「マルチメディア，分散，協調とモバイル
(DICOMO2014)シンポジウム」 平成26年7月

高精度時刻同期を分散処理制御に活用したタイムアウェア処理方式

堤 智昭[1] 大島 浩太[2] 中條 拓伯[3]

概要：ネットワークの高速化やサーバ仮想化技術の発達により、クラウドコンピューティングに代表される複数のノードが協調して動作を行う分散処理型サービスやネットワーク仮想化が普及している。一方、マイクロ秒やナノ秒レベルの精度でノード間の時刻を同期可能な IEEE1588 PTPv2 が登場した。本研究では、高精度に時刻同期されたノードを用いた分散処理制御方式を提案する。これをタイムアウェア型分散処理制御と名付ける。本方式では、予め時刻に応じていつどのような処理を実行するかを記述した動作シナリオと時刻に従ってシステム内のノードをデータが渡り歩き処理が進行する。処理状況確認パケットを送らずとも他ノードの状況を確認でき、効率的な計算資源の割り当てが可能である。方式の実現にあたり、通信遅延やノードの性能や特性に起因する遅延の影響を把握、吸収することが課題となる。そこで、動的に動作シナリオを補正するための処理時刻補正のモデル化を行った。提案方式の有効性を確認するための原理実験用ソフトウェアを開発してプロトタイプシステムを構築し、処理時刻補正のモデルに基づいた処理タイミング補正が正確な時刻に沿った処理において有効であることを確認した。

A Time-aware Control Method Using High Precision Time Synchronization

Tomoaki TSUTSUMI[1] Kohta Ohshima[2] Hironori Nakajo[3]

1. はじめに

ネットワークの高速化やサーバ仮想化技術の発達により、クラウドコンピューティングに代表される複数のノードが協調して動作を行うサービス形態が普及している。これらは、サーバやルータなどのノードが複数台連携することにより、要求される性能を備えたサービスをオンデマンドに提供可能な分散処理型の構成となっている。また、SDN(Software Defined Network) のようにコントローラがルータ群を制御することで、アプリケーションが要求するネットワークを動的に構成するネットワーク仮想化も普及し始めている。特にクラウドサービスを提供するデータセンターでは、機能をカスタマイズ可能な LSI を搭載したスイッチなどを導入することで、高速なネットワークをプログラマブルに構成することが可能となってきている。クラウドコンピューティングのような分散処理では、各ノードは独立して処理を行うため、ノード間で処理状況の把握や処理内容の同期機能が必要になる。この手段として、制御命令をネットワーク経由で交換する手法が利用される場合が多い。

ネットワークを取り巻く最近の変化の 1 つに、マイクロ秒やナノ秒レベルの精度でノード間の時刻を同期可能な IEEE1588 PTPv2[1] の登場がある。主に産業用イーサネットで用いられてきたもので、電力網の位相同期の基準情報などで利用されている。PTPv2 は、専用のハードウェアを用いることで、ナノ秒レベルで長時間安定的に時刻同期可能である。ノード間の時刻を同期する手段には、これまで NTP[2] や GPS[3] を利用する方法があった。前者はソフトウェアを導入するだけで、IP ネットワーク上で時刻同期が可能であるが精度がミリ秒レベルであり、後者はナノ秒レベルでの時刻同期が可能であるが機材の設置コスト

[1] 東京農工大学 大学院工学府
Tokyo University of Ariculture and Technology
[2] 埼玉工業大学 工学部情報システム学科
Saitama Institute of Technology
[3] 東京農工大学 大学院工学研究院
Tokyo University of Agriculture and Technology

