簡易GPSシステムと時刻同期プロトコルを用いた正確なネットワーク遅延測定手法

岩本 哲也[1,a)]　倉田 陽介[1,b)]　阿部 博[2,c)]

概要：高速大容量通信システムにおける品質指標の 1 つとして，ネットワーク遅延を計測することは重要である．これまでに Round-Trip Time を用いた単純な測定手法や専用の遅延測定プロトコルを用いた測定手法が提案されているが，これらは正確な片方向遅延を測定するという観点で技術面やコスト面で課題を抱えている．本研究では安価な簡易 GPS モジュールを用いた測定システムと，NTP プロトコルを利用したネットワークの遅延測定手法を提案する．高価な専用機器を用いずに汎用ネットワークにおける片方向遅延を実測することで，ネットワーク測定手法としての有効性を確認・評価する．

An Accurate Network Delay Measurement Method Using Low-Cost GPS System and Time Synchronization Protocol

Abstract: Measuring network delay is one of the important quality indicators in high-speed and large-capacity communication systems. The existing delay measurement methods using simple round-trip time or using delay measurement protocols have some technical and cost issues in terms of accurate one-way delay measurement. In this paper, we propose a measurement system using a simple GPS module and a method for measuring network delay using NTP protocol. We also show the capability of our system and measurement method to measure one-way delay in commercial networks with no need for expensive instruments.

1. 背景

計算機資源やネットワーク基盤の高速・大容量化はクラウドコンピューティングに大きな発展をもたらした．それに伴いマイクロサービス・アーキテクチャも急速に実用化が進んだ．また同時にインターネットに接続されるセンサー機器や駆動装置，家電等のモノ，いわゆる IoT 機器も増加した．これらインターネットに接続される機器から得られる情報は飛躍的に増加しており，このように収集されたビッグデータを機械学習や深層学習で分析してビジネスに活用しようとする動きも加速している．以上のような流れによって 5G に代表される今後のネットワーク基盤では高速・大容量化だけでなく低遅延通信や多接続がますます進んでいくと考えられる．

[1] セイコーソリューションズ（株）
Seiko Solutions Inc.
[2] トヨタ自動車株式会社
Toyota Motor Corporation
[a)] tetsuya.iwamoto@seiko-sol.co.jp
[b)] yosuke.kurata@seiko-sol.co.jp
[c)] hiroshi_abe_af@mail.toyota.co.jp

ネットワーク基盤の発展や利用方法の変遷に伴って，IT サービスの品質管理はますますと複雑化し，また管理の高精度化に対する要求も高まってきている．特にクラウドコンピューティングやエッジコンピューティングといった分散処理においては各部処理の連携をネットワークを介して行うため，End-to-End のネットワーク遅延量を出来るだけ正確／高精度に把握することが重要であり，また実際にその要望も高まってきている．

一方，このような高精度な遅延測定要望に対しては専用の測定装置を利用すれば十分な結果を得られる．しかしながら，専用装置は一般的には数百万円から数千万円という価格の非常に高価なものである．また，例えば自身で管理しているネットワークと管理外のネットワークの 2 地点間の遅延測定をする場合の調整コストや，たとえ自身の管理ネットワーク内であっても複数箇所を同時に測定したい場合などには，高価な測定装置を複数箇所に配置するためのコストなどを考慮すると容易に測定環境を構築できるとは言い難い．したがって，多地点で可能な限り正確なネットワーク遅延を把握するための低コストなソリューションが

必要であると考えられる．

ここ数年の間に Raspberry Pi [*1]に代表されるようなシングルボード・コンピュータが急速に普及した．それに伴い安価で小型の簡易 GPS モジュールも入手しやすい状況となってきている．GPS を利用すれば 1 マイクロ秒未満の精度の比較的正確な標準時刻が容易に得られる．簡易 GPS モジュールさえあれば一般的なパーソナル・コンピュータ (PC) であってもシングルボード・コンピュータと同様に手軽に高精度な時刻同期を実現できる．特に，シングルボード・コンピュータと GPS モジュールの組合せであれば数千円程度で時刻同期システムを準備できる．非常に安価にシステムを構築でき多数の配置に対応できる．

本研究では一般的な PC と簡易 GPS モジュールを用いて標準時刻に同期した測定システムを構築し，一般に普及している NTP プロトコルを利用して片方向遅延を測定する手法を提案する．本手法では，遅延測定の端点の一方にだけ測定システムを用意すれば，もう一方として（応答可能な）任意の NTP サーバを選択でき，すぐに遠方の NTP サーバまでの正確な上り・下りの両方の片方向遅延を測定することが可能となる．

