Dyas: データ転送の動的帯域制御を行うミドルウエアの提案

中山 悟[†]　長島聡志[†]　中野美由紀[‡]　寒竹 俊之[†]　菅谷 みどり[†]

概要：ロボットがネットワークにつながる時代では，インタラクティブサービスを支援するシステムにおける緊急のデータの帯域確保が重要な課題となる．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ミドルウエア Dyas（DYnamic bAndwidth control System）を提案する. Dyas 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いアプリケーションごとの応答性を確保する．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制御することで，目的を達成する．本提案ミドルウエアがない場合と比較して，レスポンスタイムは平均 37%,負荷時では約 63%減少した．また， 90%の負荷をかけた場合でもほぼ 100%帯域を確保し，有効性を示した．

キーワード：ネットワーク帯域確保, QoS, 送信バッファ制御，ミドルウエア，Dyas

Dyas: Sending Buffer Management Middleware for Dynamic Bandwidth Control to Improve Response Time

SATORU NAKAYAMA[†]　SATOSHI NAGASHIMA[†]
YUKI NAKANO[‡]　TOSHIYUKI KANTAKE[†]　MIDORI SUGAYA[†]

Abstract: In the era that robot is connected to the network, securing the bandwidth of urgent data in a system supporting interactive service is an important subject. In this research, we propose middleware Dyas (Dynamic bAndwidth control System) which secures bandwidth for emergency data transmission regardless of network load.Dyas is a subsystem that refers to a single resource such as CPU time and memory , Thereby controlling resources for each process and ensuring responsiveness for each application. The purpose is achieved by dynamically controlling the resource amount calculation for each application according to the load and the transmission buffer control for each process at the OS level. In comparison with the case without this proposed middleware, the response time decreased by an average of 37% and at the time of loading by about 63% Also, almost 100% bandwidth was secured even when 90% load was applied, Indicated.

Keywords: Network buffer management, QoS, Middleware, Dyas

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. そのため, ロボットがデータを効率よく収集し， 次の行動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]， 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．この時, 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．特に， 目の前の高齢者が倒れるなど，緊急時には，同時に送信するデータの強制的な停止を伴わずに即座にサーバに送信するには，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

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

データの送信に対して，アプリケーションごとの帯域確保を行うことを目的とする．

目的の実現にあたり，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 Dyas（DYnamic bAndwidth control System）を提案する．Dyas 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の設計，実装を行った．

シュミレーションツールを開発し，精度を調査した結果，レスポンスタイムは本提案ミドルウエアがない場合と比較して,平均 37%,負荷時では約 63%減少した.また,複数台のクライアントで評価した結果,90%の負荷をかけた場合でもほぼ 100%帯域保証を行うことができ，本提案の有効性を示した．

本論文の構成は，第 2 節にて既存研究，第 3 節にて提案の設計，4 節にて実装，5 節にてシュミレーション，6 節にて評価を行った結果について述べる．

2. 参考文献

特定のデータ転送時の帯域確保の一つとして，データ転送の帯域確保を扱う 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では，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，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．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 Dyas（DYnamic bAndwidth control System）を提案する．Dyas 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．

Dyas は，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．Dyas 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

Dyas をクライアントに適用した場合のシステム構成図を図 19 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．

Dyas 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．

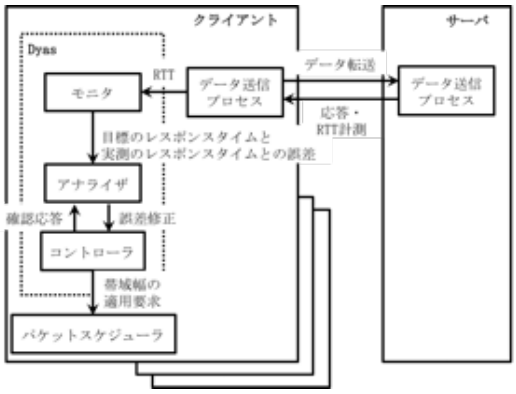

図 1　Dyas システム構成

Figure 1　Configuration of reference and chart number.