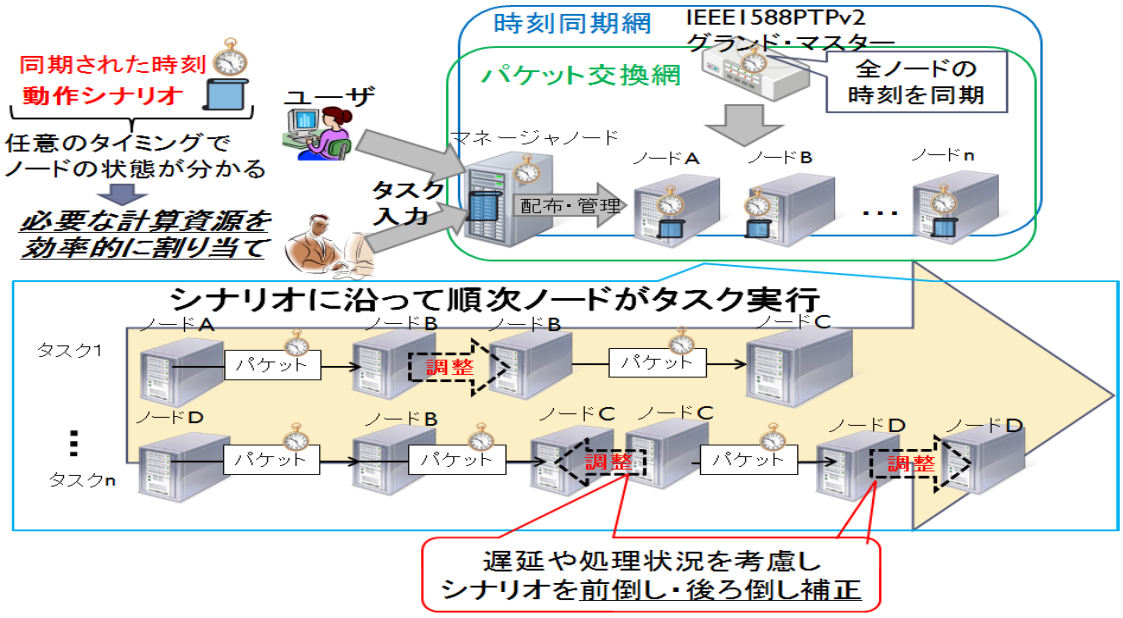

図 1　タイムアウェア型分散処理イメージ

や設置条件に制約がある。

このような背景から本研究では、分散ノードの制御に高精度に同期された時刻を用いた分散処理制御方式の提案を行う。本方式では、分散ノードで処理を行いたいタスクを処理するためのシナリオを作成、全ノードで共有し、ノードは時刻情報と動作シナリオに沿って処理を実行する。本方式をタイムアウェア型分散処理制御と名付ける。本方式を適応したシステムは、分散処理を行うノード群とそれらを管理するマネージャノードから構成される。分散ノード間の時刻同期には IEEE1588PTPv2 を導入する。システムを構成するノード群は IP ネットワークと時刻同期用のネットワークで接続される。

本論文では、タイムアウェア型分散処理制御実現の第一歩として、ノード間の通信時に生じる通信遅延等、処理の実行開始から終了までに発生する遅延が本方式に及ぼす影響を検証する。また、その結果から影響を吸収できる処理時刻補正方式を開発し、プロトタイプシステムを用いた評価を行い、方式の実現可能性を検証する。

2.　システム概要

本システムは、処理ノードが IP ネットワークで接続されており、それぞれが固有の処理機能を有することを前提としている。タイムアウェア型分散処理制御方式のイメージを図 1 に示す。本方式では、IP ネットワーク上で分散処理を実行する各ノードの時計を、IEEE1588PTPv2 を用いて高精度に同期する。処理を行うノードでは、あらかじめ時刻に応じていつどのような処理を実行するかを記述した動作シナリオを共有する。動作シナリオは、特定の時刻までに特定のノードから届くデータを自身の機能を用いて処理し、結果を次のノードへ転送するというように時刻ベースで処理シーケンスが記述されている。各ノードは動作シナリオと同期された時刻に沿って処理を行い、あるノードで行われた処理結果を次のノードに転送することを繰り返し、システム内のノードをデータが渡り歩いて行くことで処理が進行する。

IP ネットワークを介して通信を行う場合、専用のネットワークを用いても、ノード間の通信には中継するスイッチ、ルータの数や物理的な距離によって変動する不可避な通信遅延が発生する。また、ノードが行う処理内容によっては動作シナリオに記述された時刻より遅延が発生する場合や、逆に大きな余裕が発生する場合が考えられる。そこで、シナリオの実行前や実行中に予めノードにおける処理遅延やノード間の通信遅延を計測し、それらを考慮した動作シナリオの動的な補正を行う。ノードが実行中の動作シナリオ通りに処理をできない場合には動作シナリオを後ろ倒しに補正し、逆に処理に余裕がある場合は前倒しにして実行することで、計算資源を効率的に有効出来る時刻とシナリオに基づいた分散処理を実現する。

