(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2017-76334
(P2017-76334A)
(43)公開日　平成29年4月20日(2017.4.20)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 11/34	(2006.01)	G06F	11/34	B	5B042
G06F 11/30	(2006.01)	G06F	11/30	D	5K201
H04M 11/00	(2006.01)	H04M	11/00	301	

審査請求 未請求　請求項の数 15　ＯＬ　（全 28 頁）

(21)出願番号　特願2015-204723 (P2015-204723)
(22)出願日　平成27年10月16日 (2015.10.16)

(71)出願人　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　110001689
青稜特許業務法人
(74)代理人　100107010
弁理士　橋爪　健
(72)発明者　下川　功
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内
(72)発明者　花岡　誠之
東京都千代田区丸の内一丁目６番６号　株式会社日立製作所内

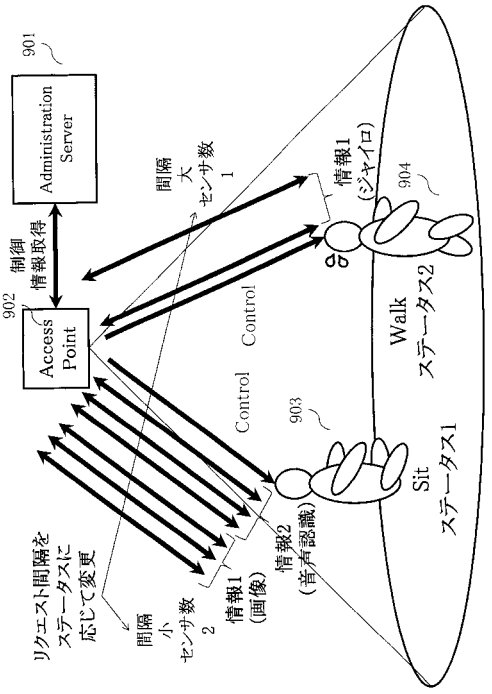
Ｆターム(参考)　5B042 GC20　JJ02　JJ29　MB02　MC29

最終頁に続く

(54)【発明の名称】管理サーバ及び管理システム及び管理方法

(57)【要約】

【課題】　管理サーバとロボット間の通信遮断及びロボット制御が不安定な状態を回避すること。
【解決手段】　管理サーバを配置し、複数のロボットを管理運用する場合において、ロボットからロボットに格納しているセンサ情報を取り出すためのリクエスト間隔及びセンサの種類、センサの個数によってロボット運用に支障をきたす場合がある。管理サーバ９０１がロボット９０３、９０４からＲＡＭに格納されている行動ステータスの情報を取得し、ロボット９０３、９０４の行動ステータスに応じてセンサの種類、センサの個数及びリクエスト間隔を決定し、ロボット９０３、９０４の動作に影響を及ぼさないロボット情報の収集及び制御を実現する。
【選択図】　図９



【特許請求の範囲】
【請求項１】
管理サーバであって、
複数のセンサを有する機器の行動状態に対して、センサ数及びセンサ種類、ＣＰＵ閾値を予め記憶したアクション定義ファイルと、
処理部と
を備え、
前記処理部は、
前記機器の行動状態を取得し、
前記アクション定義ファイルを参照し、前記機器の行動状態に対するセンサ数及びセンサ種類により、前記機器からデータを取得する複数のセンサの個数と種類を決定し、
センサ数によって前記機器のＣＰＵ使用率がＣＰＵ閾値を超えないように、複数のセンサのデータ取得を要求するリクエスト間隔を決定する、
管理サーバ。

【請求項２】
請求項１に記載の管理サーバにおいて、
前記アクション定義ファイルのセンサ種類は、最低限必要とするひとつ又は複数のセンサの種類及び個数を含み、
前記処理部は、前記アクション定義ファイルを参照し、センサ数と決定したリクエスト間隔とに従い、各リクエスト間隔において、最低限必要とするひとつ又は複数のセンサを常時確保し、最低限データを必要とする以外の複数のセンサを全センサの中から順番に取り出して、リクエストを要求することを特徴とする管理サーバ。

【請求項３】
請求項２に記載の管理サーバにおいて、
前記処理部は、複数のセンサの各々に対して、リクエスト間隔を平均値とし、予め定めた分散又は標準偏差に従い、センサ毎のリクエスト間隔を決定することを特徴とする管理サーバ。

【請求項４】
請求項２に記載の管理サーバにおいて、
前記処理部は、予め定義されたセンサ群毎に、リクエスト間隔を平均値とし、予め定めた分散又は標準偏差に従い、センサ群毎のリクエスト間隔を決定することを特徴とする管理サーバ。

【請求項５】
請求項４に記載の管理サーバにおいて、
前記処理部は、センサ群毎に予め定めた優先度毎に応じて、センサ群毎のリクエスト間隔を、優先度の高い順に短くなるように決定することを特徴とする管理サーバ。

【請求項６】
請求項５に記載の管理サーバにおいて、
前記処理部は、各センサ群の中の複数のセンサから取得した情報のデータ変化率に応じて、優先度を、データ変化率の大きい順に高くなるように決定することを特徴とする管理サーバ。

【請求項７】
請求項５に記載の管理サーバにおいて、
前記処理部は、各センサ群の中の複数のセンサから取得した情報の平均値を算出し、平

均値が予め定めた閾値以上になった場合、当該センサ群の優先度を高く設定し、当該センサ群のリクエスト間隔を短く決定することを特徴とする管理サーバ。

【請求項８】
請求項５に記載の管理サーバにおいて、
前記処理部は、予め定義されクラスタ化されたセンサ群間の予め定めた相関関係に従い、データ変化率が予め定めた閾値より高いセンサ群のデータを取得する場合、更に該センサ群と相関が高いセンサ群のデータも取得すること、又は、優先度が予め定めた閾値より高いセンサ群と相関が高いセンサ群の優先度を高く決定することを特徴とする管理サーバ。

【請求項９】
請求項５に記載の管理サーバにおいて、
前記機器は送信するクラスタ化されたセンサ情報の時系列データから瞬時時のマハラノビスの距離を求め、マハラノビスの距離が予め定めた閾値を超えた場合、そのセンサ群の優先度を高く決定することを特徴とする管理サーバ。

【請求項１０】
請求項１に記載の管理サーバにおいて、
前記アクション定義ファイルは、行動状態及び無線電話の強度に対して、センサ数及びセンサ種類、ＣＰＵ閾値を予め記憶し、
前記処理部は、さらに、前記機器との間の通信の無線電波強度を取得し、前記アクション定義ファイルを参照し、前記機器の行動状態及び無線電波強度に応じて、センサ数及びセンサ種類及びＣＰＵ閾値を取得することを特徴とする管理サーバ。

【請求項１１】
管理システムであって、
複数のセンサを有する機器と、
前記機器の行動状態に対して、センサ数及びセンサ種類、ＣＰＵ閾値を予め記憶したアクション定義ファイルを有する管理サーバと
を備え、
前記管理サーバは、
前記機器の行動状態を取得し、
前記アクション定義ファイルを参照し、前記機器の行動状態に対するセンサ数及びセンサ種類により、前記機器からデータを取得する複数のセンサの個数と種類を決定し、
センサ数によって前記機器のＣＰＵ使用率がＣＰＵ閾値を超えないように、複数のセンサのデータ取得を要求するリクエスト間隔を決定し、
リクエスト間隔に従い複数のセンサのデータ取得を前記機器へ要求し、
前記機器は、
前記サーバからの要求に従い、複数のセンサのデータを前記管理サーバに送信する、
管理システム。

【請求項１２】
請求項１１に記載の管理システムにおいて、
前記管理サーバが、前記機器から送信される情報量をスループットとして計測し、計測したスループットが予め定められた閾値以上になった場合は前記管理サーバから前記機器に対して送信パケットを集約するための制御コマンドを送信し、
前記機器は送信するクラスタ化されたセンサ情報を集約して前記管理サーバへと送信する
ことを特徴とする管理システム。

【請求項１３】
請求項１２に記載の管理システムにおいて、
前記機器は、制御コマンドを受信すると、優先度の低いセンサ群の複数のセンサの情報を集約し、前記管理サーバへと送信することを特徴とする管理システム。

【請求項１４】
請求項１２に記載の管理システムにおいて、
前記機器は、送信するセンサ群毎のセンサ情報をデータ変化率が予め定められた閾値より大きいセンサ群の複数のセンサの情報を集約し、前記管理サーバへと送信することを特徴とする管理システム。

【請求項１５】
管理サーバによる管理方法であって、
前記管理サーバは、
複数のセンサを有する機器の行動状態に対して、センサ数及びセンサ種類、ＣＰＵ閾値を予め記憶したアクション定義ファイルと、
処理部と
を備え、
前記処理部は、
前記機器の行動状態を取得し、
前記アクション定義ファイルを参照し、前記機器の行動状態に対するセンサ数及びセンサ種類により、前記機器からデータを取得する複数のセンサの個数と種類を決定し、
センサ数によって前記機器のＣＰＵ使用率がＣＰＵ閾値を超えないように、複数のセンサのデータ取得を要求するリクエスト間隔を決定する、
管理方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本実施形態は、管理サーバ及び管理システム及び管理方法に係り、特に、複数の機器を管理し、機器のモニタリング及び制御を行う管理サーバ及び管理システム及び管理方法に関するものである。
【背景技術】
【０００２】
昨今、あらゆる機器をインターネットに接続して、各種機器のモニタリング及び制御を実現するＩｏＴ（Internet of Things）が注目を浴びている。ＩｏＴ機器は、インターネットに接続され、他の機器の情報を共有し、個々の機器では提供できない新たな付加価値を提供する。ＩｏＴは新たなビジネスを創出する次世代のネットワークモデルとして期待されている。
ＩｏＴ機器の一例として、ネットワークロボット（以下、ロボットという。）が挙げられる。ロボットをネットワークに接続し、インターネットを経由してロボットから様々な情報検索及び遠隔地からのロボットの制御が可能である。経済産業省2013年度ニュースリリース(http://www.meti.go.jp/press/2013/02/20140217002/20140217002.html)によると現在高齢化の進行により高齢者介護などの生活支援分野等でのロボット技術の活用に強い期待が寄せられている。一方、生活支援ロボットは人との接触度が高くなるため、本格的な導入に向けて、対人安全の技術や基準・ルール整備と、安全対策を証明する制度の必要性が求められている。このニュースリリースからも、対人安全の技術は必要であり、対人安全を考慮した場合は、「歩行」等の行動をロボットが行う場合、人への衝突等を避けるためには、迅速な制御は勿論のこと、更に周りの状況を常にモニタリングし続けることが必要となる。このため、ロボットのようなＩｏＴ機器はスマートメータなどに代表される

IoT機器とは異なり、高頻度に情報収集や制御へのフィードバックが求められる。
【０００３】
機器のハードウェア及びソフトウェアの稼働状況を遠隔でリアルタイムに監視制御する方法として、特許文献1には、「装置の稼働状況を遠隔でリアルタイムに監視する際に、端末装置のログ情報の収集管理及び設定の手間を軽減する」（要約）ために、「異常発生の原因と想定される事象候補を示す情報と、解析に不足しているとしてリストアップされたログ情報の種別とに基づいて、サーバ装置のログ収集手段に対して、端末機器から収集するログ情報の内容及び粒度と、収集頻度との設定変更指示を行う」（段落００５７）ログ収集システムが記載されている。
また特許文献２には、「複数のセンサからの複数のセンサ信号を受信しシリアル回線経由で所定の送信周期で送信される送信フレームによりこれら複数のセンサ情報を上位制御装置に対して送信するユニット制御装置と、前記複数のセンサ信号を受信しユニット制御装置に対して動作制御信号を送信する上位制御装置により構成され」（請求項１５）、「通信効率に優れた制御システム及び信号送信方法」（段落０００５）が記載されている。

