第 12 部

NTP

第 1 章

はじめに

インターネット上に正確な時刻情報を供給すべく、文部省国立天文台のメンバと共同で活動してきた NTP ワーキンググループは、本年度でひとまずその活動を終える。したがって、本報告が NTP ワーキンググループとしての最後の報告書となる。

昨年度までに著者らは以下の活動を行ってきた [104][105]。

> 文部省国立天文台のセシウム原子時計を時刻源とする NTP stratum 1 サーバの構築と運用
>
> GPS 衛星からの電波を時刻源とする NTP サーバの性能評価

その結果、「他組織にある、stratum 1 との間で時刻を合わせるよりは、GPS 衛星からの電波を時刻源とする stratum 1 を自組織でたちあげ、これを時刻基準として組織内の時計をあわせた方がよい精度を保てる」という結論がえられた。これは、インターネット上に今後も NTP プロトコルを流通させることはあまり意味がないことを示している。本ワーキンググループの名称を否定するようなこの結論は、それ自体興味深いが、著者らはそれ以上に以下の点に興味を持った。

> それではインターネットはどの程度混みあい、通信時間はどのくらいゆらいでいるのだろうか

そこで本年度は、「正確な時刻源から時刻情報の供給を受けて稼働している計算機が複数台分散している環境」下で、この環境を生かしたネットワークトラフィックの揺らぎ計測を試み、上記の興味に自ら答えることにした。

— — — — — — — — —

ネットワークを流れるデータの損失率や通信に要する時間は、計算機間を結ぶネットワークの状態に大きく関わっている。例えば、ある計算機間で回線の能力を超える大量のパケットのやり取りを行なおうとすると、その経路はパケットの通過が困難になる。あるいは、もし物理的に線が切れてしまうとその経路は使用不能となり迂回しなければならない。したがって、ネットワークを正常に運用するためには、ネットワークの状態を的確に把握する

ことが重要である。ところが、インターネットのように大規模なネットワークでは、ネットワークの状況を的確に把握することは容易ではない。なぜなら、インターネット上では、多くの人々が常に各々異なる作業を行っており、その影響でネットワークの状態が常に変化するからである。もちろんある瞬間の状態を把握するためのソフトウェアは多く存在する。しかし、常に回線の状態が変化しているネットワークの性質を考慮に入れて、変化の様子を的確に捉える目的で設計されたソフトウェアは存在しなかった。

そこで、動的に変化するネットワークの様子を観察する方法を考え、その手段として用いるソフトウェアを実装した。このソフトウェアは、ある計算機まで経路が確保されているかどうかを調べる ping プログラムをもとに作成されたもので、仮に行きと帰りの経路が異なった場合、その事実を把握することができる。既存の ping が、ある 2 つの計算機間の往復時間しか計測できない「往復 ping」だとすれば、このソフトウェアは、片道の時間を計測することが可能なので、「片道 ping」と名付けた。往復 ping では、もし通常とは異なる経路を使用した場合でも、それを把握することは困難であった。また、送信と受信で異経路を使用している際に、トラフィックの変動が判明したとしても、送信経路と受信経路のどちらでトラフィックの変動が生じたのか分からない。しかし、片道 ping を用いれば、異経路使用の検出は容易なことであるし、片道 ping を継続的に使用することによって、送信と受信のそれぞれについて、トラフィックを動的に把握できる。

本報告では、まず既存のトラフィック計測のためのソフトウェアを紹介し、その特徴を明らかにする。次に、片道 ping を利用するために必要な時刻同期について言及し、既存のソフトウェアの欠点を補うべく考案した片道 ping の設計と実装について説明する。また、ネットワーク管理ツールとしての応用例と、ネットワーク経路の変化の様子を調べる方法について述べる。

第 2 章

片道 ping によるネットワークトラフィック計測法

本章では、まずネットワークトラフィックを計測する意義とそのために利用されているさまざまな道具について述べ、さらにその問題点を明らかにする。そして、今回新たに片道 ping を提案し、これを用いてネットワークトラフィックを測定する方法について述べる。

2.1 ネットワークトラフィック計測の意義

ネットワークは、さまざまな接続形態を持っており、また流れるパケットの種類も多い。ネットワークを正常に運用するためには、接続性の確認や、ネットワーク上でのパケット流量測定などにより、ネットワークの状態を的確に把握することが重要である。なぜなら、ネットワークの状態が把握できていないと、異常が発生しても認識できないし、また認識できたとしても、適切な対処方法が分からないからである。早期に異常の認識ができれば、異常による被害は最小限で食い止められる。また、対処法が分かっていれば、修復作業にかかる手間を軽減させることができる。

しかし、ネットワークの状態を的確に把握することはさらに難しい。なぜなら、把握する必要がある要素は数多くあるうえに数量化して評価しにくい要素も多い。さらに、必要と思われるあらゆるトラフィックを監視しても、その中から本当に必要な情報を実時間抽出するのは困難な場合がある。さらに、ネットワークの状態は動的に変化するので、ある時点の状態を把握できたとしても、一定時間経過すればそれは正しい状態ではない。

ネットワークの状態が動的に変化するのは、ネットワーク上を介して行われる通信が常に変化するからである。動的に変化するネットワークの状態を把握するためには、変化する要因であるトラフィックの計測を行わなければならない。

次節では既存のネットワークトラフィック計測ツールをいくつかとりあげ、その特徴をまとめる。

2.2 既存のネットワークトラフィック計測ツールと問題点

ネットワークの状態を把握するために、さまざまなトラフィック計測ツールを利用できる。その種類は多く、ネットワーク管理者は目的に応じてツールを使い分けている。ここ

では、ネットワークトラフィック計測ツールのうちいくつかをとりあげ、その特徴、および問題点を指摘する。

1. ping

   ある計算機から特定の計算機へ ICMP[106] パケットを送信し、その計算機への到達可能性と往復に要した時間を調べるためのコマンド。

   - 特徴
     - パケットを送受信することによって、パケットの往復に要する時間、およびパケットの到達率を調べることができる。
     - パケットサイズがデフォルトで 64 バイトである。これは、TCP パケットの標準的な大きさと比べて小さい。
   - 欠点

     経路に関する情報が把握できない。例えば以下のようなものである。
     - 経路に異常が生じていて、パケットが迂回していた場合でも、それを検出しにくい。また、異常の原因を特定できない。
     - 経路に関する情報を把握できない。

2. traceroute

   ある計算機から特定の計算機への経路を、IP[107] の TTL(Time-To-Live) と ICMP の TIME_EXCEEDED を用いて調査する。

   - 特徴
     - 経路上にある各ルータまでの往復時間を調べることができる。
     - ある計算機を指定した際に次にたどるべき計算機が記されているルーティングテーブルを無視してパケットをトレースしたり、別のルータを経由して経路を調査したりすることが可能である。
     - 回線の異常個所の発見やルーティング試験に用いることができる。また、ping と同じ測定をすることも可能である。
   - 欠点
     - IP の TTL を 1 ずつ増やしながらパケットを続々と送り続けることによって調査を行うため、ネットワークへの負荷が大きい。
     - 相手のホストが遠すぎて、パケットが到達する前に traceroute のタイムアウト時間が先に来てしまうような時、そのシステムが落ちているのか、タイムアウトになっただけなのか分からない。
     - 連続してネットワークを監視するための設計にはなっていないため、異常の検知には向かない。

