修　　士　　論　　文

並列分散計算の通信パターン設計を支援する，より高速なトポロジ推定

A Faster Topology Inference for Suppoting Communication Pattern Design of Parallel Computing

指導教員　**田浦 健次朗 准教授**

東京大学情報理工学系研究科
電子情報学専攻

氏 名　48-66421 **坂本 祥吾**

提 出 日　**平成 20 年 2 月 4 日**

概 要

並列分散計算アプリケーションの性能を高めるには，ネットワークの接続形態を考慮し，通信の分布に応じた最適化が重要である。しかし，複数のクラスタの多数のマシンを駆動するときや，計算に参加できるマシンが変動するとき，あるいは仮想化されたネットワーク環境を用いるときなど，ネットワークがどう接続されているかという情報を手動で設定するには手間がかかり，時には困難ですらある。我々は以前，ネットワークを木構造で表現するにあたり，一般的な TCP/IP プロトコルのみを用い，エンドホスト間のラウンドトリップタイム測定によって，高速にトポロジを推定する手法を提案した。

本論文はその手法を受け継ぎ，トポロジ推定アルゴリズムの並列化による高速化と，推定内容の精度の向上を図った手法を提案し，実装を行った。これにより，4 クラスタ 196 マシンのトポロジを 6.76 秒で推定した。

目 次

図 目 次

第1章　序論

1.1　背景と目的

数値計算に代表される並列分散計算において，プロセス間の通信を頻繁に行うアプリケーションの性能は，ネットワークの物理的な通信路と通信パターンに大きく影響される。マシン群が多数のスイッチで連結されている場合や，Intrigger のように複数の拠点クラスタが WAN で連結された環境など，分散計算資源を効率的に利用するには避けて通れない問題である。

集合的な通信とその最適化に関する研究の多くは，処理の局所性に対して，細かい設定を手動で与えるものである。MPI を例に挙げると，グリッドやクラスタ環境での動作に関する研究 [1, 8, 10, 11] では，各マシン間の局所性について既知であることを仮定する。このように自動的でない設定が必要なシステムは，それが静的に準備されている環境でなければ運用は困難である。クラスタ毎に異なる管理者がいるマルチクラスタ環境のように，複数の管理者が存在するマシン群について，ユーザがそれらを効率的に用いたいという望みを満たすためには，各管理者がネットワークの情報を提供し続ける手間を負わねばならない。また，ヘテロなマシン群を用いてソフトウェア的に仮想マシンを構築し，更にそれらを VLAN で接続するといった，極度に仮想化された環境 [9] では，IP アドレスやホスト名に物理的な通信路の局所性が反映されておらず，ネットワークの設定を与えることも難しくなる。

通信コストに配慮したアプリケーションを設計及び動作するためにはネットワークの情報が必要不可欠であるが，以上に見たように，それを手動で与えるようなシステムでは一般性を欠き，スケーラビリティを損なう非現実的なものであるといえる。よって，ネットワークの情報を動的に取得し，かつそれをアプリケーションに役立てられ易いように与えるための手法が求められる。

Shao らはネットワークトポロジの木構造モデルを提案し [13]，以降，トポロジを動的に推定する手法もまた多く研究されてきた。しかしそれらの研究では，長時間の測定を行い，得られた値を統計的に処理することでトポロジを推定するものがほとんどであった。これでは，任意のユーザが，その時々で自分の使い得るマシン群に関するトポロジを得る，という用途は期待できない。

[14] はこれに応え，ネットワークトポロジを木構造であると仮定するときに，ラウンドトリップタイム (小さなパケットの往復に要する時間，以下 RTT と略す) をプロセス間で測定した値のみに基づいて，トポロジを高速に推定する手法を示した。この手法は，1 クラスタ 64 台の推定を 4 秒で，4 クラスタ 256 台の推定を 15 秒程度で推定した。また，推定対象のマシンを N，ネットワークの直径

を d とするとき，推定時間は平均的なネットワークで $O(Nd)$ の効率であるとした。しかし同時に，更なる高速化と，推定トポロジの高精度化が可能であろうとした。

1.2 本研究の貢献

本研究は [14] の手法に基づき，精度を維持しつつ，より高速に動作するように最適化を施した。これによって 3 クラスタ 196 台を 6.76 秒で推定した。

1.3 本論文の構成

本論文は以下の構成から成る。
第 2 章では，トポロジ推定に関する既存の研究を述べ，本研究の立場を明らかにする。
第 3 章では，本研究がベースとした，以前の我々の手法を説明する。
第 4 章では，本研究が新しく提案する手法について述べる。
第 5 章では，本研究で提案する手法に則った実装について評価を行う。
第 6 章では，本論文を総括し，今後の課題について述べる。

第2章　関連研究

ネットワークトポロジには幾つかの用途があり，それぞれについて，ネットワークの仮定や，結果に要求される精度が大きく異なる。本章ではそれらの目的を 2 種類に大別し，それぞれについて述べる。

2.1　ネットワーク管理としてのトポロジ推定

この種類の研究は，推定したネットワークトポロジを，ネットワーク管理やトラブルシュートに利用することを目的とする。そのため，推定したトポロジは，物理的な機器と関連付けられた形で推定できることが期待される。一方，我々が欲しているような推定アルゴリズムの実行環境の柔軟性については，あまり求められることがない。

代表的な例として，traceroute を用い，広域ネットワークについてレイヤ 3 のトポロジを求める推定手法がある [5, 15, 6]。これらの研究は，非常に長時間の測定によって何百万ものマシンやルータを発見し，知識を蓄えることを目的とする。しかし我々の目的は，レイヤ 2 のトポロジに基づいて，高速に推定を行い，並列分散計算を支援することにあるため，方向性を大きく異にしている。

レイヤ 2 のスイッチを SNMP を用いて発見する研究はある [3, 12] が，スイッチなどの機器から情報を得られるようにするための権限は管理者にあるため，ユーザがもしそれを並列分散計算のために使いたいと望むなら，管理者にその都度協力を仰がなければならない難点がある。Black ら [2] は，エンドホストのパケットの監視によってレイヤ 2 のトポロジを推定したが，この手法はイーサネットに限られており，また NIC のプロミスキャスモードが必要であることから，我々の目的からすると動作環境が限られ過ぎているといえる。

2.2　エンドホスト間の測定に基づくトポロジ推定

この種類の研究は，ネットワークを木構造として仮定することにより各マシン間の通信路を一意に定め，マシン間で通信遅延やパケットロス率などの通信路に関わる特徴量を測定し，得られた値を統計的に解釈してトポロジを推定する。この手法では，レイヤ 2 のスイッチとレイヤ 3 のルータの区別ができない，分岐と関係しないスイッチの存在が認識できないなど，得られるトポロジは抽象化される。また，測定値は決定的な情報ではなく誤差を伴う値であるため，測定値の変化に応じて

推定結果も変動する。しかしトポロジの推定にあたって管理者権限が必要とされないことから，推定アルゴリズムの実行環境については柔軟である。我々の手法はこちらの考えに基づく。

Duffield ら [7] は，エンドホストを同じくする任意の 2 つの通信路について，パケットロス率を測定することによって両通信路が共有リンクを持つかどうかを判定し，最終的にトポロジを推定する手法を示した。しかし，1%にも満たないパケットロス率を充分な精度で測定するとなると，測定時間は長大となり，ネットワーク負荷も多大になってしまう。

Coates ら [4] は，通信遅延とバンド幅に基づく推定手法を示した。この手法ではまず，1 つのマシンを測定者として固定し，他のマシンは全て被測定者とする。次に，2 つの被測定者マシンに対して適切にパケットを送信することで，両通信路が共有するリンクのボトルネックとなる箇所のバンド幅を求める。これを全てのマシンのペアについて行い，得られた結果について最尤推定を行うことでトポロジを推定する。しかしこの手法では，マシン台数 n について測定回数が $O(n)$ となるため，スケーラブルな手法とはいえない。

上に掲げた 2 つの手法では測定者を 1 つのマシンに固定していたが，測定者から遠く離れた箇所のトポロジを高精度で推定することが困難である。我々の手法は，トポロジ上の位置を推定しようとする各マシンについて，適切な測定者と被測定者を設定し，測定を効率化する。

