

スマートフォンを用いた協調モニタリングのためのミドルウェア

A Middleware for Cooperative Monitoring by Smartphones

森 駿介^{†1} Yu-Chih Wang^{†2} 梅津 高朗^{†1,†3}
 Shunsuke Mori Takaaki Umedu
 山口 弘純^{†1,†3} 東野 輝夫^{†1,†3}
 Hirozumi Yamaguchi Teruo Higashino

1. はじめに

近年ではスマートフォンのような、高度な通信機能、演算能力、センサデバイスを併せ持つモバイル機器の普及が進んでいる。これらの機器は人が携帯するという特徴のため、そのセンシングデータを集めて分析することにより、人の行動やその周囲の環境などのきめ細かい環境情報を把握することが可能であると考えられる。例えば、イベントなどで混雑している地域の人々がどのように移動しているかを調べることができる群衆ナビゲーションや、人の行動圏における環境汚染調査など、長期間かつ低負荷な情報調査サービスなどへの応用が期待される。

しかし、事前に決められた特定領域を静的にカバーするワイヤレスセンサネットワーク (WSN) に対し、スマートフォンのようなモバイル端末は広域を常に移動する。そのようなモバイル端末を用いた協調モニタリングを実現する場合、全端末の存在や位置を網側で常時管理することは通信負荷やプライバシーなどの側面を考慮しても現実的でない。また、多様なセンサデバイスやカメラなどを持つスマートフォンからは、音声や動画など多様な情報が得られることが考えられるが、このようなデータを端末全てから集める場合は端末や通信帯域への負荷が大きいため、似たようなデータを持つ近い位置に存在する端末群から数ノードのみを選択してデータをアップロードさせるなど、端末間の協調により端末の適切なサンプリングを行うなどの制御ができることが望ましい。これらの課題に対し、モニタリングに対する要求だけを端末群に伝えておくことで、自律協調的にその要求を満足するような協調センシング機構が望まれる。

そこで本稿では、スマートフォン間の協調によるセンシングおよびデータの収集によるモニタリングを容易に実現するためのミドルウェアの設計について提案する。提案手法を用いた協調モニタリングのイメージを図 1 に示す。提案手法は、スマートフォンが歩行者によって携帯されており、3G などによる広域通信およびインターネットによりスマートフォンとデータ収集用サーバが通信可能な地域において利用され、WiFi-direct や Bluetooth によるスマートフォン間の近距離通信が利用可能であることを想定している。提案ミドルウェアはスマートフォン上のアプリおよびサーバ上のプログラムとして導入され、両者の連携により要求を満たすモニタリングを実現する。

ユーザは、集めたいデータを持つと考えられるノード群 (ノードグループ) を、位置やセンサの値などの条件によって指定する。また、ノードグループからのデータ収集を、期間、頻度、対象データ、モニタリングを行う端末数などによって指定することで、モニタリングの要求仕様を作る。これらの記述において、実際に必要となる細かい通信処理などは書く必要はない。

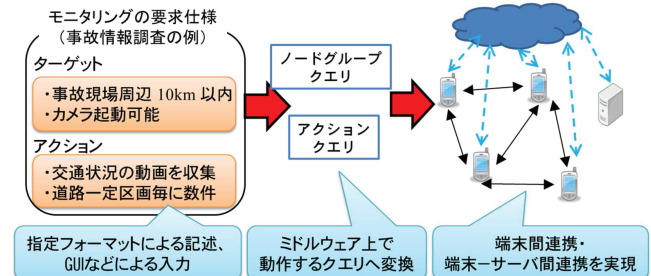


Fig.1 提案手法による協調モニタリング

この要求仕様をサーバ上のミドルウェアに与えるのみで、要求仕様の各条件および周囲の状況などに応じて各端末がモニタリングにおいて担うべき役割を判断する仕組みにより、モニタリングの実行を担当する端末を適切に絞り込み、適切な通信経路を利用して情報交換を行うことでモニタリングを行うことができる。

このように提案手法は、モニタリングの要求を情報として与えるだけで、モニタリングに伴う通信帯域および端末への負荷を抑制した上で要求を満足するモニタリングを行うことができる。

ここでは、シンプルなアプリケーション例を元にコンセプトを説明する。例えば、多くの歩行者が集まる交差点付近の混雑状況調査を行うためのモニタリングを、スマートフォンとサーバによって実行することを考える。各スマートフォンは、加速度センサや隣接ノード数などから得られる歩行者密度を常にモニタリングし、指定した領域において一定レベル以上の歩行者密度を検知した場合、その周辺領域から一定数のスマートフォンを選択し、それらが搭載されたカメラを用いて一定時間動画の撮影を行い、サーバへ報告する。サーバは、収集した道路状況の情報に基づき混雑度予想をする。

このようなモニタリングを実現するために、

- (i) 交差点から 30m 以内に存在し、歩行者密度がある閾値よりも高いノードグループ
- (ii) (i) の中心から半径 100m 以内に存在し、動画撮影が許可されているノードグループ
- (iii) (ii) から任意の 10 ノードを選択し、移動情報 (速度や方向) もしくはその平均値などを一定時間収集

といった形で、(i) と (ii) のようにその位置やセンサデータの特徴を元にノードグループを定義した上で、(iii) のようにそのノードの集合に対するデータ収集の要求を定義する形でアプリケーションの要求仕様を作成し、サーバ上で動作するミドルウェアへ与える。ミドルウェアはこの仕様記述を元にクエリを

