B-023

送信バッファの制御による優先帯域制御ミドルウエア

Priority Bandwidth Control Middleware by Control of Transmission Buffer

中山 悟† 中野美由紀‡ 寒竹 俊之† 長島 聡志† 菅谷 みどり†
SATORU NAKAYAMA† MIYUKI NAKANO‡ TOSHIYUKI KANTAKE† SATOSHI NAGASHIMA† MIDORI SUGAYA†

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サービスが検討されている．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，複数台のロボットを連携したサービスは様々な場面での活用が期待される．

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．

例えば，介護施設に適用する高齢者支援システムでは[2,3]，高齢者支援システムは複数台の移動体ロボットとサーバにより構成されることが想定され，ロボットは見守りやリハビリ支援の機能を持つことを想定している．それぞれのロボットは支援機能に応じてセンサや映像などのデータを収集し，非同期でサーバに転送することで，対象者の支援を実現する．高齢者支援システムでは機能や送信時の状況，送信するデータの内容によって通信要求は異なる．また，どのデータにも継続的な通信理由があることから，同時に送信するデータの強制的な停止を伴わずに緊急のデータを即座にサーバに送信することが重要であり，この場合の緊急データの帯域確保が必要である．

本研究では，ネットワーク負荷によらず，緊急データの送信に対して帯域確保を行うことを目的とする．目的を実現するために，ネットワーク負荷に応じたアプリケーションごとの資源量計算と，OS レベルでプロセスごとの送信バッファ制御を動的に行うミドルウエアを提案する．本ミドルウエアは，モニタ，アナライザ，コントローラの 3 つのモジュールからなる．本研究では，プロセスから情報を収集するモニタと，帯域幅分析および制御値決定アルゴリズムにより値を決定するアナライザ，決定した値をもとに OS 機能である Cgroups を通じて，CPU 時間やメモリなどの単一のリソースを指すサブシステムに接続することで，プロセスごとの資源制御を行う仕組みを提案する．シュミレーションツールを開発し，評価を行ったところ，平均値の誤差 4%の範囲で目標精度を達成していることがわかった．本論文では，2 節にて既存研究を比較して，3 節にてミドルウエアの設計，4 節にて実装，5 節にて評価，6 節にてまとめと今後の課題とした．

†芝浦工業大学 Shibaura Institute of Technology
‡産業技術大学院大学 Advanced Institute of Industrial Technology

2. 既存研究

特定のデータ転送時の帯域確保を考えた場合，データ転送の帯域確保を扱う技術として QoS（Quality of Service）制御がある．QoS 制御は特定の通信に対し優先制御や帯域制御を行うことでサービスの品質を保証する．QoS 制御においてデータの通信要求に応じた帯域を制御する研究は数多くなされている．無線通信を対象とした帯域制御に関する研究として，ネットワーク上のアクセスポイントなどデータ転送時の通過点を利用する研究と，データ転送を行うマシン自体で帯域制御する研究が存在する．無線 LAN 規格の 1 つである IEEE 802.11e（以下，802.11e とする）では，ユーザの QoS 要求を満たすためにハイブリッドコーディネーション機能（以下，HCF とする）を提供している．HCF では 4 種類の優先度のアクセスカテゴリを用いて，QoS 要件を有するアプリケーションを動作させているノードに対して送信権を与えることで動的な帯域幅割り当てを行い，QoS を保証する．しかし，帯域幅割り当ては静的な値を用いており，メディアなどのバースト性のあるデータ転送には向かない問題がある．そこで Boggia らはフィードバック型動的スケジューラを提案し，データの送信機会を調整することで遅延の保証をしている[4]．

石川らは，HTTP プロトコルを使用した WEB 閲覧に焦点を当て，ユーザがリクエスト送信後，それに対するレスポンス受信までを 1 つのフローとして扱っている．レスポンスタイムによりユーザの満足度が著しく低下することから，無線 LAN 環境におけるフロー数増大時の各ユーザの通信時間の保証を目的とし，ネットワーク混雑時においてもレスポンスタイムが閾値以下とするために，フロー間優先制御方式を提案している[5]．802.11e とフロー間優先制御方式では，アクセスポイントを通過する際に制御を行うことから，マシンからネットワークにデータを送出する必要がある．そのためマシンから送出する際の，データの送信を行うマシン内部のアプリケーションとネットワーク間の処理

