分散ロボットシステムサーバのための応答性向上手法に関する考察

中山 悟† 中野 美由紀‡ 菅谷 みどり†

†芝浦工業大学 理工学研究科 電気電子情報工学専攻〒135-8548 東京都江東区豊洲 3-7-5
‡産業技術大学院大学 情報アーキテクチャ専攻 〒140-0011 東京都品川区東大井 1-10-40
E-mail: †{ma15058, doly}@shibaura-it.ac.jp, ‡miyuki@aiit.ac.jp

あらまし 近年，安価なロボットの普及や無線への接続の普及により，分散ロボットシステムは，新たなサービス用途において大きな可能性を持つものとして改めて認識されている．複数台のロボットを利用したサービスにおいては，ロボットを利用したサービス提供者とロボット自身の間で密な通信が不可欠であることから，サービス提供者側は，個々のロボットを管理するための情報収集が必要となる．我々は分散ロボットシステムのひとつの適応例として，病院における患者，独居老人等に対する見守りロボットの開発に取り組んでいる．このシステムは，複数台のロボットが役割に応じて対象者に支援を行うもので，各ロボットが役割に応じたデータを非同期でサーバに送信するが，それぞれのデータをサーバに送る頻度や通信要求も様々である．複数台のロボットからデータを収集する仕組みに関する議論は始まったばかりで，ロボットは計算機とは異なり，ハードウェアやソフトウェアに関する制約がある．また，サーバにおけるデータの処理について研究がされているが，ネットワークの負荷が高い場合，処理対象となるデータが到達しない恐れがある．本研究では，ネットワークが高負荷の場合でも，緊急の処理を要するデータの帯域幅を確保し，必ず受け付けて最短で応答することができる仕組みをロボット分散システムに適用し，その有効性を検証することを目的とする．実現のために，ネットワークレベルでの資源制御と，データグループの割り当ての仕組みにより構成する．本論文では，提案する分散ロボットシステムの構成を概観すると共に，予備実験の結果について報告する．

キーワード 分散システム，ロボット，応答性

1. はじめに

近年，安価なロボットの普及や無線への接続の普及により，複数のロボットを連携する分散ロボットシステムは，新たなサービス用途において大きな可能性を持つものとして改めて認識されている．少子高齢化社会の日本では人の行動支援をロボットが行うことが多いに期待されている．また，Pepper[1]など人間とのコミュニケーションを目的としたロボットを使い，フロア案内をするなどにより，人的資源が限られた店舗において，待ち行列の緩和などが期待できる．このように，分散ロボットシステムは様々な場面での利用が期待される．

複数台のロボットを利用したサービスにおいては，ロボットを利用したサービス提供者とロボット自身の間で密な通信が不可欠である．つまり，サービス提供者側は，個々のロボットを管理するために，その情報収集が必要となる．ロボットは，自律制御がある，ないに関わらず，移動や会話など，動的に変化する環境へのリアルタイムでの応答が求められることから，その目的の達成についてデータを効率よく収集して直ぐに提供するサービスの内容を素早く決定するという要求は，システム構成上自然な要求である．

現在，我々の研究室では，病院における患者，独居老人等に対する見守りロボットシステムを研究開発している．見守りロボットシステムでは，複数台の移動体ロボットが，それぞれのもつ役割に応じて対象者に支援を行うものである．各ロボットは，収集したデータを非同期でサーバに送信するが，役割によって必要となるデータ及びその応答やロボット動作が異なることから，サーバに送る頻度やデータの通信要求も様々である(表 1)．また，人に対する見守りでは状況に応じて緊急度の異なる通信が必要になると考えられる．

表 1 見守りロボットシステムで収集するデータ

対象	ロボットが収集するデータ	サーバに送る頻度	通信要求
ロボット自身	動作状態	一定周期	
	電力	一定周期	
	位置	一定周期	
ロボットを介した人	生体	ストリーム（連続的）	
	位置	イベント時	高速、確実
	状態	ストリーム（連続的）	
他ロボット	位置	一定周期	