本稿では，第 2 節にて遅延測定や時刻同期に関連する研究について言及する．次に，第 3 節にて遅延測定の基本的な原理について復習する．そして，本提案の核となる簡易 GPS モジュールと PC を用いた測定システムの構築方法，そして NTP プロトコルを利用した遅延測定の方法について述べる．第 4 節では実際に構築した遅延測定システムを用いていくつかの代表的な NTP サーバを対象に片方向遅延を実測した結果を示し，ネットワーク測定手法としての有効性の確認を行い考察を行った．第 5 節では測定データに対して変化点検出を適用した結果について示した．

2. 関連研究

コンピュータネットワークにおいて End-to-End のネットワーク遅延測定には古くから，Round-Trip delay Time/Round-Trip Time (RTT) の測定が利用されてきた．古いものでは Mills による RFC889 [12] があり，その後も様々なネットワークにおいて RTT による測定が実施されてきた [1], [15], [16]．また，このような RTT をネットワークの Quality of Service (QoS) 測定に利用する研究も活発に行われてきた [9], [13].

RTT 測定は Internet Control Message Protocol (ICMP) の Echo Request/Echo Reply メッセージ（いわゆる ping）を利用すると実施でき，ほとんどのネットワーク装置や OS で ping がサポートされていることから，多地点での遅延測定としてはこの測定法がもっとも利用しやすい．しかし，もっとも単純な RTT 測定はメッセージが往復するのにかかった時間の半分を片方向遅延とするため，測定パケットの送信経路と受信経路が異なる環境においては正確なネットワーク遅延を測定するのは困難である．

これに対して，RTT 測定をもとにして上り下りのそれぞれの片方向遅延を推定する研究もおこなわれてきた．例えば [2], [6] では連続した RTT 測定の結果から片方向遅延の取りうる範囲を絞る方法を提案している．[3], [19] では 3 地点で互いの RTT 結果を交換することでさらに片方向遅延の取りうる範囲を絞る方法を提案している．しかしながら，いずれの方法においても片方向遅延を推定していることに変わりはなく，範囲の絞り込みで十分な結果精度が得られない可能性がある．

遅延測定については規約／標準も存在する．One-way Active Measurement Protocol (OWAMP) [17] あるいは Two-Way Active Measurement Protocol (TWAMP) [4] といった遅延測定に特化したプロトコルである．これらは，測定用メッセージの送信時や返信時に自身のタイムスタンプを当該メッセージに書き込むので，片方向遅延を容易に得られる．ただし，正確な測定結果を得るためには測定対象の装置間でタイムスタンプ用クロックを時刻同期させなければならない．さらに，OWAMP/TWAMP は専門性の高いプロトコルであり安価なネットワーク機器にはあまり普及していないのが実情である．

これらのクロックの時刻同期には Network Time Protocol (NTP) [11] を利用できるが，NTP による時刻同期自体がネットワーク遅延の揺らぎや上り下りの遅延量の違いの影響を受けるため，NTP による時刻同期では十分な測定精度が得られない場合がある．例えば物理ネットワーク的に遠方にある NTP サーバに NTP クライアントが時刻同期するような場合は必要なクロック同期精度が NTP クライアントで得られない可能性がある．

NTP 以上のクロック時刻同期精度を得る手法として Precision Time Protocol (PTP) [5] がある．NTP と同様にクライアントサーバモデルで時刻メッセージを交換することで時刻同期をする．PTP は標準時刻に同期したグランドマスタークロック（サーバ）の時刻をメッセージにハードウェアで打刻することで一般的には NTP とは桁違いの，1 マイクロ秒未満の時刻同期精度を実現できる．しかしながら，スレーブ（クライアント）までの中間ネットワーク装置（ルータや L2 スイッチ）がすべて PTP に対応していないと PTP 本来の精度は得られない．もちろん，低レイテンシスイッチなどを利用すれば必要な精度が出せる可能性はあるが，いずれにしてもコスト面でのパフォーマスが良いとは言えない．

時刻同期としては他に高精度（高正確度）の時刻源を直接利用する手法がある．[7], [8] では外部の安定したセシウム原子時計駆動の PC を用いてネットワーク帯域の測定を行っている．当該研究から 15 年以上の時間が経過してい

[*1] https://www.raspberrypi.org/

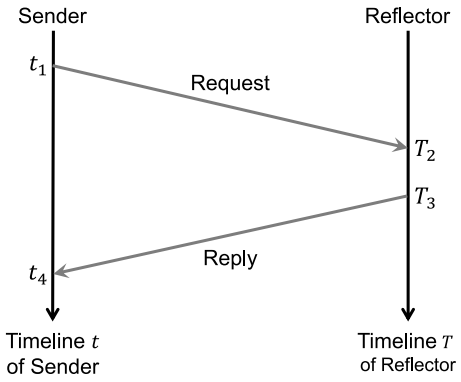

図 1 ネットワーク遅延測定の典型モデル
Fig. 1 Typical network delay measurement model

るが，現在でもこのレベルの測定インフラを整備するのは実運用レベルでは容易ではない．

一方で Raspberry Pi の普及により簡易な GPS 受信機が安価でしかも簡単に入手できるようになっている．簡易 GPS 受信機を用いて，ナノ秒からマイクロ秒レベルの正確な時刻情報を収集，利用する研究は比較的安価に手軽に行われており [10]，現在では GPS 時刻同期はこのような測定に向いていると言える．

