

WSNにおける効率的なエージェント位置管理機構

並木 勁汰^{†1,a)} 福田 浩章^{†1,b)}

概要: 無線センサネットワーク (WSN: Wireless Sensor Networks) は, センサデータを取得できる複数のノードによって構成される無線ネットワークである. WSN を構成するノードは電源が有限であり, 省電力を実現するためにノードの計算資源が極小に抑えられている. したがって, 複数のアプリケーションを運用する際, 必要なノードに対してのみプログラムを配備する必要がある. また, アプリケーションの稼働場所を変更する場合, プログラムの配備場所を変更する必要がある. WSN ではアプリケーションの開発者がプログラムの配備場所を把握した上でプログラムを記述しなければならない. 特に, プログラムの移動を考慮しなければならないモバイルエージェントアプローチではそれが困難である.

本研究では, WSN におけるプログラム間通信に起因する煩雑さを改善するため, 分散ハッシュテーブルによるエージェントの位置管理を行い, エージェントが移動しても確実に位置を探索できる機構 (CMSN: Chord for Mobile Agent on Wireless Sensor Networks) の提案と実装を行う. そして, CMSN によるエージェントの検索シミュレーションを行い, 検索に要する消費電力, 時間, 検索成功率を測定し, アドホックな検索手法と比べて効率よく探索できることを示す.

キーワード: 無線センサネットワーク, モバイルエージェントミドルウェア, エージェント検索, Agilla, CSN

An Efficient Agent Location Management on WSN

NAMIKI KEITA^{†1,a)} FUKUDA HIROAKI^{†1,b)}

Abstract: A Wireless Sensor Network (WSN) is typically deployed on a place in which no electric source is provided, meaning that its battery consumption is crucial. WSN applications require to execute many operations whose implementations are complex and consume a significant amount of battery (*e.g.*, lookup). To simplify the development of these applications, several mobile agent middlewares have been proposed (*e.g.*, Agilla). In an efficient manner, these proposals admin network hardware and react to the changes in the physical environment of a WSN. Applications for these middlewares are executed by communications among agents; thereby, a common operation is to look up agents. As existing agent lookup proposals do not support an efficient approach to look up agents, every lookup consumes a significant amount of energy and time. In addition, current approaches can fail their lookups if the target agent is able to migrate during a lookup operation. This paper proposes CMSN, an efficient and effective lookup for mobile agent middlewares. Our proposal is inspired by CSN, a distributed hash table algorithm for WSNs. We evaluate and compare CMSN in terms of performance time, effective lookups, and battery consumption.

Keywords: Wireless sensor networks, mobile agent middleware, agent lookup, Agilla, CSN

1. はじめに

無線センサネットワーク (WSN: Wireless Sensor Networks) は, センサを搭載した複数のノードによって構成される無線ネットワークである. 敷設の容易さから, 幅広い分野で WSN の応用が期待されている. 代表的なアプリ

ケーションとして, 環境や構造物のモニタリング, 対象物の検知と追跡, ヘルスモニタリング等がある. これらのアプリケーションは長期間運用することが一般的だが, WSN を構成するノードは電池で稼働することを想定しており, 電力の節約が最重要課題となる. ゆえに, 複数のアプリケーションを運用する際, 必要なノードに対してのみプログラムを配備することが重要になる.

WSN におけるミドルウェアの研究は数多く存在するが [1][2][3], モバイルエージェントアプローチ [3] は, WSN

^{†1} 現在, 芝浦工業大学
Presently with Shibaura Institute of Technology
^{a)} l09070@shibaura-it.ac.jp
^{b)} hiroaki@shibaura-it.ac.jp

に配備するプログラムをエージェントとみなし、個々のノードを渡り歩いて実行する処理を記述することで状況に応じた WSN の利用を可能にする。複数のアプリケーションを運用する際、資源の限られた一つの WSN において複数のエージェントを実行できるモバイルエージェントアプローチが適している。しかし、これを実現するミドルウェアには、効率的なエージェントの検索は実現されていない。その結果、あるエージェントが他のエージェントの場所（ノード）を検索するとき、フラッティングを行ってすべてのノードに検索をかけざるを得ない。

