時間制約の厳しいマルチメディアのためのリアルタイム通信機構

佐藤秀雄[†] 矢向高弘[††]

本論文では，遠隔地との力学的インタラクションを実現する新しいメディアを実現するための通信機構を提案する．遠隔地の物体の質感を利用者側で表現するためには，遠隔地と利用者側との間で短い周期の分散フィードバック制御を行う必要がある．そこで，リアルタイム処理が厳しい時間制約を満たしつつ利用可能な通信機構として RT-Messenger を提案する．RT-Messenger は，リアルタイム性を持たない通信媒体を用いてもリアルタイム処理内での通信を可能にする機構である．本通信機構を RT-Linux 上に実装し，制御情報を交わす分散フィードバック制御を試みた．その結果，1 [msec] 周期のリアルタイム処理ごとに通信を交わすことに成功し，また制御周期の揺れ幅を 1.5 [%] 以内に収めることができた．

A Real-time Communication Mechanism for Hard Time-constrained Multimedia

Hideo Sato[†] and Takahiro Yakoh[††]

This paper proposes a new communication mechanism for new media which can realize dynamical interactions between local and remote site. In order to express the feel of touch, a feedback control is required with short sampling period. To address this issue, RT-Messenger was proposed and implemented. RT-Messenger is a new communication mechanism which makes it possible for real-time applications to satisfy these time constraints. This mechanism is device independent, i.e., it can work without any real-time communication media. To evaluate the performance of proposed mechanism, a feedback control was implemented with RT-Messenger on RT-Linux. The results show that communication within every 1 [msec] sampling periods was well achieved, and the jitter of the control was within 1.5 [%].

1. はじめに

マルチメディアという言葉に厳密な定義はないが，一般的にはコンピュータ上で，動画，静止画，音声，文字など多様な情報形態を複合化して使うことを示す概念として用いられる言葉である．近年では，ネットワーク技術を用いてインタラクティブにマルチメディアを実現するシステムが研究されており，マルチメディア会議室[1)]や，仮想空間での建築意匠設計[2)]などをはじめとした，様々なシステムが提案されている．これらの多くは，人間の五感の中の視覚や聴覚に働きかけるものである．

本論文ではマルチメディアの表す概念を拡張し，遠隔地との力学的インタラクションを実現する新しいメディアの実現を試みている．これは，物の質感を伝達する機構であり，医療の分野において遠隔操作での触診や手術，放射線被曝や感染を余儀なくされるような空間での作業などへの応用が期待できる[3),4)]．

人間は，物体に触れたときの反作用力によって物体の感触を知覚している．そこで提案するメディアは利用者側と遠隔地側に配置する 2 台 1 対の力センサ付きマニピュレータと，それらを接続する通信機構とで構成し，利用者側の動作を遠隔地で模倣し，その模倣動作に対して遠隔地の物体から得られる反作用力をを利用者側に出力することで，遠隔地の物体の質感を表現する．この利用者側での触感を違和感なく表現するためには，主に以下に示す 3 つの理由から，遠隔地側と利用者側との間で 1 [msec] 程度の短い周期でのフィードバック制御を行う必要がある．第 1 の理由は，表面の硬い物体に触れたとき，瞬間的に大きな反作用力が

† 慶應義塾大学大学院理工学研究科総合デザイン工学専攻
School of Integrated Design Engineering, Graduate School of Science and Technology, Keio University
†† 慶應義塾大学理工学部システムデザイン工学科
Department of System Design Engineering, Faculty of Science and Technology, Keio University

物体から出力されるが，これを遅滞なく利用者側へ伝達し表現しなければ硬い物体の質感が伝えられないということである．第2の理由は，人間の指先の感覚は1 [kHz] 以上もあるため，フィードバック周期が低いとざらついた感触を人間に感じさせてしまうことである．そして第3の理由は，力フィードバック制御は，ディジタル制御の場合，位置情報の2階差分が制御情報となるため，フィードバック周期を長くした際の離散化誤差が位置の誤差として蓄積されることになり，結果として一定のはずの物体形状が時間とともに変化してしまうことである．以上の理由から，フィードバック周期は可能な限り短いことが望まれるが，マニピュレータの応答速度などハードウェア上の制約から1 [msec] 前後のフィードバック周期が広く用いられている．