3. GPS システムと NTP を用いた遅延測定

3.1 遅延測定の基本原理

典型的なネットワーク遅延測定モデルを、図 1 において 2 台のコンピュータシステム（Sender と Reflector とする）を用いて示す．

Sender と Reflector は、自身の時刻源であるローカルクロックを保有し，t_1, T_2, T_3, t_4 は次の通りそれぞれのクロックにおける時刻（タイムスタンプ）である．

$\boldsymbol{t_1}$ Sender のリクエスト送信時刻
$\boldsymbol{T_2}$ Reflector のリクエスト受信時刻
$\boldsymbol{T_3}$ Reflector のリプライ送信時刻
$\boldsymbol{t_4}$ Sender のリプライ受信時刻

RTT を用いた測定手法では，片方向遅延を

$$D_1 = \frac{t_4 - t_1}{2}$$

のように計算する．この測定手法は Sender のタイムスタンプだけを用いる．その上で，上り（リクエスト）と下り（リプライ）の遅延時間が等しく，かつ，Reflector での処理遅延が無視できるほど小さいと仮定している．当然ながら上りと下りの遅延差や Reflector での処理遅延があると正確な片方向遅延を測定することはできない．

次に，Sender と Reflector の送受信タイムスタンプを用いる手法では，

$$D_2 = \frac{(t_4 - t_1) - (T_3 - T_4)}{2}$$

のように Reflector での処理遅延を考慮した上で片方向遅延を求める．この手法の利点は，Sender と Reflector のクロックが必ずしも合っている（同期している）必要がないことである．しかし，D_1 と同様に上りと下りの遅延差の考慮はできないため，遅延差がある場合には正確な片方向遅延を測定することはできない．

したがって, 片方向遅延を正確に測定するためには Sender と Reflector のクロックを合わせておく必要がある．両者のクロックが同期していれば，上り遅延量 Δ_{forward} と下り遅延量 Δ_{reverse} は次の通り直接求めることが可能となる．

$$\Delta_{\text{forward}} = T_2 - t_1$$
$$\Delta_{\text{reverse}} = t_4 - T_3$$

当然ながら，Sender と Reflector のクロック同期精度がそのまま片方向遅延測定の結果精度になるため，できるだけ両者のクロックを高精度に同期することが望ましい．

一方，時刻同期に NTP を利用する場合は，

(1) そもそも同期する NTP サーバが協定世界時 (UTC) に正確に同期できているか．
(2) NTP サーバとクライアント間の上り・下りのネットワーク遅延差が無視できるほど小さいか．

を十分に考慮する必要がある．特に (2) について，NTP クライアントはサーバとの時差計算に際して D_2 の方法でサーバとクライアント間の片方向遅延を計算するため，上り・下りの遅延差がそのまま時差に反映される．もし遅延差が大きければ，結果的にサーバとクライアントの時刻は遅延差の分だけズレた状態となり，測定結果が信頼できなくなる．

これを避けるためには，例えば Sender，Reflector が配置される各々の LAN 内に Stratum 1 の NTP サーバを設置する機器構成が考えられる．このような構成では各々の LAN 内の NTP サーバと Sender/Reflector 間のネットワーク遅延差が小さい可能性が高い．すなわち両クロックは高精度に時刻同期し十分な測定精度を出せる可能性は高い．

3.2 GPS 同期遅延測定システム

Sender あるいは Reflector の時刻同期に GPS を利用したシステムを構築する方法を述べる．図 2 にこの GPS 同期遅延測定システムを実現するためのブロック図を示す．

GPS モジュールはビルの屋上のような全天が見渡せる環境において，通常 100ns 以下の誤差で標準時刻に同期する．GPS モジュールとシリアル (RS-232C) 回線で接続されたパーソナルコンピュータは，標準時刻が含まれた測位情報を NMEA0183 標準フォーマット [14] のテキストデータで受信し，秒の正確な切り替わりタイミングを示す Pulse Per Second(PPS) 信号を信号線 (DCD) から受信する．NTP サーバアプリケーション (ntpd) を使用すると，シリアルドライバ (TTY) および PPS ドライバから標準時刻と秒の切り替わりタイミング情報を得て，システムクロックを高精度に補正することが可能となる．

