路を利用し, それぞれのキューへ負荷を分散させるようにトラフィック制御を行うことで, ショートフローの遅延を改善

手法：

・複数経路のうちロングフローレーン(LFL), ショートフローレーン(SFL) を設置し, 優先度(通信時間、通信経路状況)によって切り換えて通信を行う

検証実験

検証実験

検証1：中継スイッチに対する負荷実験

バックグラウンドトラフィックが利用しているスイッチNICを回避しショートフローのフロー完結時間(FCT) が改善できる

トラフィックパターン：

・ショートフロー：70KB毎10ms一様分布, 3種類のルート

・バックグラウンドフロー：実験中継続してデータを転送する

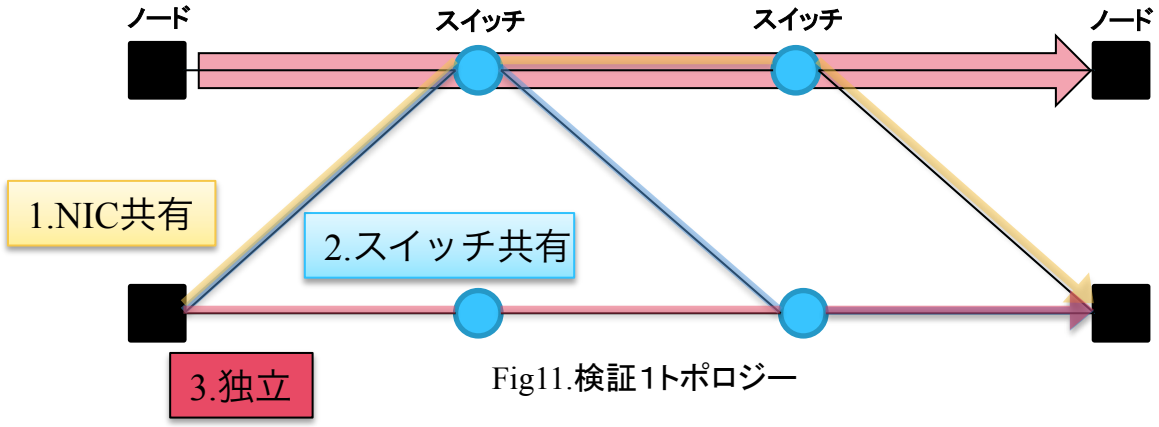
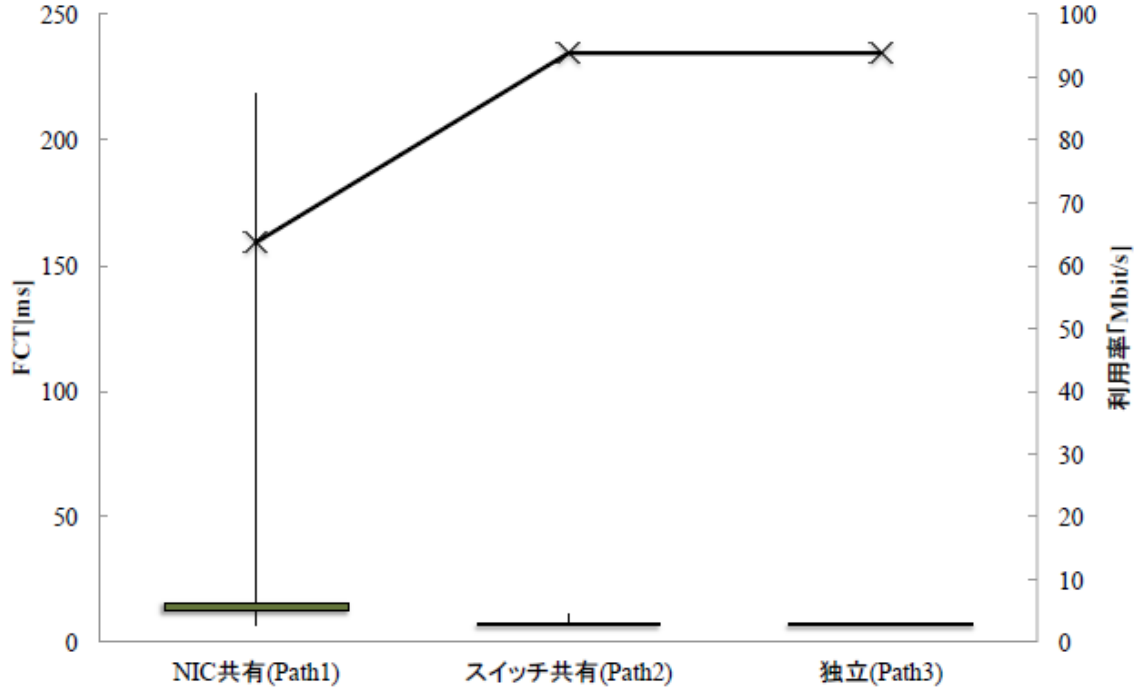