本研究では、WSN におけるモバイルエージェントミドルウェアに対し効率的なエージェント検索を行うエージェント位置管理機構 (CMSN:Chord for Mobile Agent on Wireless Sensor Networks) を提案する。これは、CSN[4] を基にした分散ハッシュテーブル (DHT:Distributed Hash Table) [5] を利用し、短時間かつ少ない電力消費でのエージェント検索を可能にする。また、検索対象のエージェントが移動中であっても、確実に検索できることを保証する。

2. モバイルエージェントミドルウェアにおけるエージェントの検索

ここでは、Agilla を例にモバイルエージェントミドルウェアについて概説し、検索の必要性について述べる。

2.1 Agilla

Agilla を利用したプログラムは、VM で解釈可能な 1byte ~ 2byte の命令セットで記述され、プログラムのコード容量を削減できる。また、メモリの制約が許す範囲で複数エージェントの同時稼働、タブルスペース [6] を用いたエージェント間通信、エージェントの配備場所指定、ノード間でのエージェント移動を実現している。その結果、プログラムは必要なノードだけにエージェントを配備し、エージェント間通信を行ってアプリケーションを実現できる。

2.2 エージェント検索

モバイルエージェントを用いた WSN のアプリケーションには様々な例 [3][7] があるが、以下に示す防火アプリケーションを例にエージェント検索の必要性を述べる。

図 1 に Agilla を用いた火災検知・追跡を行うアプリケーションの概略を示す。火災が発生すると火災検知エージェント (1) が検知し、追跡エージェントを見つけて通知を行う (2)。通知を受けた追跡エージェント (3) が、自身のクローンを火災発生場所周辺のノードへ配備する。そして、図中 3 のエージェントの存在によって火災を囲み、追跡を行うことができる。

2.3 検索における問題点

前述したように、エージェントが通知を行うには、まず

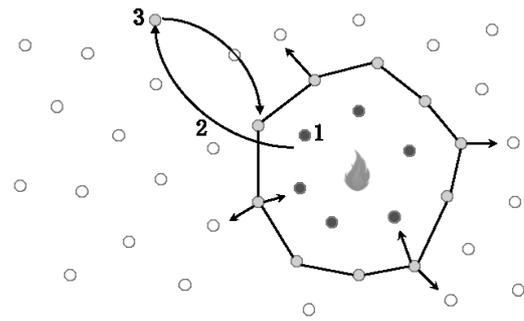


図 1 Agilla のアプリケーション例。1) 火災検知エージェント、2) 火災の通知、3) 火災追跡エージェント

Fig. 1 A WSN to detect and react to a fire. 1) An agent A detects a fire. 2) This agent notifies to an agent B. 3) The agent B travels and clones itself around the fire.

最初に通知先のエージェントを見つけなければならない。しかし、Agilla のような既存のミドルウェアでは効率的なエージェント検索は提供されていない。フラッティングを行ってすべてのノードに検索要求を転送することもできるが、エージェントは移動できるため検索結果のノードにエージェントが存在するとは限らない。そのため、基本的にはすべてのノードを移動し、アドホックに探さなければならない。これには、実行時間の増大、エージェント移動時の検索失敗、消費電力増大という 3 つの問題点が存在する。特に、電力に制約がある WSN において、消費電力の増加は重大な問題である。

3. エージェント検索のための DHT

本研究で提案する CMSN では DHT を利用する。ここでは、分散管理の必要性と DHT アルゴリズムを利用したエージェント検索について述べる。

3.1 エージェント位置情報の保管場所

WSN の情報を収集する PC をベースステーションと呼ぶ。このベースステーションで一括管理する場合、その構造ゆえにベースステーション周辺のノードにトラフィックが集中し、電力消費増大によるノード稼働時間の低下、WSN 全体の稼働時間低下という重大な問題を引き起こす。