【先行技術文献】
【特許文献】
【０００４】
【特許文献１】特開２０１２－１９８７９６号公報
【特許文献１】特開２００８－２２６２２２号公報
【発明の概要】
【発明が解決しようとする課題】
【０００５】
前述の通り、ロボットのようなＩｏＴ機器はスマートメータなどに代表されるＩｏＴ機器とは異なり、高頻度に情報収集や制御へのフィードバックが求められ、ＩｏＴ機器とサーバ側での通信頻度が高頻度になる。一般的にＩｏＴ機器は安価に提供するために高機能なCPU等を搭載していないが、高頻度な通信処理は、ＣＰＵに負荷を与え、特にロボットのように可動部をもつＩｏＴ機器等は通信以外の処理、例えばその機器の動作そのものに影響する可能性がある。特に周囲の状況を常にモニタリングしながらＩｏＴ機器が動作、制御するロボットの場合は、常時、センサの情報を管理サーバへと送信し、管理サーバでの処理に基づいてロボットの制御コマンドを受信するため、機器の動作に影響する可能性が高くなる場合が想定される。
【０００６】
特許文献１では、サーバ装置は異常状態を解析するために不足する情報をリストアップするが、もし異常状態を解析するに当り不足する情報が大量にある場合は、大量の情報を端末装置から取得する必要が生じ、端末のＣＰＵリソースを無駄に浪費する可能性がある。この場合、もし端末装置が緊急を要する事象に対応している場合、例えば緊急にメールを送信する等の動作を発生している場合、ＣＰＵの割り込み等に不具合が発生しメールを送信できない等の障害が発生する場合が想定される。また端末とサーバ間の通信において、通信遮断が発生する確率も高くなる場合が想定される。逆にリソースマネージメントを行い、ＣＰＵリソース等を考慮して取得する情報量を少なくするのであれば、障害の原因を追究できない可能性もある。これは、異常状態を解析するため必要となる情報と端末のＣＰＵリソースはトレードオフであり、さらに不足する情報またはセンサ情報の粒度と端末装置のリソースの対応付けが明確にできていないからである。これでは、質の高いリソースマネージメントを必ずしも実現することができず、ロボットのＲＡＭに格納されているセンサ情報を頻度高く収集することでロボットの動作に影響を及ぼしてしまう場合が想定される。
また、特許文献２では、ユニット制御装置のサンプリング周期の変更が各リソース(ＣＰＵ、メモリ)に影響を及ぼす可能性があり、ユニット制御装置が誤動作等を引き起こす場合が想定される。これは、サンプリング周期とユニット制御装置のリソースの対応付け

が明確にできていないからである。これでは、高いサンプリング周期の場合ユニット制御装置の誤動作を誘発してしまう場合が想定される。

【０００７】

上述の２つの文献の技術を、管理サーバから複数のロボットを遠隔で制御及び運用を行うシステムへと適用した場合、リクエスト間隔及びセンサの種類を的確に設定せず大量にロボットからロボットのＲＡＭ上に格納されているセンサ情報を引き出すため、ＣＰＵ使用率が歯止めなく上昇し、ロボット内のＣＰＵにおける割り込み等に不具合が発生しロボットの制御が不安定な状態となる確率が高くなると想定される。またロボットの高いＣＰＵ使用率によって管理サーバとロボット間の通信が遮断される可能性も想定される。この場合、ロボットシステムにおいてシステムの信頼性、可用性、保守性を示すＲＡＳ（Reliability Availability Serviceability）は低い。

これらの障害を回避するために、特にロボットは行動ステータスの変動が激しいため、これら行動ステータスの変動に対して、センサの情報をロボットから取得するためのリクエスト間隔を精度高く追従して変更するリソースマネージメントが必要となってくる。なお、本明細書ではロボットの「歩行」、「座る」、「立つ」等の行動状態を「行動ステータス」と定義する。またこの行動ステータスはパラメータとしてロボットが定期的にロボット内のRAM上に格納している。

【０００８】

本発明は、以上の点に鑑み、管理サーバからロボットへのリクエスト間隔を制御し、ロボット制御が不安定になる状態や通信遮断が発生することを回避することを目的とする。

【課題を解決するための手段】

【０００９】

本発明の第１の解決手段によると、

管理サーバであって、

複数のセンサを有する機器の行動状態に対して、センサ数及びセンサ種類、ＣＰＵ閾値を予め記憶したアクション定義ファイルと、

処理部と

を備え、

前記処理部は、

前記機器の行動状態を取得し、

前記アクション定義ファイルを参照し、前記機器の行動状態に対するセンサ数及びセンサ種類により、前記機器からデータを取得する複数のセンサの個数と種類を決定し、

センサ数によって前記機器のＣＰＵ使用率がＣＰＵ閾値を超えないように、複数のセンサのデータ取得を要求するリクエスト間隔を決定する、

管理サーバが提供される。

【００１０】

本発明の第２の解決手段によると、

管理システムであって、

複数のセンサを有する機器と、

前記機器の行動状態に対して、センサ数及びセンサ種類、ＣＰＵ閾値を予め記憶したアクション定義ファイルを有する管理サーバと

を備え、

前記管理サーバは、

前記機器の行動状態を取得し、

前記アクション定義ファイルを参照し、前記機器の行動状態に対するセンサ数及びセンサ種類により、前記機器からデータを取得する複数のセンサの個数と種類を決定し、

センサ数によって前記機器のＣＰＵ使用率がＣＰＵ閾値を超えないように、複数のセンサのデータ取得を要求するリクエスト間隔を決定し、

リクエスト間隔に従い複数のセンサのデータ取得を前記機器へ要求し、

前記機器は、
　前記サーバからの要求に従い、複数のセンサのデータを前記管理サーバに送信する、
管理システムが提供される。
【００１１】
　本発明の第３の解決手段によると、
　管理サーバによる管理方法であって、
　前記管理サーバは、
　複数のセンサを有する機器の行動状態に対して、センサ数及びセンサ種類、ＣＰＵ閾値を予め記憶したアクション定義ファイルと、
　処理部と
を備え、
前記処理部は、
　前記機器の行動状態を取得し、
　前記アクション定義ファイルを参照し、前記機器の行動状態に対するセンサ数及びセンサ種類により、前記機器からデータを取得する複数のセンサの個数と種類を決定し、
　センサ数によって前記機器のＣＰＵ使用率がＣＰＵ閾値を超えないように、複数のセンサのデータ取得を要求するリクエスト間隔を決定する、
管理方法が提供される。

【発明の効果】
【００１２】
　本発明によると、管理サーバからロボットへのリクエスト間隔を制御し、ロボット制御が不安定になる状態や通信遮断が発生することを回避することができる。

【図面の簡単な説明】
【００１３】
【図１】本実施形態のシステム概要図である。
【図２】本実施形態における管理サーバ内部のハードウェア構成図である。
【図３】本実施形態においてＩｏＴ機器であるロボット内部の簡易のハードウェア構成図である。
【図４】本実施形態においてロボットの行動ステータス毎に対するセンサの種類及び個数、CPU閾値を定義した表を示すアクション定義ファイル（１）の説明図である。
【図５】本実施形態においてロボットの行動ステータス及び無線電波強度毎に対するセンサの種類及び個数、CPU閾値を定義した表を示すアクション定義ファイル（２）の説明図である。
【図６】本実施形態においてセンサの種類、センサの個数、リクエスト間隔の関係を説明するグラフである。
【図７】本実施形態においてロボットのステータス毎のＣＰＵ負荷率の変化及びリクエスト間隔をダイナミックに変化させる制御を説明する図である。
【図８】本実施形態において管理サーバがロボットからロボットのＲＡＭに格納されているセンサ情報を取得する場合においてロボットの行動ステータスに応じてＣＰＵ閾値、センサの個数、センサの種類を定義し、設定したＣＰＵ閾値以下になるように制御する手法の説明するフローチャート図である。
【図９】本実施形態におけるシステム動作概要を説明した図である。
【図１０】本実施形態において、センサの個数が５個の場合におけるセンサの種類をダイナミック変化させ制御することを説明する図である。
【図１１】本実施形態において、取得するセンサの個数が１０個の場合における、センサ毎にリクエスト間隔を定義したリクエスト間隔テーブルを説明する図である。
【図１２】本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト周期を定義したリク

エスト間隔テーブルを説明する図である。
【図１３】本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト周期を定義し、更にクラスタ毎に優先度を定義したリクエスト間隔テーブルを説明する図である。
【図１４】本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト周期を定義し、更に取得したクラスタ毎に優先度を定義し、更に取得したセンサ情報のデータ変化率も定義したリクエスト間隔テーブルを説明する図である。
【図１５】本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト周期を定義し、更に取得したクラスタ毎に優先度を定義し、更にクラスタ毎に取得したセンサ情報のデータ変化率も定義し、クラスタ毎の平均値を定義したリクエスト間隔テーブルを説明する図である。
【図１６】本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト周期を定義し、更に取得したクラスタ毎に優先度を定義し、更に取得したセンサ情報のデータ変化率も定義し、更にロボットがセンサ情報を送信するに当りセンサ情報を集約の有無を定義したリクエスト間隔テーブルを説明する図である。
【図１７】本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト周期を定義し、更に取得したクラスタ毎に優先度を定義し、更に取得したセンサ情報のデータ変化率も定義し、更にクラスタ毎に分けられた各センサ情報の時系列データからクラスタ間の相関係数を計算し最も相関が高いクラスタを定義したリクエスト間隔テーブルを説明する図である。
【図１８】本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト周期を定義し、更に取得したクラスタ毎に優先度を定義し、更に取得したセンサ情報のデータ変化率も定義し、更に取得したセンサ情報の時系列データからクラスタ毎のマハラノビスの距離を定義したリクエスト間隔テーブルを説明する図である。
【発明を実施するための形態】
【００１４】
Ａ.概要

本実施形態による管理システムは、例えば、センサを備える機器とサーバが無線又は有線で通信を行うシステムであって、サーバは、機器の行動状態を定期的に検出し、サーバは、機器の行動状態に応じて、センサのデータを取得する個数と種類を決定し、機器のＣＰＵ使用率が所定の閾値を超えないように、センサのデータ取得を要求する周期（リクエスト間隔）を決定し、機器はサーバから定期的に送信される要求されるセンサの情報や機器の設定情報に基づき、管理サーバへとセンサの情報を送信する。
【００１５】
Ｂ.システム及び装置

図１に、第１の実施形態におけるシステム図を示す。図1は、管理サーバが複数のロボットを制御、管理する管理システムを示した図である。システムはAdministration Server(管理サーバ)(102)、Access Point(無線LAN基地局等のアクセスポイント)(103)、ロボット(104)、ＤＢ（データベース）（１０１）を備える。このシステムにおいて、管理サーバ（１０２）は、各ロボット（１０４）からロボット（１０４）のRAM上に格納されているセンサの情報を取得し各ロボット（１０４）に対して制御を行う。ロボット（１０４）は頭部にCPUや通信インターフェースを持ち、胴体や腕、足等を制御するためにロボット（１０４）の関節や胴体に備え付けられている各種Actuatorの制御や通信インターフェースを経由して外部機器との通信等をＲＯＳ(Robot Operation System)等のＯＳがＣＰＵを