Fig11.検証1トポロジー

Table2.実験環境

項目	スペック
OS	Linux 3.13.0
CPU	Intel Xeon CPU L3426
メモリ	4GB
NIC対応ドライバ	E1000(RSS off)
リンク	100M

検証実験1：結果



・箱ひげ図の上端は95パーセンタイル値

Fig12.中継スイッチに対する負荷実験での 70kb ベンチマークトラフィックに対するフロー完結時間とリンク利用率

- バックグラウンドフローと経路およびNICを共有した影響で一部のフローが大きく遅延
- NICを共有した影響は小さい(伝送遅延は小さい)

検証実験2：エンドノードに対する負荷実験

バックグラウンドトラフィックが利用しているエンドノードNICを回避しショートフローのフロー完結時間(FCT) が改善できる

トラフィックパターン：

ショートフロー：70KB毎10ms一様分布, 3種類のルート

バックグラウンドフロー：実験中継続してデータを転送する

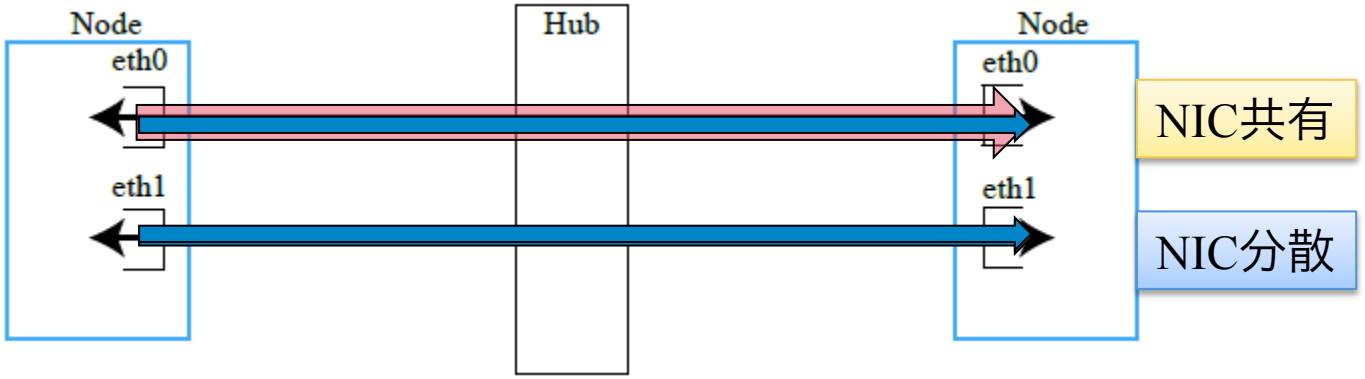