そこで、CMSN では WSN 内のノード上に分散して保管する手法を採る。この手法は、個々のノードに対する負荷の分散が期待できる。WSN におけるノード間の通信は、自身と同性能なノード同士の通信であり、P2P 通信と考えることができる。CMSN では P2P での分散管理手法として、高い探索成功率と探索効率を有する DHT を用いる。

3.2 Chord / CSN

DHT は中央サーバを必要としない純粋な P2P 型のネットワーク型分散ファイルシステムを実現することができる技術である。識別子は、ファイル名やファイル自体から

ハッシュ関数を用いて生成される数値で表現される。また、多くの DHT アルゴリズムでは $O(\log N)$ で探索可能であり、探索時の負荷が偏らないように工夫されている。

DHT には複数のアルゴリズム [8], [9] が存在するが、Chord アルゴリズム [5] を利用する。Chord は DHT を代表するアルゴリズムのひとつであり、時計回りに識別子の値が増加していくリング状の識別子空間を導入している。リングを時計回りで測定した長さが距離となり、この距離の定義からノードの遠近が定まる（ネットワーク的な近さとは無関係）。識別子は値が 160bit である SHA1 ハッシュ関数の値を用いるため、識別子空間の大きさは 160bit となる。探索対象（エージェント）の識別子も同じ識別子空間にマップする。そして、探索対象の情報（エージェントの位置）は最も近いノードが管理する。また、各ノードは自身の次に大きい識別子を持つノード（Successor）への経路とノード自身の識別子から 2 の k 乗先のノードへの経路を管理する FingerTable を用いることで、ノード総数 N に対して $O(\log N)$ での探索が可能である。

CSN (Chord for Sensor Networks) [4] は、Chord におけるノード間の物理的距離を考慮し、リング状識別子空間を階層化することでセンサネットワークに適応させたアルゴリズムである。最上層リングの参加ノード数 N 、最大階層数 M に対し、 $O(M \log N)$ での探索を可能にする。

一方、CSN ではリング作成時にベースステーションを必要とし、自律的なリング生成という性質からリングに参加できないノードが存在する可能性がある。また、最上位層のリングからでなければ探索を行うことができず、探索対象が移動することを考慮していないため、エージェントの位置管理にそのまま適用することはできない。

4. Chord for Mobile Agent on WSN

CMSN では CSN で導入された階層型リング生成をベースに、任意のノードからの検索や検索対象の移動に対応する。また、CMSN の実装には代表的な WSN 用 OS である TinyOS[10]、および実装言語である nesC[11] を利用する。

CMSN は主に PacketManager, Neighbors, DHT の 3 つのコンポーネントにより構成される。

- PacketManager: 受信コンポーネント (Receiver), 送信コンポーネント (Sender) を内包するコンポーネント。Sender はレスポンス送信の際、パケットの衝突を避けるため、乱数による送信時間の調整を行う。
- Neighbors: Successor ノードの選定のため、隣接ノードの情報収集を行うコンポーネント。ブロードキャストを用いて最近隣ノード、隣接リストの構築を行う。
- DHT: DHT を構築するため、リングの情報を管理するコンポーネント。Successor, 自身の次に小さい識別子を持つノード (Predecessor), FingerTable, ハッシュテーブルに関わる情報を扱う。

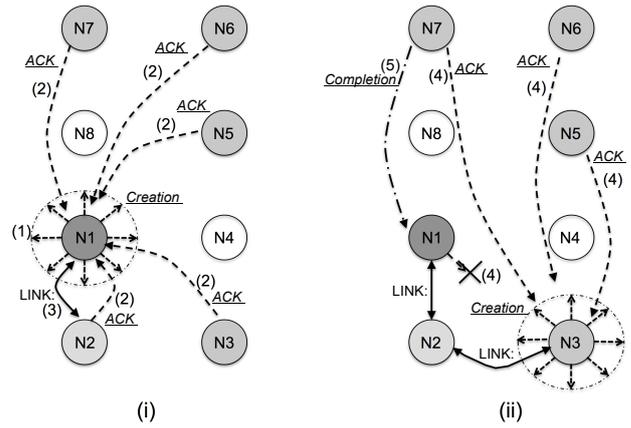


図 2 リング構築手順

Fig. 2 Procedures of a ring creation.