– 計測中に経路が変化した場合、traceroute の経路は実際の経路と一致しない。

3. netstat

ネットワークの状態に関するさまざまな情報を提供する。特に、ルーティングテーブルの統計的な検査などに用いられる。

- 特徴
  - ネットワークの構築後のテスト、トラブルの分析、特定などの用途に用いることができる。
  - ネットワークの情報を統計的に分析することもできるため、過去のトラフィック情報を保存しておけば、比較することでトラブルの予見をすることも可能である。
- 欠点
  - 自分の計算機に関する情報しか得ることができない。そのため、経路に異常があった場合、それを検知することができない。
  - 統計情報は出力されるものの、連続使用を前提とした設計にはなっていないため、ネットワークの動的な変化を調べるのことが難しい。

以上のように、いずれのツールも一長一短である。１つのツールのみでは、完全なトラフィック計測ツールとしての役目を果たすことはできない。

2.3　片道 ping によるネットワークトラフィックの計測

ネットワークに異常が起こった際にその原因を調べる、あるいは、異常を回復させた際、その異常を取り除くことができたのかを調査するツールは多くある。しかし、ネットワークに異常が発生した際それをすばやく察知するためのソフトウェアはあまり普及していない。もし、ネットワークのトラフィックを動的に検知することができれば、ネットワークに異常が発生した際、どこかで異常が発生したということが分かる可能性が大きい。また、ネットワークが正常に機能しているように見えても、トラフィックが通常とは違う傾向を示していれば、なにか異常が発生しようとしていると見ることもできる。その場合、早期にネットワークを調査し、異常があった場合それを修正すれば、重大な障害が起こるのを未然に食い止めることができる可能性がある。また、ネットワークの現在の状況が一目で把握できれば、ネットワークの成長の様子などなにか興味深いことが分かる可能性がある。

そこで、異常の検知やネットワーク調査に役立つよう、片道 ping というソフトウェアを設計し、ネットワークのトラフィック検出を行うことができるようにする。

設計に当たっては、前節に挙げたトラフィック計測ツールの欠点を補うということや、新たな特徴を持たせるということを考慮に入れて、次のような条件を満たすようにした。

- ネットワーク経路の調査ができるようにする。
- ネットワーク上でのパケットの到達の可・不可を検出できるようにし、到達可能なら、到達率や到達にかかる時間を計測できるようにする。
- 往路、及び復路の経路が異なる場合がある。これを検出できるように、パケットの往復時間だけでなく、行きと帰りの時間を別々に計測できるようにする。
- 継続的にネットワークの状態を把握するために、連続して発信ができるようにする。
- ネットワーク全体の動向を把握できるようにするため、測定点を多数設けることができるようにする。

このような特徴を持つのが、今回設計と実装を行った“ 片道 ping ”である。

片道 ping の最大の特徴は、パケットが片道に要する時間を計測することができるところにある。ネットワーク上では、必ずしも行きと帰りで所要時間が同じとは限らない。あるホスト間における、行きと帰りにかかる時間を分離することで、さらに詳しい計測が可能となっている。

片道 ping と時刻同期

片道 ping の最大の特徴である、片道の時間を計測するためには、発信側の時刻と着信側の時刻が一致していなければならない。例えば発信側の時計が着信側の時計よりも進んでいたとしたら、実際に送信にかかる時間よりも小さく計測されてしまう。

ところが、インターネット上で、全く別のところにある複数のシステムの時間を合わせることは難しい。その理由として以下がある。

- 原子時計のような時刻源となる機器が高価である。
- システムによってシステムクロックの精度が異なる。
- インターネット上の時刻同期の手段として標準的に使用される NTP[108] が、パケットの送信と受信にかかる時間の違いを考慮に入れてない。

以上のように、問題はソフトウェア上、ハードウェア上の両方にある。

時刻の概念や、時刻同期の方法などについては、既に報告してあるのでここでは述べない [104]。以下では現在のインターネット上で片道 ping を行う際、時刻同期の観点から生じる問題について述べる。

片道 ping の問題点

NTP を使って時刻同期する場合、NTP のアルゴリズムの仮定から、ネットワーク上でのパケットの行きと帰りの時間が、同じでなければ正確な同期が行えない。しかし、実際には、ネットワークの経路が違っていたり、また送信時と受信時でネットワークのトラフィックが異なっていたりして、伝達にかかる時間は必ずしも同じではない。そのずれがわずかであり、ガウス分布のような統計的に既知なばらつきであれば、何度もデータを取って補正することが可能であるが、この仮定は必ずしも成り立たない。

また、同期を行う計算機の性能も問題になる。比較的遅い計算機の場合、パケットを受信してから時間を計測するまで、時間を計測してからパケットを送信するまでの時間差が大きくなってしまい、それが一定の誤差を生みだす。それだけならば定数を足したり引いたりすることで修正できるが、計算機は時刻を合わせるためだけにあるのではなく、他のプロセスも動いている。そのため、計算機にかかる負荷は一定でなく、その影響で誤差が生じてしまう。

さらに、計算機の内部時計の粒度も問題となる。内部時計の粒度とは、その計算機が時間を計測するために採用している時間の最小単位である。粒度は、例えば PC+BSD/OS 2.0.1 だと 1 マイクロ秒であるが、SONY NWS3400+NEWS-OS 4.1.2R であると 10000 マイクロ秒になってしまう [104]。粒度が 10000 マイクロ秒であると、どんなにたくさんのデータを取ってきても精度が粒度以下にはならず、同期精度の低下を招く原因となる。

また、時計自体についても問題がある。原子時計はきわめて正確な時刻を刻むことが可能であるが、高価なためたくさん設置することが難しい。通常用いられる水晶時計では、片道 ping を行うには精度が悪すぎて、NTP による同期を行わざるを得ない。

そのため、NTP で時刻同期を行う場合、どうしても stratum1 サーバに負荷が集中してしまうという問題が生じる。stratum2 から時刻を得るにしても、stratum2 ではすでに精度は数ミリ秒程度に落ちており、十分な同期精度を得ることが難しい。

このことを解決するには、GPS 受信機を用いた、stratum1 サーバ並の精度の時計を各計算機に設置することが考えられる。GPS 受信機は、原子時計ほどの精度は無いものの、原子時計よりもずっと安価で、かつ片道 ping を行うのに十分な精度が保たれている。

ただし、現状では、GPS 受信機は普及途上にあり、片道 ping で実験を行う環境は整っていない。しかし今後急速な普及が予想されるので、今回は何らかの方法で時刻同期が十分に取れていると仮定のうえで設計・実装を行った。