^{†1} 大阪大学大学院情報科学研究科, Graduate School of Information Science and Technology, Osaka University

^{†2} Department of Computer Science National Tsing Hua University, Taiwan

^{†3} 独立行政法人科学技術振興機構, CREST, Japan Science Technology and Agency, CREST

生成し、配布を行う。各ノードは、クエリの情報をもとに位置やセンサデータの監視を常に行い、ノード間、ノード・サーバ間の連携により対象となるノードグループを決定し、モニタリングを行う。

このような提案手法について、いくつかのアプリケーション開発例について議論すると共に、実装労力を有効に抑制できることを確認し、提案手法の有用性を示している。

2. 関連研究

ワイヤレスセンサーネットワーク (WSN) におけるプログラミングを支援する手法はこれまでにいくつか提案されている。Abstract Regions¹⁾ は通信の接続性や地理的条件などにより領域ベースで定めることのできる、ノードのアドレッシングやデータ共有などの通信処理を抽象化したインタフェースとその実装を提供しており、開発者はこれらの高度なインタフェースを用いて容易に各ノードの振る舞いの実装を行うことができる。また、EnviroTrack²⁾ はオブジェクトベースの分散ミドルウェアシステムであり、環境トラッキングのための有用なインタフェース群を併せて提供している。センシングデータの特徴に基づきそれらを組み合わせることで、多数のセンサーノードを論理的な集合として扱う手法を用いている。

センサーノードの実体を隠蔽し、抽象化する手法についても提案されている。文献³⁾ においては、ノード単位の動作を規定するのではなくセンシングデータなどをデータセントリックに取り扱うアイデアについて述べられている。また、文献⁴⁾ では、ノードを抽象化したアプリケーションレベルで演算、通信、センシングといった要求を記述したスクリプトを配布し、各ノードに実装されたミドルウェアでその要求を満たす手法を提案している。

センサーネットワークの集中型プログラミングを実現するための取り組みである Kairos⁵⁾ は、集中的に個々のノード ID の管理、隣接ノードのグループ構成、任意ノードからのデータ取得を行うことができる機能をランタイムとして提供し、プログラムからこれらの処理を隠蔽することにより、ノード単位の形ではなくセンサーネットワーク全体の振る舞いを記述することを可能とする手法である。

提案手法では、地理的条件や時間、センシングデータの特徴などの条件によって端末の実体を隠蔽した形でモニタリングへの要求仕様を記述し、それを基にクエリを導出し、スマートフォンおよびサーバ上で動作するミドルウェアに配布することで、これらの端末による協調動作を行い、データ収集を実現する。既存手法の多くは、ライブラリなどの提供による実装負担を軽減することを目指しており、例えばセンサーノードのメモリ操作といった低レベル処理やセンサーノード全体へのデータ通知といったメッセージングの隠ぺいなどを実現している、これに対し提案手法は、具体的な端末やその数などを意識せずに与えられたモニタリングへの要求に対し、その要求と動的に変化する状況に応じて各端末が自律的または協調的な手順により、センシングのタイミングやセンシングデータの交換を自律的に決定できる機能によるデータ収集を実現する新しい手法で

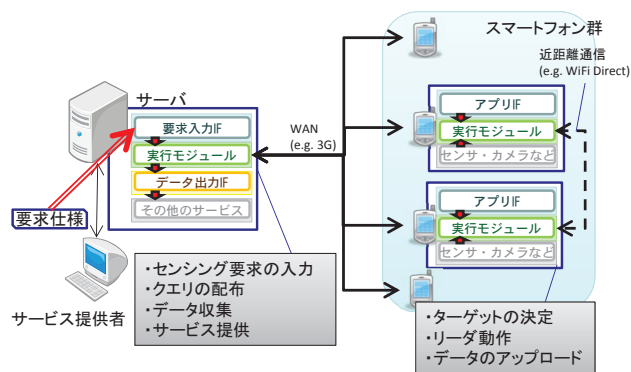


Fig.2 ミドルウェアのアーキテクチャ

ある。

3. 協調モニタリング実現のためのミドルウェアの概要

本論文では、多数のスマートフォンおよびサーバからなるネットワーク環境において、センシングおよびデータの収集を行うモニタリングの実現を支援する手法を提案する。

提案手法では、モニタリングのためのミドルウェアがスマートフォンおよびサーバ上で動作し、与えられた要求仕様に基づいてモニタリングを実現する機能を提供する。ユーザはモニタリングの要求を、領域やデータの特徴により対象となるノード群の指定と収集するデータの内容やタイミングによって定義し、サーバ上のミドルウェアに与える。入力として与えられたノードグループの定義とアクションからなるアプリケーションの仕様記述を元に、ミドルウェア上で動作するクエリを導出し配布することで、ミドルウェアが仕様記述に従った動作を行い、センシングおよびデータ収集を実現する。スマートフォンおよびサーバ上で動作するミドルウェアのアーキテクチャのイメージを図2に示す。