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．(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 3.1 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．

$$req\_bw_i = size_i \,/\, req\_rtt_i \qquad (3.1)$$

(4)式 3.2 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．誤差はモニタの手順(12)で計算するが，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．

$crt\_bw_i = size_i / req\_rtt_i * (1 - max\_rate) * error_i$ (3.2)

(5)式 3.3 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．

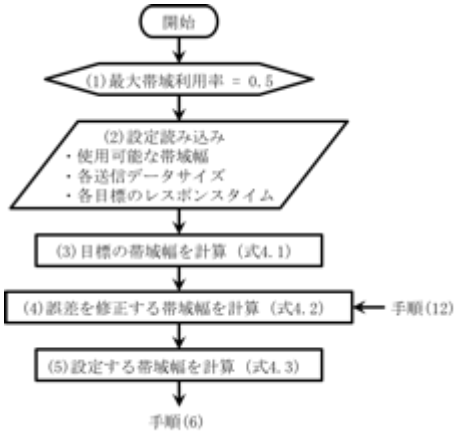

図 2　アナライザの処理
Figure 2　Flow Chart of Analyzer

$proc\_bw_i = crt\_bw_i * x$,
ただし，$avail\_bw \geq \sum_n proc\_bw_n$ (3.3)

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．(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．

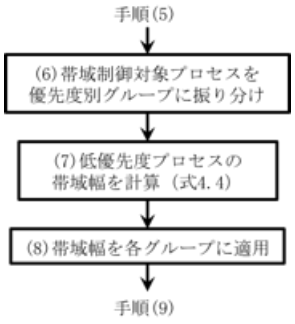

図 3　コントローラの処理
Figure 3　Flow Chart of Controller

$rem\_bw = avail\_bw - \sum_n proc\_bw_n$ (3.4)

(7)式 3.4 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．(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．

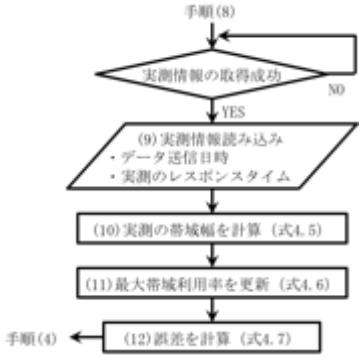

図 4　モニタの処理
Figure 4　Flow Chart of Monitor

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 3.5 から，実測の帯域幅を計算する．ここでプロセス i 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．

$real\_bw_i = size_i / real\_rtt_i$ (3.5)

(11)　最大帯域利用率を，Dyas 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 3.6 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．

$rate_i = flow\_pkt_i / avail\_bw$ (3.6)

(12)　式 3.7 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．

$$error_i' = error_i * (req\_bw_i / real\_bw_i) \quad (3.7)$$

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

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．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．Dyas 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，Dyas は低優先度のグループに含む．

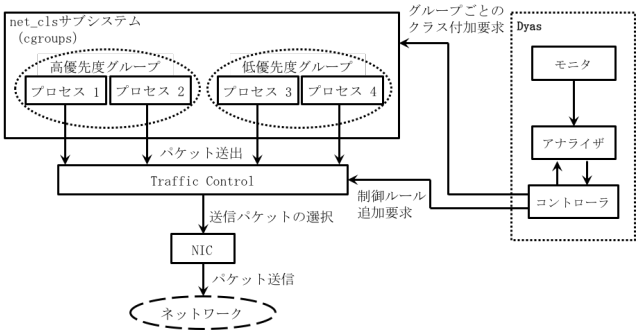

図 5　帯域制御の仕組み

Figure 5　Organization of the Dyas Traffic Control

4.2 Traffic Control

Tc[14]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. シュミレーション

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 Dyas の動作を確認することを目的として，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

Table 1　Simulation Experiment for transferring data

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	512	0.1	高
低優先度 1	1500	1.0	低
低優先度 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，Dyas で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 5.2 シュミレーションツールの設計と実装

Dyas 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 Dyas による帯域制御の動作を直感的に確認することが可能である．クライアントサーバモデルの通信では全てのクライアントの情報を収集可能なサーバと，データを送信するクライアントでのグラフ描画方法の 2 種類が考えられる．