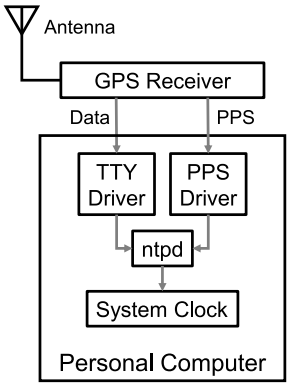

図 2 GPS 同期遅延測定システム
Fig. 2 Delay measurement system with GPS

次に，システムクロックの GPS 時刻同期に関する設定手順を述べる．PPS 信号を利用するためにはカーネルが PPSAPI をサポートしている必要がある．以下は Linux カーネルにおける PPSAPI 有効化および DCD から PPS 信号入力タイミングを取得するためのカーネルコンフィグ例である．

Linux カーネルコンフィグ
```
CONFIG_PPS=y
CONFIG_PPS_CLIENT_LDISC=m
```

また，PPSAPI および標準時刻を利用したシステムクロックの高精度補正には，ntpd の Refernece Clock Driver を用いる．PPSAPI および NMEA データで時刻補正が可能な NMEA Reference Clock Driver を用いるための ntp コンフィグ例を以下に示す．

ntp コンフィグ
```
server 127.127.20.0 minpoll 4 maxpoll 4 mode 81 prefer
fudge 127.127.20.0 flag1 1 flag2 1 flag3 1
```

以上により，GPS モジュールとパーソナル・コンピュータを用いて，片方向遅延測定に必要な時刻同期精度を有するシステムを容易に構築できる．

ここでは，シリアル (RS-232C) 回線を使用したが，Raspberry Pi などの小型シングルボード・コンピュータの GPIO ピンなどを利用しても適切に設定をすれば同様の動作をさせることが可能である．低価格のシングルボード・コンピュータと小型 GPS モジュールを利用すれば，容易かつ安価に遅延測定システムを構築可能である．

3.3 NTP を用いたネットワーク遅延測定手法

本研究では図 3 に示すように，3.2 に従い構築した遅延測定システムを Sender として用い，すでに存在するネットワーク上の NTP サーバを Reflector として利用する．

NTP プロトコルは，3.1 で触れた t_1, T_2, T_3, t_4 の 4 つのタイムスタンプを送受信するプロトコルであるから，測定

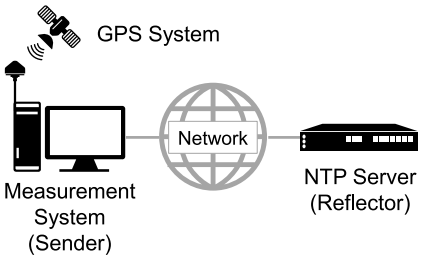

図 3 ネットワーク遅延測定の典型例
Fig. 3 Typical measurement targets

表 1 遅延測定システム
Table 1 Our instrument

GPS	
Antenna	AU-217 (FURUNO)
Module	GPSTCXO (Jackson Labs. Technologies, Inc)
PC	
Model	HP Pro 6300
CPU	Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz
RAM	4GB
OS	Ubuntu 20.04
NTP	ntp-4.2.8p14

専用の新規ソフトウェアを使わずともそのまま測定に利用できる。さらに Reflector となる NTP サーバとして，原子時計や GPS から直接時刻を取得している Stratum 1 の NTP サーバを選べばすぐに高い精度の遅延測定を行うことができる．

各種タイムスタンプの収集には，疑似 NTP クライアントとしてシステムクロック設定を実施しない ntpdate コマンドを用いる．なお，ntpdate コマンドは t_1, T_2, T_3, t_4 タイムスタンプを出力することができないため，reply を受信する receive() 関数に t_1, T_2, T_3, t_4 の値を標準出力できるよう修正をおこなった．

4. 遅延測定結果と考察

本節では，第 3 節で述べた遅延測定システムを構築し実ネットワークの測定をした結果について述べる．表 1 に今回構築した遅延測定システムの部品構成を示す．

4.1 固定回線ネットワーク遅延測定

図 4 に固定回線ネットワーク遅延測定の構成イメージを示す．本遅延測定システムを NTT フレッツ光[*2]（PPPoE 接続）回線経由でインターネットに接続し，日本標準時である原子時計に同期した Stratum 1 公開 NTP サーバ (NICT)（以降 NICT と略する），および，GPS により補正される原子時計と同期するクラウド上に構築した Stratum 1 NTP サーバ (AWS)（以降 AWS と略する）とのネットワーク遅延を 1 分周期で測定した．NICT，AWS ともに