この章では、提案手法について、前提とする環境、ミドルウェアに与える要求仕様の概要、ミドルウェアの機能概要について述べる。

3.1 対象となるネットワーク

ここでは、提案手法が対象とするネットワークのモデルについて述べる。まず、アーキテクチャとして、多数のモバイル端末（具体的にはスマートフォンを想定）とモニタリング支援用のサーバ1台が存在するものとする。WAN(3G, LTE など)による通信が利用可能であり、任意のスマートフォンとサーバ間でデータの送受信が可能である。

スマートフォンはWAN用通信デバイスの他に、近距離通信 (Bluetooth, WiFi-direct など) のデバイスを持ち、近隣端末間での情報交換のために利用される。それぞれの歩行者は任意の方向および速度で移動しているものとするが、十分な数の歩行者が存在しており、全ての端末が近距離通信を介して通信可能である状況を仮定する。また、スマートフォンはGPSデバイスが装備されており、歩行者の位置をリアルタイムに取得できるものとする。その他、加速度センサ、傾きセンサ、マイク、カメラなどのセンサデバイスを持ち、これらの入力値が利

```

nodeconfig GeneralNodeConfig{
public:
    Position INTERSECTION = (X, Y);
    DataType data;
    DataType GetDataFromSensor();
    int count;
    int GetPedestrianCount(DataType input); //歩行者数を取得
    void ExecuteAction() {
        while(){ //定期的に実行する処理
            data = GetDataFromSensor();
            count = GetPedestrianCount(data);
            sleep(1min);
        }
    }
}

```

Fig.3 ノードタイプクラスの定義

用可能であるとする。

サーバで提案手法のシステムを統括するミドルウェアが動作しており、これは端末群に対するデータ取得を行うためのプログラムである。いずれのスマートフォンとも WAN 経由の通信が可能であるが、サーバはスマートフォンの存在を把握して居らず、通信は一般的なスマートフォンアプリのように端末側からのみ開始できるものとする。スマートフォンの位置情報は持たず、位置などによる物理的な対象端末の指定は不可能である。

3.2 モニタリング要求仕様の概要

提案手法では、実フィールド上のスマートフォンの数や位置などの詳細を隠蔽した抽象的なノードの集合（ノードグループ）をベースとしてアプリケーションへの要求仕様を入力として与える。ここでは要求仕様の入力について、要求仕様入力用に設計した記法を例として述べる。この記法は、アプリケーション全体の動作を表す要求仕様を記述するためのものである。

提案手法では、どのような端末（ここではノードと表記する）群を対象として、どのようなデータを収集するかという部分のみに注力したアプリケーション実装を可能とすることを目指している。そのためにこの言語では、(a) ノード側の処理仕様の定義、(b) モニタリングの対象となるノードグループの定義、(c) ノードグループに対するモニタリングの定義、という3つの要素を元に仕様を定義する。

まず、ノード側の処理仕様の定義はノードタイプクラスの定義として与える。“nodeconfig”キーワードに続いて記述する。変数、関数を定義することができ、以降のノードグループクラスやアクションの記述において参照される。ここで参照可能な関数は、対象となるデバイスで利用可能な関数およびAPI類のみである。また、特に定期的に行われる処理も関数 *ExecuteAction* として定義することが可能であり、図3に示す例ではセンサからの定期的なデータおよび値取得などの実行すべき処理が定義されている。

アプリケーションの仕様記述はこのノードの処理の定義を参照しながら記述する。図4に前述の歩行者流調査アプリケーションの仕様記述の例を示す。*CrowdedIntersectionArea* が交

```

nodegroup CrowdedIntersectionArea:public GeneralNodeConfig {
    condition:
        WithinCircle(INTERSECTION, 30m) && SetMinSize(30);
}
nodegroup SamplingArea : public GeneralNodeConfig {
    condition:
        WithinCircle(CrowdedIntersectionArea.Center(), 100);
}
action PedestrianFlow : public GeneralNodeConfig {
    target: SamplingArea;
    value: data, Average(count);
    duration: 1h;
    interval: 10min;
    node: 10;
}

```

Fig.4 歩行者流調査アプリケーションの記述例

差点から半径 30m の円形領域に存在し、歩行者密度がある閾値よりも高いという条件を満たすノードグループ、*SamplingArea* が *CrowdedIntersectionArea* の中心から半径 100m 以内に存在するノードグループ *PedestrianFlow* は *PedestrianCounter* から任意の 10 ノードを選択し、動画などのデータをアップロードするアクションを定義したクラスである。

時間や空間、センサデータの特徴などの条件によるモニタリング対象となるノードグループの定義をノードグループクラスの定義によって行う。“nodegroup”キーワードに続いて記述する。このクラスでは、ノードグループを定めるための論理式が与えられており、この式を真とするノードの集合がこのクラスに該当するノードグループである。この定義には、後述の位置制約関数、トポロジ操作関数、グループ処理関数が利用可能であり、位置やノードグループのサイズ、所属するノードの持つデータの平均値などについての条件を与えることができる。また、ノードタイプクラスの変数および関数の他、例えば、図4の“*CrowdedIntersectionArea.Center()*”などの形で、他のノードグループクラスをグループ処理関数によって参照して条件式を書くことができる。

アクションの定義は、モニタリングの仕様を記述するものであり、“action”キーワードに続いて記述する。アクションの定義は、ターゲットとなるノードグループクラスの指定、対象となる値（グループ処理関数などを用いて指定）の他、処理を行う期間、処理を行う間隔といった項目を時間の形で与える。また、モニタリングのためのデータ取得およびアップロードを行うノード数なども指定できる。モニタリングはターゲットであるノードグループクラスに該当するノードグループが実際に存在する場合に実行される。