複数台のロボットから情報収集する仕組みに関する議論は始まったばかりである．一般に従来のロボットは計算機とは異なり，ハードウェアやソフトウェアが固有であり，リソースやネットワークの信頼性が低

いといった制約がある．また，サーバにおけるデータの処理について，スケジューリングによって，コアの最適配置を行い，計算コストの削減と応答性の向上を実現する取り組み[2]なども提案されている．しかし，ネットワーク自体の負荷が高い場合，そもそもサーバにデータが到達しない恐れがある．そのため，IO サブシステムの制御を行うことが有効であると考えられる[3, 4]．しかし，分散ロボットシステムに適用している例はあまりなく検証する必要がある．

本研究では，ネットワークが高負荷の場合でも，緊急の処理を要するデータの帯域幅を確保し，必ず受け付けて最短で応答することができる仕組みをロボット分散システムとして実現し，その有効性を検証することを目的とする．実現のために，ネットワークレベルでの資源制御と，データグループの割り当ての仕組みにより構成する．ネットワークレベルの資源制御の方法として，データの通信要求に適した通信プロトコルでデータ転送を受け付けるアクセプタと，ネットワーク上のデータ流量をみて緊急度の高いデータを受信するための帯域を確保するモニタを実装する．モニタでは，cgroup により事前にソケットごとに異なる資源グループを割り当て，制御を行うと同時に，アクセプタによりサーバ内でグループを判定し，スレッドごとに異なる資源グループとして管理することにより実現する．

今回，第一段階として，サーバクライアントモデルのプログラムを作成し，クライアントからサーバに送信する際に使用する各通信プロトコルで，データ転送にかかる時間の比較実験を行った．

本論文の構成は，以下の通りとする． 2 節にて見守りロボットシステムの概要について述べ，3 節にて既存研究を紹介する．4 節にて提案サーバの設計を提案し，5 節と 6 節で予備実験について報告し，7 節で本論文をまとめる．

2. 見守りロボットシステム

見守りロボットシステムは，例えば病院内における看護支援や独居老人の支援等での利用を目的に開発しており，配置された位置，時間，見守り対象の状況が変更されることで異なる支援を行うロボットで構成する必要がある．具体的には，次のような動作が想定される．対象となる患者が移動しない場合には 1 台で複数の患者を見守り，移動するときには個別に見守りを行う．対象となる患者が転倒した場合には，その対応が緊急度が高くなる．このような状況においてそれぞれのロボットは，ロボットに搭載されたセンサのデータやサーバに保存されたデータを利用して動作する．また，収集したデータは必要に応じて，非同期にサーバに送信する．

リハビリ支援ロボットは，杖を上部に装着させたロボットが対象者の動きに合わせて移動するものである．ロボットに搭載した超音波センサから常にデータを取得し，対象者との距離を計算し，動作している．

徘徊老人の追跡ロボットは，養介護施設内で老人が補助者を連れずに徘徊中に転倒し，危険な状態にあると判断したときに，老人の転倒した位置にロボットが向かうシステムである．老人がもつスマートフォンに搭載された加速度センサとジャイロセンサから取得したデータをもとに，屋内測位による位置の判定と転倒の判定を行う．位置情報は，老人の転倒時に緊急である場合に送られるもので，人命に関わる情報であるため，高速，かつ，確実にサーバに送られる必要がある．

また，ロボット自身のデータを取得することも重要である．ロボットの動作状態や電力消費量，位置情報を取得することで，それぞれのロボットの稼働状況の把握や故障判定ができる．また，ある範囲内のロボットの位置を把握することで，支援ロボットの過不足をみて，必要な範囲へロボットを移動させることができる．ロボット自身に関するデータはロボットの動作不良時に原因を特定するために使われることを想定し，一定周期で取得する．

見守りロボットシステムでは，データによってサーバに送る頻度や通信要求が異なることから，それぞれのデータは適切な方法で転送，処理される必要がある．

3. 既存研究

現在、分散ロボットシステムにおいて提供されているものに ROS がある[5]．ROS は，ロボットアプリケーションを作成するためのライブラリやハードウェア抽象化，デバイスドライバ等が提供されている．しかしながら，ネットワークを介して情報を取得，保持し，迅速に機能を切り替えるといった，直接サービスに適用した形での変更等は難しい．