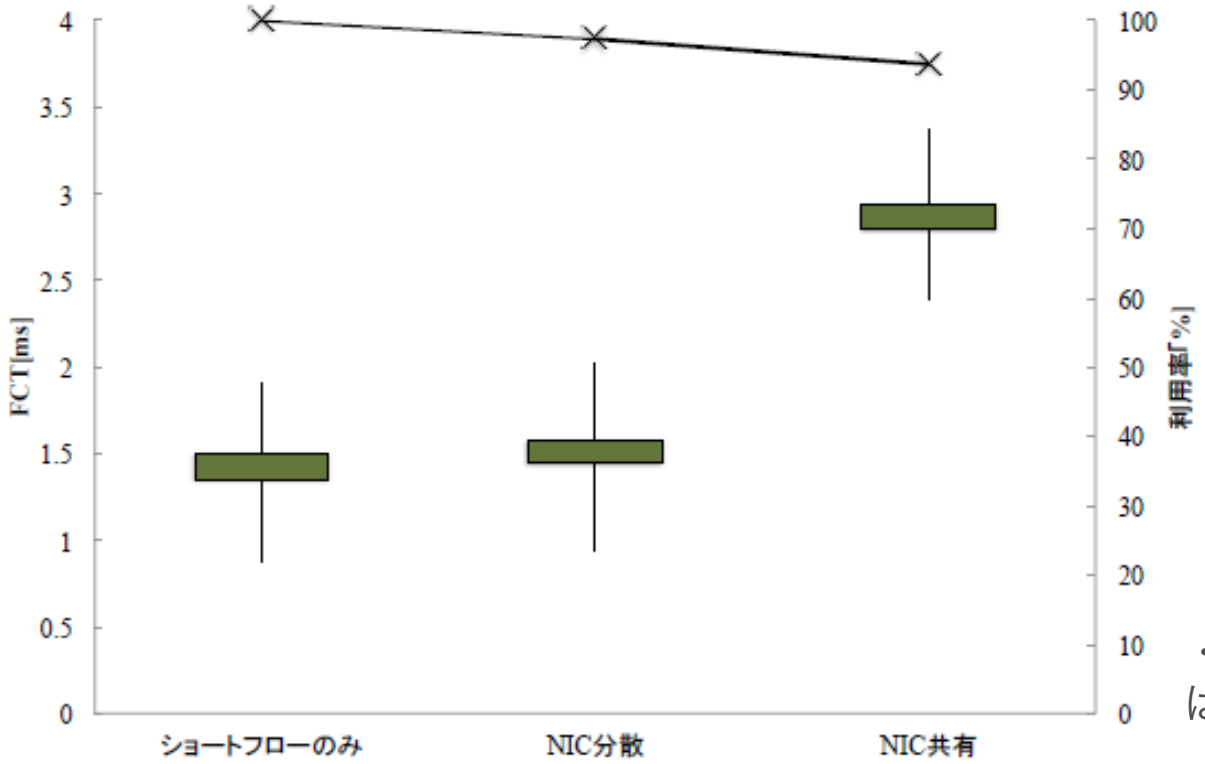


Fig13.検証2トポロジー

検証実験2：結果



・箱ひげ図の上端は95パーセンタイル値

Fig14.エンドノード に対する負荷実験での 70kb ベンチマークトラフィックに対するフロー完結時間とリンク利用率

- 単一NIC に対して二つのトラフィックが集中したことによる負荷分散の効果

結論

- データセンターにおけるショートフロー遅延の問題を解決する為に, 二つの検証を行った
  - 実環境トラフィックの解析
    - 二種類のトラフィックパターンにより汎用的なシングルキューNICを用いたネットワーク機器で機能障害が引き起こされる
  - 改善手法に向けての予備実験
    - スイッチ、エンドノードに対して複数のNICを用いた負荷分散によりショートフローのFCTを改善できた
- 今後の計画
  - 提案手法：通信時間ベースの経路切り替えアルゴリズム
  - 提案手法の実装