これらの定義により、モニタリングへの要求仕様を与える。また、位置の条件、グループの条件やノード群を対象とした演算処理のインタフェースとして各種関数を提供しており、これらについて以下に示す。

表1に位置制約関数を示す。デフォルトでは、ノードの位置については特に制約はなく、フィールド上の全てのノードが対象となるが、例えば図4のように位置制約関数 *WithinCircle*

位置制約関数	内容	通信方法	付加情報
WithinCircle(position, distance)	座標 position から一定距離 distance 以内であれば真	ジオキャスト	position, distance
WithinArea(positionA, positionB)	座標 positionA, positionB を対角とする正方形領域内ならば真	ジオキャスト	positionA, positionB
WithinEachCircle(track, distance)	track に含まれる軌跡情報の各座標から一定距離 distance であれば真	ジオキャスト	track, distance
IsNeighbor(position, k)	座標 position の一定ホップ数 k 以内のノードであれば真	k-ホップブロードキャスト	k

Table.1 位置制約関数

トポロジ制約関数	内容
SetMaxRadius(double distance)	ノードグループの重心からの距離が distance 以下
SetMinRadius(double distance)	ノードグループの重心からの距離が distance 以上
SetMaxSize(int size)	ノードグループのノード数が size 以下
SetMinSize(int size)	ノードグループのノード数が size 以上
Select(NodeGroupClass group, int size)	ノードグループ group に属するノードから size 個のノードを抽出
SelectFrac(NodeGroupClass group, double percent)	ノードグループ group に属するノードから percent%のノードを抽出

Table.2 トポロジ操作関数

を用い、交差点の座標を示す INTERSECTION から半径 30m 以内に存在するノードのみが対象となる制約を与えるような指定ができる。

トポロジ操作関数（表 2）は、ノードグループクラスによって定まるノードグループについて所属ノード数などの制約を与えるものであり、その操作が成功する場合に関数は真を返す。例えば、図 4 のように “SetMinSize(30)” という記述により、最低 30 以上のノードが存在することが条件となる。また、“Select(10, CrowdedIntersectionArea)” という記述は、ノードグループクラス CrowdedIntersectionArea により定まったノードグループに所属するノードを対象に、そこから 10 ノードを抽出したグループであることを示す（この条件で抽出できるノード群が存在しない場合は偽を返す）。

グループ処理関数（表 3）は、ノードグループを構成する各ノードの持つ値を集約し演算処理を行った結果を返す関数群である。例えば、図 4 の Average(direction); という記述では、ターゲットである PedestrianCounter のノードグループの平均速度ベクトル direction の平均を計算し、取得することができる。

3.3 ミドルウェアの機能概要

本章では、要求仕様からのクエリ生成および、クエリに基づいたミドルウェアの動作について述べる。

3.3.1 クエリの生成について

仕様記述のノードタイプクラスから導出されたクエリは、各ノード上で保持する変数および実行される動作を定めるものであるため、記述内容とほぼ等価の内容を含む。ノードグループ

処理	内容	ノードから収集する情報
Max(&var)	ノードグループ group に属する各ノードの持つ変数 var の値のうち最大値を返す	var の値
Min(&var)	ノードグループ group に属する各ノードの持つ変数 var の値のうち最小値を返す	var の値
Sum(&var)	ノードグループ group に属する各ノードの持つ変数 var の値の合計値を返す	var の値
Average(&var)	ノードグループ group に属する各ノードの持つ変数 var の値の平均値を返す	var の値、(ノード数)
Median(&var)	ノードグループ group に属する各ノードの持つ変数 var の値の中央値を返す	var の値、(ID)
CountDistinct(&var)	ノードグループ group に属する各ノードの持つ変数 var について、重複しない値を持つノード数を返す	var の値
Center()	ノードグループ group を構成するノードの重心座標を返す	座標、(ノード数)
Size()	ノードグループ group の所属ノード数を返す	ID

Table.3 グループ処理関数の関数例

クラスから導出されたクエリ（ノードグループクエリ）には、ミドルウェアがノードグループを発見するため、ノード単独で判定を行うノード条件、ノードグループの構成が満たすべきトポロジ条件、ノードグループに所属する全てのノードに渡る条件であるグループ条件の情報が含まれる。また、他のノードグループによって参照されている場合は、その参照を実現するための通知送信を行うため、そのノードグループとの位置的な依存関係の情報を含む。これは、参照を行っているノードグループクラスの位置制約関数に従って決定する。また、アクションクラスから導出されたクエリ（アクションクエリ）には、その情報収集を実現するために、収集するデータおよびそのタイミングの情報が含まれているものとする。

各ノードおよびサーバのミドルウェアは、これらのクエリに含まれる情報をもとにモニタリングを実行する。本章では、ミドルウェアがクエリの情報を元に行う動作について述べる。