マニピュレータの制御を行うためにはリアルタイムOSが必要となる．現在，リアルタイムOSにはRT-LinuxやART-Linuxなどを代表とする様々なOSが開発されている[5]．しかし，分散環境においてフィードバック制御を実現するための，厳しい時間制約を満足できる通信機構はほとんど開発されていない．そこで本論文では，リアルタイムアプリケーションが厳しい時間制約を満たしつつ利用可能な通信機構としてRT-Messengerを提案する．

本論文の構成を述べる．まず2章でリアルタイム性についての定義と分類を行い，本論文で提案するメディアに要求されるリアルタイム性について述べる．3章では，RT-Linuxとその通信機構について述べ，要求される事柄を明確にする．4章で本論文の提案する通信機構であるRT-Messengerの設計と実装について述べる．5章ではRT-Messengerの性能評価を示す．そして6章では，本論文で実装を試みている新しいメディアの実装と評価を簡単に示し，最後にまとめを述べる．

2. リアルタイム性

本章では，リアルタイム性の定義とその分類について述べる．リアルタイム性の定義は諸説あるが，本論文では，ある処理を行う際に，その処理を終了する時刻が予測可能であり，実行時にそれが保証される性質と定義する．

次にリアルタイム性の分類について述べる．本論文ではリアルタイム性をソフトリアルタイム性とハードリアルタイム性の2つに分類する．ソフトリアルタイム性とは，多少のデッドラインミスを許容できるリアルタイム性である．ソフトリアルタイム性の代表的な例は，次世代インターネットプロトコルIPv6におけるRSVP[6]が持つ性質である．RSVPは通信の帯域幅を保証し，その帯域幅の中でマルチメディアデータなどを転送することを目的としており，多少のデッドラインミスを許容する通信への用途を目的とした機構である．

これに対して，ハードリアルタイム性とはデッドラインミスを許容できないリアルタイム性であり，主に制御系で要求される性質である．本メディアのような人間と接するマニピュレータの制御では，サンプリング周期ごとに指令値を出力し続けなければ，人間に危害を加えたり，マニピュレータ自身が壊れる場合があるため，デッドラインミスを許容することはできないので，ハードリアルタイム性が要求される．

3. RT-Linux

Linux[7]はPCで動作するPOSIX 1003.1互換のOSである．この上に，Yodaikenは，RT-Linuxと呼ばれるハードリアルタイム処理を実現するための機構を開発した[8),9)]．RT-Linuxは現在，ロボット制御や計測機器，工業プラントの制御などに利用されている[5]．RT-Linuxは，デバイスドライバとLinuxのスケジューラとの間に，RT-Schedulerと呼ばれるリアルタイムスケジューラを追加することにより実装されている（**図1** 参照）．

3.1 スケジューリングアルゴリズム

RT-Linuxのタスクは RT-Taskと呼ばれ，リアルタイム処理は RT-Taskごとに行われる．RT-Taskの処理が開始される時刻は RT-Schedulerによって管理されている．

RT-Taskには5つの状態がある．このうち，スケジューリングの対象となる状態は DELAYEDと READYの2つである．DELAYEDは実行時刻が訪れるまで待機している状態である．READYは RT-Scheduler

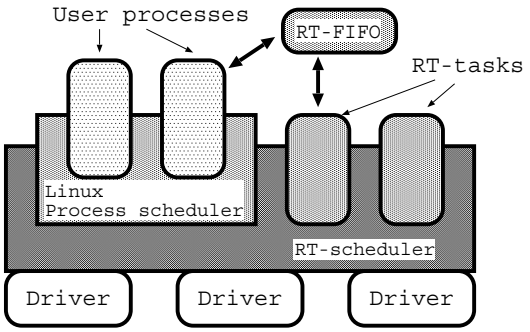

図1 RT-Linux
Fig. 1 RT-Linux.

によってコンテキストスイッチされることを待機している状態である．

ここで，RT-Scheduler のスケジューリングアルゴリズムについて述べる．RT-Scheduler は，状態が DELAYED になっている RT-Task の中から処理の開始時刻が近付いているすべての RT-Task の状態を READY にする．次に READY となっている RT-Task の中で優先度が最も高い RT-Task を記憶する．さらに状態が DELAYED である RT-Task の中から最も早く起動する RT-Task の起動時刻をタイマに設定する．最後に，記憶した RT-Task へコンテキストスイッチする．RT-Task は処理を終了すると RT-Scheduler を呼び出すので以上のスケジューリングサイクルが繰り返される．