第 3 章

片道 ping の設計と実装・実例

この章では、片道 ping の設計と実装および計測結果について述べる。

3.1 片道 ping の設計

片道 ping を用いてあるホストまでの経路変動を調べるには、相手ホストへ向けてパケットを送出し、その挙動を観察するのがよい。なぜなら、経路の変化によりもっとも大きな影響を受けるのは、その経路を使用するパケットだからである。

あるホストへパケットを送信したパケットを再び受信するには、通常は、受信したパケットを送信元に送信するサーバプロセスが送信先ホストに無ければならないし、送信を行うにはクライアントプロセスが必要である。

本節では片道 ping の設計について述べるが、それぞれの役割を明確にするために、設計をパケット部・クライアント部・サーバ部の 3 部に分けて行う。

3.1.1 パケットの設計

クライアントとサーバ間のパケット送受信時間を計測するためには、クライアントとサーバが、パケットの受信および送信時に時刻を計測しなければならない。パケットが多くの情報を運搬することは、ネットワークに負荷をかけてしまうため、以下に示す方針で実装を行うことにした。

- パケットには時刻の情報を運搬させる。
- 時刻の情報以外は極力省く。

3.1.2 クライアントの設計

クライアントは、以下の機能を持たせることにした。

- パケットを送信するホストを指定する。

- 時刻を計測し、パケットに記述して、送信する。
- 送ったパケットがサーバ側から戻ってくるのを待ち、戻ってきたら時刻を計測する。
- 送られて来たパケットが正しい物であったか確認する。
- 送られて来たパケットに含まれていたデータを使って、往路・復路・往復の時間を計算し、出力する。

3.1.3　サーバの設計

サーバは、以下のような機能を持たせることにした。

- クライアントからパケットが送られてくるのを待ち、送られてきたら時刻を計測する。
- 送られて来たパケットが正しいものかどうかをチェックする。
- 時刻を計測して、クライアントへパケットを転送する。

3.2　片道 ping の実装

上の設計をもとに、次のような実装を行った。

3.2.1　パケット部

パケットの構造を図 3.1に示す。
図中の各フィールドの意味を以下に示す。

type: パケットの種類を示す。現在パケットの種類は 2 種で、以下に示すものがある。

1 クライアントがサーバに送っているパケット。

0 サーバがクライアントに送っているパケット。

char 型で、1 バイトである。

code: 送受信されるデータの種類を示す。現在は 0 で固定である。char 型で、1 バイトである。

id: パケットの識別番号を示す。クライアントが、受信したパケットを、自分が送出したパケットであるか判断するのに用いる。クライアントのプロセス番号が記述されている。short 型で、2 バイトである。

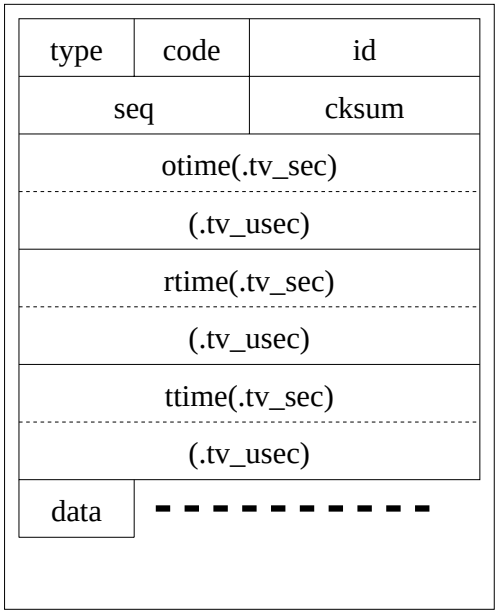

図 3.1: 片道 ping のパケット

seq: パケットの番号を示す。パケットの順番を正しく示すために用いる。また、パケットが途中で損失した場合に、何番目のパケットが損失したのか知るために用いる。クライアントからの送出順に番号が記述されている。short 型で、2 バイトである。

cksum: チェックサムを示す。経路上でパケットの改変や欠損が起こったことを知るために用いる。short 型で、2 バイトである。

なお、この部分までがパケットのヘッダである。クライアントおよびサーバは、ヘッダを見て片道 ping のパケットであることを知る。ヘッダのサイズは 8 バイトである。

以下は、パケットのデータ部である。

otime: クライアントのパケット発信時刻を示す。クライアントがパケットを発信する直前に計った時刻が記述されている。マイクロ秒までのデータを記述できる timeval 型で、8 バイトである。

rtime: サーバのパケット受信時刻を示すサーバがパケットを受信した直後に計った時刻が記述されている。timeval 型で、8 バイトである。

ttime: サーバのパケット転送時刻を示す。サーバがパケットを転送する直前に計った時刻が記述されている。timeval 型で、8 バイトである。

data: その他のデータを示す。ユーザはこの部分の大きさを変えることで、パケットのサイ

ズを変更できる。char 型で、ユーザがクライアントプログラム起動時に 0 バイトから 4088 バイトまで指定できる。

なお、パケットのサイズは、クライアントプログラム起動時に 8 バイトから 4096 バイトまで指定できる。

3.2.2　クライアント部

クライアントのプログラム名は pingpong と名付けた。pingpong が行う作業を以下に挙げる。

1. 接続先ホストを指定する。
2. パケットのヘッダにデータを書き込む。
3. 時刻を計測し、それをパケットに書き込んでサーバに発信する。
4. サーバからパケットが転送されてくるのを待つ。
5. パケットを受信すると、その直後の時刻を計測する。
6. 送信時間、受信時間、往復時間を計算して、標準出力に表示する。
7. 2. から 6. を繰り返す。
8. ユーザの指定した数のパケットを送出し終えたり、ユーザがプログラムを中断したりすると、送信、受信および往復時間の最大、最小および平均を標準出力に表示して、終了する。

起動の際、オプションを指定して、動作を制御することができる。pingpong のオプションを以下に示す。

pingpong [-drsv] [-n npackets] [-t timing] host [datasize]

`-d` デバグ用のオプション。指定すると、デバグモードになる。

`-r` 直接送信用のオプション。指定すると、ルーティングテーブルを無視して、直接ホストにパケットを送信する。指定ホストに直接送信することができない場合は、エラーとなる。

`-s` 毎回統計表示するためのオプション。指定すると、クライアントがパケットを受信するたびに、送信、受信および往復時間の最大、最小および平均を標準出力に表示する。

`-v` 冗長出力用のオプション。type が 0 でも 1 でもないパケットが送られてきた際、そのパケットに関する詳細な情報を出力する。

-n 送信パケット数指定用のオプション。次に指定する引数で、パケットの送信数を指定する。このオプションを指定しない場合、ユーザがプログラムを中断しない限り、クライアントはパケットを送信し続ける。