また，データ処理の応答性を向上させるために，コア間時間集約スケジューラ(IAS)[6]や SmartShare[2]などのスケジューリングに関する方法がある．

IAS は，メモリアドレス空間を共有するスレッド間やコア間で、使用するデータをキャッシュ上に集約することにより，スループットの向上や実行時間の削減効果が期待されるという技術である．

SmartShare は，クラウド内の動的なインスタンス割り当てを行い，異なるユーザからの複数のインタラクティブセッション間のリソース共有を可能にするものである．パフォーマンスを分離しながら，コストの削減ができる．

しかし，これらのスケジューリングは処理対象とな

るデータがそのマシン内に存在することが前提条件となる．ネットワークを介してデータを収集するシステムでは，ネットワーク負荷が高い場合に，データを送信できない問題がある．

4. 提案

4.1. 概要

本研究では，ネットワークの負荷によらず，緊急のデータに対して迅速に応答することを目的とする．そこで，ネットワークレベルでの資源制御と，データグループの割り当てするミドルウェアとして，ダイナミックアダプテーションを行う仕組みを提案する(図 1)．

本ミドルウェアでは，クライアントが送信するデータに対して，あらかじめサービスに応じて付与された優先度で通信制御を行える機構を提供する．例えば，図 1 の右側に示すように，ロボット 1 の見守り対象が転倒したことを検知する．ここで，ロボット 1 の通信優先度は高に設定され，ロボット 1 の提供する情報が最優先で処理されることとなる．このとき，帯域幅に対する優先度の低いデータの流量が，優先度の高いデータをサーバに接続されたすべてのクライアントが同時に送信するときの合計のデータサイズを上回る場合にリソース制限をかけることで，緊急のデータを常時受信可能な状態にする．受信したデータを，優先度に応じたキューに振り分けてスケジューラに処理させることで，緊急のデータに対する迅速な応答を可能とする．

本ミドルウェアは，クライアントからデータを受信して優先度ごとのキューに振り分けるアクセプタと，サーバとクライアント間で送受信しているデータの流量を監視しリソース制限をかけるモニタの 2 つから構成される．それらの設計について以降で詳細に説明する．

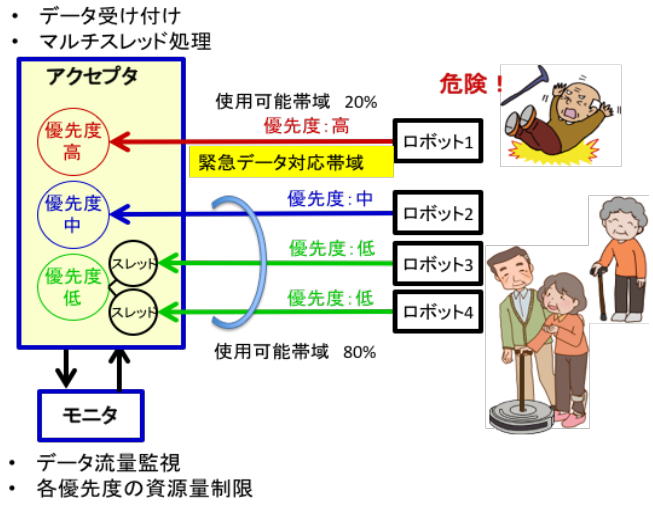

図 1　提案システム構成

4.2. アクセプタ

クライアントからデータを受信するプロセスを異なる優先度グループとして作成し，それぞれのプロセス内で，さらに，データの受信に使用する通信プロトコルごとに通信の受け口となるスレッドを作成する．クライアントは送信するデータのタイプと優先度に合ったスレッドに接続することで，データの通信要求に対して適切な通信プロトコルでデータの転送を行う．