[*2] https://flets.com/

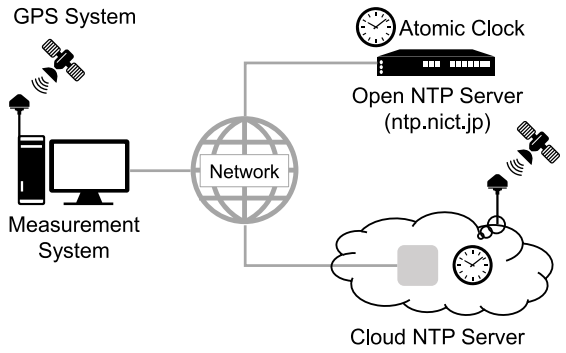

図 4 固定回線接続構成
Fig. 4 A wired network topology

表 2 公開 NTP サーバ (NICT) 測定結果（要約）
Table 2 Delays for an open NTP server (summary)

	Max.(s)	Min.(s)	Ave.(s)	Range(s)
Δ_{forward}	0.00932	0.00281	0.00464	0.00651
Δ_{reverse}	0.04853	0.00156	0.00489	0.04697
Δ_{forward} (0:00–19:00)	0.00932	0.00281	0.00467	0.00651
Δ_{reverse} (0:00–19:00)	0.02492	0.00156	0.00200	0.02336

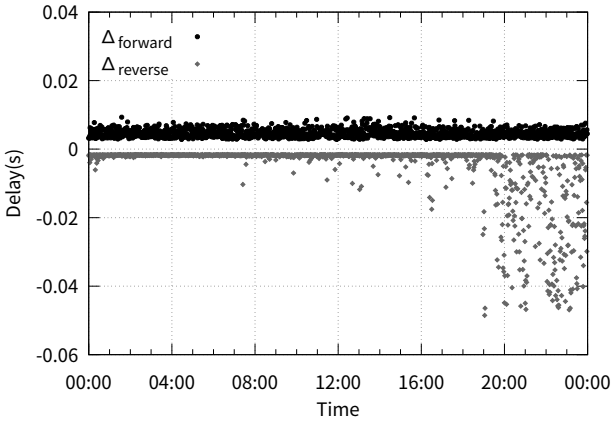

図 5 公開 NTP サーバ (NICT) 測定結果（散布図）
Fig. 5 Delays for an open NTP server (scattergram)

IPv4 UDP による測定であり，サーバは DNS による名前解決ではなく，IP アドレスを直接指定した．

NICT の測定結果を表 2，図 5 に示す．結果は午前 0 時からの 24 時間のデータである．また，図 5 においては測定データが重なりあって見づらくなることを避けるため，上り遅延量はそのままに，下り遅延量を正負を逆にして（$-\Delta_{\text{revrese}}$ の）作図をおこなった．特に断らない限り，以降の散布図でも同様に下り遅延量の正負を逆転して作図する．

次に，AWS の測定結果を表 3，図 6 に示す．NICT の場合と同様に結果は午前 0 時からの 24 時間のデータである．測定結果を確認すると，双方とも 19 時過ぎから下り遅延が大幅にばらつく傾向があることが一目で分かる．これは光回線の下りトラフィックが夕方から夜中までに集中することを示していると考えられる．

表 3 クラウド NTP サーバ (AWS) 測定結果（要約）
Table 3 Delays for a cloud NTP server (summary)

	Max.(s)	Min.(s)	Ave.(s)	Range(s)
Δ_{forward}	0.07971	0.06886	0.07317	0.01086
Δ_{reverse}	0.12621	0.07329	0.08076	0.05292
Δ_{forward} (0:00–19:00)	0.07971	0.06886	0.07328	0.01086
Δ_{reverse} (0:00–19:00)	0.09750	0.07329	0.07798	0.02422

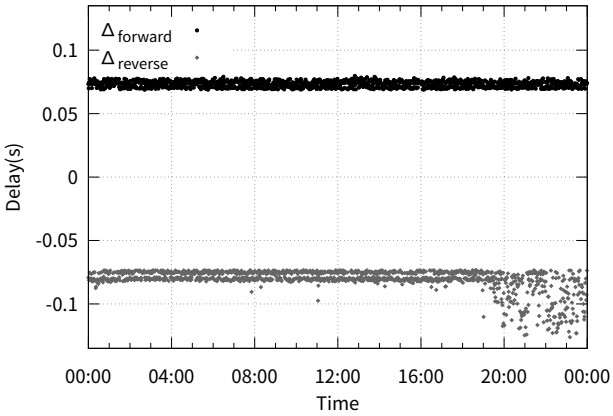

図 6 クラウド NTP サーバ (AWS) 測定結果（散布図）
Fig. 6 Delays for a cloud NTP server (scattergram)

下り遅延量のばらつきが少ない 19 時までの平均遅延量を確認すると，NICT は上りが 4.7ms，下りが 2.0ms であり上りが下りに対して 3ms 弱大きいことが分かる．また，下りは散発的に比較的大きな遅延量を観測しているが，それ以外の部分では安定しており，同時間帯の上り遅延量のばらつきの方が大きいようにグラフから見て取れる．