介して制御を行う。管理サーバ（１０２）は、ロボット（１０４）に格納されているセンサ情報を取得するためにロボット（１０４）に対してセンサ情報取得のリクエストを送信し、リクエストを受信した各ロボット（１０４）は無線通信で無線ＬＡＮ基地局を経由して、ＲＡＭ上に格納されているセンサの情報を管理サーバ（１０２）へと送信する。その場合に使用される通信プロトコルはＴＣＰ(Transmission Control Protocol)やＳＯＡＰ(Simple Object Access Protocol)等が挙げられる。センサ情報を受信した管理サーバ（１０２）は自身のＲＡＭ及びデータベースへとセンサ情報を格納する。このように複数のロボット（１０４）を管理サーバ（１０２）が管理、精度高く制御することによって、様々なサービスへとロボット（１０４）を適用することが可能となる。この管理サーバ（１０２）は、ＳａａＳのようにクラウドを経由して制御ソフトウェアを遠隔地にいるロボット管理者へと提供することも可能である。管理サーバ（１０２）がロボット（１０４）を制御するに当り、高精度な制御が求められるために、ロボット（１０４）から各種センサの情報を引き出し、その情報を基に制御を行うクローズドループ制御方式のような制御方式が必須である。管理サーバ（１０２）が制御に当り必要となるセンサ情報をAccess Point(１０３)を経由してロボット（１０４）から取得する場合、管理サーバ（１０２）が精度高く制御するためロボット（１０４）のＲＡＭ上に格納されている大量のセンサ情報をロボット（１０４）へと要求した場合は、ロボット（１０４）のＣＰＵはその要求に答えるために多く使用され、ＣＰＵ使用率が上昇し、ＣＰＵの割り込みに不具合が生じ、ロボット（１０４）の制御が不安定な状態になる可能性が想定される。ロボット制御が不安定な状態となった場合、対人安全が疎かになり、ロボット（１０４）が歩行時に人へと衝突するといった対人安全が疎かになる可能性が想定される。この場合、ロボットシステムにおいてシステムのＲＡＳは非常に低い。このようにロボット（１０４）のＲＡＭ上に格納されているセンサ情報を取得するリクエスト間隔をロボット制御が不安定な状態にならない程度に制御する必要がある。管理サーバ（１０２）で実行される制御ソフトウェアは、モニタリング画面及びロボット（１０４）を制御するための制御画面を備える。モニタリング画面上で、管理サーバ（１０２）がロボット（１０４）のＲＡＭ上に格納されているセンサ情報を取得し、取得したセンサ情報をリアルタイムに表示する。またモニタリング画面上でロボット（１０４）のカメラから定期的に画像を取得し動画を表示させることも可能である。管理者はこのモニタリング画面を見ながら、様々な状況に応じて遠隔地からロボット（１０４）を制御させることも可能となる。

【００１６】

図２は、本実施形態における管理サーバ内部のハードウェア構成図である。管理サーバはハードウェアとして、ＲＡＭ（２０１）、ＣＰＵ（２０２）、ＮＩＣ（Network Interface Card）(２０３)、ＤＢ（２０４）を備える。管理サーバで駆動する制御ソフトウェアは、RAM上に格納されＣＰＵを介して実行される。またロボットとの通信はＮＩＣ（２０３）を経由して行われる。また管理サーバはＲＡＭ上に格納されているセンサ情報をハードディスク上のＤＢへと格納ことも可能であるし、外部のハードディスクに格納することも可能である。さらに管理サーバは、ＧＵＩ、Robot Adapter部、Robot Management部、Network Interfaceを機能としてもつ。Robot Management部は複数のロボットから送信されるセンサ情報を受信し、データベースやＲＡＭ上へと格納し、複数のロボットを制御する。Robot Management部はロボット毎にセンサ情報を受信しロボットの制御を行うRobot Adapter部を複数持つ。このようにロボット毎にセンサ情報を受信、制御を行うRobot Adapter部をRobot Management部は複数持つことで複数のロボットを同時に制御することが可能となる。Robot Adapter部は、ロボットからセンサ情報を受信するCollector部、ロボットの制御を行うController部を持ち、取得したセンサ情報を基にController部が制御するに当り必要な情報へと変換、または計算を行い、計算結果をController部へと送信するCalculator部を持つ。Controller部は、Calculator部から送信される情報を基にしてロボット制御を行う。Calculator部は、予め計算していたＣＰＵリソースと機器との対応付けを明確にした上で、リクエスト間隔をＣＰＵ閾値以下になるようにCollector部へと指示する。またRobot Management部は各ロボットから送信される各種センサ情報をＧＵＩの要求に

応じてデータを送信する。
　また、ＲＡＭ２０１には、アクション定義ファイル、リクエスト間隔テーブル等が記憶される（詳細は後述）。
【００１７】
　図３は、本実施形態においてＩｏＴ機器であるロボット内部のハードウェア構成図である。ロボットは複数Actuatorを持ち、Robot OS（ＲＯＳ）等がＣＰＵを用いて各種Actuatorを制御する。またロボットは複数のセンサ(３０１)を持ち、ROS等が複数センサの制御を行い、センサ情報をＲＡＭ上へ常時格納している。またＮＩＣ(Network Interface Card)(304)を経由して管理サーバからセンサ情報を要求された場合、ＲＯＳ等がＲＡＭ上に格納されたセンサ情報を適時取り出し、管理サーバへとセンサ情報をＴＣＰやＳＯＡＰ等の通信プロトコルを用いて送信する。
　ロボットは、Sensor Collector、Actuator Controller、Network Interfaceを備える。ロボットは、Actuator Controllerを用いてロボットの各関節に組み込まれているActuatorの制御を行い、ロボットを駆動させる。またSensor Collectorは各種センサからセンサ情報を引き出しＲＡＭ上へと格納する。先程記述したように、センサ情報をＲＡＭ上へと格納する動作は、ロボット内部で行われていることでありＲＯＳ等が効率良くセンサ情報をＲＡＭ上へと格納するためにスケジューリング制御を行うのでロボットのＣＰＵに対する負荷は非常に少ない。Sensor Collectorは管理サーバからセンサ情報を管理サーバへと送信するように要求された場合は、管理サーバから送るように指示されたセンサ情報をＲＡＭ上から取り出しＴＣＰまたはＳＯＡＰ等の通信プロトコルで管理サーバへとNetwork Interfaceを経由して送信する。また、Sensor Collectorは、管理サーバから要求されたセンサ情報を管理サーバから要求される集約有無及びセンサ毎の優先度に応じて管理サーバへとセンサ情報を送り返すことができる。
【００１８】
　この場合において、センサ情報をＲＡＭ上へと格納する動作は、ロボット内部で行われている動作でありＲＯＳ等が効率良くセンサ情報をＲＡＭ上へと格納するためにスケジューリング制御を行うのでロボットのＣＰＵに対する負荷は少ない。しかし、管理サーバからロボットに対してＲＡＭ上のセンサ情報を取り出すためのリクエスト等、外部からのリクエストに対しては、ＲＯＳ等は外部からのリクエストに対してスケジューリングを制御することができないために、ロボットのＣＰＵに対する負荷は高い。このように外部からのリクエストが頻繁に発生するような状況下では、ロボットのＣＰＵ負荷率が高くなる場合が想定される。この場合、頻繁に発生するリクエストに対応してＣＰＵ負荷率が上昇し、ロボット制御に関して不安定になる状態または管理サーバとロボット間において通信遮断が発生する可能性がある。よって外部からのリクエストを、ロボットのロボット制御が不安定な状態等の障害が発生しない程度に制御する必要がある。つまり、ロボット制御が不安定な状態等の障害はＣＰＵ負荷率が高くなることによって発生する。よって、ロボットのＣＰＵリソースにおいて、ＣＰＵリソース不足に起因する管理サーバとの通信遮断やロボット制御が不安定になる状態等の障害が発生しないように所定の閾値を予め決め、所定の閾値以下のＣＰＵ使用率でロボットを駆動させる必要がある。
【００１９】
Ｃ．行動ステータスとＣＰＵ負荷、アクション定義ファイル

　本実施形態を具体的に説明するに当り、ロボットのＣＰＵ負荷に関して依存関係を定性的に考える。ＣＰＵ負荷は主に下記３つに対して大きく依存していると考えている。
（１）ロボットの行動ステータス
（２）センサの種類及びセンサの個数
（３）管理サーバが情報収集を行うためにロボットに対して行うリクエストの間隔
【００２０】
　まず、上述の（１）に関して、ＣＰＵ負荷率はロボットの行動ステータスに大きく依存し、またロボット制御が不安定になる状態が発生しないＣＰＵ閾値はロボットの行動ステ

ータス毎に違う。「行動ステータス」とは、例えば、ロボットの「歩行」、「座る」、「立つ」等の行動状態をいう。ロボットはこの行動ステータスをパラメータとして定期的にＲＡＭ上に格納しており、サーバは行動ステータスをＲＡＭから取得することができる。ロボットの行動ステータス毎にＣＰＵへと与える影響が違うために、行動ステータス毎にＣＰＵ閾値を定義する必要がある。例としてロボットの行動ステータスが「座る」場合に定期的にロボットのＲＡＭに格納されているセンサ情報を引き出しＣＰＵ負荷率が例えば５０％程度に上昇したとしても、５０％までならばロボット動作に影響は与えない。しかし行動ステータスが「座る」状態の場合におけるリクエスト間隔で、ロボットの行動ステータスが「歩行」状態の場合にセンサ情報を引き出した場合は、ＣＰＵ負荷率は例えば５０％に更に歩行する動作によって引き起こされるＣＰＵ負荷がかかり、例えば６０％以上に上昇してしまう。この場合ＣＰＵ負荷率が６０％を超えるような状態では通信遮断やロボット制御が不安定な状態を引き起こす場合が想定される。特にロボットが歩行時に制御が不安定な状態なった場合は、対人安全が疎かになり重大な損害を顧客に与えてしまうかもしれない場合が想定される。しかしロボットの行動ステータスが「座る」状態の場合は、例え通信遮断やロボット制御が不安定な状態になることが発生しても、ロボットが人や物に損害を与える可能性は低いと想定される。このように、行動ステータス毎に対人安全を考慮し、どの程度のCPU負荷ならばロボットのＱｏＳ(Quality of Service)を維持できるかを予めロボットで検査かつ定義することは必須である。

つぎに、上述（２）に関して、ＣＰＵ負荷率はセンサの種類及びセンサの個数にも大きく依存する。例えばセンサとしてロボットの腕、頭、足等の間接を動かすモータの温度とロボットの頭のカメラからの画像があり、温度と画像ではＣＰＵに与える影響もそれぞれ異なる。また情報を取得するためのセンサの個数もＣＰＵへと与える影響は大きい。例えばロボットは約４０００程度のセンサ情報をもっているが、定期的に１０個のセンサ情報を取得する場合と、４０個のセンサ情報を取得する場合とではＣＰＵの負荷はそれぞれ異なる。もちろん４０個のセンサ情報を定期的に取得する場合のほうがＣＰＵへと与える影響は大きい。

また、上述（３）に関して、センサを取得するリクエスト間隔もＣＰＵ負荷へと与える影響は大きい。センサ情報を取得するリクエスト間隔毎にＣＰＵ負荷へと与える影響はそれぞれ異なる。もちろんリクエスト間隔を早くするほうがＣＰＵ負荷へと与える影響は大きい。

つまりＣＰＵ負荷は（１）行動ステータス、（２）センサの種類、センサの個数、（３）リクエスト間隔に大きく依存する。高精度な制御を行うためロボットのＣＰＵを考慮したリソースマネージメントを行う場合、これらの関係を予め精査しておくことは特に重要となる。

【００２１】

上記を基にリソースマネージメントを行い高精度なロボット運用を実現するに当り、行動ステータス毎に、ＣＰＵ閾値（例えば、動作中に許容されるＣＰＵ負荷等の上限値）、センサの種類、センサの個数、リクエスト間隔、これら４つのパラメータを各々定義する必要がある。

行動ステータス毎に、センサの種類、センサの個数、ＣＰＵ閾値、これら３つのパラメータは、決定するに当り優先度が高いパラメータであり、管理者の裁量で決定される。なぜならば行動ステータス毎に、センサの種類、センサの個数、ＣＰＵ閾値、を定義することはサービスコンテンツ及びロボットのＱｏＳに依存するからである。つまりどのようなサービスを提供するか（行動ステータス、センサの種類、センサの個数）、提供するサービスの質（ＣＰＵ閾値）はどの程度かは管理者の裁量次第であり、提供するサービス及びサービス品質によって行動ステータス毎の、センサの種類、センサの個数、ＣＰＵ閾値、が定義される。例えば、ロボットを使って受付業務等を代行させるサービスを考えた場合、ロボットの行動ステータスは常時、「座る」状態であることが予測され、その場合、窓口越しで顧客と接するため対人安全はほぼ皆無であり、例え制御が不安定になる状態や通信遮断が発生し、ロボットを再起動する必要が発生しサービス品質が低下したとしても肉