本システムの特徴は以下の 5 つである。

(1) 時刻と共有された動作シナリオに沿って処理が進行

(2) 任意の時刻に他のノードがどういった処理を行うかが事前にわかり、タスク処理中のノード間における処理状況確認パケットを送らなくても相手の状況を確認可能

(3) ノードが後にどのような状況に遷移するかを事前に判断することが可能となり必要な計算資源の割り当てを効率的に実行可能

(4) 処理中の遅延も考慮し、その影響を吸収するように動作シナリオの動的な補正を行う
(5) 任意の時刻や任意の時間内に処理を実行・完了するという分散処理が可能

システムの適応先としては、大規模な集積回路をそれぞれ固有の機能を持つ回路に分割し分散動作させる場合の動作制御 [4][5] や、山中氏らが提案している uGrid 環境 [6][7][8] のように、多くのそれぞれ固有の機能を有するデバイスを組み合わせサービスを提供するネットワーク基盤におけるサービスフローの実行制御等に適応することを想定している。特に分割回路技術に応用する場合、参考文献 [5] では、分割された回路間を、イーサネットをはじめ様々な接続方式で回路間を接続可能な方式としているが、イーサネットを用いた場合、各回路の動作周期に比べて通信にかかる遅延が大きく、信号情報の転送や、他回路の状況確認通信の頻度増加による実行速度の低下が問題となっている。そこに本方式を適用することで、現在時刻から他ノードの処理状況を把握できるという特徴を活かし、他回路の状況確認を行うパケットの削減や、送信先回路のバッファ等の状況を把握し適切なデータ転送タイミング制御を行うことで、動作速度を向上可能であると考えられる。

3. 課題

タイムアウェア型分散処理制御の実現のためには、複数のノード間で動作シナリオ通りの目標時間にタスクを実行する必要がある。そのためには、各ノードやノード間の通信で生じる遅延を考慮し、影響を吸収することが課題となる。例として、分割された回路の動作制御に適用した場合を考える。**図 2** に示すように、暗号化を行う対象のデータを生成するデータ生成モジュール、暗号化に必要なラウンド鍵を生成するラウンド鍵モジュール、データを暗号化するデータ暗号化モジュールの 3 つに分割されたモジュール群を本方式で制御する。動作シナリオでは、次のように記述されているとする。

- データ生成モジュールは、時刻 t1 から t2 まで動作しラウンド鍵生成モジュールへ処理結果を転送
- ラウンド鍵生成モジュールは時刻 t3 から t4 までラウンド鍵生成を実行し、データ暗号化モジュールへデータを転送
- データ暗号化モジュールは、t5 に動作を開始し、t6 までに暗号化処理を行い、結果を出力

この時、ラウンド鍵生成モジュールの処理結果が遅延によって時刻 t5 までにデータ暗号化モジュールに届かなかった場合、データ暗号化モジュールは必要なデータが無いため動作を行えず、動作シナリオに沿った処理が実行できない状況に陥る。このような 1 つのタスクのみを処理するシステムの場合、最後のデータ暗号化モジュールが必要なデータが到着するまで待機すればシステムは動作するが、

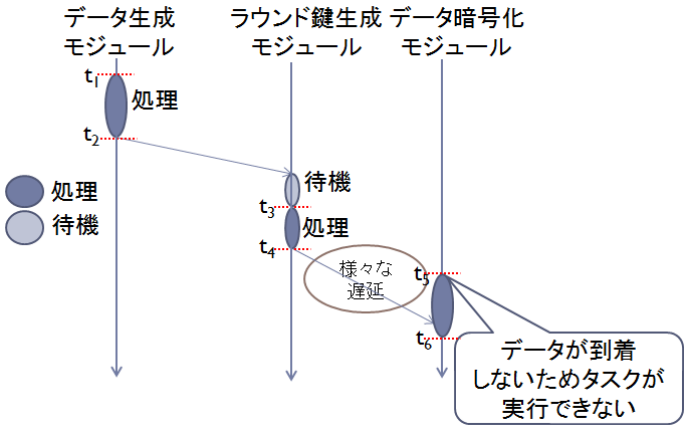

図 2 遅延の同期精度への影響

表 1 遅延の種類と要因

項目	項目番号	具体例
ノードの性能や特性に起因する遅延	1	タイムアウェア処理制御のための時刻取得処理
	2	OS による割り込み処理遅延
	3	他ノードとの時刻同期のずれ
	4	システム負荷による処理遅延
ネットワーク的要因に起因する遅延	5	ノード間の通信性能
	6	ネットワークトポロジや特性による遅延

複数タスクを同時に実行している場合、他のタスクの実行時刻に別のタスクが残っており、計算資源を割り当てられないといった問題が考えられる。こういった課題を解決するために本方式では、各ノードでの遅延やノード間の通信遅延を考慮し動的な動作シナリオの補正を行う。そのために、システム内で生じる遅延を発生原因ごとに分類し、動作シナリオ補正のためのモデル化を行った。

4. 提案方式

4.1 遅延の種類と要因

ここでは、タイムアウェア型分散処理制御を実現した場合に発生する遅延の原因を大きく（A）ノードの性能や特性に起因する遅延（B）ネットワーク的要因に起因する遅延の 2 つに分ける。

種類と要因について表 1 にまとめる。（A）は、時刻同期精度や時刻取得処理の実行にかかる時間と割り込み処理のタイミングといった各ノードのタイマ分解能が影響を及ぼすと考えられる。具体的には、