時間は考慮されておらず，個々のロボットで，アプリケーションごとの緊急時のデータ送信の帯域制御を行いたい場合適さない問題がある．毛利らは，ワイヤレス環境におけるアプリケーションに対する QoS を保証することを 1 つの目的として，リアルタイム OS を基にした次世代ワイヤレス通信システムを提案している[6]．

リアルタイム OS は，アプリケーションの実行時間の保証や計算機資源の確保に長けている特徴があり，毛利らは更にアプリケーションの動的な状態の変化によらない QoS を行うための機構を考案している．近年のロボットアプリケーション開発では，汎用 OS にミドルウエアをインストールし，その上でアプリケーションを動作させる構成が一般的である．次世代ワイヤレス通信システムはタスク単位でデータ送信を保証するメリットがある一方で，リアルタイム OS を基に設計されたため一般的なロボット制御を行う際には汎用性が低い問題がある．汎用 OS 上でパケットスケジューラを用いたアプリケーションごとの資源制御を行う仕組みに Control Groups（以下，cgroups とする）が提供されている[7]．

Cgroups はカーネルの機能で，ユーザが CPU 時間やシステムメモリ，ネットワーク帯域幅などの資源量をシステム上で実行中のプロセスに対して割り当てることが可能である．Cgroups では CPU 時間やメモリなどの単一のリソースを指すサブシステムに接続することで，プロセスごとの資源制御を行う．cgroups ではプロセスごとの資源量制御ができるが，ユーザが適宜資源を割り当てる，または事前にコンフィグを作成する，といった静的な制御が基本である．そのため，データ送信時のクライアントマシンの帯域利用時の状況に応じた動的な QoS 制御を行う機構は提供されていない．

3. 帯域制御のミドルウエアの設計

3.1 目的

本研究では，ネットワーク負荷によらず，緊急データの送信に対して帯域確保を行うことを目的とする．目的を実現するために，ネットワーク負荷に応じたアプリケーションごとの資源量計算と，OS レベルでプロセスごとの送信バッファ制御を動的に行うミドルウエアを提案する．提案ミドルウエアでは，クライアントが送信するデータに対して，予めサービスに応じて付与された優先度で通信制御を行う機構を提供する．サービスはアプリケーションごとに提供され，OS 上ではプロセスとして実行されることから，本研究ではプロセス単位でサービスを扱う．また，優先度は高と低の 2 種類で扱い，緊急のデータは高優先度，その他は低優先度とする．

実現方法として，高優先度プロセスがデータ送信後に，サーバから返答を受け取るまでのレスポンスタイムを一定時間とするための帯域制御をプロセス単位で行う．まずネットワーク負荷による影響を含めるため，実測のレスポンスタイムと目標のレスポンスタイムのそれぞれを帯域幅に換算し，実測の帯域幅と目標の帯域幅との誤差を計算する．そして，高優先度プロセスに対して，誤差に応じた帯域幅を動的に計算し，割り当てることで，高優先度データの送信に対する帯域確保を可能とする．提案システムでは実測の帯域幅と目標の帯域幅との誤差を計算するモニタと，誤差を修正し帯域幅を計算するアナライザ，パケットスケジューラを利用して帯域幅を制御するコントローラの 3 つのモジュールにより構成する．

3.2 設計

クライアントに適用した場合のシステム構成図を図 1 に示す．モニタは高優先度プロセスから実測のレスポンスタイムとしてラウンドトリップタイム（以下，RTT とする）を取得し，目標のレスポンスタイムとの誤差を計算する．アナライザは誤差を修正する帯域幅を計算し，コントローラはその計算結果の帯域幅の適用をパケットスケジューラに要求する．