Web ブラウザの対応のみが使用条件となり，様々なプラットフォームで使用可能なメリットがある．しかし，グラフにプロットするデータは JSON 形式で記述しなければならず，Dyas に統合する場合には JSON 形式に変換する処理が必要となる．

アナライザが計算した設定する帯域幅と低優先度プロセスの帯域幅でデータを送信するときのレスポンスタイムの理論値を計算するモジュールを作成した．図 6 に実測のレスポンスタイム計算モ

ジュールの処理を示し，図中の番号順に詳細を述べる．

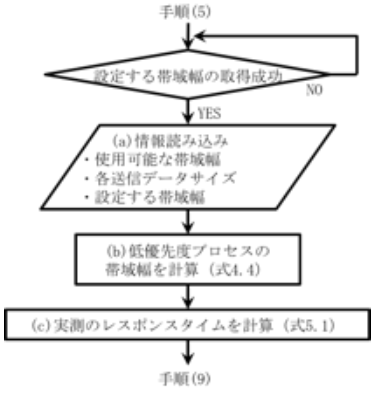

図 6　実測のレスポンスタイム計算モジュール処理
Figure 6　Organization of the Dyas Traffic Control

手順(5)はアナライザの処理である．

(a)　使用可能な帯域幅，高優先度と低優先度のプロセスの送信データサイズ，高優先度プロセスごとの設定する帯域幅を取得する．

(b)　コントローラと同様に式 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き，低優先度プロセスの帯域幅を計算する．

(c)　式 5.1 から，それぞれの帯域幅で指定したデータサイズを転送するときの実測のレスポンスタイムを理論値で計算する．ここでプロセス i に対して，real_rtt は実測のレスポンスタイム，j は経過秒数，size は送信データサイズ，proc_bw は設定する帯域幅，rate は帯域利用率を表す．設定する帯域幅に 1 秒ごとの帯域利用率をかけた，1 秒あたりの送信データサイズを合算し，各プロセスの送信データサイズと等しくなる経過時間を実測のレスポンスタイムとする．1 秒あたりの送信データサイズが各プロセスの送信データサイズを上回る場合は小数点以下を考慮して計算する．

$$real\_rtt_i = j \ ,$$
$$size_i = \sum_j (proc\_bw_i * (1 - rate_j)) \quad (4.1)$$

5.3 相対誤差の計算

式 3.7 に目標の帯域幅と実測の帯域幅を用いる誤差を修正する変数の計算式を示したが，本実験では，次の式から実測のレスポンスタイムと目標のレスポンスタイムの相対誤差を計算する．ここでプロセス i に対して error は誤差を修正する変数，real_rtt は実測のレスポンスタイム，req_rtt は目標のレスポンスタイムである．相対誤差を用いることでアナライザが帯域幅を急激に変更することを抑えることを考えた．

$$error_i = 1 - ((real\_rtt_i - req\_rtt_i) / req\_rtt_i)) \quad (4.2)$$

5.4 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．また，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．

表 2　優先度ごとのレスポンスタイム比較
Table 2　Comparison of response time by priority

	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

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ていた．しかし，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った．負荷を減少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った．

ネットワーク負荷をかけた時の結果は，誤差計算に問題があると考えた．式 4.2 では前回の誤差を考慮せずに新たに誤差計算を行っており，単純な誤差ではなく，その都度得られた相対誤差を用いたことで誤差が拡大したと考えられる．

6. 評価

6.1 概要

複数台マシンの通信環境で，ネットワーク負荷が高い時の高優先度データの送信に対するレスポンスタイムから Dyas の有効性を検証した．また，シミュレーション実験の結果より，式 4.2 の相対誤差に問題があることから，式 3.7 を用いて誤差計算を行った．

6.2 方法

クライアント 3 台とサーバ 1 台を用意し，無線

通信でデータ転送を行った．帯域制御による影響を明確にするため QVGA 動画 1 フレームに相当するデータサイズを転送した．目標レスポンスタイムは 1 秒である．レスポンスタイムは，クライアントがデータの送信を開始後，サーバから返答を受け取るまでの時間を計測した．