-t 送信間隔指定用のオプション。次に指定した引数で、パケットの送信間隔を秒で指定する。このオプションを指定しない場合、1 秒毎にパケットを送る。

オプションの指定をしたのちに、送信するホストの指定をする。送信ホストの指定がない場合は、エラーとなり終了する。

パケットのデータ部のサイズを、ホスト指定の後ろに記述できる。指定しないと、データ部の大きさは 24bytes である。

パケットの転送には、UDP[109] を用いている。ポートは、現在 12467 番を使用している。なお、片道 ping が使用するポート番号は、コンパイル時に指定できる。

3.2.3 サーバ部

サーバのプログラム名は pongd_udp と名付けた。pongd_udp が行っている作業を以下に示す。

1. クライアントからパケットが送られてくるときのための準備をする。
2. パケットを受信すると、その直後の時刻を計測し、パケットに書き込む。
3. パケットの送信直前に時刻を計測し、パケットに書き込んでクライアントに転送する。
4. 2. と 3. を繰り返す。

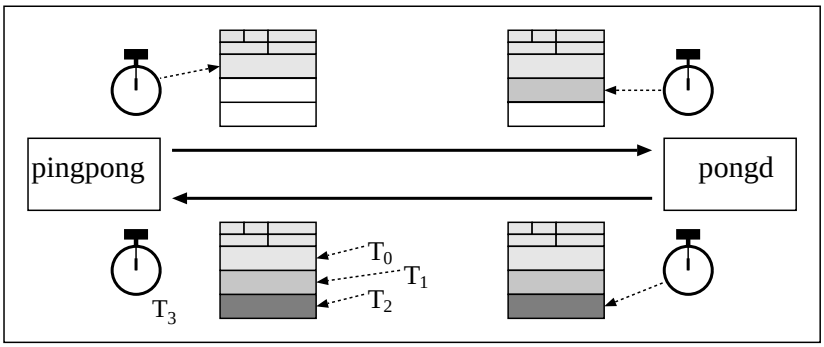

図 3.2: クライアントとサーバ間のデータ送受信モデル

起動の際、オプションを指定できる。pongd_udp のオプションを以下に示す。

pongd_udp [-dh]

-d デバグ用のオプション。指定すると、デバグモードになる。

-h ヘルプ用のオプション。指定すると、pongd_udp の使用方法を示して終了する。

すでにサーバが起動中のシステムで pongd_udp を起動すると、起動中であると警告して終了する。

3.2.4　表示形式

pingpong が前項に示したクライアントの作業 6. で行っている出力を以下に示す。

```
byte bytes from addr:  seq=seq.  stime=send ms rtime=recv ms time=rt ms
```

そのうち、下線で示されている部分は以下の意味を持つ。

- byte
  受信したパケットのサイズ。
- addr
  パケットを送って来たホストの IP アドレス。ドット記法で表される。
- seq
  受信したパケットの番号。
- send
  クライアントからサーバへのパケット送信時間。
- recv
  サーバからクライアントへのパケット受信時間。
- rt
  クライアントとサーバ間のパケット往復時間。

また、pingpong が前項に示したクライアントの作業の 8. で行っている出力を以下に示す。

```
   ----host PINGPONG Statistics----
num packets transmitted, recv received, percent
round-trip (ms) min/avg/max = rtmin/rtavg/rtmax
```

```
send (ms) min/avg/max = smin/savg/smax
receive (ms) min/avg/max = rmin/ravg/rmax
```

下線で示されている部分は以下の意味を持つ。

- host

  パケットを送信したホスト名。

- num

  送信したパケットの数。

- recv

  受信したパケットの数。

- percent

  受信できたパケットの割合。

- rt,s,r

  それぞれ往復、送信、受信を意味する。

- min,avg,max

  それぞれ最小、平均、最大時間を意味する。

3.2.5 制限事項

現在の片道 ping には、次の制限がある。

- パケットを送出するホストの時刻をそのまま得るため、時刻同期がなされていないと、正しい送受信時間を表示しない。

- パケットの送受信をしたときと、実際に時刻を計測したときには差があり、送信、受信、往復時間ともに実際の時間よりも大きく表示される。また、負荷の大きいホストでパケットの送受信を行うと、誤差が特に大きくなる。

- パケットを送信するホストは、pongd_udp もしくは pingpong がインストールできなければならない。

また、トラフィックを動的に把握し、表示するツールとして、コントローラを作成しなければならないが、今回は、その基本部分となる、片道 ping の設計・実装を行った。

3.3　片道 ping を利用した計測

本節では片道 ping を用いたネットワークトラフィック計測の可能性について検討する。

3.3.1　片道 ping の妥当性

片道 ping を、トラフィック計測ツールとして利用可能であることを示すために、前節の方法で実装した片道 ping に問題がないか調べてみた。

最初に、片道 ping から得られる時間が、送受信の時間を正しく表示しているかどうかを調べるために、以下の測定を行った。

- 正確な時計を持つ隣接したホスト間で片道 ping のパケットを送受信し合い、送信時間と受信時間を比較する。

実験には、文部省国立天文台の xenon 及び cesium を用いた。この 2 台の計算機は、以下の条件を満たす。

1. どちらも、同一の原子時計からの時刻情報をもとに計時している。
2. 同じセグメント上に存在している。このセグメントには 4 台の計算機がつながっているだけなので、コリジョン等の発生は少ない。
3. ほぼ同一のシステム構成になっている。

2.3. より、この 2 つの計算機間でパケットを送受信した場合、その経路は同じで、送信と受信の時間はほぼ一致すると予想される。

以下は、この 2 つの計算機間で、10000 個のパケットを送受信した結果である。

- xenon から cesium にパケットを送った結果。

```
----cesium.mtk.nao.ac.jp PINGPONG Statistics----
10000 packets transmitted, 10000 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 1.334/1.405/11.415
send       (ms)  min/avg/max = 0.649/0.696/10.720
receive    (ms)  min/avg/max = 0.612/0.647/9.140
```

- cesium から xenon にパケットを送った結果。

```
----xenon.mtk.nao.ac.jp PINGPONG Statistics----
10000 packets transmitted, 10000 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 1.276/1.349/11.215
send       (ms)  min/avg/max = 0.604/0.653/10.534
receive    (ms)  min/avg/max = 0.593/0.632/9.168
```

このデータから以下の点が分かる。

1. 11.415ms などの大きな時間を計測することがある。
2. xenon から cesium へのパケット送受信時間の方が、cesium から xenon へのパケット送受信時間より大きい。
3. 送信にかかる時間の方が受信にかかる時間よりも大きい。さらに、xenon から cesium へのパケット送受信の時間差の方が、cesium から xenon へのパケット送受信の時間差よりも大きい。

まず、1. について検討する。片道 ping はパケットを送受信した瞬間の時刻を計測しているわけではなく、gettimeofday システムコールを行う時間と sendto・recvfrom システムコールを行う時間の差をできるだけ小さくすることによってパケット送受信時の時刻を計っている。そのため、この隙間に、計算機がメモリ切り替えやプロセスの切り替えを行ったりすると、送受信とのタイミングに差が生じる。1. の原因は、この時間差にあると思われる。