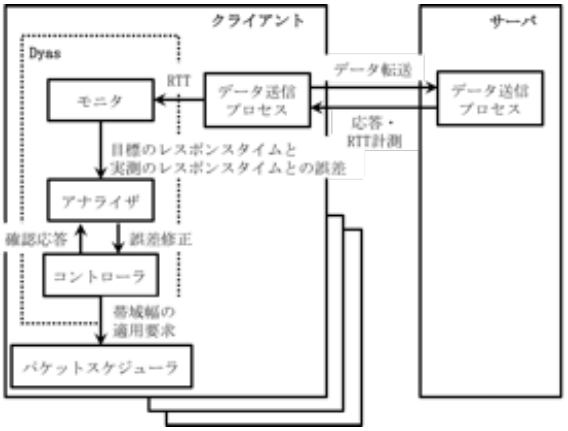

図 1　ミドルウエア構成

提案システムでは複数台のクライアントが同一のネットワーク上で動作する環境で，ネットワークの負荷軽減のためにクライアント 1 台あたりが使用可能な帯域幅が制限されることを想定した．また，クライアント 1 台あたりの使用可能な帯域幅に対して送受信している全体のデータサイズの割合を帯域利用率とする．送受信しているデータサイズの計測にはマシンのアクティビティを監視するユーティリティである System Admin Reporter（以下，sar とする）を使用する．

以降でそれぞれのモジュールについて詳細に説明する．

3.3 モジュール構成

3.3.1 アナライザ

アナライザの処理手順を図 2 に示した．

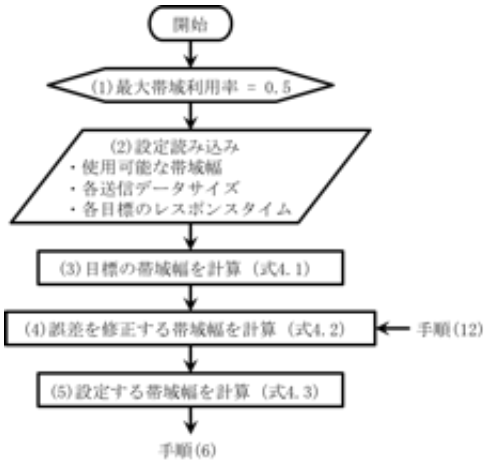

図 2　アナライザーの処理

図に従って手順を述べる.

(1)　最大帯域利用率の初期値を設定する．最大帯域利用率は起動後のデータ送信時の帯域利用率の最大値とする.

(2)　ユーザが設定した，クライアント 1 台あたりが使用可能な帯域幅と高優先度プロセスの送信するデータサイズ，目標のレスポンスタイムを読み込む.

(3)　式 4.1 から，高優先度プロセスの目標の帯域幅を計算する．ここでプロセス i に対して req_bw は目標の帯域幅，size は送信データサイズ，req_rtt は目標のレスポンスタイムを表す.

$$req\_bw_i = size_i \,/\, req\_rtt_i \qquad (4.1)$$

(4)　式 1.2 から，実測の帯域幅と目標の帯域幅との誤差を修正する帯域幅を計算する．ここでプロセス i に対して crt_bw は誤差を修正する帯域幅，req_rtt は目標のレスポンスタイム，max_rate は最大帯域利用率，error は誤差を修正する変数を表す．計算方法は後述のモニタの処理で説明する．また，初回は誤差を考慮しない.

$$crt\_bw_i = size_i \,/\, req\_rtt_i * (1 - max\_rate) * error_i \quad (4.2)$$

(5)　式 4.3 から，高優先度プロセスの帯域幅の合計が使用可能な帯域幅以下となるよう，誤差を修正する帯域幅に一定の倍率を乗算し，設定する帯域幅とする．ここで，プロセス i に対して proc_bw は設定する帯域幅，crt_bw は誤差を修正する帯域幅，x は倍率，avail_bw は使用可能な帯域幅，n は高優先度のプロセス数を表す.

$$proc\_bw_i = crt\_bw_i * x \quad \text{ただし，} \; avail\_bw \geq \sum_n proc\_bw_n \quad (4.3)$$

3.3.2 コントローラ

図 3 にコントローラの処理を示し，図中の番号順に詳細を述べる.