もし，READY 状態の RT-Task がなかった場合には，RT ではない通常の Linux にコンテキストスイッチされるが，その場合にも RT-Task が起動する時刻はタイマに設定されており，タイマ割込みにより RT-Scheduler の処理が開始されるので，リアルタイム性はつねに保たれる．このようにして RT-Linux のリアルタイム処理と通常の Linux との共存が実現されている．

3.2 通信機構

RT-Linux には RT-FIFO と呼ばれる通信機構が用意されている．これは，1 台のコンピュータ上で動作している RT-Task と通常の Linux プロセスとの間で通信を行うためのものである．したがって，RT-Task が 2 台のコンピュータ間で通信を行うには，現在の RT-Linux の実装においては RT-FIFO を経由した Linux のユーザプロセス間でネットワーク通信を行うしか方法がない．

現在最も普及しているネットワーク通信機構として TCP/IP がある．TCP/IP は 4 つの階層で構成されており，それぞれ，アプリケーション層，トランスポート層，インターネット層，データリンク層と呼ばれる．アプリケーション層はアプリケーションプログラム内の通信部分に相当し，データリンク層は通信を行うハードウェアとそのドライバに相当する．残りの 2 つの層は，TCP/IP を特徴付けている階層である．

インターネット層は，大きなデータをハードウェアが取り扱える大きさのパケットに分割したり，通信の衝突制御や経路の決定などを行う．トランスポート層は信頼できる接続を実現するための階層である．これらの機能は多くのアプリケーションにとって非常に有用である．しかしながら，これらの機能はデータをバッファリングすることにより実装されており，予測できない遅延を発生することになる．このため，リアルタイムアプリケーションに TCP/IP を利用するのは適切ではない．そこで，リアルタイムアプリケーションが通信を行うための新たな通信メカニズムが必要となる．

4. RT-Messenger

本章では，リアルタイムアプリケーションが厳しい時間制約を満たしつつ利用可能な通信機構として RT-Messenger を提案し，その実装方法について述べる．

リアルタイムアプリケーションは実行時の振舞いが時間制約を満たさなければならないため，処理時間が予測可能な作業しか行うことができない．したがって，これまでは通信時間が予測可能というリアルタイム性を持つ通信媒体しか利用できないと考えられてきた．ところで，一般的にコンピュータ単体よりも通信相手や通信経路などの通信系の方が障害の発生する確率が高いため，通信を行う際には障害対策が行われる．通信における障害を検知する一般的な方法はタイムアウト処理であり，リアルタイム性を持つ通信媒体を利用している場合でもタイムアウト処理を行うのが一般的であった．これをリアルタイムアプリケーション内で行うためには，その時間制約よりもタイムアウト時間が十分に短いことが必要がある．逆にいえば，タイムアウト時間を十分に短く設定できるのであれば，通信媒体がリアルタイム性を持つか否かにかかわらず，リアルタイムアプリケーション内で利用することが可能となる．

もちろん，リアルタイム性のない通信媒体を用いてタイムアウトを短く設定すれば，定常状態でさえも，ある頻度でタイムアウトによる障害対策処理が行われることになる．しかし，データの欠損が致命的か否かはアプリケーションに依存する．マニピュレータの制御においては，ロバストな制御系を用いればデータの欠損に耐えて処理を継続することが可能である．以上の論理に基づいて，本論文では，リアルタイムアプリケーションの中で利用できる通信機構 RT-Messenger を提案する．RT-Messenger はリアルタイムタスクから通信を行う際のオーバヘッドを最小にした機構であり，これにより短いタイムアウト時間を設定しても高い確率でデータの送受信が可能となる．タイムアウト処理は，リアルタイム OS の機能を用いて容易に実現できることから，RT-Messenger はタイムアウト機構を提供していない．本論文では RT-Messenger を RT-Linux 上に実装し，通信媒体にイーサネットを用いた．次に RT-Messenger の特徴をあげる．

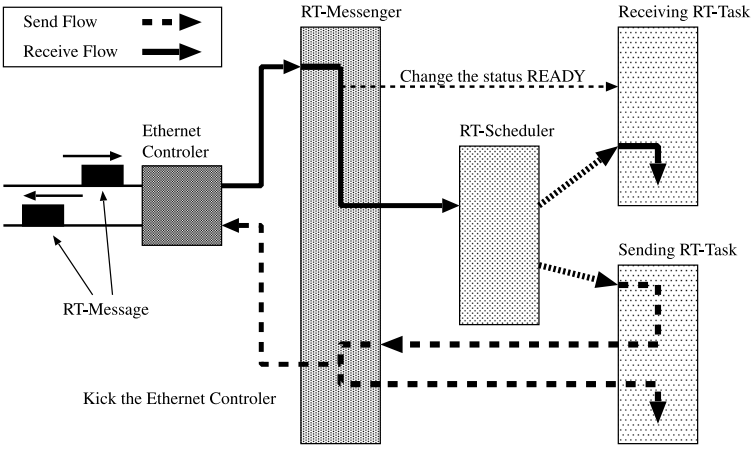

図 2 RT-Messenger の処理
Fig. 2 Process flow in RT-Messenger.