第3章　RTTに拠るネットワークトポロジの推定

3.1　トポロジ推定アルゴリズムの要件

我々が想定する大規模並列計算環境ネットワークのトポロジを推定するにあたり，以下の点が要求される。

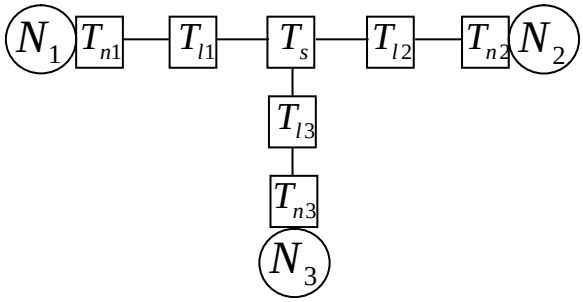
- ヘテロネットワークを横断できること。複数のクラスタからなる環境では，それぞれでOSやCPUの種類が異なることも充分考えられ，それらに対応できる柔軟性を持ったアルゴリズムが望まれる。またVLANやVPNが用いられている環境でも，実際の通信遅延に基づいたトポロジを得られたい。
- なるべく手動設定を要さないこと。複数のクラスタがそれぞれ異なる管理者によって運用されているとき，各クラスタのトポロジを管理者にそれぞれ尋ねたり，またその回答を手作業で設定するのは大きな手間である。
- なるべく高速に動作すること。トポロジの推定に要する時間が，並列分散計算の実行時間に並んだり，あるいはそれ以上となってしまっては本末転倒である。

我々の手法はこれらの要求に応じ，マシン間のTCPパケットの往復から測定されたRTTのみに基づき，高速かつ適切な精度でネットワークトポロジを推定することができる。

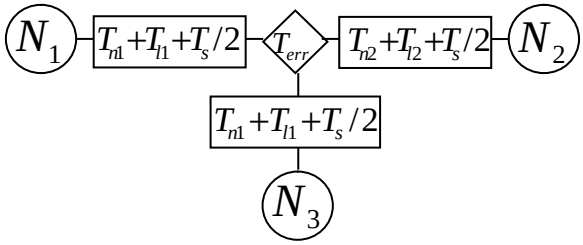
3.2　ネットワークモデル

3.2.1　ネットワークの仮定

我々の手法では，ネットワークは木構造を成しているものとする。我々の仮定する木構造において，辺はリンクである。葉はマシンであり，葉でない頂点はスイッチである。スイッチにはルータ等を含む。ここで，単一のクラスタ内の短期的な接続状況は木構造に近いものと考えられる。並列分散計算の設計において，クラスタ毎のトポロジを正確に取得できることは，クラスタ同士の接続の様子を正確に知ることよりも優先的に望まれるため，この仮定には妥当性がある。



(a) 現実の通信遅延モデル



(b) 本手法における通信遅延モデル

図 3.1: 通信遅延モデル

更に，どのマシン間の通信でも，微小なパケットが同じ通信路部分を通過するとき，同じ遅延時間が発生するものという仮定を設ける。具体的には，全ての通信路部分が通信方向に対して対称であり，またバッファリング機能を持つ機器が存在しないことを意味する。

3.2.2 通信遅延のモデル

ネットワークの通信遅延は，現実には図 3.1(a) に示すように，NIC やプロセスなどの個々のマシンに関する遅延 T_n，ケーブルに基づく遅延 T_l，スイッチに基づく遅延 T_s の 3 種類に区分される。しかし，プロセス間の RTT だけでは，これらを厳密に区別することはできない。そこで我々は，全ての種類の通信遅延をまとめ直し，遅延の基本値を辺の属性として，遅延の誤差をスイッチの属性として与える。このときの通信遅延モデルを図 3.1(b) に示す。

3.3 推定アルゴリズムの概要

本節では，[14] による，本稿がベースとする手法の大略を説明する。

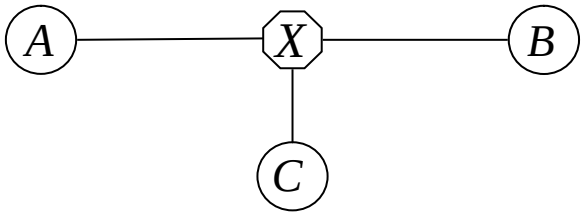


図 3.2: 3 マシンのトポロジ

3.3.1 推定のスケジュール

マシンの総数を N とする。マシン群は，各マシンの状態に依存しない，ランダムな列 $\{M_i\}(0 \leq i < N)$ によってスケジュールされており，この順にトポロジを推定してゆくものとする。M_i までの推定を終えたトポロジを T_i とする。

3.3.2 3 マシンのトポロジ推定

3 マシン A, B, C によるトポロジは，それらを結ぶスイッチ X を挿入することで，次の連立方程式の解として図 3.2 のように求まる。AB, BC, CA はそれぞれの RTT に基づく実測値である。

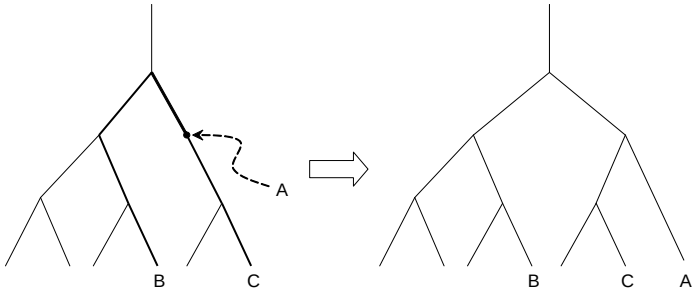
$$\begin{aligned} AX &= (\mathrm{AC} + \mathrm{AB} - \mathrm{BC})/2, \\ BX &= (\mathrm{AB} + \mathrm{BC} - \mathrm{CA})/2, \\ CX &= (\mathrm{BC} + \mathrm{CA} - \mathrm{AB})/2, \end{aligned} \tag{3.1}$$

M_0, M_1, M_2 のトポロジはこのようにして求められる。

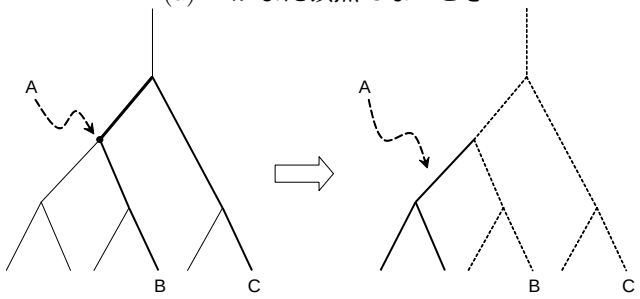
3.3.3 4 マシン以上のトポロジ推定

4 マシン以上のトポロジは，既に推定した部分トポロジ $T(= T_{i_1})$ に，新しいマシン $A(= M_i)$ を加えることによって推定する。これは，T 上から，A が直接接続している分岐点 X_A を求めることによって行う。

まず，T の中から任意の 2 マシン B, C を選択する。A から B, C のそれぞれについて RTT を測定し，式 3.1 を用いると，A, B, C のみによる部分トポロジと，その分岐点となるスイッチ X が求まる。これにより X_A は，X であるか，または X から分岐する経路のうち，経路 BC とリンクを共有していないものの中のどこかに存在することが分かる。この様を図 3.3 に示す。従って，ここからの処理は次のように分岐する。



(a) X がまだ頂点でないとき



(b) X が既に頂点であるとき

図 3.3: 2 マシンとの測定による結果

- X が T 中の頂点でないとき，$X_A = X$ である。T の経路 BC 中に頂点 X_A を挿入し，X_A に A を接続して処理を完了する。
- X が T 中の頂点であるとき，T 中の，経路 BC を経由せずに X へと連絡できるマシンを 1 つ選択し，D とする。
  - D が見つからなかったとき，$X_A = X$ である。X_A に A を接続して処理を完了する。
  - D が見つかったとき，C を D によって差し替える。改めて 3 マシン A, B, C による部分トポロジを測定・算出し，その結果について再び検討する。