3.3.2 ミドルウェアの動作方針

ミドルウェアは、相手の ID が既知である場合のサーバ・ノード間の広域通信、ノード間のブロードキャストや位置情報を利用したジオキャスト通信といったネットワーク層の機能の実装を持ち、以降で述べる各プロセスにおいて利用可能であるものとする。また、全てのノードおよびサーバ上でミドルウェアが動作しており、各クエリはサーバから各ノードに配布され、全てのノードがクエリを保持しているものとする。ノード上のミドルウェアは、マルチタスクで動作しており、クエリの情報を元に、ノードタイプクラスに記述した関数 ExecuteAction の実行、およびノードグループの構築やモニタリングなどの処理を並行して実行する。サーバも同様にマルチタスクで動作しており、各ノードと連携してノードグループの構築やモニタリング処理を行う。仕様記述中には複数のノードグループクラスや

アクションが記述されているが、それらに対応する処理は並行に行われる。

このようなミドルウェアによって、モニタリングを実行する端末を適切にサンプリングすることで、通信帯域や端末への負荷を抑えた上で要求仕様を満たすモニタリングを実現するしくみについて述べる。ここではまず、要求仕様で定義された各ノードグループクラスに対応するノードグループを決定するための処理について述べる。各ノードは、ノードグループクエリ毎に定期的にノード条件の監視を行う **Monitor**、ノードグループの候補となりサーバとの連携を開始する **Candidate**、ノードグループとなりアクションの処理を実行する **Execution** といった状態を持ち、処理の進行と共に状態も推移する。サーバは、ノードグループクエリ毎に、特に処理を行わずに待機する **Waiting**、一定時間ノードグループ候補からの通知を受け付け、候補からノードグループを構成する **Organization**、アクションの処理を実行する **Execution** の状態を持つ。各ノードおよびサーバは、それぞれがノードグループクエリ毎にこれらの状態の間での遷移と対応する処理を実行する。各ノードの動作のフローチャートを図 5 に示した。

まず、各ノードは Monitor 状態の間でノードグループクエリに含まれるノード条件を常に監視しており、定期的に近隣ノードとの間で ID やノード条件の情報を共有している状態であるとする。ノード条件が真となった場合、確率 p によって Candidate 状態へ移行する。 p は与えられた仕様記述と情報共有で得た周囲のノードに応じて決定する。例えば、ある領域に存在するノードのうち 10 ノードという条件が与えられ、隣接ノード数などから推定したその領域全体のノード数が 100 であった場合は、 $p = \frac{10}{100} = 0.1$ のように、必要なノード数に近い数のノードのみが Candidate 状態へ移行するよう確率を導出する。Candidate 状態では、サーバに対してノードグループ候補となった通知である **Join Request** パケットを、グループ条件の成否を検証するために必要なデータ群と共に送信しそのまま待機する。

サーバは、最初に JoinRequest パケットを受け取ったタイミングで、Waiting 状態から Organization 状態へ移行する。一定時間（センシングの間隔などを元に決定）の間、他のノードからの Join Request パケットの受付を行う。一定時間経過後、JoinRequest の送信元のノードである候補ノード群から、ノードグループに属するノード群のリストを生成する。まず各候補ノードの情報を元にノードグループクエリに含まれるグループ条件を満たすようにノードを選択していき、ノード数や広さといったトポロジ条件を満たす組み合わせとなった場合にそれらのノード群がノードグループとなる。ノードを選択していく方針は、ランダムや先着順なども考えられるが、基本的には最初に JoinRequest パケットを受け取ったノードを起点に、より近いノードを優先的に選択していく方針を取ること、地理的条件に近いデータを 1 つのノードグループで取り扱い、通信や情報の集約におけるオーバーヘッドを減らすことを狙っている。残りの候補ノード群に対しても、ノードグループを見つけられる限り同様の処理を繰り返し、条件によっては

複数のグループが作られる。ノードグループが決定したとき、サーバは **Join Confirmation** パケットをノードグループに属する各ノードへ送信する。**Join Confirmation** パケットは、収集するデータの種類収集開始および終了時刻、収集の間隔の情報が含まれている構造体であり、アクションクエリに基づいて含む内容が決定する。

また、この時決定したノードグループが他のノードグループに参照されている場合、ノードグループクエリに位置的な依存関係の情報が含まれているため、それに従ってそのノードグループの候補となるノード群へ通知を行う。サーバは各ノードの位置を管理していないため、迅速な通知の受け渡しを位置関係に対応したジオキャスト通信により行う。この通知がノード間通信によって到達することで、ノードグループ間の情報伝達が行われる。なお、位置関係によりジオキャスト通信による受け渡しが困難だと考えられる場合は、参照されているノードグループがサーバに情報をアップロードし、参照を行うノードグループではその情報をサーバにおいてグループ条件と同じタイミングで参照する方法を取る。

通知を受け取ったノードは、その情報に基づいてセンサによるモニタリングを行い、終了時には取得した一連のデータをサーバに渡す。また、そのノードがノードグループの条件を満たさなくなった場合は、サーバに収集したデータを送信し、ノードグループから離脱する。サーバは、候補ノードのリストからランダムに新たなノードを選択し、通知を送信することで、次のノードにモニタリング処理が引き継がれる。

この後、ノードグループに所属するノードがその条件を満たさなくなった場合にはサーバへ **Leave Request** パケットを送信し、グループを離脱する。また、定期的（アクション実行時などのタイミング）にグループの再構成を行う。既にノードグループに所属するノードと共に、後からそのノードグループのノード条件を満たし Join Request パケットをサーバへ送信したノード群を併せて新たなノードグループ候補とし、上記のノードグループ構成の処理を行う。また、グループ全体が条件を満たさなくなった場合は各ノードへ **Eliminate Group** パケットを送信し、各ノードは初期状態へ戻る。