次に、2. および 3. について検討する。まず、往復時間が 3ms より大きいものは、上に示した、ネットワークとは関りのない原因によって生じたものと考えて除去し、その上で平均を取ってみた。

まず、xenon から cesium に送った場合には、20 個のパケットが棄却され、平均は 1.395ms ほどになった。逆に、cesium から xenon に送った場合には、26 個のパケットが棄却され、平均は 1.337ms ほどになった。

送信、受信時間についても、2ms より大きいものは異常と見なして、平均を取ってみることにした。xenon から cesium の送信に関して、棄却パケットは 10 で平均時間は 0.648ms、受信に関しては、棄却パケット 18 で平均時間は 0.624ms となる。cesium から xenon の送信に関して、棄却パケットは 8 で平均時間は 0.690ms、受信に関しては、棄却パケットは 11 で平均時間は 0.642ms となる。

1. により生じたと考えられる物を除いても、やはりクライアントのホストがどちらであるか、送信・受信のどちらの作業であるかによって、時間が異なっている。

3. で生じた誤差の原因となっているのは、「送信か受信か」ということであり、機種に依存した話ではない。そのため、片道 ping を実装した際の仕様上の問題であると考えられるが、他の問題を取り除かない限りはっきりしない。

2. は、計算機に依存した違いが、この差を生み出しているものと思われる。cesium と xenon はパケットの送受信に同じ経路を使用しているので、送受信の時間に大きな差は出ない。そこで、対処療法として、計算機毎に定められた定数を、送信時間と受信時間に加減することで、補正を行うことにする。本来ならば、トラフィック変動を長期的に調査したうえで、時間や曜日などに依存した変数により補正すべきである。しかし、送受信する瞬間の時刻を得られるよう工夫をするほうが、この短期的な実験で用いるには効率的で実用的な手段である。よって現段階では、もっとも典型的でデータ量も多い、上に示したデータを利用して補正を行う。

補正する際、どちらか一方のみから数値を足したり引いたりするのでは、往復にかかる時間との矛盾が生じる。そこで、両方から同じ量を加減して調整する。

上の例をこの方法で補正してみる。

xenon から cesium へ片道 ping を行った際の送信時間は、測定結果から 0.025ms 引いたものとし、受信時間は、その結果に 0.025ms 足したものとする。

cesium から xenon への送信時間は、測定結果から 0.011ms 引いたものとし、受信時間は、その結果に 0.011ms 足したものとする。

実際には、ネットワーク的距離が数十から数百 ms のホスト間で測定を行うのが有効であり、また実用的であるため、この機種依存誤差の影響は小さいと思われる。

ただし、長期に渡る統計的測定や、隣接ホスト間で測定を行う場合など、この誤差が無視できなくなる状況も生じうる。この補正は暫定的なものであり、送受信の時間のずれを取り除くための、新たな工夫をする必要がある。

次に、ping との比較評価を行う。

ping の出力の例を以下に示す。

```
64 bytes from 133.40.40.131: icmp_seq=995 ttl=255 time=0.976 ms
64 bytes from 133.40.40.131: icmp_seq=996 ttl=255 time=0.973 ms
64 bytes from 133.40.40.131: icmp_seq=997 ttl=255 time=0.978 ms
64 bytes from 133.40.40.131: icmp_seq=998 ttl=255 time=0.983 ms
64 bytes from 133.40.40.131: icmp_seq=999 ttl=255 time=0.984 ms

--- xenon.mtk.nao.ac.jp ping statistics ---
1000 packets transmitted, 1000 packets received, 0% packet loss
round-trip min/avg/max = 0.961/0.992/6.585 ms
```

これに対し、片道 ping は以下のような出力を行う。

```
32 bytes from 133.40.40.131: seq=9995. stime=0.641 ms rtime=0.625 ms time=1.329 ms
32 bytes from 133.40.40.131: seq=9996. stime=0.637 ms rtime=0.630 ms time=1.331 ms
32 bytes from 133.40.40.131: seq=9997. stime=0.646 ms rtime=0.625 ms time=1.335 ms
32 bytes from 133.40.40.131: seq=9998. stime=0.631 ms rtime=0.626 ms time=1.321 ms
32 bytes from 133.40.40.131: seq=9999. stime=0.636 ms rtime=0.625 ms time=1.323 ms

----xenon.mtk.nao.ac.jp PINGPONG Statistics----
10000 packets transmitted, 10000 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 1.276/1.349/11.215
send       (ms)  min/avg/max = 0.604/0.653/10.534
receive    (ms)  min/avg/max = 0.593/0.632/9.168
```

なお、これをさきほど定義した方法で補正すると、

```
send       (ms)  min/avg/max = 0.593/0.642/10.523
receive    (ms)  min/avg/max = 0.604/0.643/9.179
```

となる。

片道 ping の使用により、往復時間だけでなく、送信時間と受信時間が計測可能となっている。

ところで、片道 ping では、送信にかかる平均時間と受信にかかる平均時間を足しても、往復にかかる平均時間との間に差が生じる。この理由は、片道 ping のサーバが、パケットを受信してから転送するまでの間に、エラーチェックなどの処理を行っているためである。

また、サーバが行っている作業量の差を考慮に入れても、片道 ping の往復にかかる時間の方が大きい。上の二つの出力で、使用している経路は同一である。時刻を得て送信する、受信して時刻を得るという作業も、片道 ping と ping とでは同一なので、受信や送信、時刻に関するシステムコールとも関わりがない。したがって、この理由は、使用しているプロトコルに関係していると考えられる。なぜなら、現在片道 ping は UDP/IP を使用しているのに対し、ping は ICMP を使用しているからである。UDP はメッセージを一旦ユーザが扱えるレベルまでもどすが、ICMP はネットワークソフトウェア自身が処理をする。そのため、往復にかかる時間は、ping の方が速くなっている。

この差がトラフィックの計測に関して支障をきたす恐れがある。そこで、cesium の代わりに東京工業大学の drums とパケットの送受信を行ってみた。

```
--- xenon.mtk.nao.ac.jp ping statistics ---
1000 packets transmitted, 833 packets received, 16% packet loss
round-trip min/avg/max = 75.005/180.355/363.785 ms

----xenon.mtk.nao.ac.jp PINGPONG Statistics----
1000 packets transmitted, 847 packets received, 15% packet loss
round-trip (ms)  min/avg/max = 62.614/181.545/381.406
send       (ms)  min/avg/max = -10123.988/-9918.724/-9672.661
receive    (ms)  min/avg/max = 9895.513/10100.221/10304.095
```