体的に顧客に損害を与えることはまずないと想定できる。その場合、ＣＰＵ閾値を高く設定し、センサの種類及びセンサの個数を多数定義し、それらの情報を取得することで質の高いサービスを顧客へと提供するも可能であるし、ＣＰＵ閾値を低く設定し、ロボット制御が不安定になる状態や通信遮断が絶対に発生しないようにセンサの種類及びセンサの個数を定義しロボットの可用性を確保することも可能である。これらは全て管理者、サービス提供者の裁量によって定義される事柄である。

ここでリクエスト間隔は、決定されたＣＰＵ閾値、行動ステータス、センサの種類、センサの個数から定義される。なぜならばＣＰＵ負荷は行動ステータス、センサの種類、センサの個数、リクエスト間隔に大きく依存するため、予めこれらの関係を精査し関係式を導くことでリクエスト間隔を算出することが可能であるからである。これらの関係式からリクエスト間隔を初めに定義して行動ステータス、センサの種類、センサの個数を決めることも可能である。しかしロボットサービスを考慮に当り、一般的にリクエスト間隔は他のパラメータに比べ優先度は低いと思われる。

【００２２】

図４は、本実施形態においてロボットの行動ステータス毎に対するセンサの種類及び個数、ＣＰＵ閾値を定義した表を示すアクション定義ファイル（４０１）の説明図である。管理者はどのようなサービスを提供するか（行動ステータス、センサの種類、センサの個数）、提供するサービスの質（ＣＰＵ閾値）はどの程度かを定義する必要がある。管理サーバはそれらの関係を表としてあらかじめ定めたアクション定義ファイル（４０１）を持つ。管理サーバはアクション定義ファイルを読み込み、それら定義ファイルから各行動ステータスに対するセンサの種類、センサの個数、ＣＰＵ閾値をパラメータとして定義する。その後ロボットから定期的にロボットの行動ステータスをセンサ情報として取得し、取得した行動ステータスを基に定義ファイルと比較参照し、センサの種類、センサの個数、ＣＰＵ閾値を定義していく。

【００２３】

Ｄ.行動ステータスに応じた制御

ロボットを高精度に制御するに当り、ＣＰＵを考慮したリソースマネージメントは必須である。よって、ロボットのＣＰＵリソースにおいて、ＣＰＵリソース不足に起因する管理サーバとの通信遮断やロボット制御が不安定な状態等の障害が発生しないように所定の閾値を決め、所定の閾値以下のＣＰＵ使用率でロボットを駆動させる必要がある。ＣＰＵ負荷はロボットの行動ステータスに依存するため、行動ステータス毎に設定する必要がある。もしロボットがある行動ステータス時に、ＣＰＵ使用率が所定のＣＰＵ閾値を超えてしまうと、通信遮断やロボット制御が不安定な状態等の障害が発生する確率が高くなる場合が想定される。よって所定のＣＰＵ閾値をＣＰＵ使用率が超えないように制御するには、ダイナミックにロボットの行動ステータスに応じて、ロボットを制御するに当り必要なセンサの個数及びセンサの種類を変化させ、そのセンサの個数及びセンサの種類、ロボットの行動ステータスに基づいてセンサ情報を取り出すためのリクエストの数つまりリクエスト間隔をダイナミックに変化させる必要がある。

【００２４】

図７は、本実施形態においてロボットのステータス毎のＣＰＵ負荷率の変化及びリクエスト間隔をダイナミックに変化させる制御を説明する図である（７０１）。ＣＰＵ負荷率を実線、ＣＰＵ閾値を破線、リクエスト周期を二重破線でそれぞれ示す。

ここで、もしも一定のセンサの個数及びリクエスト間隔（例、センサの個数３０、リクエスト間隔４００ｍｓ）で管理サーバがロボットからＲＡＭ上に格納されているセンサ情報を取得する場合を想定する。つまりセンサの個数及びリクエスト間隔をダイナミックに変化させるように制御しない。しかしロボットの行動ステータスはダイナミックに変化する。もしロボットの行動ステータスが「立つ」状態から歩行速度が速い「歩行」状態へと変化した場合、もし一定のセンサの個数及びリクエスト間隔であれば、ＣＰＵ使用率が所定のＣＰＵ閾値を超えてしまい、管理サーバからロボットへの通信遮断、ロボット制御が

不安定な状態となる可能性が想定される。その後、ロボットの行動ステータスが歩く速度が遅い「歩行」の状態へと変化した場合、ＣＰＵ使用率が所定のＣＰＵ閾値近傍で動作する。この場合においても管理サーバ～ロボットの通信遮断、ロボットの制御が不安定な状態となる可能性は低いわけではない。このようにロボットから取得するセンサの個数及びリクエスト間隔を制御しないときは、管理サーバからロボットへの通信遮断、ロボット制御が不安定な状態となる可能性があり、ロボット運用に支障をきたす場合が想定され得る。

【００２５】

先程は一定のセンサの個数及びリクエスト間隔（センサの個数３０、リクエスト間隔４００ｍｓ）で管理サーバがロボットから取得することを想定していたが、本実施形態ではセンサの個数及びリクエスト間隔をダイナミックに変化させることを想定している。この場合先程とは違い、もしロボットの行動ステータスが「立つ（Ｓｔａｎｄ）」状態から歩く速度が速い「歩行」（Walk High Speed、Walk Slow Speed）の状態へと変化した場合、センサの個数及びリクエスト間隔をＣＰＵ使用率が所定の閾値以下になるようにダイナミックに制御する場合、管理サーバからロボットへの通信遮断、ロボット制御が不安定な状態となる可能性は限りなく低く、質の高いリソースマネージメント及び高精度なロボット制御が可能となる。

例えばロボットの行動ステータスが「立つ（Ｓｔａｎｄ）」状態である場合における、センサの個数及びセンサの種類、ロボットからロボットのＲＡＭに格納されているセンサ情報を取り出すためのリクエスト間隔をダイナミック変化させ制御することを考える。この場合、行動ステータス毎にあらかじめ取得するセンサの個数を定義しているアクション定義ファイルから、ロボットの行動ステータスが「立つ」状態時に取得するセンサの個数を定義する。定義したセンサの個数内において、更にロボットの行動ステータス毎において最低限必要なセンサの種類及び個数を定義し、それ以外のセンサ情報はベストエフォートにダイナミックに変化して取得する。その関係式を下記に記す。

$N_s >= N_e + N_b$　・・・（１６）

N_s：センサの個数　N_e：最低限必要なセンサの個数　N_b：ベストエフォートに取得するセンサの個数

【００２６】

ロボットを運用してサービスを提供する場合、サービス内容によって、ロボットの行動ステータス及びロボットの行動ステータスに応じて最低限取得する必要があるセンサの情報は定義される。例えば、ロボットを使って受付業務等を代行させるサービスを考えた場合、ロボットの行動ステータスは常時「座る」状態であることが予測される。この場合、受付業務を行うに当り、顧客の要望を聞く必要があり、音声認識機能は最低限起動させる必要がある。その場合顧客が話した言葉をロボットは聞いて認識した上で、その言葉を管理サーバへと送信することも考えられる。その言葉を基にして管理サーバ上で、どのような返答が望ましいのかをDeep Learningを用いて検索することも考えられる。またどのような表情をしたのか、男性なのか女性であるのか等顔を認識することも、サービスを提供する場合に様々な判断を行う場合において重要な情報であるため、画像データを最低限必要とするセンサ情報を考えることができる。しかしその２つ以外は受付業務を行うに当り常時必要でなりセンサと定義できる。その場合適時状況に応じて、２つ以外のセンサは管理サーバが指示して取得するといったことを考えられる。その場合、アクション定義ファイルからロボットの行動ステータスが「座る」場合におけるＣＰＵ閾値を定義し、センサの種類、個数が決まればリクエスト間隔も関係式から取得することができる。

【００２７】

図８は、本実施形態において管理サーバがロボットからロボットのＲＡＭに格納されているセンサ情報を取得する場合においてロボットの行動ステータスに応じてＣＰＵ閾値、センサの個数、センサの種類を定義し、設定したＣＰＵ閾値以下になるように制御する手法の説明するフローチャート図である。管理者はどのようなサービスを提供するか（行動ステータス、センサの種類、センサの個数）、提供するサービスの質（ＣＰＵ閾値）はど

の程度かを定義する必要がある。それらの関係を表として予め定め、管理サーバはアクション定義ファイルとして予め記憶する。管理サーバのＣＰＵ（２０２）は、初めにアクション定義ファイルを読み込んでいるかを確認し（８０２）、読み込んでいないのであれば、アクション定義ファイルを読み込み、それら定義ファイルから各行動ステータスに対するセンサの種類、センサの個数、ＣＰＵ閾値をパラメータとして定義する。ＣＰＵ（２０２）は、もしアクション定義ファイルの読み込みに失敗した場合は再びアクション定義ファイルを読み込みに行く（８０３）。その後、管理サーバのＣＰＵ（２０２）はロボットから定期的にＲＡＭ（２０１）上に格納されているロボットの行動ステータスをセンサ情報として取得する（８０４）。ＣＰＵ（２０２）は、取得した行動ステータスを基にアクション定義ファイルを参照し、ロボットの行動ステータスに応じてセンサの種類、センサの個数、ＣＰＵ閾値を読み取り、決定する（８０５）。ＣＰＵ（２０２）は、その後予め、ロボットを精査して算出した行動ステータス、センサの種類、センサの個数、リクエスト間隔、ＣＰＵ閾値の後述の関係式から、ロボットのＣＰＵ使用率が所定のＣＰＵ閾値以下になるようにリクエスト間隔を算出する。ＣＰＵ（２０２）は、その後、算出されたリクエスト間隔で管理サーバがロボットからセンサ情報を取得する。

【００２８】

Ｅ．関係式

図６は、本実施形態においてセンサの種類、センサの個数、リクエスト間隔の関係を説明するグラフである（６０１）。図６に示すグラフは横軸をリクエスト間隔、縦軸をＣＰＵ負荷率としている。これらの情報は予めロボットを精査し、取得したデータである。ロボットの行動ステータスは「座る」状態で取得したデータである。リクエスト間隔がＣＰＵ負荷に与える影響は定性的に考えるならば、リクエスト間隔が長ければＣＰＵ負荷に与える影響は少ない。よって対数的に減少していくことが予測されるため、取得したデータを対数近似しリクエスト間隔とＣＰＵ負荷率の関係式を導出する。関係式は下記のように定義する。

$C_r = A * \ln(req) + B$　　・・・（１）

ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率　Ａ：傾き　Ｂ：切片

図６のグラフ及び定性的に考えるならばセンサの数が多ければ多いほどＣＰＵ負荷に与える影響は大きい。よって傾きも切片もセンサの個数が多ければ多いほど大きくなる傾向があり、よってこの２つも対数近似でセンサの個数との関係式を導き出せることが可能である。傾きと切片の関係式も下記のように定義する。

$A = \alpha_A * \ln(N) + \beta_A$　　・・・（２）

$B = \alpha_B * \ln(N) + \beta_B$　　・・・（３）

Ｎ：センサの個数　Ａ：傾き　Ｂ：切片　α_A：Ａの関係式における傾き　β_A：Ａの関係式における切片　α_B：Ｂの関係式における傾き　β_B：Ｂの関係式における切片

（１）に示す関係式からＣＰＵ負荷率からリクエス間隔を導出することは可能であり、その場合は下式から求めることが可能である。

$req = \exp((C_r - B) / A)$　　・・・（４）

Ａ：傾き　Ｂ：切片　ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率

（４）式は（２）（３）式を用いて算出することが可能であり、つまりセンサの数からリクエスト間隔を算出することが可能である。

【００２９】

例として、３６センサと表記されているグラフは３６種類のセンサ情報を横軸に示す各リクエスト間隔でロボットから取得した場合において、ロボットのＣＰＵ負荷率を示した図である。この場合におけるリクエスト間隔とＣＰＵ負荷率の関係式は、次式の通りである。