(1) タイムアウェア分散処理制御のための時刻取得処理やタイムスタンピング
(2) NIC 等のデバイスとのやりとりで生じる OS による割り込み処理による遅延
(3) 他ノードとの時刻同期のずれに起因する処理タイミングのずれ
(4) システム負荷による処理遅延

が考えられる。（B）は、ノード間の通信性能やホップ数の

違い、ルータによる輻輳といったネットワークトポロジや特性が原因となり引き起こされる。具体的には、次の 2 つが考えられる。

（5）ポートからパケットが送出されるまでの伝送遅延やノード間の伝搬遅延による通信遅延

（6）ルータによる再送やキューイングといった転送処理にかかる遅延

4.2 ネットワーク的要因に起因する遅延の計測方式

本システムでは、各ノードにおいてノード間の通信遅延の測定を行う。送信ノードは、送信パケットに送信した時刻をタイムスタンプする。ノード間の時刻が同期されているため、受信ノードはパケットを受信した時刻とパケットにタイムスタンプされた時刻との差分から通信遅延を計測する。測定のタイミングは（1）システムの起動時（2）パケット通信発生時（3）一定時間ノード間の通信がなかった時に行う。測定した遅延時間から、平均遅延時間と標準偏差を計算し、動作シナリオの補正に利用する。

4.3 ノードの性能や特性に起因する遅延の計測方式

ノードの性能や特性に起因する遅延として、今回はノードが行うタイムスタンプにかかる時間について計測を行い、動作シナリオの補正に利用する。また、本方式ではノードはタスク処理中に指定された実行時刻まで待機する場合があるため、どの程度の時間単位で待機状態から復帰しタスクを実行可能かについても計測を行った。タイムスタンピングにかかる時間はシナリオの実行前に計測し、平均時間と標準偏差を求め定数として利用する。待機状態からの復帰については、システムが待機可能な最短の時間待機を行い、待機する前と復帰後の 2 回、現在時刻を取得しその差分から求める方法をとり計測する。

4.4 動作シナリオの補正方式

本補正方式では、タスクを実行しているノードが動作シナリオの補正を行うことで、動的に実行タイミングを変更する。ノードは補正した動作シナリオをマネージャノードへ送信し、マネージャノードから他ノードへ転送を行う。補正は、実測した遅延時間データを元に行い、補正の種類には**図 3** に示したように、処理を後ろ倒しにする補正と、前倒しにする補正の 2 種類がある。自ノードがパケットを受信した時刻を Tr、動作シナリオに記述された自ノードがタスク実行を開始する時刻を Tep、タスクが終了する時刻を Tfi、自ノードから次ノードへパケットを送信する時刻を Ts、次ノードにおけるタスクの開始時刻を Tep2、タスクの終了時刻を Tfi2、パケットの送信時刻を Ts2 とする。また、測定したネットワーク的要因に起因する遅延を dn、ノードの性能や特性の起因する遅延を dl、ノードとマネージャノード間でシナリオを配布するのにかかる時間を m と