3.3.4 予備推定トポロジ

前小節の手順において，始めの B, C を A となるべく近いマシンから選択することができれば，A の位置を高速に推定しやすくなる。そのために，各 M_i が T_i の推定を終えたとき，M_{i+1} に T_i を送信するだけでなく，M_{i+10} に T_i を送信して，なるべく M_{i+10} が T_{i+9} を受け取るよりも前に B, C の初期値を選択できているようにする。このように，B, C の選択のために M_{i+10} に送信されるトポロ

ジを予備推定トポロジと呼ぶものとし，一方 M_{i+1} に渡すものを主推定トポロジと呼んで区別することにする。予備推定トポロジに含まれるマシンが多い，即ち i が大きいときほど，M_{i+10} が受け取る予備トポロジ T_i が主トポロジ T_{i+9} の近似として機能することを期待できる。

予備推定トポロジ T' を受け取ったマシン A が B, C の初期値を選択する手法は以下の通りである。まず，T' 中に含まれているマシン群を抽出し，集合 M とする。M からランダムにマシンを 1 つ選択して P とし，AP の RTT を測定し，記憶しておく。P を M から除外する。次に，M に含まれる各マシン Q について，通信遅延が $AQ < AP/3$ または $AQ > 3AP$ であるとき，これを M から除外する。この操作によって M には，A からみて，P と同程度か，あるいはより近いマシン群が残される。なぜならば，もし A, Q が同一のクラスタに所属しており，かつ P がそれとは異なるクラスタに所属しているとき，通信遅延 AP と PQ の値が近いためである。その後，M が空集合でなければ，M から再びランダムに P を選択して上記の操作を繰り返す。M が空集合になったとき，記憶していた RTT の中から最も小さな値を 2 つ選ぶことによって，A となるべく近いマシン B, C を決定できる。

第4章　推定アルゴリズムの解析と拡張

4.1　検証に用いたクラスタ群

我々が推定アルゴリズムの解析に用いたクラスタ群を表 4.1 に示す。各クラスタは図 4.1 のように構成されている。図中の各スイッチは全て論理スイッチを表す。okubo クラスタと suzuk クラスタはいずれも，クラスタに所属する各マシンと外部への通信路とが，1 つの論理スイッチによって接続されている。また hongo0 クラスタと hongo1 クラスタは近い位置にあり，両方を共に扱う際は，これを hongo クラスタと呼ぶものとする。

hongo，chiba，okubo，suzuk の 4 クラスタについて，各クラスタ間は冗長に接続されている。

4.2　RTT 測定値分布の解析

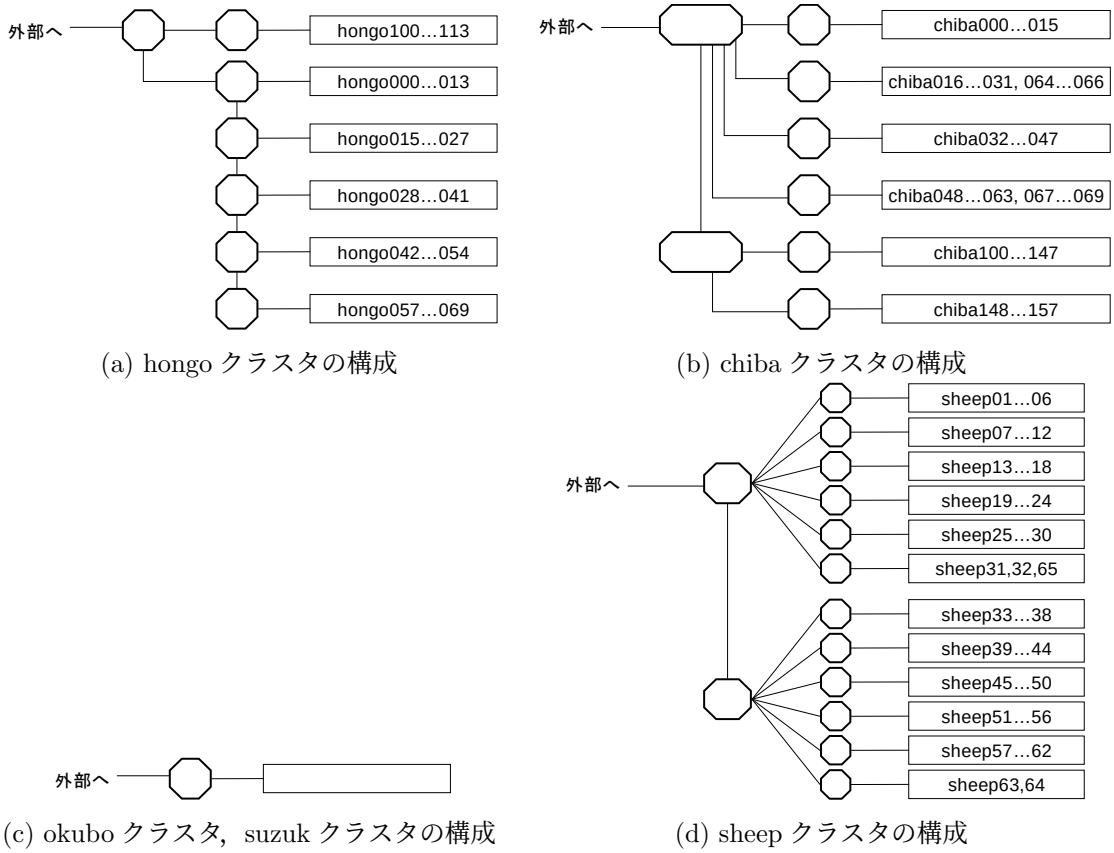
この節では実際のクラスタ群における RTT 測定値の分布を解析し，測定アルゴリズムの改定を提案する。

4.2.1　単一クラスタのマシンペアにおける RTT

大量の通信を伴う並列分散計算の設計を行う際，互いに近い位置にあるマシン群のトポロジをより正確に推定できるほど，その計算性能をより高めることができる。従ってまず，単一クラスタのト

表 4.1: 使用したクラスタ

cluster	machine	#	OS	CPU
hongo0	hongo000...069	63	Linux 2.6.18	Pentium M 1.8GHz
hongo1	hongo100...113	14	Linux 2.6.18	Core2 Duo 2.1GHz
chiba	chiba000...066	47	Linux 2.6.18	Pentium M 1.8GHz
chiba	chiba100...157	58	Linux 2.6.18	Core2 Duo 2.1GHz
okubo	okubo000...013	14	Linux 2.6.18	Core2 Duo 2.1GHz
suzuk	suzuk000...035	36	Linux 2.6.18	Core2 Duo 2.1GHz
sheep	sheep01...65	61	Linux 2.6.8	Xeon 2.4GHz



(a) hongo クラスタの構成

(b) chiba クラスタの構成

(c) okubo クラスタ，suzuk クラスタの構成

(d) sheep クラスタの構成

図 4.1: 各クラスタの構成

ポロジを推定するときの精度の確保を目指す。

図 4.2 は，hongo クラスタ内の複数のマシンペアについて，それぞれ 1000 個の RTT を調べた結果の分布である。RTT はマイクロ秒単位で取得した。測定者は hongo0 クラスタの単一のマシンである。被測定者マシンの 6 台のうち 5 台は hongo0 に所属しており，それぞれ，測定者からみて物理的なスイッチを 1 個，2 個，3 個，4 個，5 個隔てた位置にあるマシンについて，1 台ずつ選んだものである。被測定者の最後の 1 台は hongo1 に所属するマシンから 1 台を選んだものであり，測定者から約 10 個の物理的なスイッチを隔てた位置にある。この図に見られるように，互いに近い位置にあるマシンペアにおける RTT は，非常に高い確率で，ある狭い範囲に含まれる値として観測される。一方，ごく稀に大きな値として観測されることもあるが，これらの値から規則性を読み取ることは困難である。従って，単一クラスタ内の 1 つのマシンペアにおける RTT の測定は，以下の要領で行うものとする。