$C_r = -10.9 \ln(req) + 77.346$　　・・・（５）

ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率

例えば、ＣＰＵ負荷率が３０％の場合は、管理サーバは３６種類のセンサ情報を７０ｍｓのリクエスト間隔で定期的にロボットから取得している。つまり３６種類のセンサ情報を取得する場合は、ＣＰＵ負荷率を３０％以下にしたいのであれば、７０ｍｓ以上のリクエスト間隔で定期的にロボットから取得すれば良い。つまり３０％以下における最小のリクエスト間隔は７０ｍｓであることを意味している。

他の例として１０センサと表記されているグラフは、１０種類のセンサ情報を横軸に示す各リクエスト間隔でロボットから取得した場合において、ロボットのＣＰＵ負荷率を示した図である。この場合におけるリクエスト間隔とＣＰＵ負荷率の関係式は、下記の通りである。

$C_r = -7.006\ln(req) + 50.756$　　・・・（６）

ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率

３６種類同様、ＣＰＵ負荷率が３０％の場合は、管理サーバは１０種類のセンサ情報を２０ｍｓのリクエスト間隔で定期的にロボットから取得している。つまり１０種類のセンサ情報を取得する場合、ＣＰＵ負荷率を３０％以下にしたいのであれば、２０ｍｓ以上のリクエスト間隔で定期的にロボットから取得すれば良い。つまりＣＰＵ負荷率３０％以下における最小のリクエスト間隔は２０ｍｓであることを意味している。

ここで（５）（６）を用いて、傾きと切片のセンサの個数との関係式を導出する。この場合における、センサの個数と傾き及び切片の関係式は（５）（６）及び（２）（３）式から下記の通りである。

$A = -2.427\ln(N) - 1.925$　　・・・（７）

$B = 14.67\ln(N) + 21.214$　　・・・（８）

Ａ：傾き　Ｂ：切片　Ｎ：センサの個数

（７）（８）式及び（４）式を用いて、センサの個数とＣＰＵ負荷率が定義できるのであれば、その時における最小のリクエスト間隔を定義することができる。

【００３０】

ここで取得しているセンサ情報は、動画のような画像データ、音声のような音声データを考慮せず、温度データのように値のみを管理サーバへと送信するセンサ情報を想定している。もし動画のような画像データ及び／又は音声のような音声データを考慮するのであれば、これらのセンサ情報は他のセンサ情報と情報量が違いＣＰＵ負荷に大きな影響を与えるために、画像データ及び／又は音声データによるリクエスト間隔とＣＰＵ負荷率の関係を精査し関係式を新たに導出する必要がある。もし他のセンサとは違う画像データを取得する場合は、例えば１０％程度負荷率が高くなることが様々な実験から分かっている。この場合におけるリクエスト間隔とＣＰＵ負荷率の関係式は、画像データのＣＰＵ負荷率に与える影響を考慮した場合、下記の通り算出することができる。

$C_r = -10.9\ln(req) + 77.346 + C_p$　・・・（９）

ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率　C_p：画像によるＣＰＵ負荷率（例、１０％）

（４）に示す関係式においてＣＰＵ負荷率からリクエス間隔を導出することは可能であり、その場合は下式から求めることが可能である。

$req = \exp((C_r - B - C_p)/A)$　　・・・（１０）

Ａ：傾き　Ｂ：切片　ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率　C_p：画像によるＣＰＵ負荷率（例、１０％）

また（１０）式は（２）（３）式を用いて算出することが可能であり、つまりセンサの数からＣＰＵに大きな影響を与える動画データをロボットから取得する場合においてもリクエスト間隔を算出することが可能である。

もし音声認識機能を起動させた場合は１０％程度負荷率が高くなる。この場合におけるリクエスト間隔とＣＰＵ負荷率の関係式は、音声認識のＣＰＵ負荷率に与える影響を考慮した場合、下記の通り算出することができる。

$C_r = -10.9\ln(req) + 77.346 + C_S$　　・・・（１１）

ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率　C_S：音声認識によるＣＰＵ負荷率（

１０％）
（４）に示す関係式からＣＰＵ負荷率からリクエス間隔を導出することは可能であり、その場合は下式から求めることが可能である。
ｒｅｑ＝ｅｘｐ（（C_r－Ｂ－C_S）／Ａ）　・・・（１２）
Ａ：傾き　Ｂ：切片　ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率　C_S：音声認識によるＣＰＵ負荷率（１０％）
また（１２）式は（２）（３）式を用いて算出することが可能であり、つまりセンサの数からＣＰＵに大きな影響を与える音声認識機能を起動させた場合においてもリクエスト間隔を算出することが可能である。
【００３１】
もしロボットが「座る」状態から「歩行」状態へと移行した場合、例えば１０％程度負荷率が「歩行」状態のほうが「座る」状態に比べ高くなる。この場合におけるリクエスト間隔とＣＰＵ負荷率の関係式は、ロボットの行動ステータスが「歩行」がＣＰＵ負荷に与える影響を考慮した場合、下記の通り算出することができる。
C_r＝－７．００６ｌｎ（ｒｅｑ）＋５０．７５６＋C_W　・・・（１３）
ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率　C_W：歩行によるＣＰＵ負荷率（１０％）
（４）に示す関係式を用いてＣＰＵ負荷率からリクエス間隔を導出することは可能であり、その場合は下式から求めることが可能である。
ｒｅｑ＝ｅｘｐ（（C_r－Ｂ－C_W）／Ａ）　・・・（１４）
Ａ：傾き　Ｂ：切片　ｒｅｑ：リクエスト間隔　C_r：ＣＰＵ負荷率　C_W：歩行によるＣＰＵ負荷率（例、１０％）
（１０）（１２）（１４）式をまとめて、センサの個数からリクエスト間隔を算出する関係式は下記の通りである。
ｒｅｑ＝ｅｘｐ（（C_{Th}－Ｂ－C_W－C_p－C_S）／Ａ）　・・・（１５）
Ｎ：　センサの個数Ａ：傾き　Ｂ：切片　C_{Th}：ＣＰＵ閾値　C_W：歩行によるＣＰＵ負荷率
C_p：画像によるＣＰＵ負荷率　C_S：音声認識によるＣＰＵ負荷率　ｒｅｑ：リクエスト周期
上記（１５）式を用いてセンサの個数からリクエスト間隔を算出する。また上記式によってリクエスト間隔からセンサの個数を定義することも可能である。
【００３２】
Ｆ．ロボットへのリクエスト要求

図９は、本実施形態のシステム動作概要を説明した図である。システムはAdministration Server(管理サーバ)(９０１)、Access Point(無線ＬＡＮ基地局）（９０２)、ロボット(９０３、９０４)から構成される。管理サーバは複数のロボットを管理、制御を行う。高精度に制御するのであれば、ロボットから様々な情報を管理サーバが取得し、その情報を基に制御を行うクローズドループ制御方式のような制御が必須となる。しかし、様々かつ詳細なセンサ情報を管理サーバが取得する場合は、ロボットに対して頻繁にリクエストを出す必要がある。その場合、多数かつ詳細なセンサ情報を取得するためにリクエスト間隔が非常に短く、ロボットのＣＰＵ使用率が歯止めなく上昇し、管理サーバとロボット間の通信遮断やロボット制御が不安定な状態になる可能性が高く、システムの信頼性、可用性、保守性を示すＲＡＳ(Reliability Availability Serviceability)は非常に低い。ＲＡＳの高い高精度な制御を行うにはＣＰＵを考慮したリソースマネージメントは必須であり、よってロボットのＣＰＵリソースを効率よく運用する必要がある。よってＲＡＳを高めるに当り、ロボットの行動ステータス毎にセンサの種類や個数を予め定義し、管理サーバがロボットからセンサ情報を引き出すリクエスト間隔を予め精査し導出した関係式から算出して、ＣＰＵ使用率が所定のＣＰＵ閾値以下になるように制御する。
【００３３】

ロボットは有線通信だけではなく、無線通信でも管理サーバから制御することが可能である。この場合無線電波の強度によって、管理サーバがロボットからセンサ情報を取得するためのセンサの個数や種類、リクエスト間隔をダイナミックに変化させる。無線電波の強度によってセンサの個数や種類、リクエスト間隔をダイナミックに変化させることはシステムのＲＡＳを高めるために非常に重要である。つまり無線強度が強い場所ではパケットロスが発生する確率が低いため、沢山の種類と個数のセンサを高頻度で管理サーバはロボットから取得するように制御する。逆に無線強度が弱い場所ではパケットロスが発生する確率が高いために、センサの種類は優先度が高いもののみで、取得するセンサの個数は出来るだけ少なくし、管理サーバがロボットからセンサ情報を取得するためのリクエスト間隔を出来るだけ長くとるように制御する。しかしあまりリクエスト間隔を長くとるとロボットサービスのＱｏＳが低くなるため、最低限のＱｏＳは確保できる程度にリクエスト間隔を長く設定するように制御する。

【００３４】

図５は、本実施形態においてロボットの行動ステータス及び無線電波強度毎に対するセンサの種類及び個数、ＣＰＵ閾値を定義した表を示すアクション定義ファイル（５０１）の説明図である。管理者はどのようなサービスを提供するか（行動ステータス、センサの種類、センサの個数）、提供するサービスの質（ＣＰＵ閾値）はどの程度かを定義する必要がある。更にロボットは有線だけではなく無線通信によっても制御されるために、無線電波の強度も考慮する必要がある。例えば無線電波が強い場合は、取得するセンサの個数を多く定義し、無線電波の強度が弱い場合は、最低限必要とするセンサの情報のみを取得するように定義する。これは無線電波の強度が強い場所では、パケットロスが発生する確率が低く高信頼に通信を行うことができる。よって電波強度が強い場合には、沢山のセンサ情報を取得することが望ましいからである。逆に言えば無線電波の強度が弱い場合は、パケットロスが多く発生するために、最低限必要とするセンサ情報のみを取得することが望ましい。

無線電波の強度を考慮する場合、図８のフローチャートにおいて、図５のアクション定義ファイルを用いる。そして、ステップ８０５でＣＰＵは、無線電波強度を取得し、行動ステータスと電波強度に従い該当するセンサ種類、センサ数、ＣＰＵ閾値を読み取り決定する。

【００３５】

図１０は、本実施形態において、センサの個数及びセンサの種類をダイナミック変化させ制御することを説明する図である（１００１）。上記までは、リクエスト間隔を決定するアルゴリズムを記載しているが、リクエスト間隔を決定後にセンサの情報を取得する制御手法に関して記載する。この場合、適時ロボットが状況に応じてセンサの個数及びセンサの種類を決定する手法もあるが、ここではベストエフォートに取得するセンサは、ロボットが持つ全てのセンサの情報をRound Robin的に取得することを示している。ここでは、一例として、アクション定義ファイルにより、センサ数が５個である場合、それらセンサ種類として最低限必要なセンサが２個（例、画像、音声認識）、他のベストエフォートのセンサが３個である場合について説明する。図では５個のセンサを確保することを想定し、管理サーバのＣＰＵは、最低限必要なセンサは２個常時確保するが、ベストエフォートに取得する箇所は図に示すように、初めはセンサ３、センサ４、センサ５と確保し、次の時はセンサ６、センサ７、センサ８と確保していく。このようにロボットがもつ最大のセンサ数まで順番に取得していくことをこの例では想定している。そしてＣＰＵは、確保した各センサに対して、リクエスト間隔に従い、リクエストを要求する。

【００３６】

Ｇ．センサ毎又はセンサ群毎のリクエスト間隔

以下では、管理サーバ（ＣＰＵ）は、算出したリクエスト間隔に基づき、さらに、個々のセンサ毎又は個々のセンサ群（クラスタ）毎のリクエスト間隔を設定する手法を説明する。管理サーバ（ＣＰＵ）は、設定した個々のリクエスト間隔をリクエスト間隔テーブル