このように、各スマートフォンが自律的に状況を監視し、要求仕様の条件を真とする端末のみがモニタリング担当候補となる、また、モニタリング担当候補は、確率的にサーバへ報告を行う。さらにサーバは、報告を受けたモニタリング担当候補から必要最低限の数のモニタリング担当の端末を決定する。以上のような段階的なサンプリングによりモニタリング担当端末を決定することで、負荷を抑制したモニタリングを実現する。

4. 考 察

ここでは、協調モニタリングの実現における提案手法の有用性を示す。まず、提案手法によるモニタリングを用いたアプリケーションのいくつかの例を挙げると共に、提案手法を用いたデータ収集における要求仕様の解析により、提案手法がアプリケーション実現において有用であることを示す。

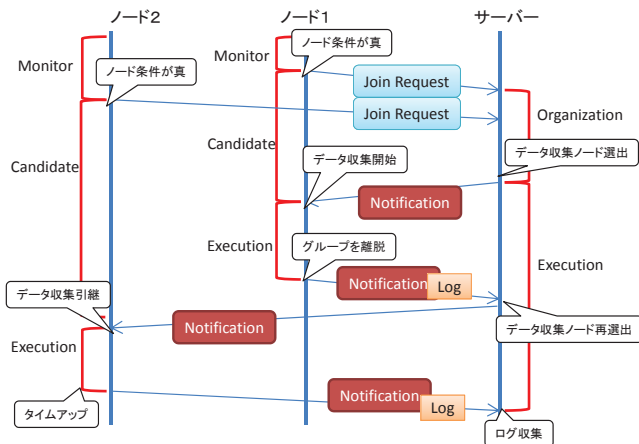


Fig.5 サーバ・スマートフォン間連携のフローチャート

4.1 アプリケーション例

本稿で提案する設計支援手法は、多くのモバイル端末が存在するネットワークにおいて、継続的なセンシングを行う際に非常に有用である。ここでは、いくつかのアプリケーション例とそのモニタリング要求仕様の記述を示し、提案手法の汎用性を確認する。

4.1.1 騒音検知アプリケーション

まず、騒音検知を目的としたシンプルなアプリケーションを例として示す。アプリケーションのイメージ図を図6に示す。フィールド上に存在する多数のスマートフォン端末は、それぞれがマイクデバイスを用いて定期的に騒音レベルの計測を行う。騒音レベルの計測値が閾値 ($NOISE_THRESHOLD$ とする) を超えた場合、それを検知した中心から周辺 $AREA_RADIUS$ m 以内の端末のうち 30 ノードを選択し、通知を渡す。1 時間の間、30 秒毎に騒音レベルを調査し、最終的に時間毎に平均値を算出する。このアプリケーションのモニタリング仕様記述を図7に示す。

ノードタイプクラス *ThermoSensorNodeType* には、騒音レベルの取得に関わる変数・関数を定義している。ノードグループクラス *Initiator* は騒音レベルの計測値が一定値を超えた場合に生成されるグループである。*SensingArea* は *Initiator* の周辺領域であり、これら 2 つのノードグループの定義により、まずは各ノードによるノイズの検知、続いてその周囲でのセンシングというように、段階的にターゲットを決定することができる。

4.1.2 来場者分布調査アプリケーション

次に、テーマパークやイベント会場などの比較的混雑した領域において、来場者がどの位置にどのくらい存在するか、といった情報を調べるため、来場者の持つスマートフォンなどの端末とのやりとりによって情報を収集するためのアプリケーションについて述べる。システムの概要を図8に示す。プライバシーの問題などを考慮し、各端末の位置情報などを収集するのではなく、領域毎におよそのノード数を調べる形で来場者の分布を調べるものとする。

このアプリケーションのモニタリング要求仕様が図9であ

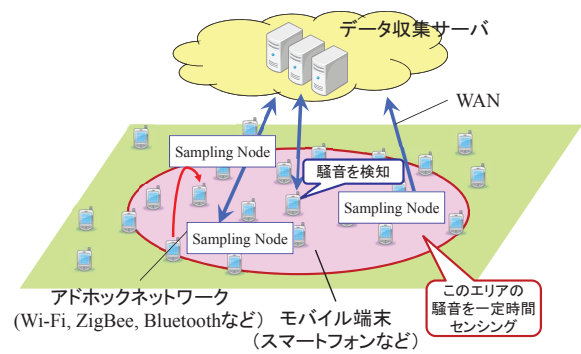


Fig.6 騒音検知アプリケーションのシステム概要

```

nodetype ThermoSensorNodeType {
double NOISE_THRESHOLD = 50;
double AREA_RADIUS = 100;
double GetNoiseLevel(); //騒音レベルを取得
double myNoiseLevel;
void ExecuteAction() {
while(){
myNoiseLevel = GetNoiseLevel();
}
}
}
nodegroup Initiator : public ThermoSensorNodeType{
condition:
GetNoiseLevel() > NOISE_THRESHOLD
&& SetMaxRadius(AREA_RADIUS);
}
nodegroup SensingArea : public ThermoSensorNodeType{
condition:
WithinCircle(Initiator.Center(), AREA_RADIUS)
}
action PedestrianFlow : public GeneralNodeConfig{
target: SensingArea;
value: Average(myNoiseLevel);
duration: 1h;
interval: 30sec;
node: 30;
}