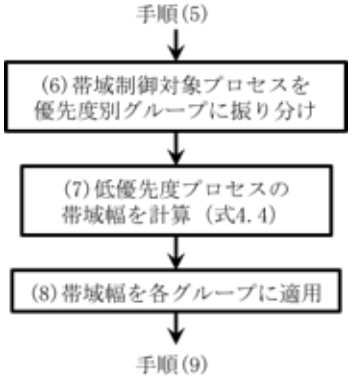

図 3　コントローラの処理

(6)　高・低のそれぞれで異なるグループを作成し，高優先度プロセスとその他のプロセスを各グループに振り分ける.

(7)　式 4.4 から，使用可能な帯域幅から高優先度プロセスの設定する帯域幅の合計を差し引いて，低優先度プロセスの帯域幅を計算する．ここで rem_bw は低優先度プロセスの帯域幅，avail_bw は使用可能な帯域幅，n は高優先度のプロセス数，proc_bw は高優先度プロセスの設定する帯域幅を表す.

$$rem\_bw = avail\_bw - \sum_n proc\_bw_n \quad (4.4)$$

(8)　アナライザの手順(5)とコントローラの手順(7)で計算した，高優先度プロセスの設定する帯域幅と低優先度プロセスの帯域幅を，各グループに適用し帯域制限する.

3.3.3 モニタ

図 4 にモニタの処理を示した.また，図中の番号順に詳細を述べる.

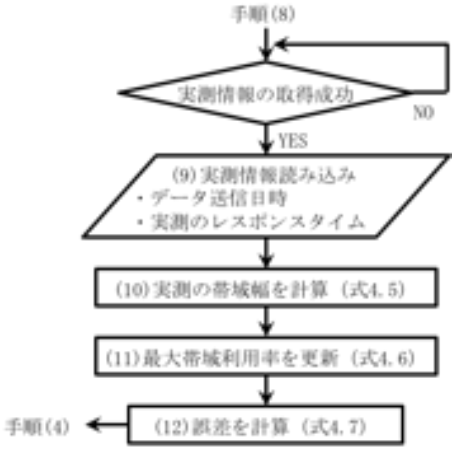

図 4　モニタの処理

手順(8)はコントローラの処理である.

(9)　高優先度プロセスがデータ送信を完了した後，データ送信日時と実測のレスポンスタイムを読み込む.

(10) 式 4.5 から，実測の帯域幅を計算する．ここで i 番目のプロセスに対して real_bw は実測の帯域幅，size は送信データサイズ，real_rtt は実測のレスポンスタイムを表す．

$$real\_bw_i = size_i \,/\, real\_rtt_i \quad (4.5)$$

(11) 最大帯域利用率を，本ミドルウエア起動後のデータ送信時の帯域利用率が上回った場合に更新する．帯域利用率は式 4.6 からクライアント 1 台あたりの帯域幅に対し送受信しているデータサイズの割合を計算する．ここでプロセス i に対して rate は帯域利用率，flow_pkt はデータ送信時の送受信しているデータサイズの合計，avail_bw は使用可能な帯域幅を表している．

$$rate_i = flow\_pkt_i \,/\, avail\_bw \quad (4.6)$$

(12) 式 4.7 から，前回の誤差に目標の帯域幅と実測の帯域幅との誤差を乗算し，新たな誤差補正変数 error’を求めることで前回の誤差を考慮した誤差の修正を行う．ここでプロセス i に対して error は誤差を修正する変数，req_bw は目標の帯域幅，real_bw は実測の帯域幅を表す．