に記憶し、そのテーブルを参照して、各センサ又はセンサ群に、リクエスト要求する。

【００３７】

図１１は、本実施形態において、取得するセンサの個数が１０個の場合における、センサ毎にリクエスト間隔を定義したリクエスト間隔テーブルを説明する図である（１１０１）。上記まで説明したセンサの取得の手法は、取得するセンサ全てに対して同じリクエスト間隔であったが、この例では個々のセンサに対してリクエスト間隔を設定できることもできる。この場合、管理サーバ（ＣＰＵ）は、前述の式（（４）式、（１０）式、（１２）式、（１４）式又は（１５）式等）を用いてセンサの個数からリクエスト間隔を算出し、その算出したリクエスト間隔を平均とし、更に予め定められた標準偏差又は分散を定義した上で分布を定義し、分布に従うようにランダムに割り当てる手法も考えられる。この場合具体的に割り当てる手法としては、分布は平均を中心として左右対称である分布を想定し、初めにセンサの個数を定義する。その場合、分布の左側の集合を負集合として定義し、分布の右側の集合を正集合として定義する。正集合の数と負集合の数は等しく定義するので、下式のように定義することができる。

Ｘ　＝　Ｙ　＝　Ｎ／２　　・・・（１７）

Ｘ：正集合の数　Ｙ：負集合の数　Ｎ：センサの個数

次に、分散に正集合の数または負集合の数を掛けてばらつきの総和を算出する。関係式を下式（１８）（１９）（２０）のように定義することができる。

σ＾２＊Ｎ／２＝σ＾２＊Ｘ＝σ＾２＊Ｙ＝Σ（x_i－ｍ）^{2}　・・・（１８）

σ＾２：分散　Ｎ：センサの個数　x_i：各センサのリクエスト間隔　ｍ：正集合または負集合のリクエスト間隔の平均　Ｘ：正集合の数　Ｙ：負集合の数

Σσ＾２＊R_i＝Σ（x_i－ｍ）^{2}＝σ＾２＊Ｘ＝σ＾２＊Ｙ　・・・（１９）

ΣR_i　＝　Ｘ　＝　Ｙ　＝　Ｎ／２　・・・（２０）

σ＾２：分散　R_i：各センサのばらつきを定義するための分散に対する割合　x_i：各センサのリクエスト間隔　ｍ：正集合または負集合のリクエスト間隔の平均

管理サーバ（ＣＰＵ）は、ばらつきの総和を算出後に、各々のばらつきを定義し、正集合の各々のリクエスト間隔を算出する。ここで各々のばらつきは分散を基にある割合を掛けて計算する。各々のリクエスト間隔は、リクエスト間隔の平均に乱数を用いて算出する標準偏差を足したものする。ここで正集合ならば足す標準偏差は正であり、負集合であれば足す標準偏差は負である。標準編を計算するために各々のばらつきを、乱数を用いて計算する。各々のばらつきの総和が先程算出したばらつきの総和に等しくなるように乱数を用いて計算する。それらの関係式を（２１）式のように定義する。ここで正集合、負集合の各々のリクエスト間隔を計算するに当り、各リクエスト間隔を順番に計算していくが、最後のリクエスト間隔を計算する場合のみ乱数を用いず（２２）式を用いる。

（x_i－ｍ）^{2}＝（ΣR_i－ΣR_j）＊ｒａｎｄｏｍ（０－１）　・・・（２１）

（x_i－ｍ）^{2}＝（ΣR_i－ΣR_j）　・・・（２２）

σ＾２：分散　（x_i－ｍ）^{2}：　ΣR_i：正集合または負集合のばらつきの総和　正集合の各々のリクエスト間隔の平均に対するばらつき　ΣR_j：計算されたばらつきの総和　x_i：各センサのリクエスト間隔　ｍ：正集合または負集合のリクエスト間隔の平均　Ｘ：正集合の数

【００３８】

管理サーバ（ＣＰＵ）は、これらの式を用いて各々センサに対してリクエスト間隔を定義することは可能である。下記に具体的な例を示す。例えば、図１１に示しているように、一例として、リクエスト間隔の平均を６［ｍｓ］、分散を４と定義する。この場合（１９）式を用いて、正集合、負集合の数は１０／２＝５である。次に（１７）式を用いて各センサのばらつきを定義するための分散に対する割合の総和は５である。管理サーバ（ＣＰＵ）は、次に各々のリクエスト間隔の平均に対するばらつきを計算する。（２１）式、（２２）式を用いて、乱数は今回適当に定義し、計算した結果は下記ようになる。

センサ１の分散に対する割合：(5-0)*乱数(0.4)=2

センサ２の分散に対する割合：(5-2)*乱数(0.166)=0.5

センサ３の分散に対する割合：(5-2.5)*乱数(0.2)=0.5
センサ４の分散に対する割合：(5-3)*乱数(0.5)=1
センサ５の分散に対する割合：(5-4)=1
上記割合を基に標準偏差を下記のように計算できる。
センサ１の標準偏差:　標準偏差=sqrt(4*2)=2.82[ms]
センサ２の標準偏差:　標準偏差=sqrt(4*0.5)=1.4[ms]
センサ３の標準偏差:　標準偏差=sqrt(4*0.5)=1.4[ms]
センサ４の標準偏差:　標準偏差=sqrt(4*1)=2[ms]
センサ５の標準偏差:　標準偏差=sqrt(4*1)=2[ms]
よって正集合におけるリクエスト間隔は、
センサ１のリクエスト間隔:リクエスト間隔=6[ms]+2.82[ms]=8.82[ms]
センサ２のリクエスト間隔:リクエスト間隔=6[ms]+1.4[ms]=7.4[ms]
センサ３のリクエスト間隔:リクエスト間隔=6[ms]+1.4[ms]=7.4[ms]
センサ４のリクエスト間隔:リクエスト間隔=6[ms]+2[ms]=8[ms]
センサ５のリクエスト間隔:リクエスト間隔=6[ms]+2[ms]=8[ms]
上記を用いて負集合におけるリクエスト間隔は、
センサ６のリクエスト間隔:リクエスト間隔=6[ms]-2.82[ms]=3.18[ms]
センサ７のリクエスト間隔:リクエスト間隔=6[ms]-1.4[ms]=4.6[ms]
センサ８のリクエスト間隔:リクエスト間隔=6[ms]-1.4[ms]=4.6[ms]
センサ９のリクエスト間隔:リクエスト間隔=6[ms]-2[ms]=4[ms]
センサ１０のリクエスト間隔:リクエスト間隔=6[ms]-2[ms]=4[ms]
上記のようにしてセンサ毎にリクエスト間隔を定義することは可能である。これらは一例であり別の設定方法も十分に考えられる。

【００３９】

図１２は、本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト周期を定義したリクエスト間隔テーブルを説明する図である（１２０１）。管理サーバ（ＣＰＵ）は、センサの個数から算出したリクエスト間隔を平均とし、上記記載の（１７）～（２２）式を用いて、更に予め定められた標準偏差又は分散を定義した上で分布を定義し、分布に従うように、クラスタ毎にリクエスト周期を設定することも可能である。

【００４０】

図１３は、本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト間隔を定義し、更にクラスタ毎に優先度を定義したリクエスト間隔テーブルを説明する図である（１３０１）。管理サーバ（ＣＰＵ）は、上記記載の（１７）～（２２）式を用いてクラスタ毎にリクエスト間隔を設定し、更にクラスタ毎に優先度も設定することが可能である。例えば、管理サーバ（ＣＰＵ）は、標準偏差又は分散を予め定義し上記記載の（１７）～（２２）式を用いて、クラスタ毎にリクエスト間隔を予め計算する。また、リクエスト間隔テーブルには、ロボットの行動ステータスが「歩行」の状態では、対人安全を考慮してロボットに備えつけられている複数のソナーから取得するセンサの情報をクラスタ化したクラスタ２は優先度を予め高く設定し、各種Actuatorの温度をクラスタ化したクラスタ3は優先度を予め低く設定する。そして、管理サーバ（ＣＰＵ）は優先度に従って、優先度の高い順に予め計算されたリクエスト間隔の中から短いリクエスト間隔を設定することも可能である。この場合は優先度が初めに決定されており、その後にリクエスト間隔を割り当てていく。

【００４１】

図１４は、本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト間隔を定義し、更に取得したクラスタ毎に優先度を定義し、更に取得したセンサ情報のデータ変化率も定義したリクエスト間隔テーブルを説明する図である。（１４０１）。ここでデータ変化率は

各々取得したセンサ情報の時系列データにおいて、ひとつ前の時刻のデータと比較しどの程度変化したかを示したデータである。データ変化率の関係式を下式に示す。

$D_r = D_c / D_p * 100$・・・（２３）

D_r：データ変化率　D_c：瞬時値　D_p：時系列データでひとつ前のデータ

管理サーバ（ＣＰＵ）は、各々データ変化率の情報を基にして、クラスタ毎に平均値を算出し、平均値に応じてクラスタ毎の優先度を設定することも可能である。例えば、管理サーバ（ＣＰＵ）は、あるクラスタの平均のデータ変化率が例えば、５００％になれば、そのクラスタのデータを取得する優先度を高く設定し、短いリクエスト間隔を設定し管理サーバがロボットから詳細にそのクラスタのデータを取得することも可能である。

【００４２】

図１５は、本実施形態において、管理サーバがロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト間隔を定義し、更に取得したクラスタ毎に優先度を定義し、更にクラスタ毎に取得したセンサ情報のデータ変化率も定義し、クラスタ毎の平均値を定義したリクエスト間隔テーブルを説明する図である。例えば、管理サーバ（ＣＰＵ）は、各Ａｃｔｕａｔｏｒの温度情報で構成された温度情報クラスタがある場合は、その温度の平均値を算出してその平均値が所定の閾値以上になった場合、温度情報クラスタの優先度高く設定し、短いリクエスト間隔を設定し管理サーバがロボットから詳細にそのクラスタのデータを取得することも可能である（１５０１）。

【００４３】

Ｈ．集約機能

上述の実施形態は管理サーバ側で、ロボットからＲＡＭに格納されているセンサ情報を取り出すリクエスト間隔を変化させるＣＰＵリソースのリソースマネージメントを行っているが、以下の実施形態は上述の実施形態に対してロボットが送信する情報を集約する集約機能をロボットに対して追加したものである。

ロボットがセンサ情報を管理サーバへ送信する場合、複数のロボットを管理するために大量のセンサ情報が管理サーバへと送信されてくる。その場合、管理サーバがロボットから送信される情報量をスループットとして計測し、計測したスループットが所定の閾値以上になった場合は管理サーバからロボットに対して送信パケットを集約するための制御コマンドを送信する。制御コマンドを受信したロボットは、優先度の低い情報をロボット側が集約し、管理サーバへと送信することで複数のロボットを運用する管理サーバの運用負荷を低減することができる。もし詳細な情報を知りたいのであれば、集約せずにセンサ情報を送信することも可能である。その場合は、管理サーバからロボットに対して集約しない制御コマンドを送信する。

【００４４】

図１６は、本実施形態において、管理サーバが、ロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト間隔を定義し、更に取得したクラスタ毎に優先度を定義し、更に取得したセンサ情報のデータ変化率も定義し、更にロボットがセンサ情報を送信するに当りセンサ情報を集約の有無を定義したリクエスト間隔テーブルを説明する図である（１６０１）。集約の有無はクラスタ毎のデータ変化率や、平均値によって決定されるクラスタ毎の優先度に大きく依存する。ここでは、管理サーバ（ＣＰＵ）は、例えば、優先度の低いクラスタを集約有りとして定義する。

【００４５】

図１７は、本実施形態において、管理サーバが、ロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト間隔を定義し、更に取得したクラスタ毎に優先度を定義し、更に取得したセンサ情報のデータ変化率も定義し、更にクラスタ毎に分けられた各センサ情報の時系列データからクラスタ間の相関係数を計算し最も相関が高いクラスタを定義したリクエスト間隔テーブルを説明する図である（１７０１）。ここでクラスタ間の相関係数を算出するに当り、クラスタ毎の平均値の