同様に 19 時までの AWS の平均遅延量を確認すると，上りが 73.3ms，下りが 78.0ms であった．こちらは下りが上りに対して 5ms 弱大きいと分かる．遅延量の絶対量が NICT に対して大きいために比較的に遅延差を小さく感じるが，実際には NICT の上り下りの遅延差よりも大きい．また，グラフからは上り，下りともに遅延量が 2 層に分かれている．これは通信経路が時間帯によって異なることを示していると考えられるが，これ以上の調査はタイムスタンプだけでは不可能である．

4.2 モバイル回線ネットワーク遅延測定

図 7 にモバイル回線ネットワーク遅延測定の構成イメージを示す．

本遅延測定システムを IIJ IoT サービス[*3]のモバイル回線経由でインターネットに接続し，モバイル閉域網でサービス提供される NTP サーバとのネットワーク遅延を 1 分周期で測定した結果を表 4，図 8 に示す．結果は午前 0 時からの 24 時間のデータである．なお，モバイル回線も IPv4

[*3] https://www.iij.ad.jp/biz/iot/

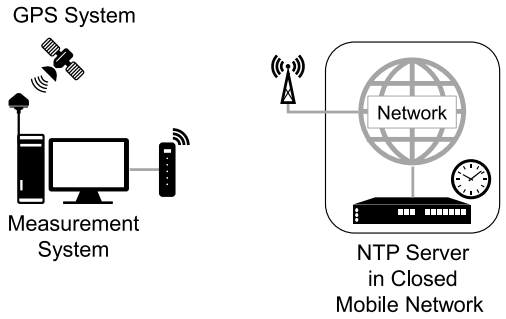

図 7 モバイル回線接続構成
Fig. 7 A closed mobile network topology

表 4 モバイル閉域網 NTP サーバ測定結果（1 分周期 要約）
Table 4 Delays for a NTP server in closed mobile network (every 1 min. summary)

	Max.(s)	Min.(s)	Ave.(s)	Range(s)
Δ_{forward}	0.24280	0.06196	0.10124	0.18084
Δ_{reverse}	0.12506	0.02324	0.02994	0.10183

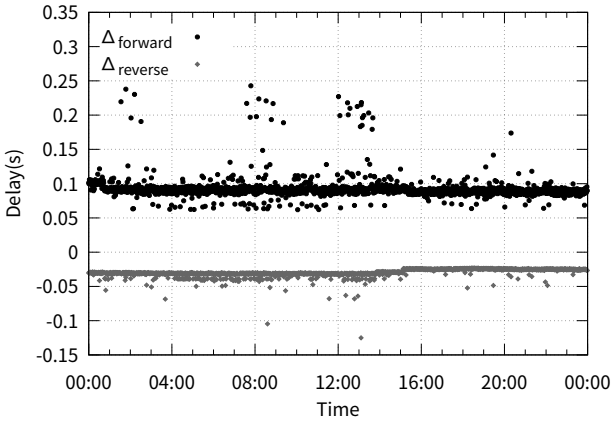

図 8 モバイル閉域網 NTP サーバ測定結果（1 分周期 散布図）
Fig. 8 Delays for a NTP server in closed mobile network (every 1 min. scattergram)

UDP での測定であり、サーバは IP アドレスを直接指定した。測定結果の平均遅延量は上り 101.2ms，下り 29.9ms であり，上りと下りの遅延差が約 71ms であった．また，上りにおいて散発的に 200ms 程度の比較的大きな遅延が発生していることが確認できる．

続いて同構成において測定周期のみを，30 分ごとに 30 秒間，1 秒おきに連続的に遅延測定をするように変更する．例えば 13 時ちょうどから 30 秒間遅延測定を実施すれば，次は 13 時半から 30 秒間遅延測定を実施するという具合である．この結果を表 5，図 9 に示す．表 5 おいて “Δ_{forward}: 1st delay” は 30 秒測定の初回の Δ_{forward} 値を対象とし，“Δ_{forward}: otherwise” は 30 秒測定の 2 回目以降の Δ_{forward} 値を対象としている．

図 9 の破線で囲んだ部分は，1 分周期の同期では散発的にしか見られなかった大きな遅延量が 30 秒測定では毎回発生していることを示している．実際，30 秒測定の初回の

表 5 モバイル閉域網 NTP サーバ測定結果（30 分周期 要約）
Table 5 Delays for a NTP server in closed mobile network (every 30 min. summary)

	Δ_{forward}: 1st delay	Δ_{forward}: otherwise
Max.(s)	0.25300	0.16264
Min.(s)	0.16055	0.02020
Ave.(s)	0.18978	0.03702
Range(s)	0.09246	0.14244