- 有効値：取得した RTT の最低値から順に，ある個数までを有効値とみなし，残りを破棄する。

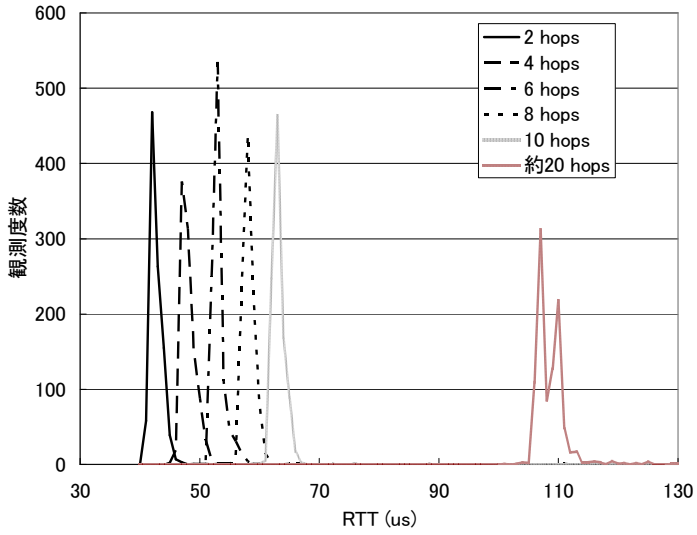


図 4.2: 近距離のマシンペア間の RTT 測定値の分布

その個数は，RTT を取得した総数 n と定数 r について，nr として定める。r は 1 以下である。

- 代表値：全ての有効値の平均とする。
- 誤差直径：有効値の最高値と最低値の差とする。

このように，RTT を n 個取得した中から nr 個を有効値として採用する RTT 測定モデルを，$\Gamma(n,r)$ と呼ぶこととする。$\Gamma(n,r)$ モデルに基づいてトポロジの推定を行うとき，n が小さいほど推定に必要な通信回数が少なくなり，推定アルゴリズムは高速になる。しかし小さくし過ぎると，代表値及び誤差直径が不安定になり，推定したトポロジの精度が損なわれる。即ち，推定したトポロジの精度を高水準に維持できるように n,r を定めなければならない。

以上に基づき，幾つかのクラスタについて適正とみられる n,r を調べた。これは，各クラスタについて，それぞれ次の手順で行った。まず $n = 100$ について，r を変化させたときのトポロジ推定精度の変化を見，適切な r を求めた。次に，その値で r を固定し，n を変化させたときのトポロジ推定精度の変化を見，適切な n を求めた。

図 4.3 は，これを hongo0 クラスタについて検討した結果である。このとき負荷となるプロセスは動作していない。図 4.3(a) は，$n = 100$ とし，r を変化させたときのトポロジ推定精度の変化である。これにより，$r = 0.9$ が適正であるといえる。ここで，r が 0.9 を超えてから false negative が増加している。これは，$r = 0.9$ によって測定値の範囲を設定するとき，通信路上のスイッチの個数が実際と異なると仮定したときにあり得る測定値の範囲とは衝突せず，かつその条件下で最大の誤差

直径を得られているものと考えられる。これは図 4.2 の分布と一致する。図 4.3(b) は，$r = 0.9$ とし，n を変化させたときのトポロジ推定精度の変化である。図では $n = 30$ が適正であるが，他に負荷となるプロセスが存在するときは精度が低下するという脆弱性があったため，$n = 40$ とするのが妥当とみられる。以上により，hongo0 クラスタは $\Gamma(40, 0.9)$ が適正であることが分かった。

同様に，chiba クラスタについて検討した結果を図 4.4，sheep クラスタについて検討した結果を図 4.5 に示す。chiba クラスタでは $\Gamma(40, 0.9)$ が，sheep クラスタでは $\Gamma(100, 0.8)$ が適正である。このように，トポロジ推定の対象とするマシン群によって適切な測定モデルが異なる。

4.2.2 互いに異なるクラスタに所属するマシンペアにおける RTT

次に，互いに異なるクラスタに所属するマシンペアの RTT 測定値について，その分布を確認した。ここで，測定者は hongo0 クラスタにある固定されたマシンとし，被測定者は chiba クラスタと okubo クラスタからそれぞれ 1 台を選択した。これらの被測定者に対してそれぞれ 1000 個の RTT を取得し，10 マイクロ秒単位で度数を集計し，分布を得た。結果を図 4.6 に示す。このように，異なるクラスタのマシンペア間の場合も，それぞれ一定範囲に集中する分布を成しており，$\Gamma(n, r)$ モデルに準じることが予想される。

これに基づき，異なるクラスタに所属するマシンペアに関する RTT 測定モデルを調べた。ここで，用いるマシン群は hongo，chiba，okubo，suzuk の 4 クラスタとし，同一クラスタ内のマシンペアは $\Gamma(100, 0.9)$ に固定した。この条件下で，互いに異なるクラスタに所属しているマシンペアを $\Gamma(n, r)$ とし，n, r を変化させたときのトポロジ推定精度の変化を見た。結果を図 4.7 に示す。図 4.7(a) は，$n = 100$ とし，r を変化させたときのトポロジ推定精度の変化である。これにより，r はトポロジ推定精度にほとんど影響を与えないことが読める。また図 4.7(b) は，$r = 0.8$ とし，n を変化させたときのトポロジ推定精度の変化である。トポロジ推定精度は n によって緩やかに変動している。

以上のように，異なるクラスタに所属するマシンペアの測定値の精度は，トポロジ推定の精度に大きな影響を与えることはないものとみられる。今回，我々は $\Gamma(10, 0.8)$ を採用するものとした。

4.2.3 推定トポロジの精度の評価

推定したトポロジ 1 つの精度は,「互いに異なる 4 台のマシン P, Q, R, S について，通信路 PQ と通信路 RS がリンクを共有しているか」という問いに対する正答率で求める。しかし，100 台を越えるマシン群について，全ての P, Q, R, S の組み合わせについて調べるのは困難であるため，ランダムに P, Q, R, S を選択した 100,000 問によって代替する。また，トポロジ推定に用いるパラメータを変化させるときの各パラメータにおける推定の精度は，そのパラメータについて 20 回の推定を行った 20 個の推定結果についてそれぞれ正答率を求め，それらを累計したもので表す。

各設問に対する解答は次の 3 通りに区分して評価する。一つは正しい解を得られたときの correct，一つは共有リンクがあるという解を得たが現実には存在しないときの false positive，一つは共有リンクがないという解を得たが現実には存在しているときの false negative である。定性的に述べて，false positive は，現実には同一のスイッチに接続されている複数のリンクが，推定されたトポロジでは異なるスイッチに接続されているものとみなされていることを意味する。即ち，現実の構成では図 4.8(A) のように，通信路 PQ と通信路 RS が 1 つのスイッチ X を共有しているのみであるにも拘らず，図 4.8(B) として推定されたために，両通信路がリンク X_0X_1 を共有しているという解答を得たのである。これは，RTT 測定値の誤差を狭く取り過ぎたために，単一のスイッチ X が，複数のマシンによってそれぞれ異なる位置 X_0, X_1 で観測されてしまったために起こる。また false negative は，現実には複数あるスイッチを区別することができず，単一のスイッチとして推定されてしまったことを意味する。つまりこのときは先とは逆に，図 4.8(B) が現実の構成であるが，図 4.8(A) として推定されている。これは測定値の誤差を広く取り過ぎてしまったことにより，隣り合った 2 つのスイッチ X_0, X_1 を結ぶリンクの存在を認識できず，両者が単一のスイッチ X として推定されたために生じる。今回の我々の手法の 1 つは，各スイッチをこれらのように誤って認識することのないように適切なパラメータを設定することにあり，これは false positive と false negative がバランスするところを求めるということでもある。