時系列データを用いる。管理サーバ（ＣＰＵ）は、クラスタ毎の平均値を時系列に取得し、そのクラスタの平均値の時系列データにおいての平均値を算出する。管理サーバ（ＣＰＵ）は、２つのクラスタ間において、時系列データの平均値からピアソンの積率相関係数を計算する。ピアソン積率相関係数は下式（２４）から計算する。

【００４６】

【数１】

$$r=\frac{\sum(x-m)(y-n)}{\sum\sqrt{(x-m)^2}\sqrt{(y-n)^2}} \quad \cdots(24)$$

x:ある時系列データ１　y:ある時系列データ2　m:xの平均　n:yの平均

【００４７】

管理サーバ（ＣＰＵ）は、全てのクラスタ間の相関係数を計算し、正の相関係数が最も高いクラスタを算出する。このように相関が高いクラスタをリクエスト間隔テーブルに予め定義しておくことで、データ変化率が非常に高いクラスタのデータを詳細に取得する場合、更にそのクラスタと相関が高いクラスタのデータも詳細に取得することも可能となる。例として表のクラスタ１はクラスタ２と相関が高いため、もしクラスタ１のデータ変化率が非常に高く、そのクラスタのセンサ情報を取得するための優先度が高くなった場合、相関が高いクラスタ２の優先度も高く設定することも可能である。

【００４８】

図１８は、本実施形態において、管理サーバが、ロボットからロボットのＲＡＭに格納されている複数のセンサ情報をクラスタ化して、クラスタ毎にリクエスト間隔を定義し、更に取得したクラスタ毎に優先度を定義し、更に取得したセンサ情報のデータ変化率も定義し、更に取得したセンサ情報の時系列データからクラスタ毎のマハラノビスの距離を定義したリクエスト間隔テーブルを説明する図である（１８０１）。マハラノビスの距離を計算するに当り、クラスタの平均値の時系列データを用いる。管理サーバ（ＣＰＵ）は、クラスタ毎の平均値を時系列に取得し、クラスタ毎の平均値の時系列データから平均値及び標準偏差を計算する。マハラノビスの距離は下式（２５）から計算する。

ｄ　＝　（ｘ－ａ）／σ　・・・（２５）

ｄ：マハラノビスの距離　ａ：時系列データの平均値　σ：標準偏差　ｘ：時系列データの瞬時値

管理サーバ（ＣＰＵ）は、クラスタ毎に時系列データからマハラノビスの距離を算出し、マハラノビスの距離が所定の閾値を超えた場合において、そのクラスタの優先度を高く設定する。優先度が高く設定されたクラスタのリクエスト間隔を短く設定し、管理サーバがロボットから詳細にそのクラスタのデータを取得することも可能である。

【００４９】

Ｉ．実施形態の効果

本実施形態によれば、ＩｏＴ機器のセンサの種類及びセンサの個数及びＣＰＵ閾値を機器の状態に応じて決定し、ＩｏＴ機器のＲＡＭ上に格納されているセンサ情報を引き出すためのリクエスト間隔を所定のＣＰＵ閾値を超えない程度に設定することによりリソース不足によるＩｏＴ機器の制御が不安定になる状態や通信遮断を防止することができる。

【００５０】

Ｊ．付記

なお、本発明は上記した実施例に限定されるものではなく、様々な変形例が含まれている。例えば、上記した実施例は本発明を分かりやすく説明するために詳細に説明したものであり、必ずしも説明した全ての構成を備えるものに限定されるものではない。また、ある実施例の構成の一部を他の実施例の構成に置き換えることが可能であり、また、ある実

施例の構成に他の実施例の構成を加えることも可能である。また、各実施例の構成の一部について、他の構成の追加・削除・置換をすることが可能である。

また、上記の各構成、機能、処理部、処理手段等は、それらの一部又は全部を、例えば集積回路で設計する等によりハードウェアで実現してもよい。また、上記の各構成、機能等は、プロセッサがそれぞれの機能を実現するプログラムを解釈し、実行することによりソフトウェアで実現してもよい。各機能を実現するプログラム、テーブル、ファイル等の情報は、メモリや、ハードディスク、ＳＳＤ（Ｓｏｌｉｄ　Ｓｔａｔｅ　Ｄｒｉｖｅ）等の記録装置、または、ＩＣカード、ＳＤカード、ＤＶＤ等の記録媒体に置くことができる。

また、制御線や情報線は説明上必要と考えられるものを示しており、製品上必ずしも全ての制御線や情報線を示しているとは限らない。実際には殆ど全ての構成が相互に接続されていると考えてもよい。

【符号の説明】

【００５１】

101 DB(データベース) 102 Administration Server(管理サーバ) 103 Access Point(無線LAN基地局)

201 RAM 202 CPU 203 NIC 204 DB

301 sensor1 302 RAM 303 CPU 304 NIC

401 アクション定義ファイル

501 アクション定義ファイル

601 リクエスト間隔とCPU使用率の関係を示したグラフ

701時系列に変化するリクエスト間隔とCPU使用率を示したグラフ

801 Start 802 アクション定義ファイル読み込み済 803 アクション定義ファイル読み込み 804 行動ステータス取得 805 CPU閾値、センサ種類、センサの個数決定 806 リクエスト間隔決定

901 Administration Server(管理サーバ) 902 Access Point 903 Robot

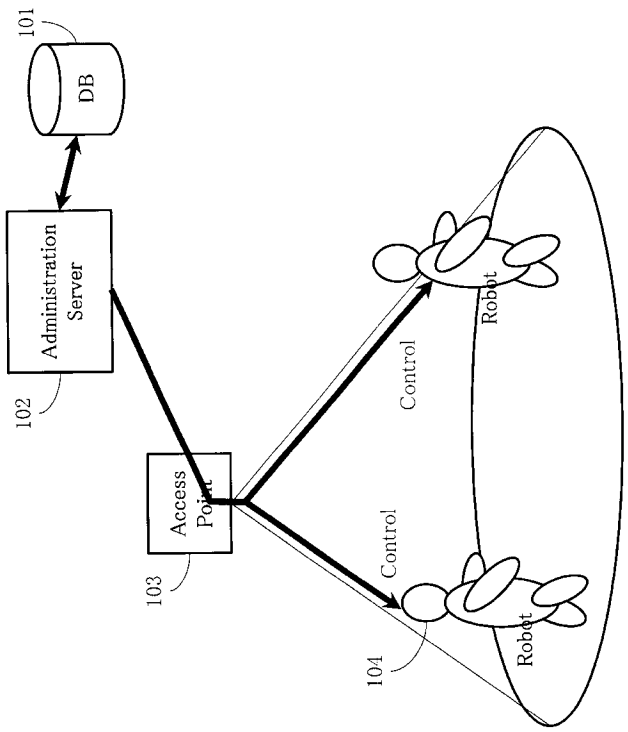
1001 Administration Server(管理サーバ) 1002 Access Point 1903 Robot

1001 Round Robin方式

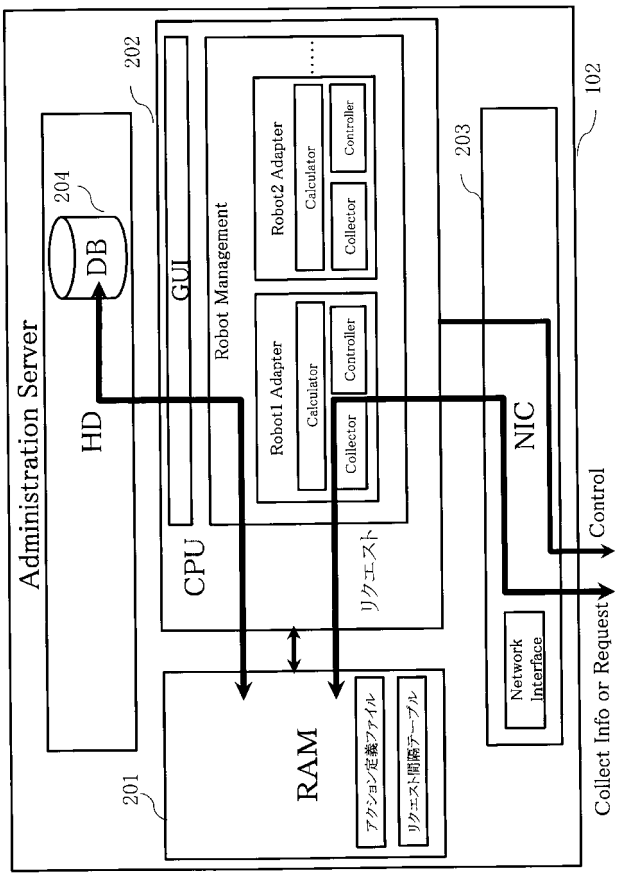
1101 センサ毎のリクエスト間隔を定義した表

1201～1801 センサをクラスタ化してクラスタ毎のリクエスト間隔を定義した表(1)～(7)

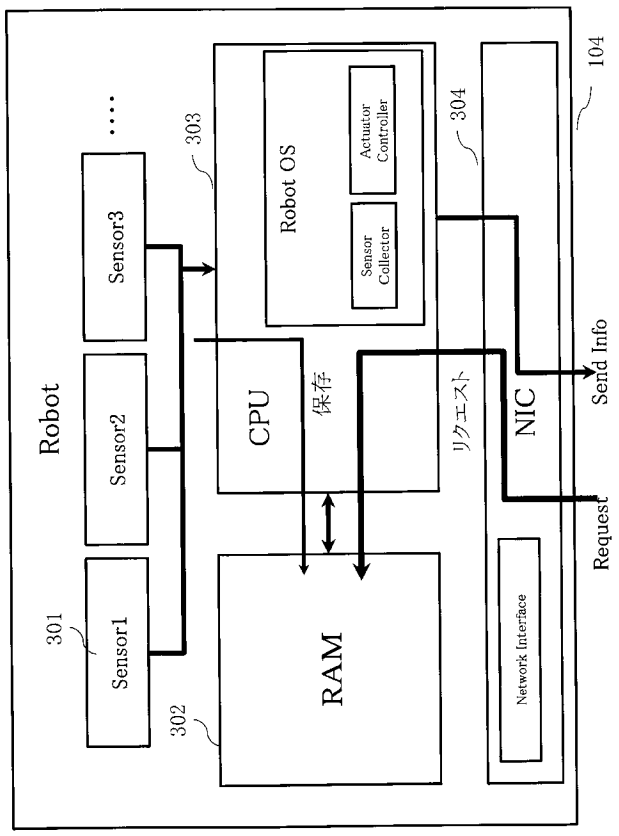
【図１】



【図２】



【図３】



【図４】

アクション定義ファイル　401

行動ステータス	センサ種類	センサ数(最大4000)	CPU閾値
座る	画像 音声認識 他ベストエフォート	20	50%
歩く	ジャイロ 他ベストエフォート	10	30%
立つ 声を聞く	画像 音声認識 音声方向認識 他ベストエフォート	20	40%
立つ 踊る	画像 他ベストエフォート	10	30%
寝る 声を聞く	音声認識(Sound Detect) 音声方向認識 他ベストエフォート	15	30%
座る 顔を見る	画像 音声認識 顔認識 他ベストエフォート	30	60%