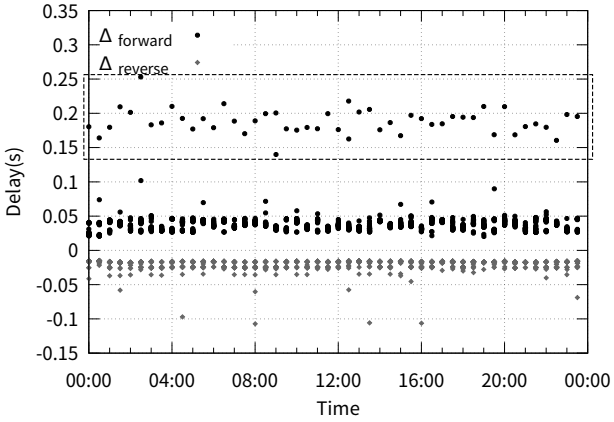

図 9 モバイル閉域網 NTP サーバ測定結果（30 分周期）
Fig. 9 Delays for a NTP server in closed mobile network (every 30 min.)

上りは平均 189.8ms と非常に大きな遅延量であるが，2 回目以降は 37.0ms 程度と大幅に小さくなることがわかる．

これは，モバイル回線において無通信状態が継続した際に自動的に回線が切断され，再接続時の通信経路確立のオーバーヘッドが影響していると考えられる．1 分周期の場合でも同様の回線切断が起こっていたものと思われるが，データからは回線切断は頻繁ではなかったと推察される．

このように，測定インターバルによって得られる結果が大きく変わることがあるため，測定対象のネットワークの特徴を把握した上で，目的に合った測定をおこなう必要がある．

5. 変化点検出

実運用環境においては，ネットワーク異常などをリアルタイムに検出し適切な対処を迅速におこなうことで安定したサービス提供をおこなっている．本小節では，本手法で得られたデータからリアルタイムに変化点の可視化が可能であるか検証する．

図 10 は公開 NTP サーバ (NICT) の測定結果（図 5）に，オンライン処理可能な変化点検出アルゴリズムである Change Finder [18] を用いて算出した変化点スコアを重ね合わせたグラフである．なお，変化点スコアは Python Change Finder ライブラリ[*4]のパラメータを r=0.01, or-

[*4] http://argmax.jp/index.php?changefinder

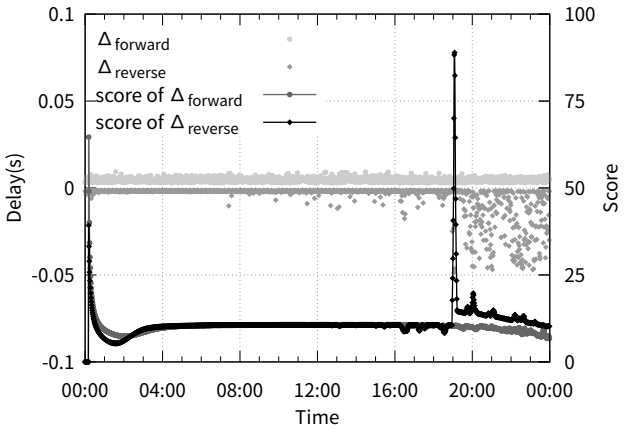

図 10 公開 NTP サーバ測定結果/変化点スコアスコア
Fig. 10 Open NTP server delays vs change-point score

der=1, smooth=7 と設定して算出した．

目視でも大きな変化がなかった上りの変化点スコアは，測定開始直後に表れるピークを除き，24 時間を通して小さな値を示している．一方，夕方以降に大きな変動があった下りの変化点スコアは，19 時頃に最大値をとり，遅延量が大きく変動しはじめる時刻と一致している．

このように，測定値そのものではなく変化点検出アルゴリズムのスコアを追うことで，瞬間的に発生する測定誤差などの影響を受けず効率的な監視が可能となると考えられる．また，毎日あるいは同曜日の同時間帯の遅延量の記録の変化スコアを追うことで “いつもと違う” 状況をいち早く検出することもできるようになるだろう．

6. まとめと今後の課題

本研究では，簡易 GPS モジュールを用いて標準時刻に同期した測定システムを構築し，NTP を用いたネットワーク遅延測定手法を提案した．本測定システムは汎用機器のみを用いて安価に構築することができた．また，本システムによる実ネットワークでの実験結果からネットワーク測定手法としての有効性を確認した．

今回の研究では測定システムの対向は既存の公開 NTP サーバであったが，測定システムに NTP サーバを導入すれば本測定システム同士で容易に遅延測定をすることができる．複数構築すれば任意 2 点間を同時に測定することも可能となる．また，今回は IPv4 ネットワークでのみ測定を行ったが IPv6 で同様の測定をすると異なる結果になる可能性がある．