(1) RT-Task がパケットを送受信可能である．
(2) パケットの送信は，最小の処理時間で行われる．
(3) パケットの受信は，RT-Scheduler のリアルタイムスケジューリングを乱さない範囲で可能な限り早く行われる．
(4) RT-Messenger は TCP/IP を用いない実装をしている．

RT-Messenger は，通信に要する処理をできるだけ少なくするためと，バファリングよる予測不可能な遅延を防ぐために，TCP/IP を用いないデータリンク層上に実装している．データリンク層から RT-Messenger のデータを受信するために，RT-Messenger 用のパケットタイプとして RT-Message と定義する．さらに，新しい RT-Task の状態として MESSAGE を追加した．MESSAGE は RT-Task が RT-Message の到着を待機している状態である．RT-Messenger 処理の流れを**図 2** に示す．

4.1 送信機構

RT-Messenger では，可能な限り高速に送信処理を行うため，RT-Task から送信されたパケットを，ソフトウェアによって 1 度もコピーすることなく，通信デバイスが DMA により読み出して送信するというゼロコピー方式を採用している．そのため，パケットを作成するメモリ領域は DMA 可能な空間に確保しなければならない．そこで RT-Messenger は，パケット用のメモリ領域の確保および解放のプリミティブを提供している．一般にメモリを確保する作業はリスト検索をともなうため処理時間が予測できず，リアルタイム処理の中で行うことはできない．そこで RT-Messenger は，いったん確保したメモリ領域を再利用できるようにしている．そのためリスト検索をともなうメモリ領域の確保および解放処理は RT-Task の最初と最後にそれぞれ行うだけでよくなり，周期的にパケットを送信する場合にリアルタイム性を乱すことはない．

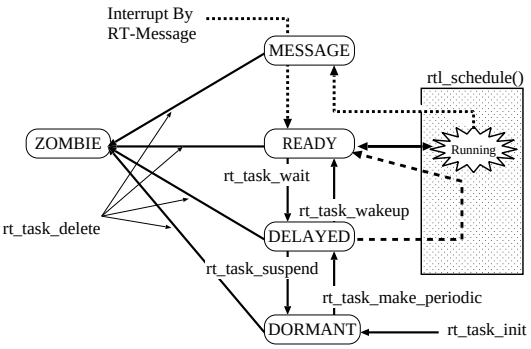

図 3 RT-Task の状態遷移図
Fig. 3 State transition diagram for RT-Task.

なお，この方式ではパケットヘッダを RT-Task 自身が設定しなければならないため，Linux のカーネル内で提供されているいくつかの関数を利用して，イーサヘッダの作成を行うサービス関数を用意している．

4.2 受信機構

RT-Message が通信デバイスに到着するとその通信デバイスが割込みを発行する．この割込みに応じて RT-Messenger の受信処理が呼び出される．RT-Messenger は，RT-Message を受信する RT-Task をパケットの中身から識別し，当該 RT-Task へ RT-Message の先頭アドレスを渡すとともに状態を MESSAGE から READY へ変更し，RT-Scheduler を呼び出す（**図 3** 参照）．その結果，リアルタイムスケジューリングを乱さない範囲で可能な限り早く当該 RT-Task へコンテキストスイッチされ，RT-Task が RT-Message を

受信することになる．

なお，IRQ（割込み番号）は通常，割込みを発生するデバイスごとに割り振られ，同時に複数の割込みが発生した場合には，IRQの小さい割込みが優先される．RT-Messageの受信処理が，他の割込みによって遅延させられることなく可能な限り早く行われるためには，通信デバイスのIRQを，他のデバイスのIRQより小さい値に設定する必要がある．ただし，RT-Linuxはタイマ割込みによってリアルタイムスケジューラの時を刻んでいるため，イーサネットデバイスのIRQはタイマのそれよりも大きく設定しなければならない．