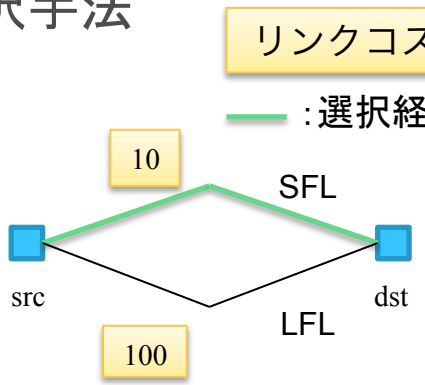
提案手法

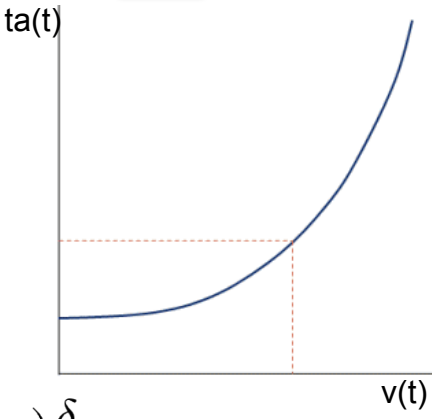
提案手法

- 提案手法：エンドノードから見たときのキュー遅延
  - RTTを用いたリンクコストベースのマルチパス選択手法
  - データセンターモデル：レーンごとの通信切り替え