4.3. モニタ

アクセプタ内のそれぞれのプロセスをサービスの緊急度等の情報に基づき，異なる優先度のグループとして利用可能な資源量を管理する．管理方法として，あるプロセスをグループとみなし，プロセス内で使用する資源量を制限する cgroup を用いる．サーバが送受信しているデータ量を sar コマンドで出力し，帯域幅に占める割合をみて，各グループが使用できる資源量を動的に管理することで、緊急のデータの転送に必要な帯域を確保する。

5. 予備実験

5.1. 予備実験環境

分散ロボットシステムにおいて，ロボットが持つ通信特性を考慮して設計を行う必要がある．そこで，ロボットに性能の異なる計算機を搭載した場合の通信性能の違いを調査した．計算機を搭載したロボット（以下，ロボットクライアントとする）は 2 種類用意し，一方のロボットクライアントに搭載する計算機として，小型のシングルボードコンピュータで，Debian OS が動作する Raspberry Pi 2 Model B[7]（以下，Raspberry Pi と略す）を採用した．実験環境は表 2 の通りである．

表 2　実験に使用したマシン性能

	機種	OS	プロセッサ	プロセッサ速度	コア	キャッシュ	メモリ
ロボットクライアント（PC）	VOSTRO 3550	ubuntu 14.04 LTS	Intel Core i3	2.1GHz	2	L3: 3MB	4GB
ロボットクライアント（Raspberry Pi）	Raspberry Pi 2 Model B	Debian	ARM Cortex-A7	900MHz	4	L2: 512KB	1GB
サーバ	MacBook Pro	OS X	Intel Core i5	2.6GHz	2	L3: 3MB	8GB

5.2. ロボット動作とデータ転送方法

ロボットクライアントとサーバの操作手順は以下の通りである．

(1)　ロボットクライアントを起動する．
(2)　サーバを起動する．
(3)　ロボットクライアントで，別のプロセス上でロボットの動作を開始する．
(4)　ロボットクライアントでデータ転送を開始し，

終了まで待機する．

(5) 手順 4 をもう一度行う．

(6) サーバを停止する．

(7) ロボットクライアントでロボットの動作を停止する．

手順 1 から手順 3 の間ではロボットクライアント上で，ユーザによる他の作業を一切行わない．手順 3 ではロボットクライアントに Create Open Interface Library[8]を予めインストールし，C 言語プログラムでロボットに動作命令を送る．プログラムは以下の動作命令を順番に実行することを，無限回繰り返すものである．直前の動作命令が終了次第，次の動作命令をロボットに送り，実行する．

① 速度 250mm/s で 10mm 前進する．

② 速度 250mm/s で反時計回りに 10 度回転する．

③ 速度 250mm/s で 10mm 後退する．

④ 速度 250mm/s で時計回りに 10 度回転する．

5.3. データの流れ

OSI 参照モデルにおけるトランスポート層の通信プロトコルは複数存在する．今回の見守りロボットシステムでは，ロボットの定期的なステータス情報に加え，見守る対象の動画ストリームが送られる．そこで，大容量動画情報と小容量のデータ転送の 2 つが混在することを想定した．そのうち主に使用される UDP と TCP について，それぞれのプロトコルで，送信側と受信側の基本的なソケットプログラムを作成した．送信側がデータを送信し始めた時刻を T_{C1}，受信側からの返答を受信した時刻を T_{C2}，受信側がデータを受信した時刻を T_{S1}，返答を送信し始めた時刻を T_{S2} とする．送信側がデータを送信し，受信側から返答を受け取るまでの時間のうち，データの転送にかかった時間 $((T_{C2}-T_{C1})-(T_{S2}-T_{S1}))$を計測した(図 2)．本実験以外は使わないローカルネットワークを使用し，ネットワークの負荷は一定であるとする．

また，データ転送の流れとして、マシンの場合、送信されるデータは、まずプログラムからシステムバッファにコピーされる(図 3)．その際に、データのメモリアドレスが変わらないときはキャッシュに保存されたものを使うため，処理時間が短縮される．しかし、最終的に実装する際は，ロボットからデータを収集することから、キャッシュによる影響を受けない．そこで，本実験では，毎回の送受信で，データの先頭アドレスを 3 次キャッシュのサイズ以上ずらすことで，キャッシュの影響を排除した．