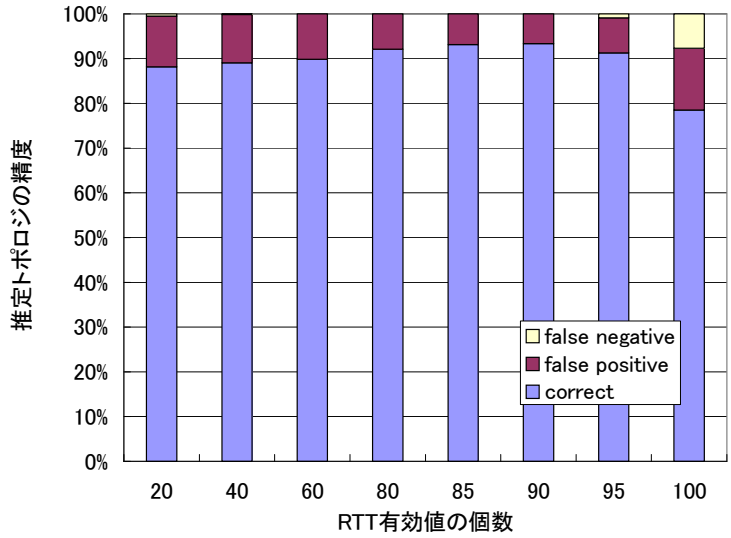
4.3 予備推定トポロジの高速化への応用

各マシンについて，主推定トポロジを受信してから，それに自身を追加するまでに要した時間は，トポロジを推定するために要した時間にそのまま反映される。特に RTT の測定を行うことによる影響は大きい。そのため，主推定トポロジの到着を待機している間に，主推定トポロジを処理するに当たって必要となる RTT を収集できていれば，推定時間を大幅に軽減できると予想される。本拡張では，このために予備推定トポロジを利用する。

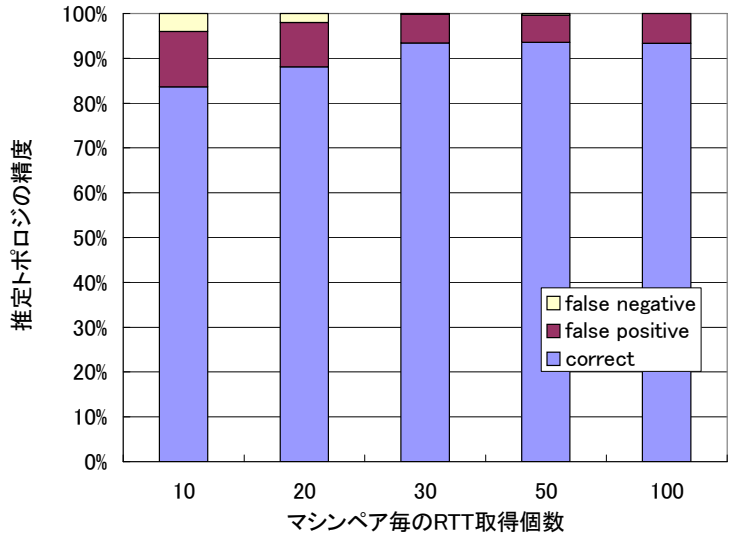
予備推定トポロジは本来，3.3.4 節のように，推定トポロジの高精度化に用いられるものであった。しかしそこで既に述べたように，予備推定トポロジが充分多くのマシンを含んでいるときは，主推定トポロジの近似とみなすことができる。このため，予備推定トポロジを受け取った各マシンが，それに自身を追加する処理を試行することによって，主推定トポロジに自身を追加する際に必要となる可能性の強い RTT 測定を，予め実行できるものと考えられる。

推定したトポロジの精度を確保するという点では，各マシンは，なるべく直前にスケジュールされたマシンが算出したトポロジを予備推定トポロジとして受け取ることが望ましい。しかし，直前のマシンから得た予備推定トポロジに自身を追加する試行をすると，その間に主推定トポロジを受信してしまい，高速化の点において充分な意味を成さなくなる恐れがある。本拡張では，予備推定トポロジを複数設けることによって，これらに対応する。

以上に述べた手法を，次の要領で行う。マシン群が 3.3.1 節のようにスケジュールされており，マシン M_i が T_i を得たとき，非負の整数 n について $M_{i+10\cdot 2^n}$ に対して，M_i が T_i を予備推定トポロジとして送信する。この様子を図 4.9 に示す。また，各マシンは予備推定トポロジを受け取る度に，それに対して自身を追加する試行を行うものとする。この，予備推定トポロジを $10\cdot 2^n$ 先にスケジュールされたマシンに転送するという設定は，一般的な環境において最適である保証はない。しかしこのように段階的な試行を行うことにより，各マシンが主推定トポロジを受け取るに先立って RTT 測定値を準備し易くなり，それに自身を追加する処理に要する時間の軽減を期待することができるものと思われる。