$$error_i' = error_i * (req\_bw_i \,/\, real\_bw_i) \quad (4.7)$$

以降は手順(4)のアナライザの処理から同様に行う．

4. 帯域制御実装

4.1 特定のプロセスの帯域制御

帯域制御は cgroups と Traffic Control（以下，tc とする）を使用して行う．帯域制御の仕組みを図 5 に示す．

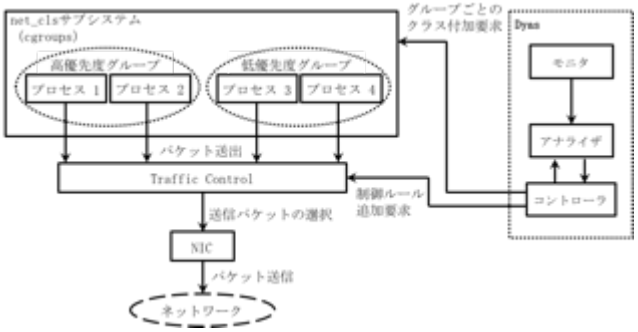

図 5　帯域制御システム構成

cgroups では複数のプロセスを 1 つのグループとし，サブシステムごとのリソースの動的割り当てが可能である．帯域制御は cgroups の net_cls サブシステムを用いてクラス ID をグループに付加し，tc が識別，制御することで実現する．本システムではコントローラが cgroups と tc また，現在の実装では高優先度以外のプロセスとして，システム自体は低優先度のグループに含む．

4.2 Traffic Control

Tc[7]は Linux カーネルでトラフィック制御を行うユーティリティである．トラフィック制御の要素として，パケットの出力時に行うシェーピングとスケジューリング，入力時に行うポリシング，そして入出力の双方で行われるドロッピングがある．トラフィックの処理は qdisc（queuing discipline），クラス，フィルタの 3 つのオブジェクトで行う．qdisc はカーネルが送出するパケットを一時的に格納するキューである．qdisc はネットワークインタフェースに応じて設定する．qdisc にはクラスを付加することが可能で，クラス内に更に qdisc を設定できる．特定のパケットを優先的に送出する場合はクラスによりデキューする順序を変える．また，パケットの分類はフィルタで行うものとした．

5. 評価

5.1 方法

評価に先立ち，単一のマシンで優先度の異なる複数のプロセスを動作させてデータ送信を行う環境を想定し，ネットワーク負荷をかけた場合の動作確認を目的として，シミュレーションツールおよび視覚化ツールを設計，実装し，実験を行った．

表 1　シミュレーション実験の転送データ

送信データ	データサイズ [byte]	目標のレスポンスタイム[sec]	優先度
高優先度データ	512	0.1	高
低優先度データ 1	1500	1.0	低
低優先度データ 2	231000	1.0	低

本実験ではネットワークを介したデータ送信は行わず設定した帯域幅で指定したデータサイズを送信するときの RTT を理論値で計算した．クライアント 1 台が使用可能な帯域幅を 300Mbps とし，送信するデータを**表 1** の通り設定した．低優先度データ 2 は QVGA 動画の 1 フレーム分のデータサイズとした(表 1)．

5.2 シュミレーションツールの設計と実装

提案システムでは，ネットワーク負荷の動的な変化に応じて，データ送信を行うプロセスのレスポンスタイムを制御する．そのため視覚化ツールを開発することにより経過時間ごとの帯域制御の動作を直感的に確認することが可能である．

Gnuplot は Linux や OSX，Windows などの様々な OS をサポートするグラフ描画ユーティリティである．描画は X11 や AquaTerm などのターミナルを使って行う．

5.2 シュミレーション結果

優先度ごとのレスポンスタイムの平均，標準偏差，最悪値を**表 2** に示す．レスポンスタイムの平均はいずれの優先度も目標のレスポンスタイムに対して 5%以内の範囲であった．標準偏差は高優先度プロセスが最も小さく，高い精度を示した．

ネットワーク負荷を動的に変化させた時の経過時間ごとのシミュレーション結果を図 7 に示す．第 1 縦軸はレスポンスタイム，第 2 縦軸はネットワーク負荷，横軸は経過時間を示している．表 2 のデータの内，緊急データを赤色，その他の通常データを青色で示している．

表 2　優先度ごとのレスポンスタイム比較

	目標のレスポンスタイム[sec]	平均[sec]	標準偏差	最悪値[sec]
高優先度	0.1	0.104	0.020	0.132
低優先度 1	1	1.032	0.057	1.102
低優先度 2	1	1.044	0.120	1.274