4.3 プリミティブ

RT-Messengerは，以下に示す4つのプリミティブにより実装されている．これらは，RT-Linuxカーネルのローダブルモジュールとして実装されている．

- `int rtm_task_wait(task, rtm_packet)`
  `RT_TASK *task;`
  `rtm_packet **rtm_packet;`
  この関数は，RT-Taskの状態をMESSAGEに遷移させる．このRT-Taskの状態はRT-Messageを受信したときにREADYに遷移し，スケジューリングの対象となる．
- `void rtm_send(dev, packet, ipaddr)`
  `struct device *dev;`
  `struct rtm_packet *packet;`
  `const char *ipaddr;`
  この関数は，RT-Messageの送信を行う．関数が呼ばれるとただちに送信を開始する．
- `void *rtm_alloc(dev, size)`
  `struct device *dev;`
  `int size;`
  この関数は，RT-Messageを作成するためのメモリを割り当てる．
- `void rtm_free(rtm_buffer)`
  `void *rtm_buffer;`
  この関数は，`rtm_alloc()`関数で割り当てられたメモリを解放する．

5. RT-Messengerの評価

本章では，RT-Messengerの性能評価について述べる．測定に用いたコンピュータのCPUはPentium-II 300 [MHz]，Linuxのバージョンは2.0.36，RT-Linuxのバージョンは1.2である．イーサネットのコントローラにはDEC製Tulip-21140Aを用い，2台のコンピュータを10BASE-Tのシェアードハブを介して接続した．より実環境に即した評価をするならば，他のトラフィックが存在する場合の測定を行うべきである．しかし，その場合のコリジョン発生率などの研究はすでに行われている．また，パケットの再送信を行わないRT-Messengerの場合にはパケット欠損時のロバスト制御の安定性を評価することになってしまい，通信機構自体の評価とはならない．そこで本論文では，これら2台のコンピュータだけがハブに接続されており，RT-Message以外のトラフィックが存在しない場合だけを評価対象とした．測定に用いたコンピュータのIRQの設定を**表1**に示す．

表1 性能評価に用いたコンピュータのIRQ
Table 1 IRQs of the computer for experiments.

IRQ	name of interrupt	IRQ	name of interrupt
0	timer	8	rtc
1	keyboard	9	eth0
4	serial	15	aicxxx

測定は，RT-Linuxで用意されているrt_get_time()関数を用い，タイマの値を読み取った．測定精度はこのタイマに入力されるクロック（単位はTick）に依存し，1 [Tick]は，約0.84 [μsec]である．

RT-Messengerの評価は次の方法で行った．2台のうち一方のコンピュータのRT-Taskから1 [msec]ごとにRT-Messageを送信し，他方のコンピュータのRT-Taskでそれら受信する．10 [sec]の間に10,000のRT-Messageの送受信を行った．また，通常のLinuxプロセスによるRT-Messengerの処理時間への影響を調べるために，最も影響があると思われるDMA転送をLinuxプロセスから発生し続けてている場合においても同様の評価を行った．なお，他のトラフィックが存在しないため評価結果には再現性があることが予想されるが，実際に同じ測定を10回試行して再現性があったことを確認してある．

5.1 測定結果

まずRT-Messengerの送信遅延の測定方法について述べる．これは，rtm_send()関数が呼ばれてから，イーサネットデバイスに送信命令を送るまでの時間である．**図4**に測定結果を示す．最大送信遅延は5 [μsec]であった．また，並行してLinuxプロセスからDMA転送を発行させながら同様の測定を行った．この測定結果を**図5**に示す．この場合の最大送信遅延は，8 [μsec]であった．

次にRT-Messengerの受信遅延の測定方法について述べる．これはRT-Messageの受信によるイーサネットデバイスからの割込みによるトップハーフが呼び出されてから，RT-Taskが起動するまでの時間である．**図6**に測定結果を示す．最大受信遅延は35 [μsec]で

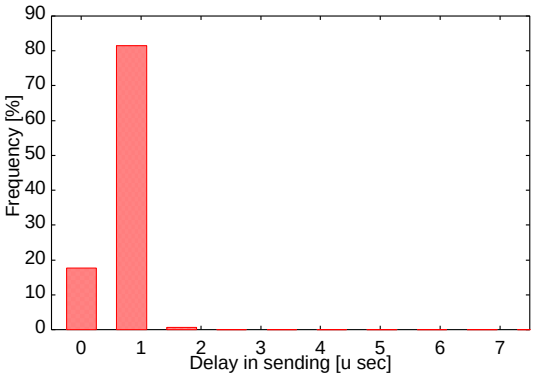

図 4 送信遅延
Fig. 4 Delay in sending.