$$\mathrm{RTT}: \tau(t) = \sum_{i=1}^{m} \frac{q_i(t_i)}{c_i} + bias$$

$q_i(t_i)$：キュー長　c_i：リンク容量



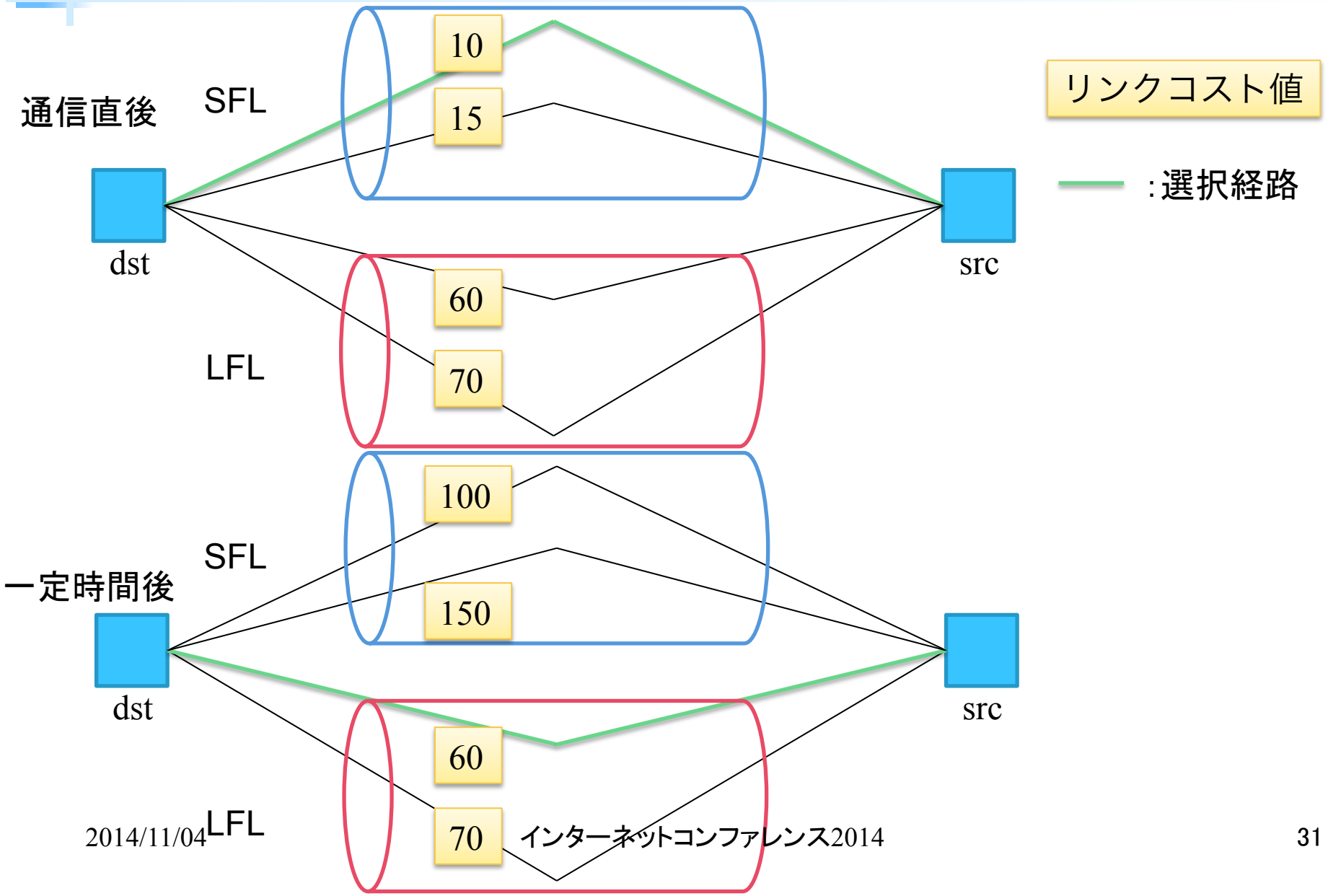


リンクコスト関数

$$t_a(t) = t_{a0} \cdot \{1 + \alpha \cdot \nu(t)^{\beta}\} + \gamma \cdot (t - t_{deadline})^{\delta}$$

Fig10.リンクコスト関数

想定するシナリオ


通信直後
SFL
10
15
dst
src
60
70
LFL
リンクコスト値
:選択経路
一定時間後
SFL
100
150
dst
src
60
70
LFL

提案手法

- 優先対象とするフロー
  - 通信時間ベースでの優先付け：通信開始時は優先度が最大
  - 通信時間が長くなれば, LFLへ移動(デッドラインベース)
- リンクコストの更新
  - RTTを用いて中継スイッチのNICキュー長の影響を考慮
  - 通信が行われていない経路については定期的に監視
- 実装方法について
  - Multipath TCPを用いて経路情報を保持, リンクコストに基づき用いる経路を選択する

提案手法の実現可能性について

- TCP/IP層のみの実装
  - 既存のアプリケーションも意識せずに通信ができる
  - エンドノード同士の情報のみ使用
    - スイッチに対して特別な実装を必要としない
  - データセンター内トラフィック経路のみに適用