また，直線は目標レスポンスタイム，X 印は実測レスポンスタイム，四角印は設定帯域幅をレスポンスタイムに換算した値である．シミュレーション開始後の初回とネットワーク負荷が 0%の時，途中ネットワーク負荷を最大約 20%かけた時，最後，再びネットワーク負荷を 0%に下げた時の帯域制御の結果である．

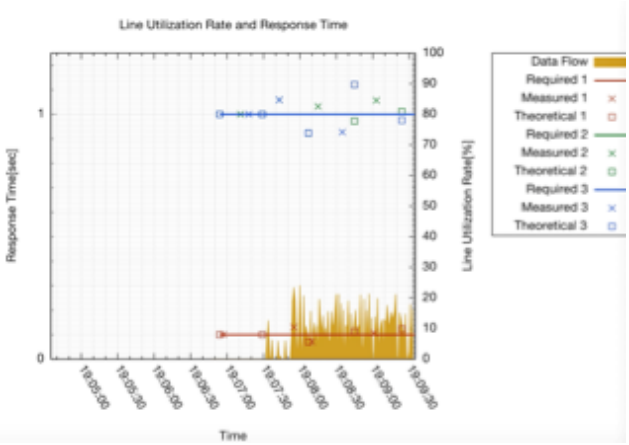

図 7　レスポンスタイムの計測区間

ネットワーク負荷がない時は目標レスポンスタイムと等しい値の実測レスポンスタイムが得られたが，ネットワーク負荷をかけた時は目標のレスポンスタイムに対して実測のレスポンスタイムの誤差が大きくなり，ネットワーク負荷を減少させた時は徐々に目標のレスポンスタイムと実測のレスポンスタイムの誤差が小さくなる傾向がみられた．

6. 結論

本研究では，ネットワーク負荷によらず，緊急データの送信に対して帯域確保を行うことを目的としアプリケーションごとの資源量計算と，0S レベルでプロセスごとの送信バッファ制御を動的に行うミドルウエアを提案した．シュミレーションにおいては，ネットワーク負荷を減少させた時は徐々に目標のレスポンスタイムと実測のレスポンスタイムの誤差が小さくなる傾向がみられた．ネットワーク負荷をかけた時に誤差が大きくなる問題については，前回の誤差を考慮せずに新たに誤差計算を行っていることなどが考えられる，単純な誤差ではなく，相対誤差を用いたことも影響した可能性があることから，今後，これらを解決する．

謝辞

本研究は JSPS 科研費 15K00105 の助成を受けて実現したものです．ここに厚く感謝申し上げます．

参考文献

[1] 製品情報 | Pepper（一般販売モデル） | ロボット | ソフトバンク，SoftBank，http://www.softbank.jp/robot/consumer/products/
[2] 住谷拓馬．“IXM:ロボット制御ソフトウエア向けプロセス間通信ミドルウエア”，芝浦工業大学 2015 年度修士論文，2016.
[3] 岡崎純己，保科篤志，池田悠平，菅谷みどり，歩幅推定によるリハビリテーション促進ロボットの実現,研究報告高齢社会デザイン（ASD）,2016-ASD-5(12),1-8 (2016-07-28)
[4] Gennaro Boggia， Pietro Camarda， Luigi Alfredo Grieco and Saverio Mascolo．Feedback-Based Control for Providing Real-Time Services With the 802.11e MAC ， IEEE ・ ACM TRANSACTIONS ON NETWORKING， Vol.15， No.2， pp.323-333，April 2007.
[5] 石川圭也，妙中雄三，中山雅哉．“レスポンスタイムを一定時間内とするための帯域使用率に基づくフロー間優先制御方式の提案と評価”．電子情報通信学会技術研究報告 SITE 技術と社会・倫理，Vol.112，No.488，pp.127-132，2013.
[6] 毛利公一，前田忠彦，大久保英嗣．“次世代ワイヤレス通信を指向するオペレーティングシステムの提案”．情報処理学会研究報告システムソフトウェアとオペレーティング・システム（OS），Vol.2003，No.19，pp.107-114，2003.
[7] Chapter 1. Introduction to Control Groups (Cgroups)， Red Hat，https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch01.html