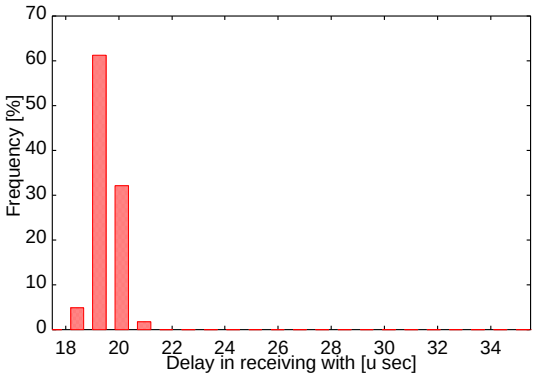

図 6 受信遅延
Fig. 6 Delay in receiving.

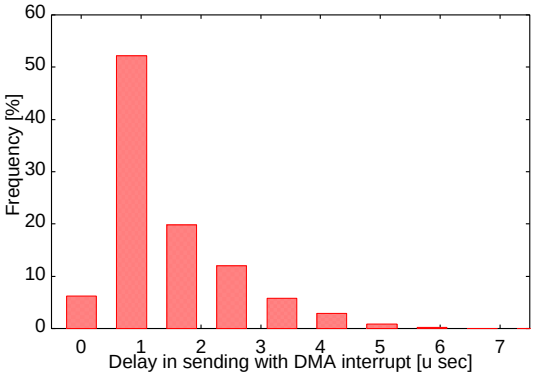

図 5 割込み発生時の送信遅延
Fig. 5 Delay in sending with DMA interrupts.

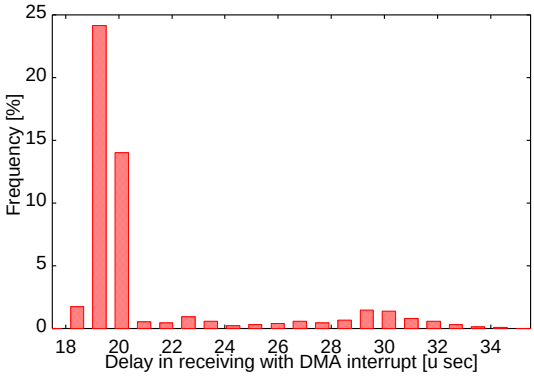

図 7 割込み発生時の受信遅延
Fig. 7 Delay in receiving with DMA interrupts.

あった．次に，送信遅延と同様に，通常の Linux プロセスから DMA 転送を発行させながら測定した場合の結果を**図 7** に示す．この場合の最大受信遅延は，120 [μsec] であった．

5.2 遅延発生原因の検討

まず，送信における遅延について検討する．RT-Messenger の処理中，すなわち RT-Task が実行中の CPU は割込み禁止状態であるため，デバイスからの割込みによる影響を受けないはずである．しかし，DMA 転送が発生しているとき（図 5）は，発生していないとき（図 4）に比べて若干の遅延が見受けられる．この理由は，プロセッサのコンテキストが切り替わっても，DMA コントローラによる DMA はすぐには停止しないため，DMA と RT-Task とでバスの競合が発生し，見かけのプロセッサ性能が低下するために RT-Messenger の処理時間が延びるものと考えられる．

次に，受信における遅延について検討する．受信では，Linux プロセスによる DMA 転送の発生により，大きな遅延が発生している（図 6，図 7）．これは，RT-Messenger の受信処理がボトムハーフの処理を経由していることに起因する．ボトムハーフのコードは割込み許可の状態で実行されるため，DMA コントローラから割込みが発生すると，その RT-Messenger のボトムハーフ処理に割り込んで DMA 処理のトップハーフが動作してしまう．そのため，受信時においては DMA 転送の影響が送信時より大きくなっている．

6. 新しいメディアの実現

RT-Linux による制御と RT-Messenger による通信を用いることによって，遠隔地との力学的インタラクションを実現する新しいメディアを実現した．**図 8** はその概観である．2 台のマニピュレータはそれぞれ 3 自由度のアームを持っており，これらを 1 [msec] のサンプリング周期で制御する[10]．ここで，2 台のコンピュータそれぞれに誤差を持つクロックが実装されていることによる問題が存在する．タイマには水晶体によって発生させるクロックが入力される．しかし，水晶体は工業的に同一のクロックを発生するものを作ることは困難であることに加え，クロックの精度は利用環境にも著しく影響されるので，それぞれのコンピュー