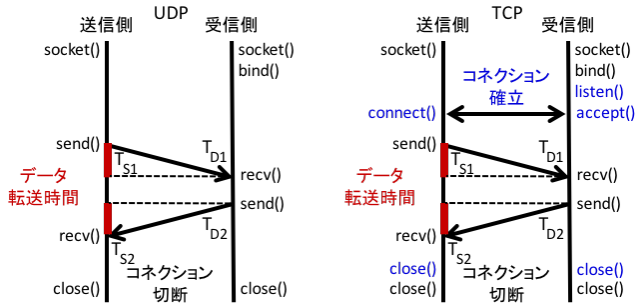

図 2 データ転送時間

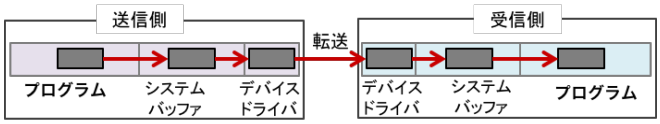

図 3 データ転送の流れ

ロボットクライアントに搭載した計算機と，データ転送に使用したプロトコル，別のプロセスでのロボット動作の並列実行の有無の組み合わせを変えて，512 バイトのデータを 1024 回ずつ転送したときの，データ転送時間の平均を図 4 に示した．横軸は通信プロトコルとロボットクライアントの計算機の組み合わせを示しており，縦軸はデータ転送にかかった平均時間をマイクロ秒単位で示している．

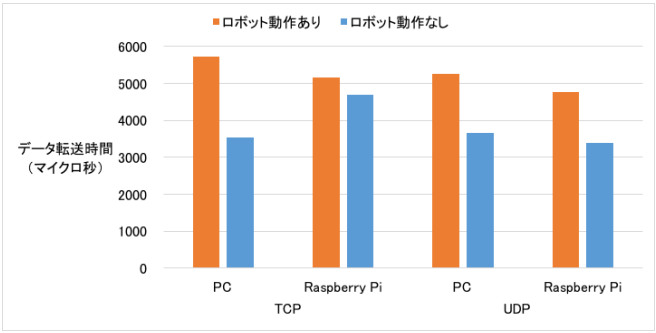

図 4 各環境での平均データ転送時間

全体的に，ロボット動作を並行してデータ転送した場合に時間がかかっていることが分かった．ロボット動作をした場合としなかった場合とで，TCP を用いて PC をロボットクライアントに搭載した場合は約 40%の差が現れ，一方で，Raspberry Pi を搭載した場合は約 10%の差がみられた．また，Raspberry Pi を搭載した場合は PC を搭載した場合に比べて，ロボット動作をしないときにデータ転送時間がかかっていた．更に，UDP の場合は，PC，Raspberry Pi ともに約 30%のオーバーヘッドがあり，TCP に比べると UDP はデータ転送が速く行えることが分かった．

以上のことから，平均転送時間で見られるオーバーヘッドの差について，個別の転送時間の傾向について調べたい．

5.4. データ転送時間

プロトコルとロボットクライアントのそれぞれの組み合わせごとの計測結果について，横軸を経過時間としたデータ転送時間を図 5 から図 8 に示した．縦軸はデータ転送時間をマイクロ秒単位で示している．

ロボット動作時に，全体的にデータ転送時間が延びており，ばらつきが激しくなっていた．このことから，ロボットの動作が負荷となり，データ転送に影響を与えていることが分かった．

まとめとして，ロボットが動作することで，確かにオーバーヘッドは生じ，大きく遅延している通信もあった．一方で，今回は 1 対 1 の通信のみ行ったため，複数台のクライアントと通信する場合にはさらにオーバーヘッドや遅延が増大すると予想される．また，ロボット動作時の通信コストは一定ではなく，大きくばらつきが存在しており，この原因の一つとしてロボット動作時のモータの駆動による影響等も考えられる．