また，iPerf3[16]でクライアント 3 台がサーバ 1 台に同時通信をした際の通信速度の最悪値は，約 3Mbps であった．そのため公平性を考え，Dyas 適用の有無によらず予め Tc を用いて各クライアントの使用可能な帯域幅を 3Mbps に設定した．ネットワーク負荷は使用可能な帯域幅に対して 80%とした．

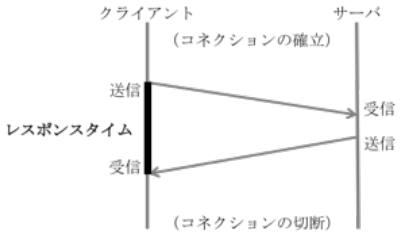

図 7　レスポンスタイムの計測区間

Figure 7　Measurement interval of response time

6.3 レスポンスタイムの比較

3 台のクライアントが同時に 100 回のデータ転送を行った時の優先度が高いデータ転送に対するレスポンスタイムの平均，標準偏差，最悪値について，3 台のクライアントのデータを合算した結果を表 3，およびそのグラフを図 8 に示した．

表 3　レスポンスタイム比較（3 台分合算）

Table 3 Response Time Comparison (Total of 3 units)

	平均[sec]	標準偏差	最悪値 [sec]
通常	1.450	0.284	3.510
通常+負荷	5.184	1.071	13.480
Dyas	1.567	0.269	3.070
Dyas+負荷	1.913	0.742	5.780

負荷をかけない状態のデータ転送では，通常の場合と比べて Dyas を適用した場合は平均のレスポンスタイムが約 9%増加した．これは Dyas の設定帯域幅計算によるものと考えられる．Dyas では帯域幅 500Kbps を高優先度以外のデータ転送プロセスのために確保した上で，高優先度プロセスが使用可能な帯域幅を計算している．500Kbps は SSH などの通信に利用され，実測値に基づいて設定した．

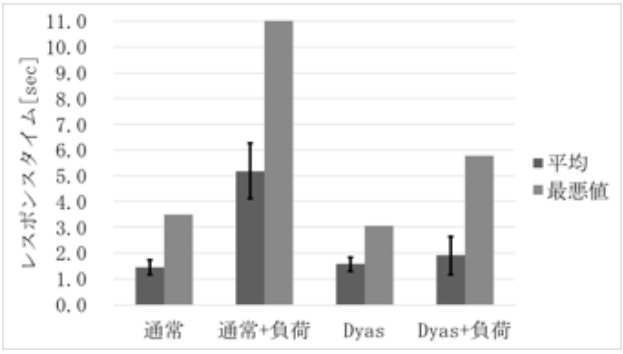

図 8　レスポンスタイム比較（3 台分合算）

Figure 8　Response Time Comparison (Total of 3 units)

これにより，通常のデータ転送プロセスは Dyas のものに比べて 500Kbps の帯域幅を多く使えることとなり，レスポンスタイムに影響したと考えられる．

また，負荷をかけた状態のデータ転送では，通常の場合と比べて Dyas を適用した場合は平均のレスポンスタイムが約 63%減少した．高優先度プロセスのレスポンスタイムを優先的に調整することに加えて，負荷をかけるプロセスが使用可能なデータ転送帯域幅が減少し負荷が低減することでレスポンスタイムが向上した．

レスポンスタイムの累積度数分布を図 9 に示した．0.5 秒ごとのレスポンスタイムで区切り，度数を計算した．実線は負荷をかけずにデータ転送を行った場合，点線は 80%の負荷をかけてデータ転送を行った場合の結果である．

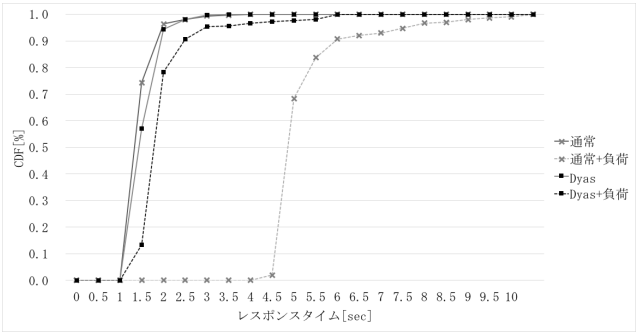

図 9 レスポンスタイム累積度数分布（3 台合算）

Figure 9 Response time cumulative frequency distribution (3 units combined)