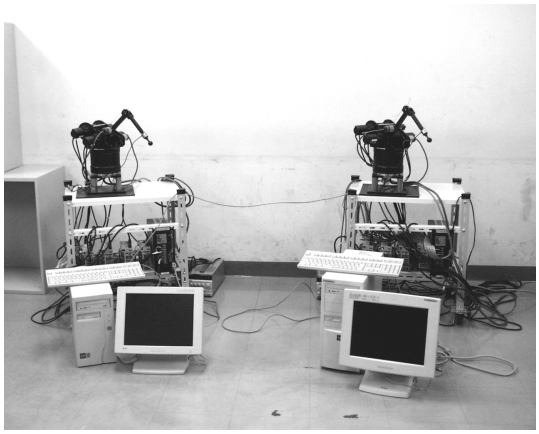

図 8 提案するメディアの概観
Fig. 8 The overview of the proposed media.

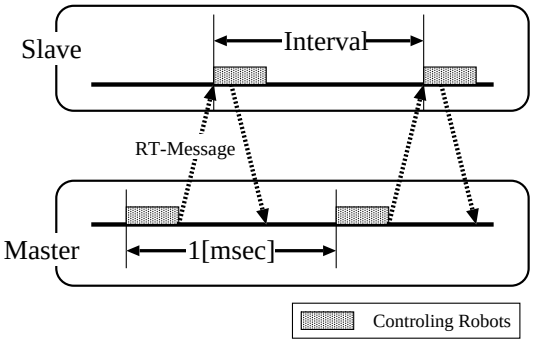

図 9 2 つのコントローラのタイミングチャート
Fig. 9 Timing chart of two controllers.

タで 1 [msec] を刻み続けると，数分ごとに 1 周期程度の頻度で位相差が生じてしまう．

これを解決するために，本実装では時間管理をマスタとスレーブとで共有する方法を用いた．これは，スレーブの時間管理をマスタ側に委ねる方法である（**図 9** 参照）．マスタ側では RT-Scheduler によって 1 [msec] 周期の RT-Task を実行し，各周期の終了時にマスタ側の情報を RT-Messenger を用いてスレーブに送信する．スレーブ側ではマスタからの RT-Message の受信をトリガとして RT-Task を開始し，1 周期分の処理の終了時にスレーブの情報をマスタ側へ送信する．この方法だけではパケットの欠損がスレーブ側の周期停止を引き起こす．そこで，用いた制御系から算出した安定限界よりも小さい揺れ幅として制御周期の ±10 [%] を周期の揺れ幅の目標値と設定し，1.1 [msec] にタイムアウトを設定することにより制御の安定継続性を確保している．

この制御方法において，スレーブ側の起動間隔を測定した．マスタとスレーブの両方から 3 軸の力と

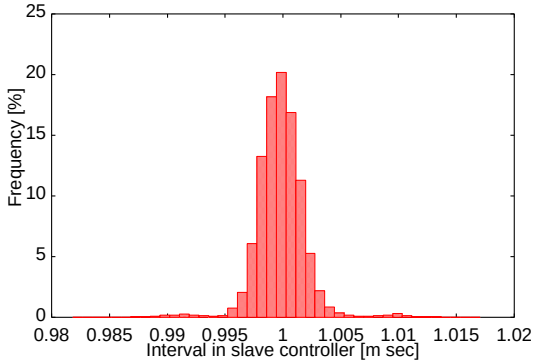

図 10 スレーブ側の制御間隔
Fig. 10 Interval of slave controller.

位置の情報を送っており，RT-Message の大きさは 48 [Byte] である．**図 10** に測定結果を示す．最長の間隔は 1.015 [msec]，最短では 0.985 [msec] の間隔となっており，制御の揺れ幅は 1.5 [%] 以内に収まっている．この揺れ幅は設定したタイムアウトよりも十分に短いため，パケットの欠損は発生せず，結果として安定した制御を継続することができた．