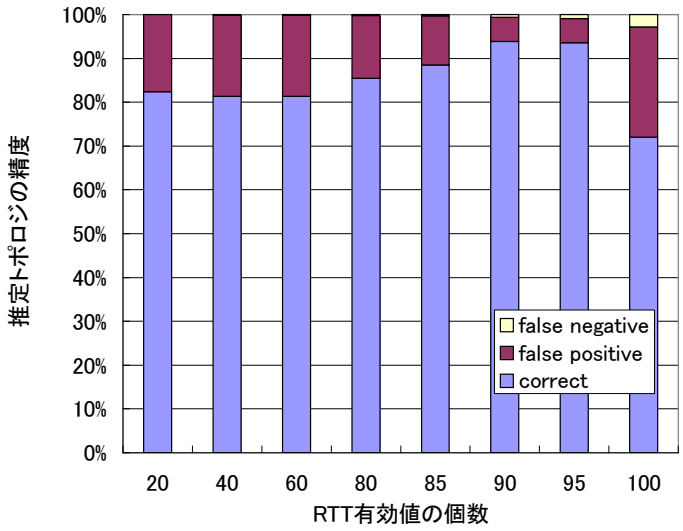


(a) 有効値の個数の検討

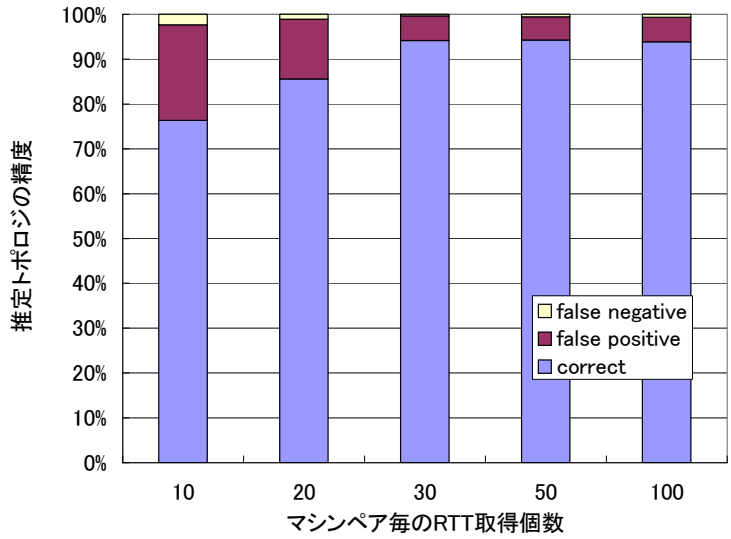


(b) RTT 取得個数の検討

図 4.3: hongo0 クラスタの RTT 測定モデル

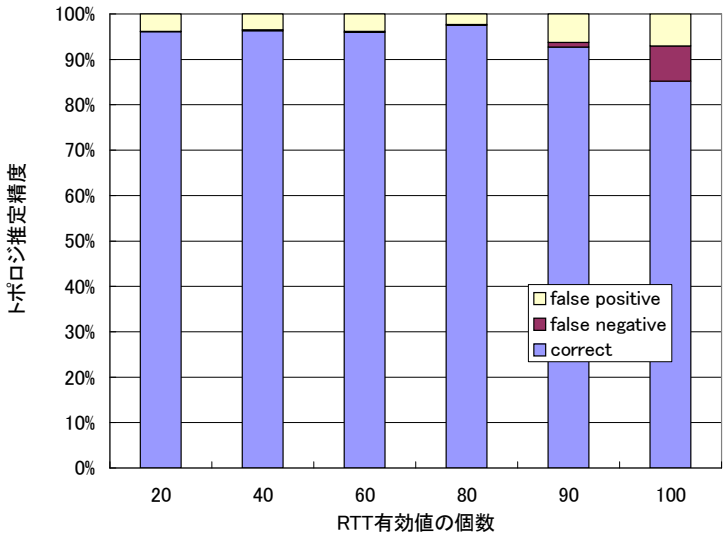


(a) 有効値の個数の検討

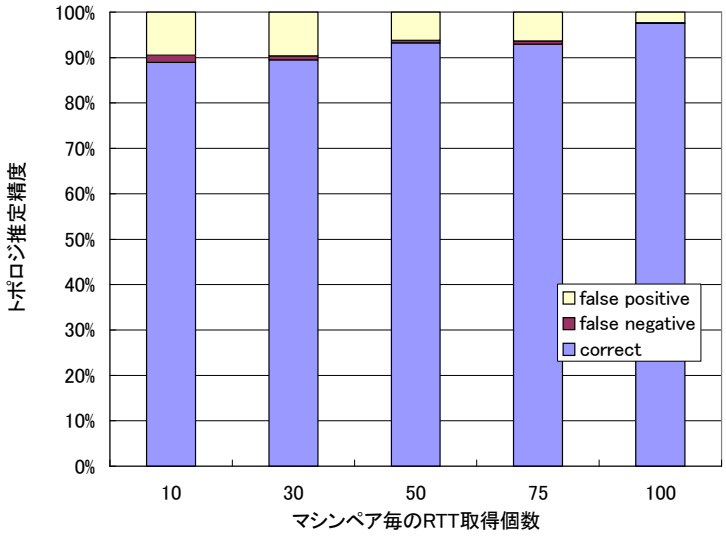


(b) RTT 取得個数の検討

図 4.4: chiba クラスタの RTT 測定モデル

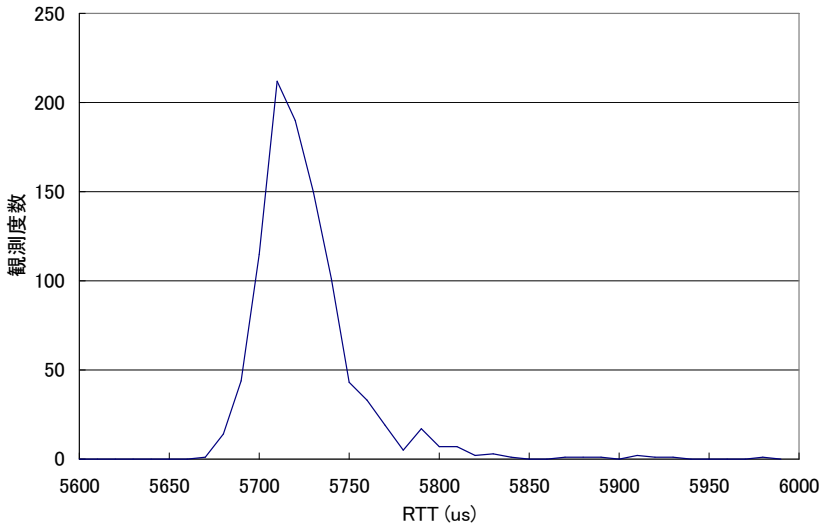


(a) 有効値の個数の検討

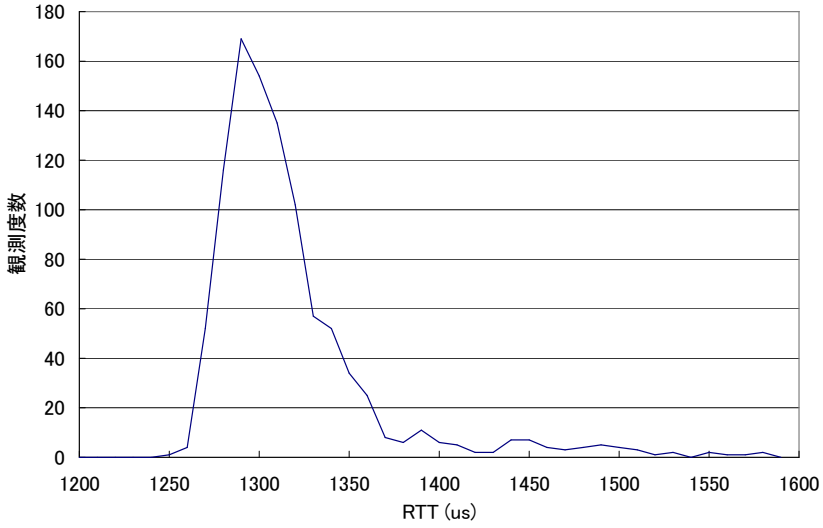


(b) RTT 取得個数の検討

図 4.5: sheep クラスタの RTT 測定モデル

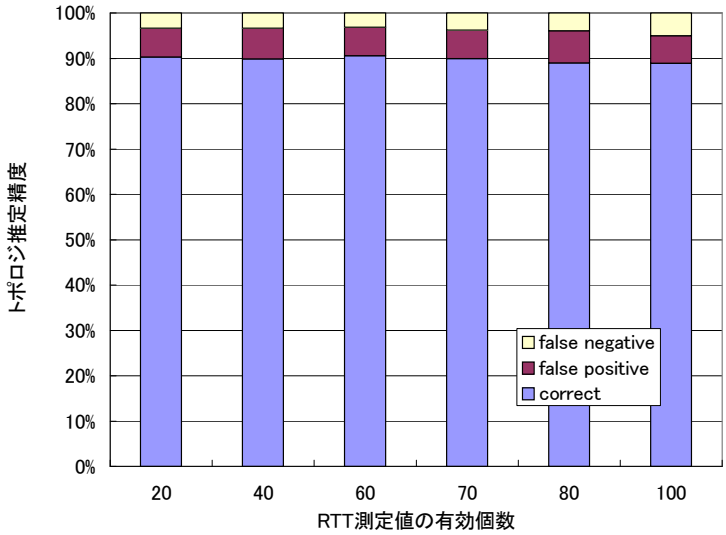


(a) hongo0-chiba 間

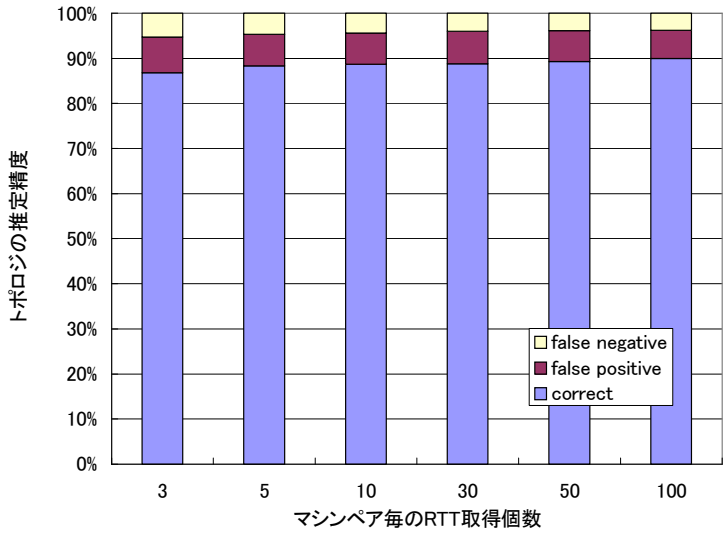


(b) hongo0-okubo 間

図 4.6: 所属クラスタの異なるマシンペア間の RTT 測定値の分布



(a) 有効値の個数の検討



(b) RTT 取得個数の検討

図 4.7: 遠距離の RTT 測定モデル

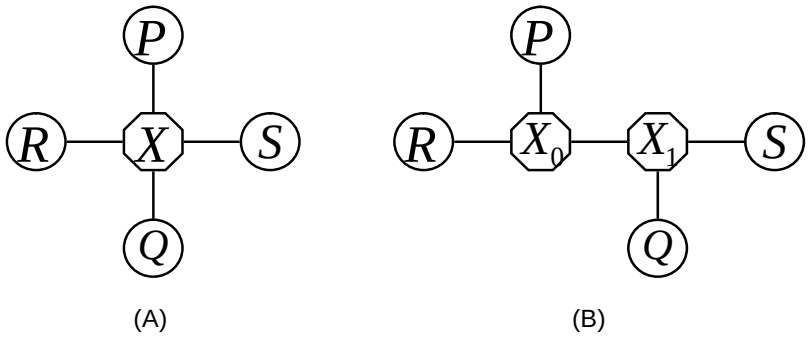


図 4.8: 推定トポロジの誤りのモデル

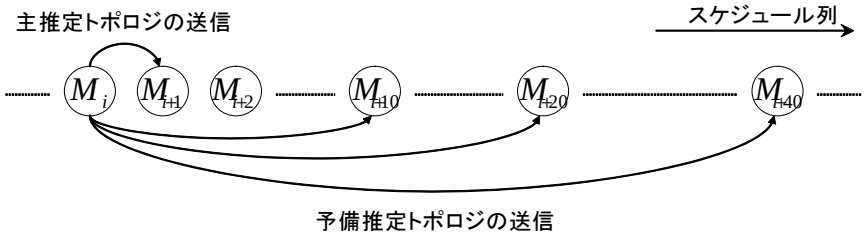


図 4.9: 予備推定トポロジの多段化

第5章　評価

5.1　実装と実験内容

我々は前章までに述べた手法をそれぞれ C++言語で実装した。我々のプログラムは，起動した直後，トポロジ推定に参加している全マシンの IP アドレスとポート番号を共有する。その後，いずれかの任意のマシンからトポロジ推定を開始し，いずれかのプロセスが全体のトポロジを構築したとき，プログラムが停止する。プロセスの起動や，各プロセスにおける IP アドレスとポート番号の共有化には，GXP[16] を用いた。GXP はグリッド環境向けのシェルプログラムであり，任意の複数のマシンでプロセスを起動する機能や，ssh 通信網を用いた簡易なメッセージパッシングの機能を有している。前者の機能で各マシンのトポロジ推定プログラムを起動し，後者の機能でマシンの IP アドレス等を共有するものとした。

以上のように構築したプログラムと，[14] に従う元手法とについて，4.1 節で述べたクラスタ群のうち幾つかを用いた環境について，その性能を比較した。以下，各手法は次のように略するものとする。

- BS (Base): 元手法に従うプログラムを，そのまま動作させたもの。
- MI (Measure Improvement): 4.2 節に従い，RTT 測定手法に調整を施したもの。RTT 測定モデル $\Gamma(n, r)$ の n, r は，4.2 節で得た値に従い，環境に応じて与えるものとする。
- MI-TR (Test Run): MI を踏まえつつ，4.3 節に従い，予備推定トポロジを多段化し，それぞれにマシンを追加する試行を行うもの。

各手法の性能は，推定して得られたトポロジの精度と，推定に要した時間の 2 つに基づいて評価するものとした。これらは，推定の手法と，推定の対象とするネットワーク環境が与えられているとき，次の要領で実験を行い，その結果について調べる。

- 精度: 4.2.3 節に従い，20 回の推定を行って得られた 20 個のトポロジの精度を累計した。
- 所要時間: 20 回の推定に要した時間を平均した。また併せて，各マシンが主推定トポロジを受け取ってからそれに自身を追加するまでの間に，RTT を取得する目的で送信したパケットの数について，全てのマシンについて集計を行い，20 回の推定における平均を取得した。

表 5.1: 各手法における単一クラスタの推定時間