上が ping の出力結果で、下が片道 ping の出力結果である。

片道 ping 結果は、時刻が同期されていない環境のため、送信及び受信時間の結果が奇妙なものとなっている。ここでは往復時間にのみ着目する。ping と片道 ping の往復時間の差は、xenon-cesium 間で 0.357ms、xenon-drums 間では 1.19ms である。これは、xenon と drums の間で途中いくつかのゲートウェイを通過する際に、プロトコルの差がさらに大きく影響したものと考えられる。ICMP を使用した場合は、プロトコル処理にかかる時間はずっと小さいが、UDP/IP を使用した場合は、UDP を処理する速度が影響してくる。

今回は UDP を用いて計測を行ったが、TCP や UDP のパケットが主な構成要素であるインターネットでは、今回の計測も、トラフィックの計測として有効なものと思われる。

3.3.2　片道 ping の応用例

ここでは、片道 ping のさまざまな応用例を示す。

例 1 トラフィックの変化

実際に動的に測定したときの例を示す。章末の図 3.4から図 3.7は、xenon と cesium との間で測定した送信時間と受信時間のグラフである。縦軸がパケットの送信又は受信にかかった時間、横軸が実験開始時からの経過時間である。ただし、この図は送受信時間の変動を見やすくするため、1ms 以上のデータは省略している。

図を見ると、まず xenon から cesium への送信時のデータでは、送受信時間は増加と減少をくり返しているのが分かる。また、いくつか不連続な部分が見える。

受信の方を見ると、安定しているが、送受信にかかる時間が変化するときがある。受信の方も送信と同じような増減傾向を示している。

ネットワーク上で、何らかの変化が起こった影響が出ていると思われる。

今度は cesium から xenon への送信時のデータを見てみる。

受信の方は安定しているが、送信の方では、いくつか突出した部分がある。

ここでもネットワーク上で何らかの変化が起こった影響が出ていると思われる。

このように、片道 ping を用いると、トラフィックの変化の様子や、ネットワークに何かが起こったという事実を把握できる。

例 2 ntpdate を用いた時刻同期の精度評価

時刻同期が不完全な環境でも、ntpdate コマンドを用いて、時刻を同期できる。現状では時刻が完全に同期している環境は少ない。しかし、完全に時刻同期が実現している環境での同期精度と、ntpdate で時刻同期をした際の同期精度を比較すれば、ntpdate での同期精度を評価できる。

評価は、以下のやり方で行うことを想定している。

- ntpdate を、片道 ping のパケットを送信するごとに実行し、時刻同期をした直後に片道 ping のパケットを送信する。

例 3 異常の種類の検出

ある異常に関しては特有の変化があり、それを把握できていれば、異常の種類の検出に用いることも可能である。例えば、サーバ側の計算機に負荷がかかっているとき、片道 ping が見せる変化をあらかじめ把握していれば、サーバ側の計算機に直接触れることなく負荷がかかっていることを知ることができる。

3.3.3 動的トラフィック掲示板システムの概要

現段階では、ホスト間のトラフィック変化を調べることはできても、出力がテキストであるため、トラフィックの変化を把握しにい。また、ネットワーク全体を見渡すことはできない。

そこで、あるホストから見たネットワークの状態を、容易に把握できるような表示を行う、図 3.3のような動的トラフィック掲示板を考えた。

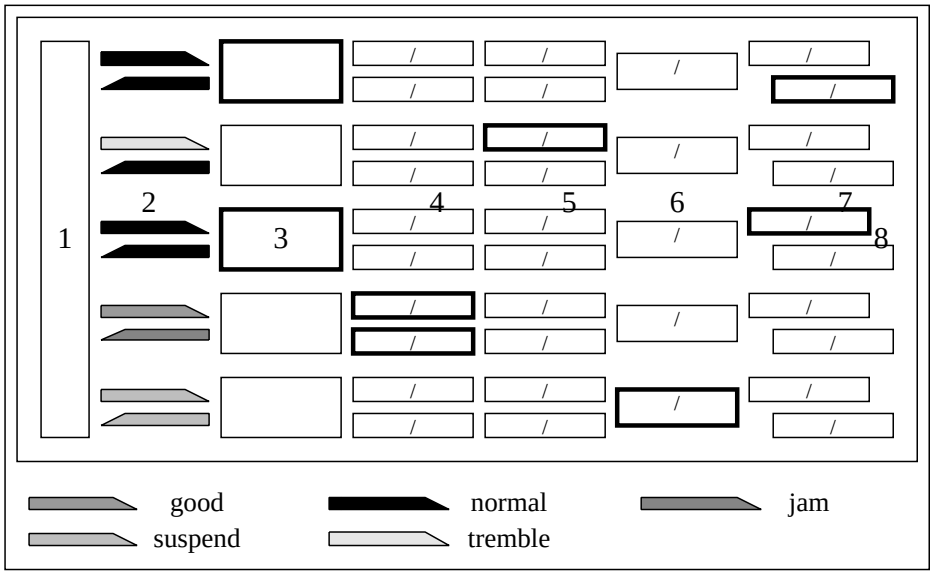

図 3.3: 掲示板の表示例

図 3.3のそれぞれの項目について説明する。

1. 片道 ping のパケットを送信するホスト名を示す。
2. 送信および受信経路の状態を示す。(詳細は後ほど述べる。)
3. 片道 ping のパケットを受信するホスト名およびホストの状態を示す。
4. 送信経路および受信経路に関する送受信時間の、最近の平均と異常がないときの平均を示す。
5. 送信経路および受信経路に関する送受信時間の、最近の分散と異常がないときの分散を示す。
6. 片道 ping のパケットが透過できた最近の割合と、異常がない場合の割合を示す。
7. 送信先ホストで経過した時間の、最近の平均と異常がないときの平均を示す。

8. 送信先ホストで経過した時間の、最近の分散と異常がないときの分散を示す。

1から3までは、ネットワークの状態を一目で見て分かるようにするためのもので、4以降で詳細な情報を表示する。

経路に関する情報を示す矢印は、状態に応じて色や太さが変化する。経路の状態には、不通、混雑、不安定、通常、良好がある。それぞれ、図3.3中では suspend, jam, tremble, normal, good と記されている。それぞれの意味を以下に示す。

通常 ネットワークは正常。

混雑 パケットの透過率が通常と比べ低い、または送受信にかかる時間が通常と比べ長い。

不通 パケットの透過率がきわめて低い。

良好 パケットの透過率が通常と比べ高い、または送受信にかかる時間が通常と比べ短い。

不安定 送受信にかかる時間の揺らぎが通常と比べ大きい。

図3.3中の枠の形は、ネットワークやホストの状態によって変化する。例えば、図3.3中の上から二番目のホストで、パケットの送信時間が揺らいでいることを示す矢印が出ている。このとき、図3.3中の5で、上から二番目のホストで、パケット送信時間の分散を表す枠が変化し、送信時間の揺らぎに関する詳細情報を強調する。