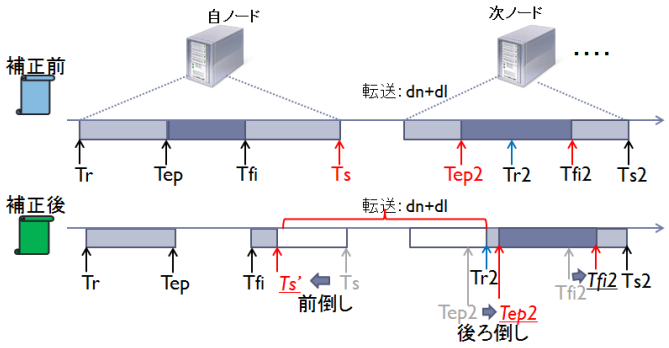

図 3 シナリオ補正の実行例

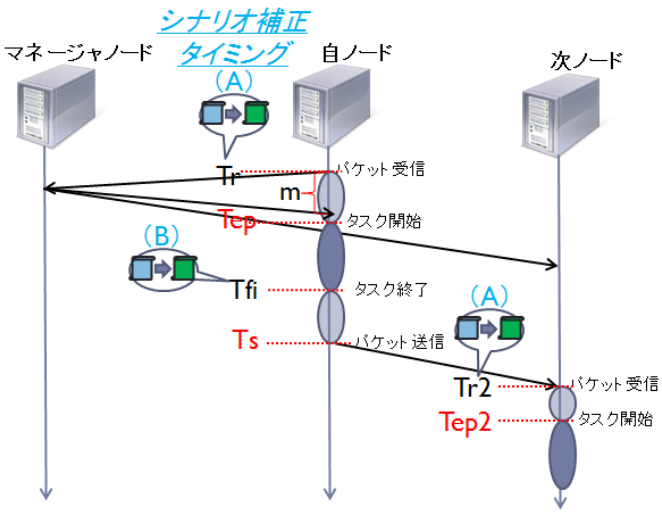

図 4 シナリオ補正の実行タイミング

する。それぞれの時刻について、実際にノードで実行されるタイミングが動作シナリオの順序から入れ替わることがないように動作シナリオの補正を行う。図中の例では遅延 dn、dl によって指定されたタスク実行時刻 Tep2 にパケットが届かない自体を避けるため、Ts に余裕があるので前倒しを行い、さらに Tep2 を後ろ倒しにしている。 補正を行うタイミングは**図 4** に示すように、次の 2 つとする。

（A）ノードが前のノードからパケットを受信した時

（B）ノードがタスクを終了した時

シナリオの補正は、自ノードと次のノードが行うタスクに対してのみ行い、自ノードに対する補正を優先して実行する。例えば、自ノードから次ノードへのデータ転送時間が長く、自ノードからの転送開始を前倒しするか、次ノードのタスク実行を後ろ倒しするかのどちらか一方が必要な場合、優先的に自ノードが前倒しを行う。（A）ノードが前のノードからパケットを受信した時では、時刻 Tep、Ts、Tep2 について確認し、補正を行う。（B）ノードがタスクを終了した時では、Ts、Tep2 について確認し、補正を行う。

4.4.1 Tep の補正

（A）の時に

$$Tep < Tr < Tfi \tag{1}$$

となる場合 Tr-Tep 分以降のシナリオの実行時刻を後ろ倒しする。それに伴い、Tfi も後ろ倒しされ Tfi' となる。

$$Tr + m > Tfi' \quad (2)$$

となる場合、他ノードがマネージャノードから新しいシナリオが配布されるまで、つまり時刻 Tr+m になるまで次のタスクに移行しないよう Ts の時刻を補正することで、マネージャノードとの通信についても考慮する。また

$$Tr + dl + m < Tep \quad (3)$$

を満たすとき、Tfi、Ts 及び Tep が前倒し可能である場合 Tep を Tep-Tr 分前倒しする。それに伴い、Tfi、Ts も Tep-Tr 分前倒しされ、Tfi'、Ts' となる。

4.4.2 Ts、Tep2 の補正

（A）（B）の時に

$$Ts + dn + dl > Tep2 \quad (4)$$

となり遅延の影響により次ノードの動作時刻にパケットが届かないと予測される場合、Ts が前倒し可能で（Ts+dn+dl）- Tep2 分 Ts を前倒しを行う。前倒しにより変更になる Ts' は

$$Tfi + dl < Ts' \quad (5)$$

を満たす時刻まで前倒し可能である。Ts が前倒しできなかった場合や、Ts の前倒し後も

$$Ts' + dn + dl > Tep2 \quad (6)$$

となる場合、（Ts+dn+dl）- Tep2 分 Tep2 を後ろ倒しする。

5. プロトタイプ実装

提案方式の有効性を確認するため、プロトタイプシステムを開発し評価を行った。プロトタイプシステムのシステム構成を図 5 に示す。プロトタイプシステムでは、3 台のノードとマネージャノード 1 台を用い、PTPv2 による時刻同期精度を保証するため、専用のスイッチを用いた時刻同期用ネットワークと、ノード間のデータ通信用ネットワークの 2 つを用いる構成とした。タスクを実行するノードには表 2 に示す汎用 PC を用い、時刻同期ネットワークに接続するインターフェースは，PCI Express 接続の PTPv2 OC（Ordinary Clock）機能搭載 NIC（Meinberg 社製 PTP270PEX　以下 PTPNIC）を使用した。これにより、ノード間でナノ秒レベルの時刻同期精度を確保し、マイクロ秒レベルでのシナリオ実行には時刻同期のずれは無視できるものとした。現在時刻の情報は、この PTPNIC から専用の API を用いて取得する。また、動作シナリオで指定された時刻まで待機をするために、usleep システムコールを用いた。ノードには、Kernel バージョン 2.6.31 に CONFIG PREEMPT　RT パッチを適用した Ubuntu10.10 を用いた。プロトタイプシステムは時刻とシナリオに応じてノード間で通信を行うソフトウェアとして実装し、開発には C 言語を用いた。

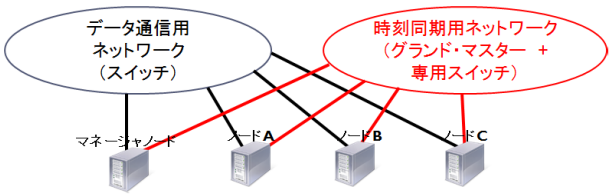

図 5　プロトタイプシステムのシステム構成

表 2　ノード仕様

項目	仕様
CPU	Core2Duo　2.66GHz
Memory	2GB（DDR2）
OS	Ubuntua10.10（Kernel2.6.21）
NIC	Marvell 製 1Gbps Meinberg 製 PTP270PEX（時刻同期ネットワーク接続）