```

Fig.7 要求仕様例：騒音検知アプリケーション

る。ノードタイプクラス *CountAppNodeType* は、隣接ノード数を計測できる端末の変数および関数の定義である。ここでは、近距離通信により隣接ノードの存在を把握する機能が利用可能と想定している。ノードグループクラス *CrowdedArea* により 30 ノード以上が隣接しているような過密な状態になった場合、*PeopleCounter* がその周囲 100m の領域のノードをグループとして定義している。アクションは *PeopleCounter* からのデータ収集を定義しているが、この例は 1 時間に 1 回のみ、全ノードを対象とする形で定義しており、ノードによる継続的なデータ収集ではなく、各条件が成立したときのスナップショットとしてのデータを取得したいというシンプルな要求も対応可能な例である。

4.2 実装労力の削減

提案手法では、モニタリングの要求仕様に基にクエリを生成

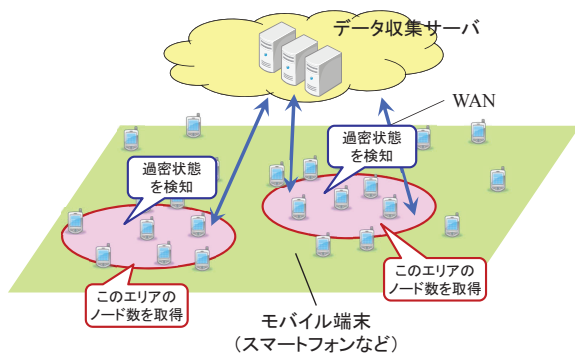


Fig.8 来場者分布調査アプリケーションのシステム概要

```

nodetype CountAppNodeType {
    int neighborNum;
    int GetCountNeighbor(); //隣接ノード数を返す
    int myId = GetMyId(); //端末の ID
    void ExecuteAction() {
        while() {
            neighborNum = GetCountNeighbor();
        }
    }
}

nodegroup CrowdedArea : public CountAppNodeType{
    condition :
        neighborNum > 30 && SetMaxSize(1);
}

nodegroup PeopleCounter : public CountAppNodeType{
    condition :
        WithinCircle(CrowdedArea.Center(), 100m);
}

action PeopleDistribution : public GeneralNodeConfig{
    target: PeopleCounter;
    value: Size();
    duration: 1h;
    interval: 1h;
    node: all;
}

```

Fig.9 要求仕様例：来場者分布調査アプリケーション

し、ノードおよびサーバで動作するミドルウェアに与えることで、協調モニタリングを実現する。この手法によって本来実装すべきプログラムの複雑さや煩雑さの軽減を行っていることについて示す。

まず、開発者が記述する必要のある仕様記述に含まれる内容は、ノードにおける通常時のセンシング処理、ノードグループの条件として与えられたノード単独で判定を行う条件およびノードグループ全体で満たすべき条件、期間や頻度により定まるセンシングデータ収集のためのアクションといった項目であり、これらはモニタリングによって達成したい項目が中心となっている。

それに対し、3.3.2 節で述べたように、ノード条件の成立時のサーバへの通知、通知の届いたノード情報の管理とノードグループの決定、ノードグループの各ノードに対する通知、ノードによるセンシング、サーバへのログ送信、データ収集担当の



Fig.10 提案手法による Android アプリ

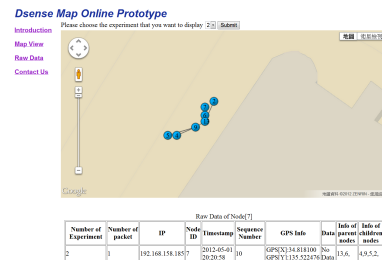


Fig.11 ウェブインタフェース

ノードの交代など、ノード間およびノードサーバ間の連携を実現するための複雑な手順に基づく通信処理によってデータ収集を実現している。

提案手法を用いることで、これらの処理を隠蔽し、モニタリングへの要求に注力した要求仕様の作成を行うことにより、労力を削減した上で協調モニタリングを実現することができると考えられる。

4.3 実環境における性能評価

提案手法の協調モニタリングによるアプリケーションのプロトタイプとして、サーバおよびスマートフォン端末用の情報収集アプリケーションプログラムの実装を行った。スマートフォン用プログラムは、図 10 のような、データ収集対象のスマートフォンの位置やその情報を分かりやすく表示する GUI を開発した。また、図 11 のように、情報収集対象のノード群の位置情報や所持データを Web インタフェースを通して分かりやすく表示することができる。これにより、データ収集作業におけるスマートフォン端末とデータとの対応付けが直感的に確認できるため、多数端末を管理運用する場面において煩雑さを抑制することができる。

このプロトコルは、3.2 節に示した歩行者流調査アプリケーションをよりシンプルなプロトコルにより実装している。このアプリによるデータ収集の性能について評価・確認を行った。

まず、性能評価実験の実施環境について示す。スマートフォン端末として、Samsung 社 Galaxy Nexus 12 台を用いた。各端末は GPS 機能により自身の位置情報を把握できる。また、IEEE802.11g 方式で Wi-Fi のアクセスポイントを 1 台設置し、各端末はこれを経由することで端末間の通信を行うことができる。ただし端末間の通信は、パラメーターとして与えられた通信可能距離 R_m 以内に両端末が存在するときのみ可能であり、 R_m 以上離れた端末間の通信は、複数の端末によるマル