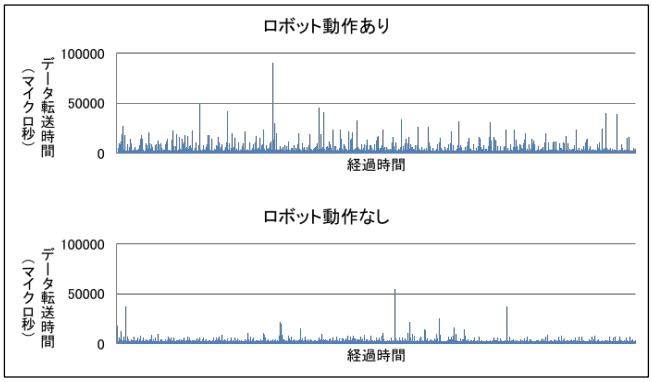

図 5　データ転送時間（TCP，PC 搭載時）

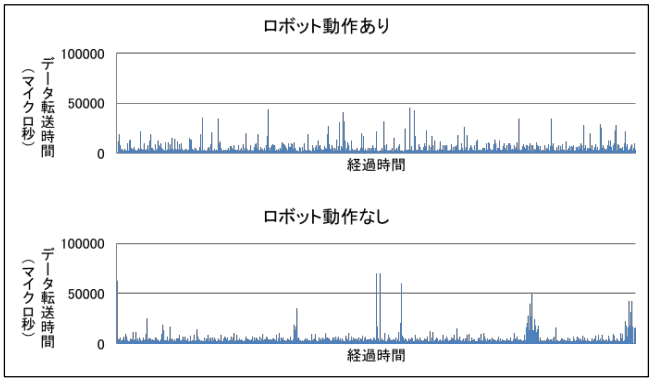

図 6　データ転送時間（TCP，Raspberry Pi 搭載時）

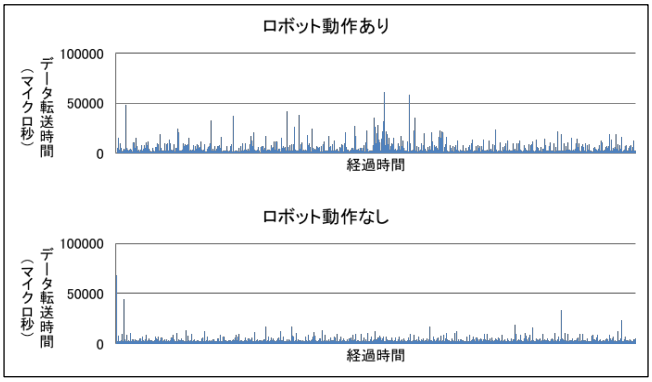

図 7　データ転送時間（UDP，PC 搭載時）

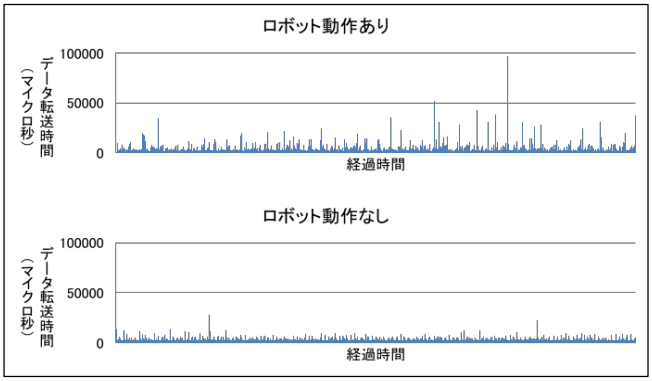

図 8　データ転送時間（UDP，Raspberry Pi 搭載時）

6. 分散ロボット通信における考察

6.1. 実験環境

5 節で示したように，ロボット動作時の通信コストはばらつきが大きい．その原因の一つとして考えられる，ロボット動作時のモータの駆動によるデータ転送時間への影響を調べた．

ロボットに動作命令を送る 1 台のマシンと，サーバ・クライアントモデルのデータ転送を行う 2 台のマシンを用意する．動作しているロボットに 1 cm の距離までクライアントを近づけてデータ転送を行った場合と，ロボットを動作させずにデータ転送を行った場合のデータ転送時間を計測した．