システムを起動すると、まず、指定されたファイルに記述されたホストすべてに、片道 ping のパケットを送出する準備をする。そして、何秒毎かにいずれかひとつのホストに pingpong を使用してパケットを送出する。例えば、一秒毎にホスト A, B, C, D, E, F, G, A,... と各ホストにパケットを送出する。これにより、ネットワークにあまり高い負荷をかけずに、トラフィックの連続的な変化を観察できる。

次に、ネットワークの状態が普段と変わらないときのトラフィック情報と、pingpong の出力を比較する。

そして、この情報に、先程の pingpong の出力から得られるデータを追加する。

情報の内容は、3.3.2 項の例 1 に示されたようなもので、トラフィックの一日、一週間、一年間および全体としての変動傾向に関するデータである。この情報をもとに、掲示板システムは、ネットワークの状態を比較評価することができる。この情報は、システムを起動すれば自動的に作成・更新されていく。

比較し終わると、調査対象の経路やホストの状態、詳細な情報を、上で説明した掲示板の表示部に渡す。

以上が動的トラフィック掲示板及びそのシステムの概要である。

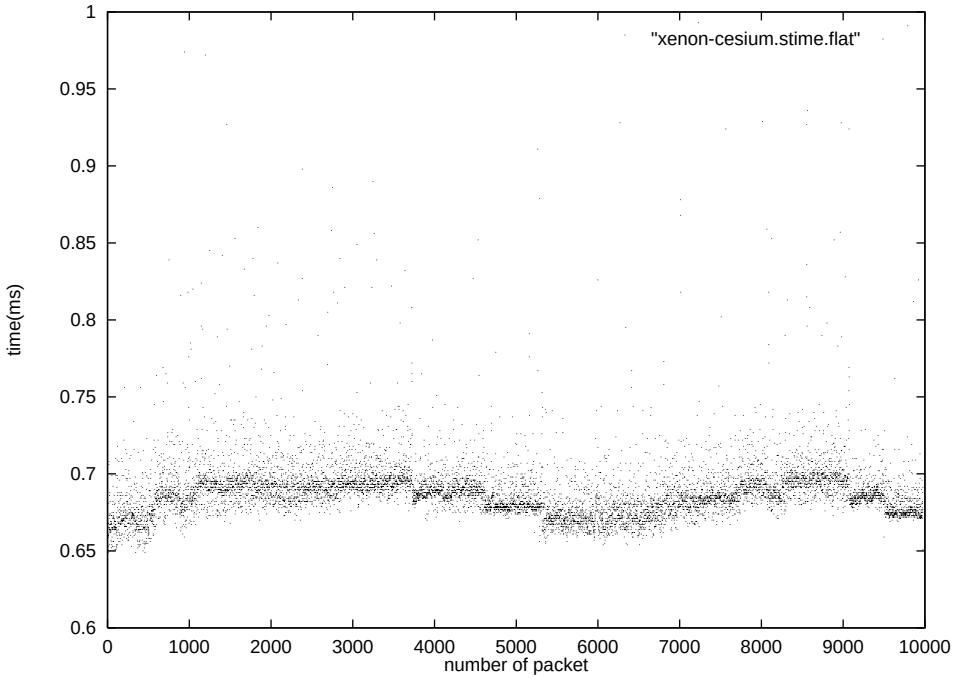

図 3.4: xenon から cesium へ片道 ping を行った際の送信時間のグラフ

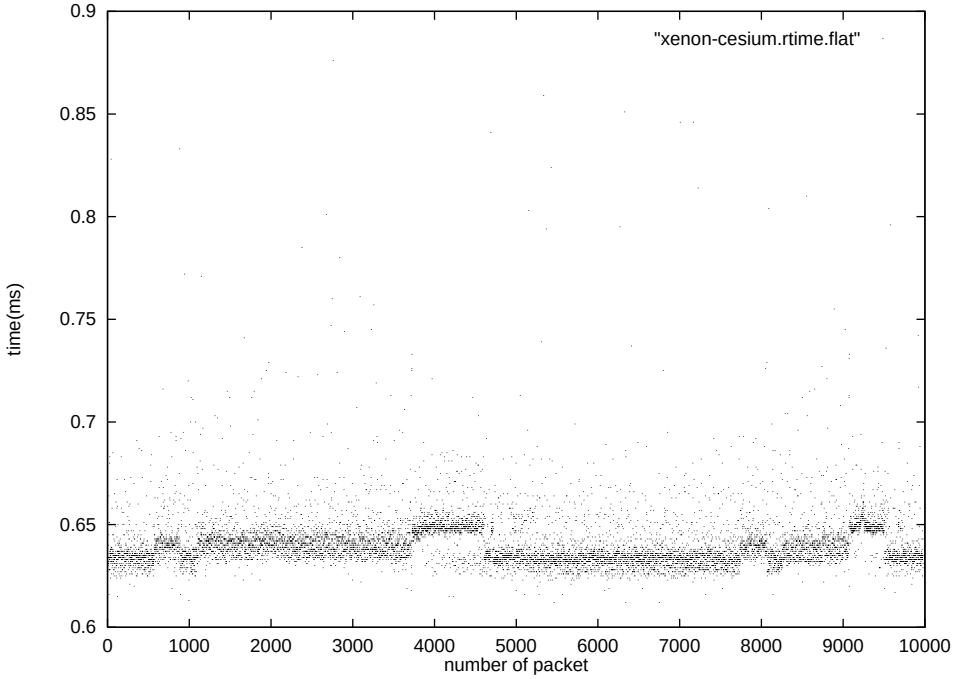

図 3.5: xenon から cesium へ片道 ping を行った際の受信時間のグラフ

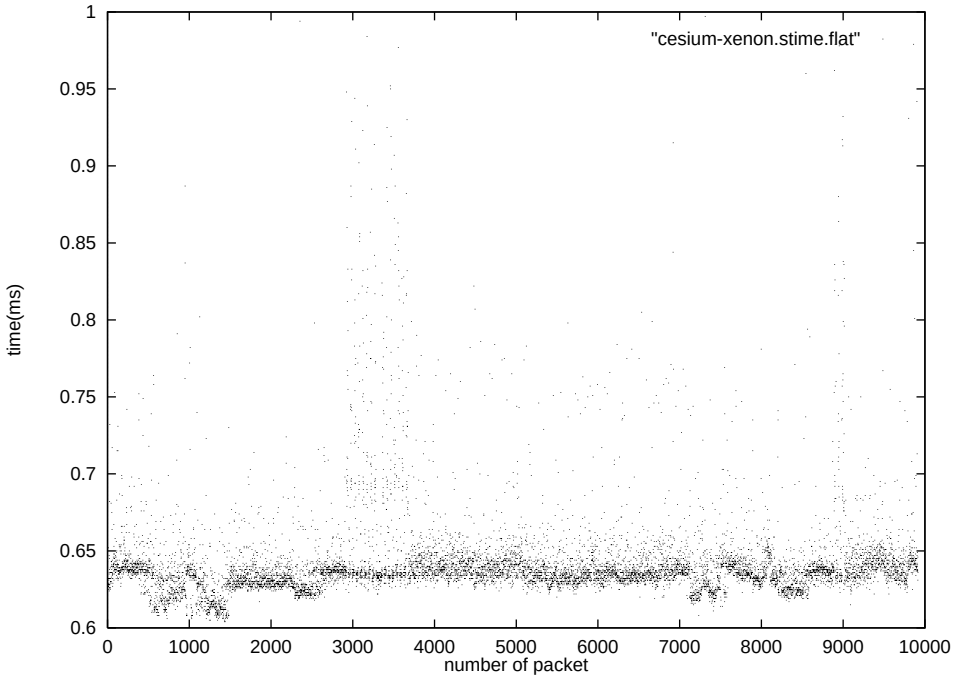

図 3.6: cesium から xenon へ片道 ping を行った際の送信時間のグラフ

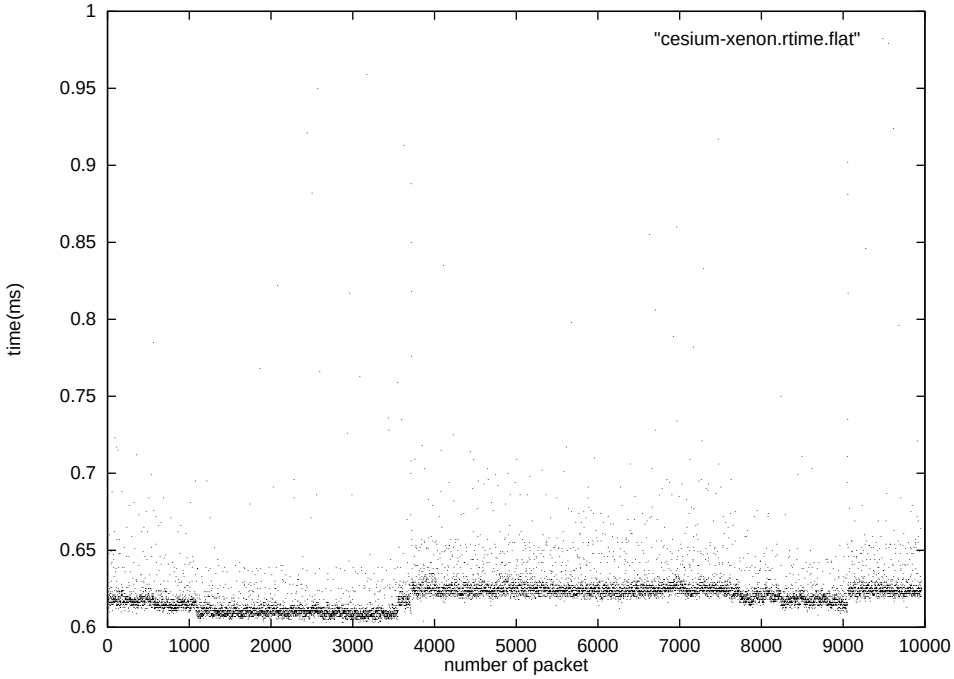

図 3.7: cesium から xenon へ片道 ping を行った際の受信時間のグラフ

第 4 章

考察

前章に示した片道 ping の図 3.4から図 3.7を見れば、ネットワークのトラフィックに関する情報が把握できる。一方、課題も多く残された。そこで、今後の課題を整理して、片道 ping の有効活用に役立てることにする。

4.1 トラフィック計測に関する課題

まず、片道 ping がネットワーク計測ツールとしての有用性を確かめるために、ネットワークの状態に応じたトラフィックの変化の傾向を評価する作業が必要となる。

また、動的トラフィック掲示板システムのように、分かりやすく動的にトラフィックを表示する方法を検討していく必要もある。さらに、このシステムをさまざまな計算機で起動し、全システムの情報を管理するホストを設置すれば、ネットワーク全体のトラフィックを計測できる。動的トラフィック掲示板システムは、システムの存在するホストと他のホストとの間のトラフィックを計測するという、一対多の計測を行うものである。ネットワーク全体のトラフィックが明らかとなれば、任意の 2 ホスト間のトラフィックを平行して動的に測定することが可能となる。