提案手法

- RTTモデル化

$$\tau(t) = \sum_{i=1}^{m} \left(d_{l,i}(t) + d_{p,i}(t) + d_{q,i}(t) \right)$$

$d_{p,i}(t)$:伝播遅延 $d_{l,i}(t)$:リンク伝送遅延 $d_{q,i}(t)$:キュー遅延 $\tau(t)$:キュー遅延

m:リンク上のパス数

単純化

$$\tau(t) = \sum_{i=1}^{m} \frac{q_i(t_i)}{c_i} + bias$$

$q_i(t_i)$:キュー長 c_i:リンク容量

提案手法

- 相対キュー遅延

$$\nu(t) = \tau(t) - \tau_0 = d_t - d_0$$

$\nu(t)$:相対遅延量 τ_0:最小RTT d_0:最小キュー遅延

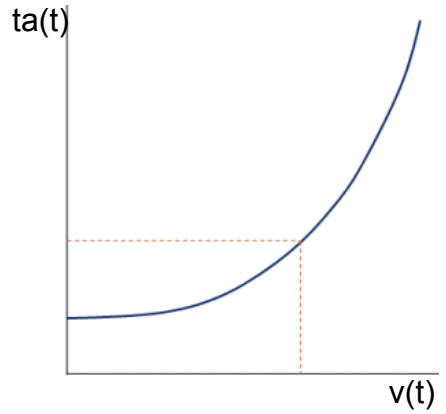
- リンクコスト関数

For SFL: $t_a(t) = t_{a0} \cdot \{1 + \alpha \cdot \nu(t)^{\beta}\} + \gamma \cdot (t - t_{deadline})^{\delta}$

For LFL: $t_a(t) = t_{a0} \cdot \{1 + \alpha \cdot \nu(t)^{\beta}\}$

$t_a(t)$:リンクコスト t_{a0}:リンクコスト α~δ:パラメータ

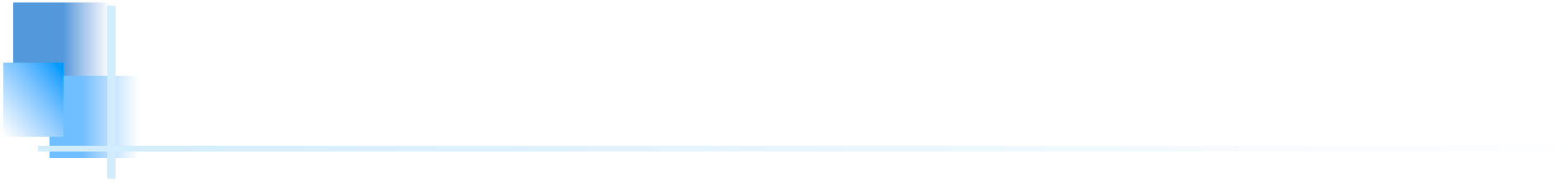
リンクコスト値によって経路を決定し、
ロングフローであるかどうかの判定を行う

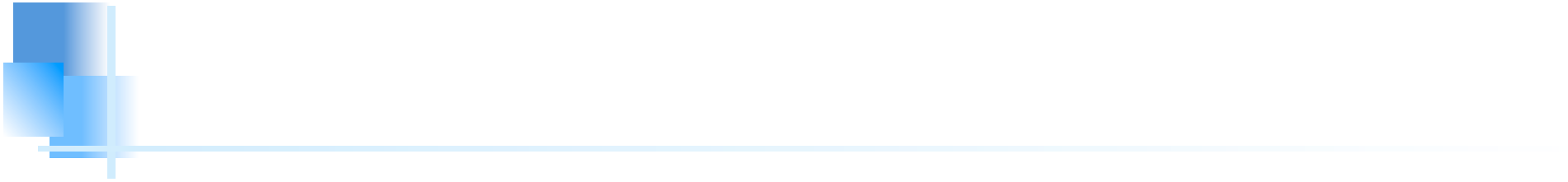


今後の課題

今後の課題

- 提案手法
  - RTTを用いたリンクコストベースのマルチパス選択手法
  - データセンターモデル：レーンごとの通信切り分け
    - 有効性の検証
- 今後の課題
  - 理論に基づいた提案手法の有効性の検証
  - 提案手法の実装





性能障害：エンドノード

エンドノードでのボトルネック：受信処理での割り込み

- i.e. 1GbEの64バイトフレームの最大受信可能数：毎秒150万

割り込み回数を抑え、CPUリソースを効率的に利用する

- 割り込み処理：interrupt coalescing, ポーリング
  - 一定期間待ってからまとめて割り込ませる
  - 高負荷時に即座に処理することができない
- プロトコル処理：Receive Side Scaling(RSS),
  - 複数の受信キューを持ったNICを用いてキューを分散させる
  - マルチコアCPU環境ではCPUを効率的に利用できる
  - 一般にRSS機能を持つNICは高価である