表 2 に示したマシンのうち，ロボットクライアントに使用した PC をロボットに動作命令を送るマシンとし，Raspberry Pi をデータ転送のクライアントマシンとした．データ転送のプログラムは 5 節と同じものであり，512 バイトのデータを 1024 回転送した．

6.2. 計測結果

データ転送に使用したプロトコルとロボット動作の実行の有無の組み合わせを変えてデータ転送したときの，データ転送時間の平均を図 9 に示した．横軸は通信プロトコルを示しており，縦軸はデータ転送にかかった平均時間をマイクロ秒単位で示している．

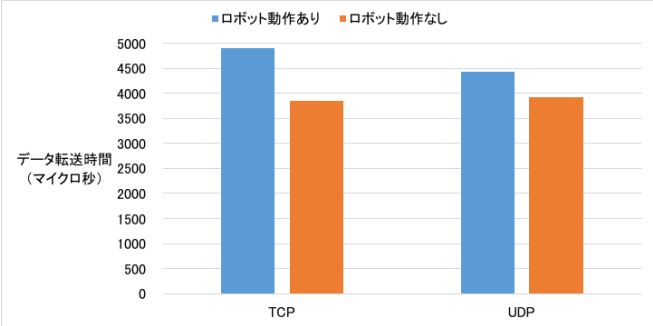

図 9　モータ駆動時の平均データ転送時間

動作しているロボットにクライアントマシンを近づけてデータ転送を行うことで，TCP では約 30%，UDP では約 10%のデータ転送の遅延がみられた．このことから，ロボットのモータの駆動がデータ転送時間に影響を与えていることが分かった．1 台の場合は動作時と停止時のみ考えればよいが，分散ロボットシステムでは互いの影響等について検討する必要がある．

7. まとめ

分散ロボットはこれから広く利用が期待されているが，サービスに応じた通信優先度等を制御できるミドルウェアはいまだ提供されていない．本報告では，分散ロボットシステムのミドルウェアとして，複数の優先順位があるデータ送信を受け付けるサーバにおける，緊急のデータに対する応答性を向上させるための機構を示した．

基本的な通信コストを調べるための予備実験として，サーバ・ロボット間の一対一通信を行いロボット動作時には，通信オーバーヘッドが TCP で 40%ほど高くなることが分かった．また，個別の転送コストを時間経過に沿って見ることで，ロボット動作時の通信コストは全体的にばらつきがあることが分かった．更に，モータ駆動による通信コストへの影響として 20%程度のオーバーヘッドがみられた．

今後，複数のロボットが動作している環境においてサーバ・ロボット間の通信実験を行い，分散ロボットシステムの通信要件を明らかにする．

参　考　文　献

[1] ソフトバンク：Pepper ロボット，http://www.softbank.jp/robot/

[2] Tao Zou, “Optimizing Response Time For Distributed Applications In Public Clouds”, A Dissertation Presented to the Faculty of the Graduate School of Cornell University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy, January 2015

[3] Gaurav Banga, Peter Druschel and Jeffrey C.Mogul, “Resource Containers: A New Facility for Resource Management in Server Systems”, Proceedings of the 3rd Symposium on Operating Systems Design and Implementation New Orleans, Louisiana, February, 1999

[4] Eric W.Anderson and Joseph Pasquale , “The Performance of the Container Shipping I/O System”, SIGOPS ‘95, December, 1995

[5] ROS Wiki, Open Source Robotics Foundation, http://wiki.ros.org/ja

[6] 山田賢，日下部茂，“集約制御機構を持つコア間時間集約スケジューラの実装と評価”，情報処理学会論文誌コンピューティングシステム(ACS), Vol.3, No.3, pp.235-247, September, 2010

[7] Raspberry Pi , Raspberry Pi Foundation , https://www.raspberrypi.org/

[8] Google Code Archive - Long-term storage for Google Code Project Hosting , Google , https://code.google.com/archive/p/libcreateoi/