6. 評価

タイムアウェアな処理に影響を与える遅延について計測し、提案するモデルに従って動作シナリオの補正を行い、補正モデルの検証を行った。

6.1 ノードの性能や特性に起因する遅延の計測

6.1.1 PTPNIC からの時刻取得実行時間

本システムを構成する各ノードにおいて、タスクを処理する時に動作の判断基準となる現在時刻を取得する必要がある。ここでは、1 台のノードを用いて、PTPNIC から時刻取得にかかる時間を計測した。計測では、PTPNIC から 2 回連続で時刻を取得し、その差分から 1 回の時刻取得処理にかかる時間を計測した。計測は 1000 回実行し、平均実行時間と標準偏差を測定した。結果、平均実行時間は 4.0 マイクロ秒、標準偏差は 0.1 となった。

6.1.2 待機状態からの復帰にかかる時間

usleep システムコールで待機可能な最短時間の測定を計測することで、一度待機状態に入ってからタスクを再開するまでに発生する遅延を測定した。計測では、usleep システムコールを実行する前後に時刻を取得し、2 つの差分を計算する。その差分から、6.1.1 で求めた時刻取得にかかる時間の平均を減算し、計測した。試行を 1000 回行った結果、平均実行時間は 55.8 マイクロ秒、標準偏差は 9.8 となった。この結果から、本プロトタイプシステムでは、usleep システムコールで待機可能な最短時間よりも短い時間待機処理を行う場合、例えば 20 マイクロ秒の待機処理を行うシナリオがあった場合、約 55.8 - 20 マイクロ秒の遅延が発生すると考えられる。そこで、このようなシナリオが実行された場合、次の動作が前倒し可能ならば 20 マイクロ秒前倒しするか、不可能ならば 35.8 マイクロ秒後ろ倒しする補正が必要になる。

表 3　ノード間転送遅延評価結果

パケットサイズ	平均転送時間 (マイクロ秒)	標準偏差 (マイクロ秒)
64 バイト	143.8	4.6
128 バイト	145.6	4.5
256 バイト	149.0	4.0
512 バイト	154.8	4.6
1024 バイト	168.1	4.3
All	152.3	9.8

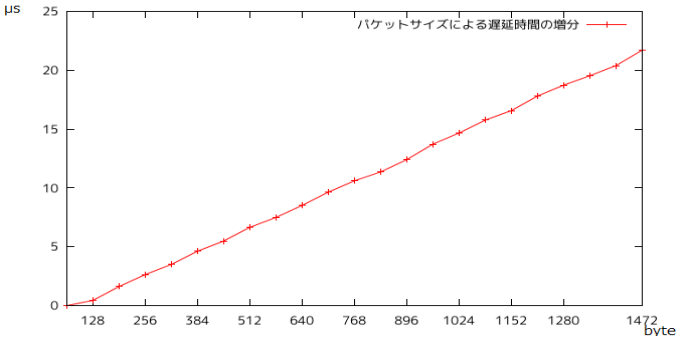

図 6　パケットサイズによる遅延時間の増分

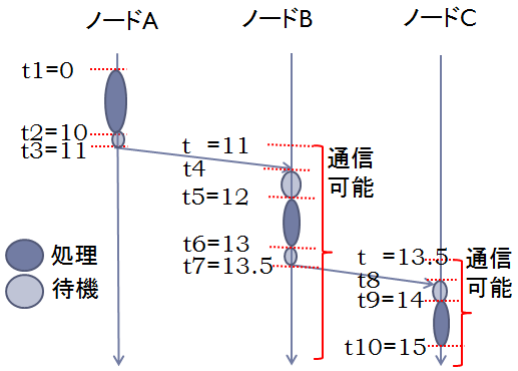

図 7　実験シーケンス図

6.2　ネットワーク的要因に起因する遅延の計測

6.2.1　ノード間の通信遅延

UDP 通信を行いノード間の 1 ホップの転送にかかる通信遅延の計測を行った。パケットサイズは 64、128、256、512、1024 バイトとし、それぞれ 300 回、合計 1500 回試行した。結果を表 3 に示す。64 バイトの時は平均転送時間 143.8 マイクロ秒、標準偏差 4.6、1024 バイトの時では平均転送時間は 168.1 マイクロ秒、標準偏差は 4.3 となっており、パケットサイズによって標準偏差はほぼ変わらないが、平均転送時間は変動することがわかる。