負荷をかけない状態のデータ転送で，通常の場合と Dyas を使用した場合を比較すると，どちらも 1 秒から 2 秒に 90%の度数が集中しており，レスポンスタイムの分布に大きな差は見られなかった．負荷をかけた状態のデータ転送では，Dyas を適用していない場合は 4 秒以下の範囲にレスポンスタイムは分布しておらず，4 秒から 6 秒の範囲で 90%のレスポンスタイムを占めていた．一方で

Dyas を適用したデータ転送の場合は 1 秒から 2.5 秒の範囲で同等の割合のレスポンスタイムが得られており，負荷をかけない状態に近い精度でデータ転送を行うことが可能であった．Dyas を適用した高優先度プロセスでは，ネットワークの負荷が高い場合にも，目標のレスポンスタイムに近い時間でデータ送信を行うための帯域確保が可能であることを確認した．

4.4 複数台クライアントでの評価

4.3 節では，ネットワークの負荷は 80%のみ扱っており，20%の帯域幅が空いている状態で行ったことから，0%から 90%のネットワーク負荷を与えた場合でも同様のレスポンスが得られるか課題であった．そこで，20 台までクライアント台数を増やし，また，ロボット制御に適した高性能組込みシステム(Rasberry Pie)を利用し，100Mbps のルータにより効果を検証した．テスト時に使用した環境を表 4 に示した．

表 4　テスト環境

Table 4 Testing Environment

	CPU	Mem	OS	台数
サーバ	Corei5, 1.2GHz(4Core)	4GB	Ubuntu 10.4	1
クライアント	Coretex-A53 1.2GHz(4Core)	1GB	Rasbian	10

Dyas 適用/不適用の差分を確認するために,4.2 節同様， QVGA 動画 1 フレームに相当する 230kb データサイズを転送し，その際の平均レスポンスタイムを確認した．4.3 節に示した予備調査では，無線を用いた計測を行ったが，無線は距離や障害物の有無や位置などにより電波状態が異なることから，理論値により近い値を計測することを目的にイーサネットを用いた接続による調査を行った．結果を表 5 に，平均値のみの結果を図 11 に図示した．

表 5 負荷時のレスポンスタイム(sec)

Table5 Response Time Comparison (sec) with under load

（Dyas 有り(W), Dyas 無し(W/O), 平均，標準偏差，最悪値）

		0%	10%	30%	50%	70%	90%
平均	With	0.37	0.38	0.46	0.38	0.39	0.38
	W/O	0.96	1.09	1.35	1.71	2.72	8.81
標準偏差	With	0.03	0.03	0.06	0.03	0.04	0.04
	W/O	0.04	0.06	0.06	0.05	0.06	0.07
最悪値	With	0.49	0.51	0.7	0.5	0.56	0.56
	W/O	1.13	1.3	1.53	1.89	2.94	9.17

結果より，クライアント台数 10 台での負荷を変更した場合の評価結果から，Dyas 有り（With Dyas）ではネットワークの負荷が高い場合にも，目標のレスポンスタイムを達成していることがわかった．Dyas 無し(Without(W/O) Dyas) では，負荷が高くなるにつれて，レスポンスタイムが悪化していることがわかった（図 10）．このことから，ルータが保証している帯域内については，無線ではない場合には理論値が保証されることがわかった．