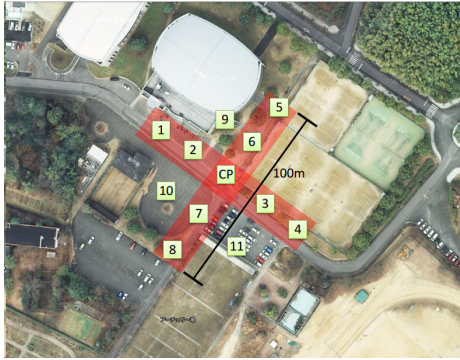


Fig.12 実機実験の実施領域

チホップ通信によって行われる。これらは、送信端末がパケットに位置情報を載せ、受信時にそれと自身の位置情報を元に通信可能かを判定することで実装している。

評価実験は大阪大学吹田キャンパス構内のおよそ 100m 四方の交差点周辺 (図 12 に示す) にて行った。交差点の中心に交差点ノードとして端末 1 台を配置し、他の 11 台の端末は周辺領域にグリッド上に配置する。この実験では、交差点ノードが周囲一定距離に位置する歩行者ノードへ Notification パケットを送信し、それを受信した端末群は、周囲の端末とツリー構造を構築しデータ収集を実行する。なお、ツリーの根ノードは、交差点ノードへ最も距離が近い GPS 座標を持つノードが選択される。

性能評価は、データ収集範囲が $R = 20, 30, 40, 50$ の各値に設定された場合のそれぞれの性能を対象とし実験を行った。まず、平均遅延および収集成功率については表 4 に示した。遅延については、一般に収集範囲が大きい場合にデータ収集に要するホップ数が大きくなるのが原因であると考えられる。ただし、 $R = 40$ から 50 への変化については、対象ノード数は増加しているが、必要な最大ホップ数は同程度だったため、遅延も同程度に収まっていると考えられる。また、収集成功率については全体として 70%以上の値を示している。ここで発生したロスについては、パケットの送信タイミングの衝突などが要因であり、それらを回避する機構を実装に含めることで、性能の改善は可能と考えられる。

また、発生パケット数についても図 4 に示した。シミュレーション上での $R = 50$ 時の平均発生パケット数が、制御パケットが 90、データパケットが 10.8 であったことと比較し、ほぼ同程度のパケット数で動作が行われていることが分かり、想定した通りの動作が行えていることが分かる。

今回示した実装は、提案手法に対して非常にシンプルなプロトコルを用いたプロトタイプである。今後はサーバとの連携、データ収集を行うノードの選出および引き継ぎアルゴリズムなどを拡充し、実践的な評価に取り組んでいきたい。

5. まとめと今後の課題

本稿では、スマートフォン間の協調によるセンシングおよびデータの収集によるモニタリングを容易に実現するためのミドルウェアの設計について提案した。提案ミドルウェアはス

R	20	30	40	50
Average Delays (Sec.)	13.963	17.812	21.443	20.598
Success Rate (%)	100	87.5	73.2	84.4
Control Packets	33	52	139	164
Data Packets	2	2	9	13

Table.4 実機実験における性能評価

マートフォンおよびサーバ上で動作し、入力として与えられた要求仕様からクエリを生成してスマートフォン群に配布する機能、および、クエリを元にスマートフォン間およびスマートフォン・サーバ間の連携によるモニタリングを実行する機能を提供する。ユーザは、位置やセンサの値などの条件によるノードグループの定義、およびノードグループに対するモニタリングするデータ内容、タイミングなどによるアクションの定義によって、モニタリングに対する要求仕様を与える。この要求仕様をサーバ上のミドルウェアに与えるのみで、スマートフォン群とサーバとの連携によりモニタリングの実行を担当する端末を適切に絞り込み、適切な通信経路を利用して情報交換を行うことでモニタリングを行うことができる。

今後は、提案手法による実装労力削減についての定量的評価を進めると共に、与えられた要求に対して、本稿で述べたデータ収集の手順の他、端末によるクラスタリングを利用したデータ収集など、ネットワーク環境や要求に適した手順を選択し、パラメータを決定してプログラムに組み込むような、システムの最適化手法について取り組みたい。また、ノードやサーバをモデル化の上で、サービス実行に伴う性能の保証を行う手法などについて検討していきたいと考えている。

References

- 1) M.Welsh and G.Mainland, "Programming Sensor Networks Using Abstract Regions," in *the Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation (NSDI 2004)*, 2004, pp. 29–42.
- 2) T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru, and A. Wood, "Envirotrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks," in *the Proceedings of 24th International Conference on Distributed Computing Systems (ICDCS 2004)*, 2004, pp. 582–589.
- 3) F.Zhao and L.Guibas, *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, July 2004.
- 4) A.Boulis, C.-C. Han, and M.Srivastava, "Design and Implementation of a Framework for Efficient and Programmable Sensor Networks," in *the Proceedings of the 1st international conference on Mobile systems, applications and services (MobiSys 2003)*, 2003, pp. 187–200.
- 5) R.Gummadi, O.Gnawali, and R.Govindan, "Macro-programming Wireless Sensor Networks using Kairos," in *the Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2005)*, 2005, pp. 126–140.