	hongo クラスタ		chiba クラスタ		sheep クラスタ	
手法	推定時間	測定パケット数	推定時間	測定パケット数	推定時間	測定パケット数
BS	1.73 秒	15682	2.38 秒	20939	1.11 秒	7683
MI	1.52 秒	14640	2.89 秒	24852	2.13 秒	20120
MI-TR	1.23 秒	9417	2.26 秒	16484	1.67 秒	11766

表 5.2: 手法 BS の精度に従うときの単一クラスタの推定時間

	chiba クラスタ		sheep クラスタ	
手法	推定時間	測定パケット数	推定時間	測定パケット数
MI	1.85 秒	12886	0.83 秒	5000
MI-TR	1.66 秒	10727	0.81 秒	5052

5.2 単一クラスタの推定

hongo クラスタ，chiba クラスタ，sheep クラスタのそれぞれについて，各手法に基づいて推定を行ったときの精度に関する実験結果を図 5.1，所要時間に関する実験結果を表 5.1 に示す。手法 MI 及び手法 MI-TR の RTT 測定モデルは，hongo クラスタと chiba クラスタでは $\Gamma(40, 0.9)$ に，sheep クラスタでは $\Gamma(100, 0.8)$ に従うものとした。

hongo クラスタでは，全ての手法が高い推定精度を示しているが，手法 MI-TR では主推定トポロジの処理に要する通信量が大きく低減しており，高速に動作していることが分かる。

chiba クラスタと sheep クラスタでは，手法 MI は手法 BS と比べて精度の改善がなされている一方，通信量が多く，推定速度が低下している。手法 MI-TR はいずれにおいても手法 MI の精度を維持しながらその高速化を果たしており，chiba クラスタでは手法 BS の速度を若干上回っている。しかし sheep クラスタではそれでも手法 BS の速度に達していない。これは RTT を取得する個数について，手法 BS では各マシンペア毎に最悪でも 90 個であるが，手法 MI 及び手法 MI-TR では $\Gamma(100, 0.8)$ に従うことにより，常にそれぞれ 100 個のデータを採取していることが負担になっていると思われる。

このことを踏まえ，chiba クラスタと sheep クラスタにおいて，手法 MI 及び手法 MI-TR を，手法 BS の推定精度を維持する程度の RTT 測定モデルで動作させ，推定に要した時間を確かめた。このとき，chiba クラスタでは図 4.4 に基づき $\Gamma(20, 0.9)$，sheep クラスタでは図 4.5 に基づき $\Gamma(10, 0.8)$ とした。結果を表 5.2 に示す。

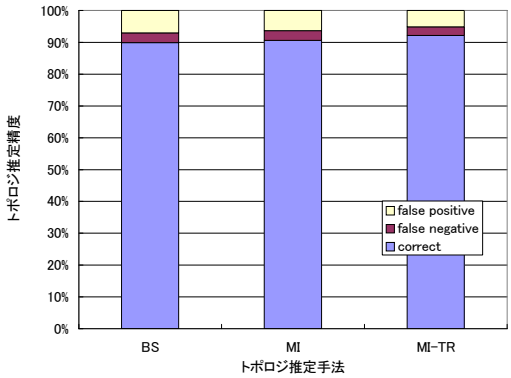
表 5.3: 3 クラスタの推定時間

推定手法	推定時間	測定パケット数
BS	9.50 秒	39474
MI	7.98 秒	31048
MI-TR	6.76 秒	13820

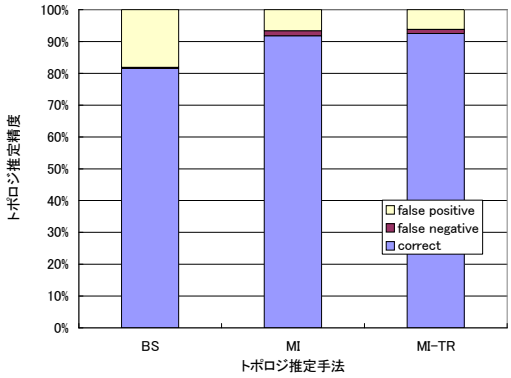
5.3 3 クラスタの推定

次に，hongo クラスタ，chiba クラスタ，okubo クラスタの 3 クラスタからなるマシン 196 台のネットワーク環境について，各手法の性能を調べた。手法 MI 及び手法 MI-TR の RTT 測定モデルは，同一クラスタ内のマシンペアについては $\Gamma(40, 0.9)$ に，互いに異なるクラスタに所属しているマシンペアについては $\Gamma(10, 0.8)$ に従うものとした。推定の精度に関する実験結果を結果を図 5.2 に，所要時間に関する結果を表 5.3 に示す。

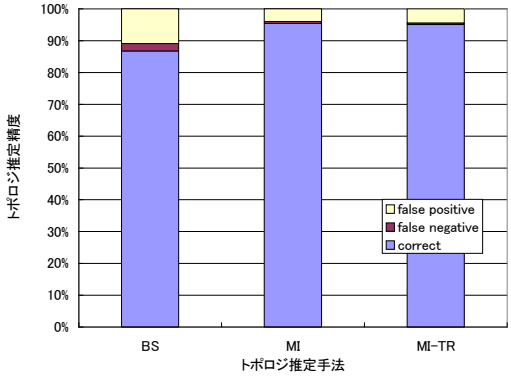
全ての手法について約 9 割の推定精度が達成されているが，手法 MI 及び手法 MI-TR によって測定パケット数の軽減がなされており，複数のクラスタからなる環境においても，これらの手法が推定時間の短縮について有効に機能していることが分かった。