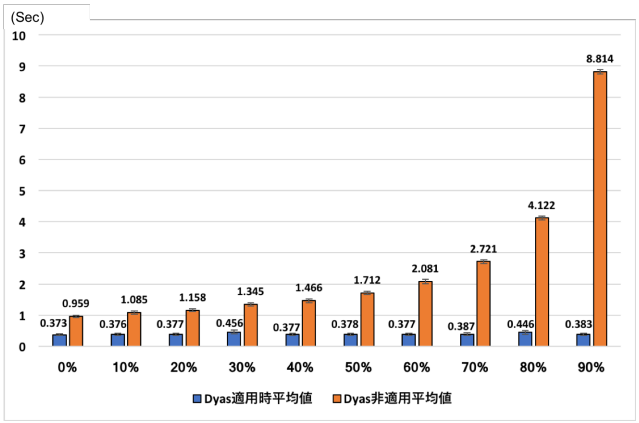

図 10　負荷 0%から 90%時のレスポンスタイム(10 台)

Response time (10 units) when load is 0% to 90%

7. まとめと今後の課題

本研究では， アプリケーションごとの目標レスポンスタイムに応じて動的に帯域制御を行うミドルウエア Dyas の設計，実装，評価を行った．Dyas 適用の有無での緊急のデータ送信に対するレスポンスタイムを比較した結果，Dyas を適用しネットワーク負荷を与えてデータ送信した時のレスポンスタイムは，ネットワーク負荷を与えない時のレスポンスタイムと同等の精度であったことから，Dyas を用いることで緊急のデータ送信に対する帯域確保が十分に実現できることを確認した．今後は高齢者支援システムに適用するために，更なる改善が必要である．

要件の再検討: 高齢者支援システムではロボットの支援機能に基づいて収集するデータと通信要求を定めた．しかし，実際に高齢者支援システムを介護施設などで運用する場合は，使用するマシンの性能やネットワーク性能に制限がかかることが考えられる．現在の Dyas の実装ではマシン内部の処理を扱っており，ネットワークにデータを送出した後の対応は別で考える必要がある．既存研究に示したように，QoS 制御は様々な分野で行われており，ネットワーク送出後にはアクセスポイントなどでの制御が適していると考える．そこで高齢者支援システムの要件を抽出し，既存の帯域制御技術を組み合わせることで，資源の限られた環境でのアプリケーションのレスポンスタイムに対す

る達成度を高めることが可能である．

設計の改善：帯域制御は，実測のレスポンスタイムの更新を監視して取得でき次第行うが，監視間隔は 1 秒に固定している．実測のレスポンスタイムを監視する間隔を早めることで，ネットワーク負荷に対する帯域制御の精度は更に向上する可能性があるが，CPU 使用率が増大する問題がある．これは，現在の実装で固定の監視間隔ごとに実測のレスポンスタイムの更新を確認しているためである．この問題は，通常時は処理を一切行わない待機状態を維持し，実測のレスポンスタイムが更新されたことをイベントとして検知することで改善できる．

無線利用時の帯域保証：今回は最終的にはイーサネットで保証を確認したが，無線などを利用せざるおえない環境なども想定される．そうした場合であっても保証ができるように動的な制御を行う方式を検討することも重要である．

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．“”，芝浦工業大学 2015 年度卒業論文，2016.
[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
[8] iRobot Create Open Interface ， iRobot Corporation ，https://www.irobot.com/filelibrary/pdfs/hrd/create/Create%20Open%20Interface_v2.pdf
[9] Raspberry Pi 2 Model B ， Raspberry Pi Foundation ，https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
[10] 無線 LAN 親機（Wi-Fi ルーター） : WZR-600DHP シリーズ | BUFFALO バッファロー ， BUFFALO ，http://buffalo.jp/product/wireless-lan/ap/wzr-600dhp/#spec
[11] Chart.js | Open source HTML5 Charts for your website，Chart.js，http://www.chartjs.org/
[12] gnuplot homepage ， Thomas Williams，Colin Kelley ，http://www.gnuplot.info/
[13] A.Botta，A.Dainotti，A.Pescapè．A tool for the generation of realistic network workload for emerging networking scenarios，Computer Networks (Elsevier)，Volume 56，Issue 15，pp 3531-3547，2012.
[14] Packet Sender - The Free UDP and TCP Network Test Utility，Dan Nagle，https://packetsender.com/
[15] Ostinato Network Traffic Generator，Srivats P.，http://ostinato.org/
[16] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool，iperf.fr，https://iperf.fr/