6.2.2　パケットサイズによる通信遅延への影響

イーサネット経由でノード間の通信を行う場合、転送するパケットが NIC で処理され、送出され初めてから最後のビットが送出されるまでの時間は、パケットサイズによって異なる。6.2.1 の結果からもわかるようにパケットサイズの違いはノード間の通信遅延に影響を及ぼすと考えられる。ここでは、パケットサイズを 64 バイトから 1472 バイトまで 64 バイト刻みで通信にかかる時間を計測した。計測結果から、各パケットサイズの時の通信時間と 64 バイトの時の通信時間との差分をとった結果を**図 6** に示す。この結果から、ノード間の通信遅延はパケットサイズに比例して増加し、64 バイト増加するごとに通信時間が約 1 マイクロ秒増加していることが分かる。最大のパケットサイズ 1472 バイトの時は、最小のパケットサイズ 64 バイトの場合に比べて約 22 マイクロ秒長くなっている。

この結果と、6.2.1 の 64 バイト〜1024 バイトまでの試行 1500 回分の標準偏差が大きくなっていることから、通信遅延を計測する時にパケットサイズによる影響を考慮する必要があると考えられる。そこで、4.4 の（4）（6）式を用いて動作シナリオの補正を行う時、ネットワーク的な要因に起因する遅延 dn の値を、パケットサイズによる影響 pd と 64 バイトのパケットを送った時の通信遅延 td に分け、補正を行うこととする。pd は送信するパケットサイズを S、64 バイトごとに増加する遅延時間の増分を ad とすると、

$$pd = (S/64) \times ad \tag{7}$$

で表すことができる。

6.3　測定結果に基づいた動作シナリオの補正

データ通信用ネットワークの遅延時間を変動させた場合に、4.4 で述べた動作シナリオの補正モデルが正常に稼働していることを確認するため 3 台のノード、ノード A・ノード B・ノード C を用いタイムアウェア型分散処理を行い動作結果を検証する。3 つのノードは課題に挙げたデータ生成/ラウンド鍵生成/データ暗号化モジュールのように、3 つのモジュールから構成されるシステムを想定し、ノード A、ノード B、ノード C の順でデータを転送し処理を行う。今回は、ノード A とノード B 間のネットワークの遅延時間を 500 マイクロ秒から 1900 マイクロ秒まで 200 マイクロ秒刻みに変動させ、動的に動作シナリオの補正を行い検証した。動作シナリオの補正に必要な事前に計測する遅延情報は 6.1、6.2 の結果を利用した。初期の動作シナリオのシーケンス図を**図 7** に示す。動作シナリオは次の通りとした。ここでは tn をシステム動作後からの経過時間とする。

（1）ノード A は t1=0 から t2=10ms まで動作後 t3=11ms にノード B へ転送する
（2）ノード B は t = 11ms までに通信用インターフェースを通信可能な状態にする
（3）ノード B がパケットを受信した時刻を t4 とし、t5=12ms から t6=13ms まで動作後 t7=13.5ms にノード C へ転送する
（4）ノード C は t=13.5ms までに通信インターフェースを通信可能な状態にする
（5）ノード C がパケットを受信した時刻を t8 とし、t9=14ms から t10=15ms まで動作後結果を出力する

ノード B、C が通信可能になる時刻は前倒し不可能とする。

表 4　動作シナリオ補正評価結果

時刻 (ms) ＼遅延時間 (μ s)	500	700	900	1100	1300	1500	1700	1900
t1	0	0	0	0	0	0	0	0
t2	10.10	9.95	10.00	9.99	9.98	9.99	9.99	10.10
t3	11.00	11.00	11.00	10.83	10.64	10.40	10.22	10.10
t4	11.52	11.72	11.88	11.95	11.97	11.93	11.94	12.03
t5	12.00	12.04	12.04	12.05	12.05	12.03	12.03	12.06
t6	13.12	13.16	13.15	13.17	13.16	13.15	13.15	13.18
t7	13.53	13.51	13.50	13.52	13.52	13.50	13.50	13.52
t8	13.86	13.83	13.82	13.84	13.83	13.82	13.82	13.84
t9	14.05	14.04	14.01	14.03	14.03	14.00	14.00	14.03
t10	15.00	15.04	15.01	15.03	15.03	15.05	15.03	15.04