以上から，リアルタイム性を持たない通信媒体を用いてもリアルタイム処理内での通信を可能にする新しい通信機構 RT-Messenger の提案の有効性が示された．また RT-Linux 上へ実装した RT-Messenger を用いることにより，制御周期 1 [msec] で安定限界が ±100 [μsec] という実時間制約の厳しいマルチメディアを，リアルタイム性のないネットワーク経由で安定に実現できることが示された．新しいメディアの性能については本論文の範囲を越えるが，先端に設置されている力学センサの分解能は 49 [mN] であり，利用者はスポンジ表面の柔らかさや机表面の硬さなどを感じることができている．

7. おわりに

本論文では，遠隔地との力学的インタラクションを実現する新しいメディアについて述べた．この実現には，リアルタイム OS と厳しい時間制約を満足する通信機構が必要である．そこで厳しい時間制約を満足する通信機構として RT-Messenger を提案し，リアルタイム OS RT-Linux 上に実装した．RT-Messenger は，リアルタイム性を持たない通信媒体を用いてもリアルタイム処理内での通信を可能にする画期的な通信機構である．

提案する機構の有効性を確認するために，サンプリング周期ごとに通信を行う分散制御システムを実装し，性能評価を行った．その結果，低速な 10BaseT のイー

サネットを用いても 1 [msec] 周期で制御を行うことが可能であることが示され，しかも制御周期の揺れ幅が ±1.5 [%] 程度に収まることが分かった．この揺れ幅は制御系の安定限界から求めた目標値である ±10 [%] に比べて十分に小さな値であり，ネットワークを介して安定した分散制御を実現することができた．以上から，リアルタイム性を持たない通信媒体を用いてもリアルタイム処理内での通信が可能であることが示された．

現在，受信処理時間が他の割込みに影響されにくくするためにボトムハーフを経由しない実装へ変更し，また RT-Linux Version2.x への対応，および通信系での同期機構の開発を手がけている．

謝辞 本メディアのマニピュレータ部および制御系に関しては慶應義塾大学理工学部システムデザイン工学科大西研究室の協力を得た．大西公平教授および同研究室の学生諸氏に感謝の意を表したい．

参 考 文 献

1) 橋本浩二，柴田義孝，白鳥則郎：利用者環境の違いと QoS 保証を考慮したマルチメディア会議システム，情報処理学会研究報告マルチメディア通信と分散処理，Vol.94, No.28, pp.155–160 (1999).
2) 高坂幸春，宮川明大，橋本浩二，柴田義孝：拡張仮想空間の伝統工芸への応用，情報処理学会研究報告マルチメディア通信と分散処理，Vol.94, No.10, pp.53–58 (1999).
3) 橋詰博行，藤原一夫：21 世紀の病院と手術室，日本ロボット学会誌，Vol.18, No.1, pp.24–28 (2000).
4) 生田幸士：医療ロボティクスへの道，日本ロボット学会誌，Vol.18, No.1, pp.49–52 (2000).
5) 加賀美聡：ロボット研究のための PC/AT 互換機上のリアルタイム OS，日本ロボット学会誌，Vol.16, No.8, pp.1036–1041 (1998).
6) 寺岡文男，塩野崎敦：IPv6 とリアルタイム通信プロトコル，コンピュータソフトウェア，Vol.15, No.2, pp.94–95 (1998).
7) Beck, M., Böhme, H., et al.: Linux Kernel Internals, Addison-Wesley (1996).
8) Yodaiken, V.: *The RTLinux Manifesto*, Department of Computer Science, New Mexico Institute of Technology, Socorro NM87801.
9) Yodaiken, V.: *The RT-Linux approach to hard real-time*, Department of Computer Science, New Mexico Institute of Technology, Socorro NM87801.
10) Hayashida, N., Yakoh, T., Murakami, T. and Ohnishi, K.: A Friction Compensation in Twin Drive System, *6th International Workshop on Advanced Motion Control*, pp.187–192 (2000).

(平成 12 年 5 月 15 日受付)
(平成 12 年 12 月 1 日採録)

佐藤 秀雄

1977 年生．2000 年 3 月慶應義塾大学理工学部システムデザイン工学科卒業．リアルタイム通信に関する研究に従事．現在，慶應義塾大学大学院理工学研究科総合デザイン工学専攻前期博士課程在学中．電気学会会員．

矢向 高弘（正会員）

1994 年 3 月慶應義塾大学大学院理工学研究科計算機科学専攻博士課程修了．同年 NKK 入社，1996 年慶應義塾大学メディアネット本部を経て 1998 年 4 月より慶應義塾大学理工学部システムデザイン工学科助手．工学博士．日本ロボット学会，電気学会，IEEE 等会員．