(a) hongo クラスタの場合



(b) chiba クラスタの場合



(c) sheep クラスタの場合

図 5.1: 各手法における単一クラスタの推定精度

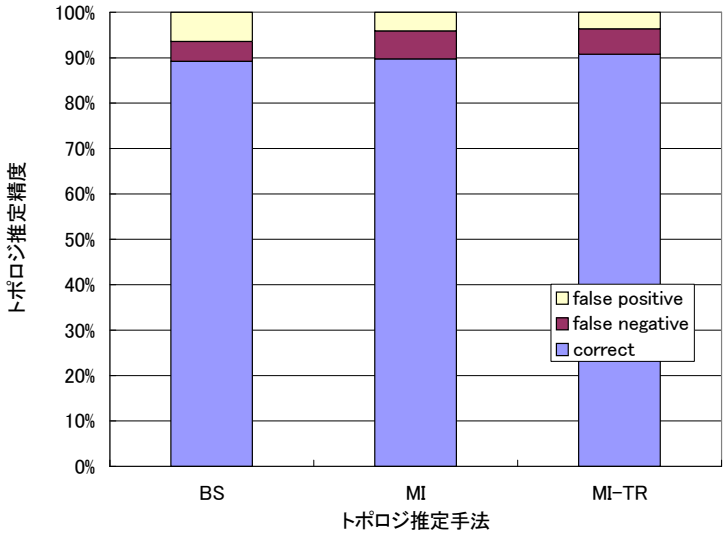


図 5.2: 3 クラスタの推定精度

第6章　結論

6.1　本論文のまとめ

本論文では，以前の我々のトポロジ推定アルゴリズムを踏まえ，その精度の高さを維持しつつ，更なる高速化を施す手法について述べた。

我々はそこで，RTT 測定値の分布と，予備推定トポロジの応用可能性の 2 点に配慮し，以下の手法を提案した。

- RTT 測定値の分布モデルに基づき，RTT 測定値の取得個数と，その中で有効とする個数を必要最低限の値に設定した。
- 主推定トポロジについて各マシンの位置を推定する処理に先立ち，予備推定トポロジについて同様の処理を行うことで，主推定トポロジの処理に要する可能性のある RTT 測定値を事前に取得するものとした。

以上の手法によって，3 クラスタ 196 台の環境を推定するにあたり，以前の手法では 9.50 秒を要したが，今回提案した手法を全て利用することによりこれを 6.76 秒に短縮でき，1.4 倍の高速化を達成したことを確認した。また各手法はトポロジの推定精度を害することなく，高速化を図りながらも従来の高い推定精度を維持できていたことを確認した。

6.2　今後の課題

今回提案した手法により，トポロジ推定の精度に配慮した高速化については一つの達成を見た。次は推定精度に関わる諸問題に対する改良を試みたい。

4.2 節にて述べた RTT 測定モデル $\Gamma(n, r)$ について，パラメータ n, r として適切な値は各クラスタについて固有であり，ある定数を一般的なトポロジ推定手法へと組み込むことはできない。しかし，多数の RTT を観測したとき，観測値の大数が狭い範囲に集中するという分布は，一般性の高い知識として応用できるものと予想される。このとき，RTT 測定値の代表値及び誤差直径をより適切に取得するための手法として，トポロジ推定または RTT 測定をマルチパス化したり，または過去に行った推定の履歴を参照するなどがあり得る。我々は次に，これらの手法を用いることによって，より一般的なネットワークのトポロジを，より少ない知識に基づき，より高精度に推定することを目指す。

参考文献

[1] Oliver Aumage and Guillaume Mercier. MPICH/MADIII: a Cluster of Clusters Enabled MPI Implementation. *3rd International Symposium on Cluster Computing and the Grid*, pages 26–33, 2003.

[2] Richard Black, Austin Donnelly, and Cédric Fournet. Ethernet Topology Discovery without Network Assistance. In *ICNP*, pages 328–339, 2004.

[3] Yuri Breitbart, Minos Garofalakis, Ben Jai, Cliff Martin, Rajeev Rastogi, and Avi Silberschatz. Topology discovery in heterogeneous IP networks: the NetInventory system. *IEEE/ACM Trans. Netw.*, 12(3):401–414, 2004.

[4] Mark Coates, Rui Castro, Robert Nowak, Manik Gadhiok, Ryan King, and Yolanda Tsang. Maximum likelihood network topology identification from edge-based unicast measurements. *SIGMETRICS Perform. Eval. Rev.*, 30(1):11–20, 2002.

[5] Mathijs den Burger, Thilo Kielmann, and Henri E. Bal. "TOPOMON": A Monitoring Tool for Grid Network Topology. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part II*, pages 558–567, London, UK, 2002. Springer-Verlag.

[6] Benoit Donnet, Philippe Raoult, Timur Friedman, and Mark Crovella. Efficient algorithms for large-scale topology discovery. *SIGMETRICS Perform. Eval. Rev.*, 33(1):327–338, 2005.

[7] N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Multicast topology inference from measured end-to-end loss. *IEEE Transactions in Information Theory*, 48:26–45, 2002.

[8] Edgar Gabriel, Michael Resch, Thomas Beisel, and Rainer Keller. Distributed Computing in a Heterogeneous Computing Environment. In *Proceedings of the 5th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 180–187, London, UK, 1998. Springer-Verlag.

[9] Arijit Ganguly, Abhishek Agrawal, P. Oscar Boykin, and Renato Figueiredo. WOW: Self-organizing wide area overlay networks of virtual workstations. In *Proceedings of the 15th*

IEEE International Symposium on High Performance Distributed Computing (HPDC), pages 40–41, 2006.

[10] Toshiyuki Imamura, Yuichi Tsujita, Hiroshi Koide, and Hiroshi Takemiya. An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers. In *Proceedings of the 7th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 200–207, London, UK, 2000. Springer-Verlag.

[11] Nicholas T. Karonis, Brian Toonen, and Ian Foster. MPICH-G2: a Grid-enabled implementation of the Message Passing Interface. *J. Parallel Distrib. Comput.*, 63(5):551–563, 2003.

[12] Bruce Lowekamp, David O'Hallaron, and Thomas Gross. Topology discovery for large ethernet networks. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 237–248, New York, NY, USA, 2001. ACM Press.

[13] Gary Shao, Fran Berman, and Rich Wolski. Using Effective Network Views to Promote Distributed Application Performance. In *Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 2649–2656, 1999.

[14] Tatsuya Shirai, Hideo Saito, and Kenjiro Taura. A Fast Topology Inference — A building block for network-aware parallel computing. In *Proceedings of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 11–21, June 2007.

[15] Skitter. http://www.caida.org/tools/measurement/skitter.

[16] Kenjiro Taura. GXP: An Interactive Shell for the Grid Environment. In *IWIA '04: Proceedings of the Innovative Architecture for Future Generation High-Performance Processors and Systems (IWIA'04)*, pages 59–67, Washington, DC, USA, 2004. IEEE Computer Society.

謝辞

本研究を進めるに当たり，近山隆教授ならびに田浦健次朗准教授より貴重なご助言をいただきました。自分の身勝手さにお付き合い下さった田浦健次朗准教授には感謝の念に絶えません。

横山大作特任助教授と三輪誠さんからは，研究の方針を見定めるための助言をいただきました。

鴨志田良和さんと斉藤秀雄さんからは，プログラムの高性能化に関する重要な部分について，多くの有用なテクニックを教えていただきました。

要所で激励してくれた斉藤大君を始め，関谷岳志君，高橋慧君，吉田慎一郎君には何度となく相談や雑談に応じてくださいました。

InTrigger がなければこの研究を行うことはなかったでしょう。各拠点の管理者の皆様に感謝いたします。

そして最後に，手触りの楽しいアルゴリズムを遺した白井達也さんにありがとうの意を表します。

平成 20 年　2 月　4 日