測定システムについて，Linux カーネルの PPS kernel consumer support 機能を用いて PPS 信号入力処理ジッタの低減や，汎用 NIC の H/W タイムスタンプ機能を用いたタイムスタンプの打刻タイミングの揺らぎの排除などの改良を行えば，さらなる高精度な遅延測定が可能となると考えられる．

ただし，GPS の受信状態は GPS モジュールやアンテナの設置環境に大きく影響されることを注意しておく．これに対しては，例えば 3.1 で触れたように LAN 内に Stratum 1 NTP サーバがある場合はそれを利用することも考えられるだろう．他には PTP を利用してシステムクロックを標準時刻に同期させる手段も考えられる．第 2 節でも触れたように PTP に対応したネットワーク上で，さらに上に述べたような汎用 NIC の H/W タイムスタンプ機能を備えていた場合は，これを利用すれば GPS に近い標準時刻への同期精度が得られると考えられる．

参考文献

[1] Acharya, A. and Saltz, J.: A Study of Internet Round-trip Delay, Technical Report CS-TR-3736, University of Maryland (1996).
[2] Choi, J.-H. and Yoo, C.: One-way delay estimation and its application, *Computer Communications*, Vol. 28, No. 7, pp. 819–828 (2005).
[3] Gurewitz, O. and Sidi, M.: Estimating One-Way Delays From Cyclic-Path Delay Measurements, *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, Vol. 2, New York, IEEE, pp. 1038–1044 (2001).
[4] Hedayat, K. et al.: A Two-Way Active Measurement Protocol (TWAMP), RFC 5357, RFC Editor (2008).
[5] IEEE Instrumentation and Measurement Society: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Std 1588-2008, IEEE-SA Standards Board (2008).
[6] Kim, D. and Lee, J.: One-way delay estimation without clock sychronization, *IEICE Electronics Express*, Vol. 4, No. 23, pp. 717–723 (2007).
[7] 北口善明, 町澤朗彦, 箱崎勝也, 中川晋一: 高精度時刻 PC による片道遅延時間によるネットワーク帯域推定手法, 電子情報通信学会論文誌 B, Vol. 87, No. 10, pp. 1696–1703 (2004).
[8] Kitaguchi, Y., Okazawa, H., Shinomiya, S., Kidawara, Y., Hakozaki, K. and Nakagawa, S.: Development of a High-Accurate Time Server for Measurements of the Internet, *Information Networking: Wireless Communications Technologies and Network Applications*, Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 351–358 (2002).
[9] Kocak, C. and Zaim, K.: Performance measurement of IP networks using Two-Way Active Measurement Protocol, *2017 8th International Conference on Information Technology (ICIT)*, New York, IEEE, pp. 249–254 (2017).
[10] 桑野 茂: 簡易 GPS モジュールを用いた LAN 用高精度時刻同期技術に関する検討, 大同大学紀要, Vol. 54, pp. 39–44 (2018).
[11] Mills, D. et al.: Network Time Protocol Version 4: Protocol and Algorithms Specification, RFC 5905, RFC Editor (2010).
[12] Mills, D. L.: Internet Delay Experiments, RFC 889, RFC Editor (1983).
[13] Mnisi, N. V., Oyedapo, O. J. and Kurien, A.: Active Throughput Estimation Using RTT of Differing ICMP Packet Sizes, *2008 3rd International Conference on*

Broadband Communications, Information Technology & Biomedical Applications, New York, IEEE, pp. 480–486 (2008).

[14] National Marine Electronics Association: NMEA 0183 Interface Standard, Standard, NMEA (2018).

[15] Pointek, J., Shull, F., Tesoriero, R. and Agrawala, A.: NetDyn Revisited: A Replicated Study of Network Dynamics, *Computer Networks and ISDN Systems*, Vol. 29, No. 7, pp. 831–840 (1997).

[16] Sanghi, D., Agrawala, A. K., Gudmundsson, O. and Jain, B. N.: Experimental assessment of end-to-end behavior on Internet, *IEEE INFOCOM '93 The Conference on Computer Communications, Proceedings*, New York, IEEE, pp. 480–486 (1993).

[17] Shalunov, S. et al.: A One-way Active Measurement Protocol (OWAMP), RFC 4656, RFC Editor (2006).

[18] Takeuchi, J. and Yamanishi, K.: A unifying framework for detecting outliers and change points from non-stationary time series data, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, No. 4, pp. 482–492 (2006).

[19] Vakili, A. and Gregoire, J.: Accurate One-Way Delay Estimation: Limitations and Improvements, *IEEE Transactions on Instrumentation and Measurement*, Vol. 61, No. 9, pp. 2428–2435 (2012).