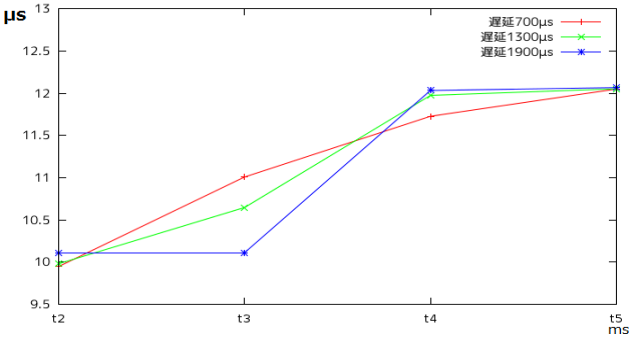

図 8　補正実験結果

ノード間でやりとりするパケットサイズは 1024 バイトとする。また、ノード B とノード C 間、各ノードとマネージャノード間の遅延時間は 300 マイクロ秒とした。

全ての実験結果を表 4 に示す。**図 8** にノード A とノード B 間のネットワークの遅延時間が 700 マイクロ秒、1300 マイクロ秒、1900 マイクロ秒の時の t2、t3、t4、t5 の値を示す。結果から、ノード A とノード B 間の遅延時間が 500 マイクロ秒から 900 マイクロ秒の時は、初期シナリオ通り送信時刻 t3=11ms で実行され、ノード B が通信不可能な t=11ms より前には転送が実行されていないことがわかる。一方、2 ノード間の遅延時間が 1000 マイクロ秒を超えた場合、ノード B の動作開始時刻 t5=12ms に間に合うように t3 が書き換えられ、遅延時間を考慮して事前に送信されている。具体的に、通信遅延が 1300 マイクロ秒だった場合、t3 が 10.40ms に書き換わり、ノード B はシナリオに記述された動作実行時刻 12ms の前 t4=11.93ms にデータを受信することができている。この結果から、本処理時刻補正方式がノード間で生じる遅延の影響を緩和できることを確認した。

7.　まとめ

本論文では、分散ノードの制御に高精度に同期された時刻を用い、あらかじめ時刻に応じていつどのような処理を実行するかを記述した動作シナリオと現在時刻に沿って、データがシステム内のノードを渡り歩いて行くことで処理が進行するタイムアウェア型分散処理制御を提案し、実現の第一歩として、ノード間の通信時に生じる通信遅延等、処理の実行開始から終了までに発生する遅延が本方式に及ぼす影響を検証した。その結果から、処理時刻を補正する方式を開発し、原理実験のためのプロトタイプシステムを用いて、方式の実現可能性を検証した。評価の結果、データ通信用ネットワークにおいて、ノード間の通信にかかる遅延時間を変動させた場合に時刻補正モデルが正常に稼働していることを確認し、本補正方式が正確な時刻に沿った処理において有効であることを確認した。

今後は複数のルータを経由する複雑なネットワークにおける本方式の検証、効率的な初期シナリオ生成手法の開発、提案方式を適応したシステムの開発を予定している。

謝辞

本研究の一部は、文部科学省特別経費「持続可能社会にむけた知的情報空間技術の創出」および JSPS 科研費基盤研究 (C) 25330067、 若手研究 (B) 24700060 による支援を得た。ここに記して感謝する。

参考文献

[1] IEEE Standard 1588,"IEEE Standard for a Precision Clock Synchronization Protocol for Networked Mesurement and Control Systems", November 2002.

[2] D. L. Mills,"Internet Time Synchronization: The Network Time Protocol",IEEE Trans. On Communication, Vol.39, No.10, October 1991.

[3] トランジスタ技術編集部編,“GPS のしくみと応用技術”, CQ 出版社、2009 年 11 月.

[4] 加藤佑典, 渡邊大輔, 中條拓伯,” スケーラブル・ハードウェア機構における信号情報格納・再生方式”, 信学技報, vol. 113,no. 417,CPSY2013-77,pp. 25-30,2014 年 1 月.

[5] 渡邊大輔, 加藤佑典, 中條拓伯,” スケーラブル・ハードウェア機構におけるハードウェア拡張プロトコル”, 信学技報,vol.113,no.417,CPSY2013-78, pp.31-36,2014 年 1 月.

[6] 岡本聡, 荒川豊, 山中直明“ ユビキタスグリッドネットワーキング環境 (uGrid) の提案”, 電子情報通信学会ソサエティ大会, B-7-15, September 2007.

[7] 岡崎裕介， 須佐雄輝， 碓井亮太， 荒川豊， 岡本聡，山中直明,” uGrid におけるダイナミック光パスを用いた映像サービスパーツ選択”, 電子情報通信学会技術研究報告. PN, フォトニックネットワーク 108(476), 29-34, 2009-03-02 .

[8] 中原健太, 菊田洸, 山田翔太, 石井大介, 岡本聡, 山中直明,” ユビキタスネットワーキング環境 (uGrid) におけるスケーラブルなサービス提供の実現へ向けたルーチングプロトコルの拡張”, 電子情報通信学会技術研究報告. NS, ネットワークシステム 110(190), 49-54, 2010-08-26.