しかし、片道 ping を用いた動的トラフィック掲示板は、まだ設計の段階であり詳細が固まっていない。例えば、掲示板の表示方法について、見やすい配置を検討しなければならない。また、ネットワークの状態を示す記号や印を出す基準を検討しなければならない。そのためにも、ネットワークの傾向を把握するためのデータの取り方を検討しつつ、実装に入りたい。

また、時刻同期が十分に取れている環境のうち、使用できるものが国立天文台の同一セグメントにある cesium と xenon に限られていたため、片道 ping の評価が十分に行えなかった。より多くの、時刻同期が十分取れている計算機に片道 ping のサーバを置けば、多様なトラフィック計測が可能となる。例えば、遠くの計算機とパケットを送受信し合い、トラフィックの不安定を近くで計ったときと比較したり、送信と受信とで異なる経路を使用している計算機間でパケットを送受信し合い、各経路のトラフィックの差を比較したりできる。

いずれにせよ、トラフィックの計測ということに関する当面の課題は、多様な環境と条件で、大量のデータを取り、検討していくという作業を行うということである。

4.2　片道 ping に関する課題

片道 ping では、時刻に関して非常に細かい点まで考慮しなければならない。例えば、送受信をする時間と時刻を得る時間との間に誤差が生じているため、何らかの方法で誤差を直してやらなければならない。現状では、実装を容易に進めるために誤差の修正を省略している。これを直すには、UDP などで相手ホストと接続せずに送受信を行うのためのシステムコール `sendto` と `recvfrom` を改造し、システムコールを行うと時刻を記録するようにすればよい。

評価の段階で明らかとなった、パケットの送信と受信の時間が一致しない問題もある。これについては、まず送信と受信のシステムコールを改造したのち、問題が解決しているかどうか確かめなければならない。もし解決すれば、cesium と xenon の差を完全に解消し、計測し直してみることが必要である。この方法で解決しないのであれば、片道 ping のソースコードの見直しをはかる必要がある。

以上のように、片道 ping に関する課題は、現在実装の容易さのため省略している部分を厳密に調べ、実装に組み込む作業を行うということである。

4.3　まとめ

ネットワークをリアルタイムで正確に把握したいと言う思い付きから、片道 ping という物を考えだした。これを利用すれば、ネットワークの全体の様子を動的に把握する事が可能となる。しかしながらその実現はまだ始まったばかりであり、色々と課題が残された。最後に、今後より正確にネットワーク全体を見渡せるツールの製作するにあたり、それに向けての課題を列挙して締めくくる。

- さまざまな条件下での片道 ping の使用、及びその結果の評価を行う。
- 片道 ping により何が分かり、また何を把握する事は出来ないのかを明らかにする。
- 動的トラフィック掲示版の作成を行う。
- より正確に計時するために、片道 ping をより厳密に再評価し、実装に盛りこむ。