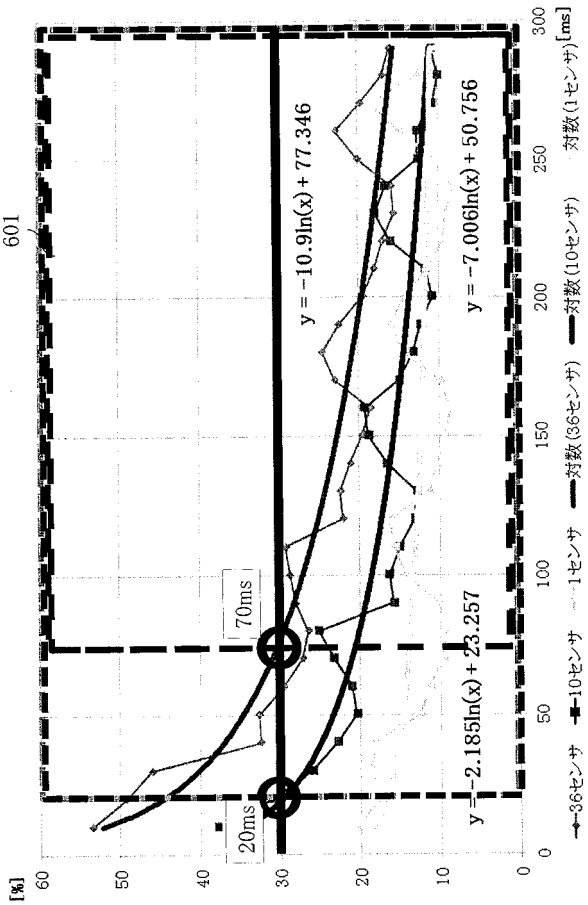
【図５】

501

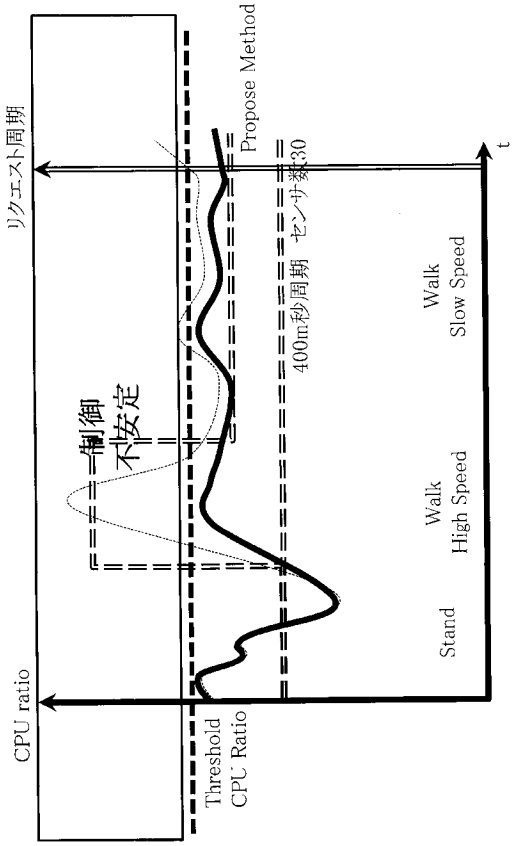
アクション定義ファイル

行動ステータス	電波強度	センサ種類	センサ数(最大4000)	CPU閾値
座る	強	画像 音声認識 他ベストエフォート	20	50%
座る	弱	画像 音声認識 他ベストエフォート	2	20%
歩く	強	ジャイロ 他ベストエフォート	20	50%
歩く	弱	ジャイロ 他ベストエフォート	2	20%
立つ 声を聞く	強	画像 音声認識 音声方向認識 他ベストエフォート	20	40%
立つ 声を聞く	弱	画像 音声認識 音声方向認識 他ベストエフォート	2	20%

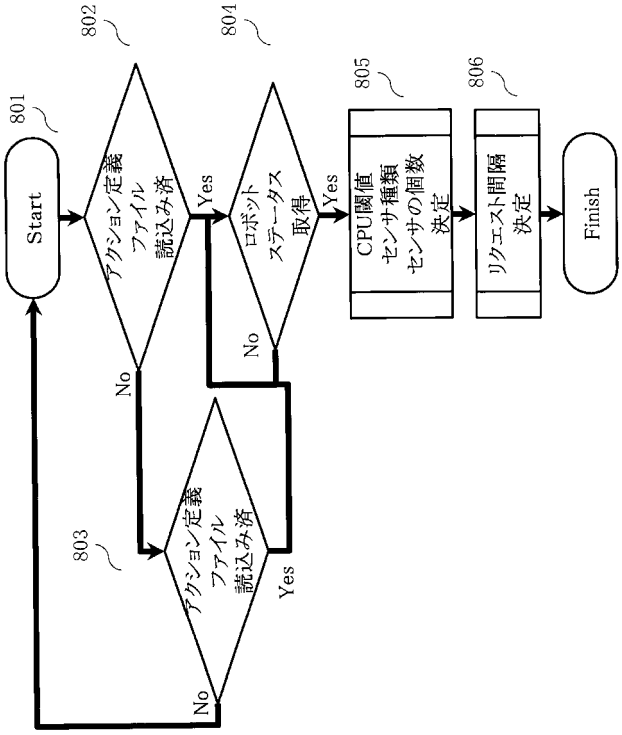
【図６】



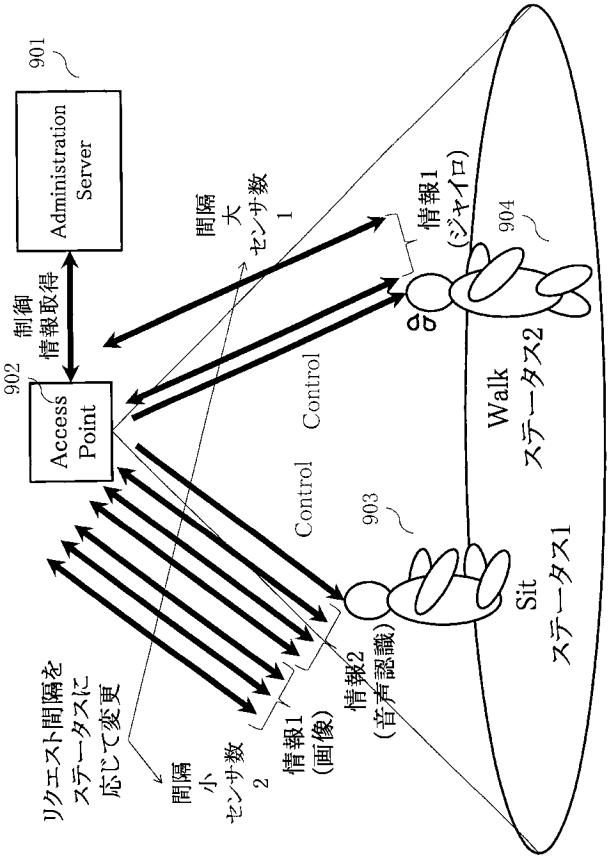
【図７】



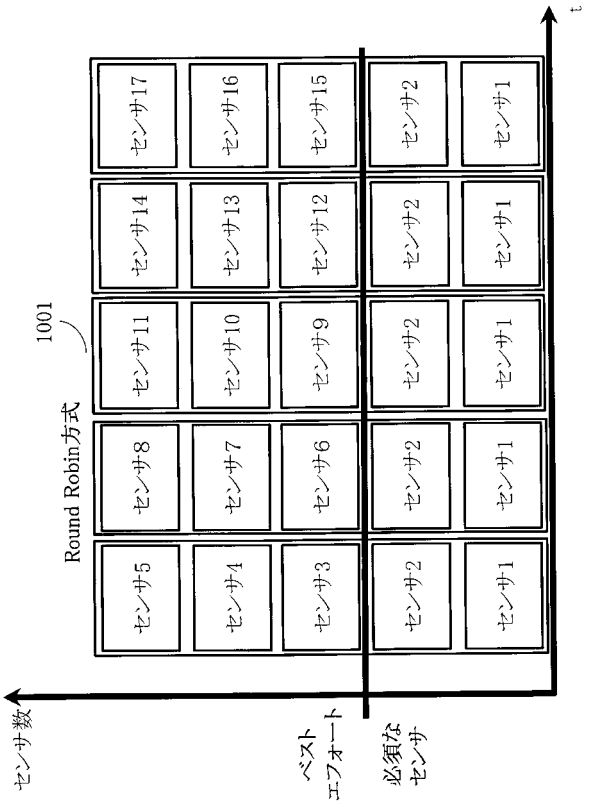
【図８】



【図９】



【図１０】



【図１１】

リクエスト間隔=6ms 標準偏差=2ms

1101

センサ	リクエスト間隔
センサ1	8.82ms
センサ2	7.4ms
センサ3	7.4ms
センサ4	8ms
センサ5	8ms
センサ6	3.18ms
センサ7	4.6ms
センサ8	4.6ms
センサ9	4ms
センサ10	4ms

【図１２】

リクエスト間隔=6ms　標準偏差=2.93ms

1201

センサクラスタ	センサ	リクエスト間隔
クラスタ1	センサ1	3ms
	センサ2	
クラスタ2	センサ3	10ms
	センサ4	
	センサ5	
	センサ6	
	センサ7	
クラスタ3	センサ8	5ms
	センサ9	
	センサ10	

【図１３】

1301

センサクラスタ	センサ	優先度	リクエスト間隔
クラスタ1	センサ1	高	3ms
	センサ2		
クラスタ2	センサ3	高	5ms
	センサ4		
	センサ5		
	センサ6		
	センサ7		
クラスタ3	センサ8	低	30ms
	センサ9		
	センサ10		

【図１４】

1401

センサクラスタ	センサ	データ変化率	優先度	リクエスト間隔
クラスタ1	センサ1	0.1%	低	300ms
	センサ2			
クラスタ2	センサ3	30%	高	5ms
	センサ4			
	センサ5			
	センサ6			
	センサ7			
クラスタ3	センサ8	50%	高	2ms
	センサ9			
	センサ10			

【図１５】

1501

センサクラスタ	センサ	平均値	データ変化率	優先度	リクエスト間隔
クラスタ1	センサ1	10	0.1%	低	300ms
	センサ2				
クラスタ2	センサ3	30	30%	高	5ms
	センサ4				
	センサ5				
	センサ6				
	センサ7				
クラスタ3	センサ8	20	50%	高	2ms
	センサ9				
	センサ10				

【図１６】

1601

センサクラスタ	センサ	平均値	データ変化率	集約	優先度	リクエスト間隔
クラスタ1	センサ1	10	0.1%	有	低	300ms
	センサ2					
クラスタ2	センサ3	30	30%	無	高	5ms
	センサ4					
	センサ5					
	センサ6					
	センサ7					
クラスタ3	センサ8	20	50%	無	高い	2ms
	センサ9					
	センサ10					

【図１７】

ピアソンの積率相関係数

$$r=\frac{\sum(x-m)(y-n)}{\sum\sqrt{(x-m)^2}\sqrt{(y-n)^2}}$$

m：xの平均
n：yの平均

1701

センサクラスタ	センサ	平均値	データ変化率	集約	相関	優先度	リクエスト間隔
クラスタ1	センサ1	10	0.1%	有	クラスタ2と高い	高	300ms
	センサ2						
クラスタ2	センサ3	30	30%	無	クラスタ1と高い	高	5ms
	センサ4						
	センサ5						
	センサ6						
	センサ7						
クラスタ3	センサ8	20	50%	無	全部低い	低	2ms
	センサ9						
	センサ10						

【図１８】

マハラノビスの距離 d：マハラノビスの距離

$$\mathbf{d}=\frac{(\mathbf{x}-\mathbf{a})}{\boldsymbol{\sigma}}$$

a：平均
σ：標準偏差

1801

センサクラスタ	センサ	平均値	マハラノビスの距離（絶対値）	集約	優先度	リクエスト間隔
クラスタ1	センサ1	10	1.5	無	高	300ms
	センサ2					
クラスタ2	センサ3	30	1	無	高	5ms
	センサ4					
	センサ5					
	センサ6					
	センサ7					
クラスタ3	センサ8	20	0.5	有	低	2ms
	センサ9					
	センサ10					

フロントページの続き

Ｆターム(参考) 5K201 BA01 BA02 CC02 CC10 DA02 DC02 DC04 DC05 EB07 EC06
EC08 ED09 EE02 EE04 EF10 FA01 FA03 FA04 FB01 FB03
FB06