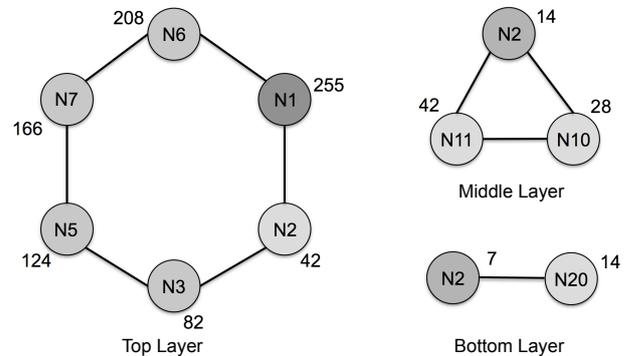


図 3 識別子割り当て

Fig. 3 Assign Identifiers.

4.1 DHT 構築手順

CSN ではクラスタヘッドの選出まで自律的に行い各階層のリングを生成する。一方 CMSN ではクラスタヘッドの選出は予め行い、実際のリング生成は自律的に行う。これは、1) 本研究の想定としてベースステーションを必要としない、2) ノードは一度設置したら動かないため、トポロジーは変化しない*1、3) 敷設した全てのノードをいずれかのリングに参加させる、という点を考慮したためである。

4.1.1 リングの構築

図 2 にリング構築手順を示す。ノード N1 がクラスタヘッド、N2, N3, N5, N6, N7 はひとつ下の階層でのクラスタヘッド候補である。図 2 (i) のように、リング構築はクラスタヘッドがリング構築メッセージ (Creation) をブロードキャストすることで開始される (1)。Creation を受け取ったノードのうち、Predecessor が決まっていない*2 ノードが Ack を返す (2)。返ってきた Ack の中で最も近いノード*3、N2 を Successor に選び、その旨を N2 に対して通知する (3)。このメッセージには現在リングに参加

*1 電池切れによるトポロジーの変化は起こる可能性があるが、本論文では対象外としている。

*2 Predecessor が決まっていないということは、別のノードの Successor になっていないことを意味する。

*3 現在の実装では、電波強度を示す RSSI を利用している。

しているノード数およびクラスタヘッドのノード ID の情報が含まれる。次に、図 2(ii) に示すように、Predecessor が決まった N2 は、同様の方法で Successor を選択するが、クラスタヘッドだけは *Creation* に対して Ack を返さない (4)。こうすることで、他に候補があるにも関わらずリング構築が終了してしまうことを防いでいる^{*4}。リング参加ノード数が限度数に到達するか *Creation* に対して Ack が帰ってこなかったときはリング構築完了メッセージを示す *Completion* をクラスタヘッドに送信する (5)。この操作を Top Layer から順に Bottom Layer まで繰り返し行う。

4.1.2 識別子割り当て

各 Layer のリング構築後、リング参加ノードの識別子割り当てを行う。図 3 に 8bit 識別子空間の例を示す。Top Layer では識別子空間をリング参加ノード数で割り、識別子を割り当てる (N2 の担当範囲は 0 から 42)。Middle Layer 以下では、一つ上の層の担当範囲をリング参加ノード数で割って割り当てる。図 3 で、N2 は Top Layer で 42 が割り当てられているが下位 Layer ではクラスタヘッド (N2) の Predecessor (N11) がその値を引き継いでいる。(Middle Layer における N2 の担当範囲は 0 から 14 となる)。これは、どの層のリングでも時計回りで検索を行うためである。

4.2 検索実行手順

図 3 において、N10 がエージェント A50 の検索リクエストを受信した場合を例に説明する。N10 は A50 のハッシュ値 (50) と自身のハッシュ値 (28) を比較する。50 は 28 よりも大きいため、Successor である N11 にリクエストを転送する。N11 も同様の手順を踏み、リクエストを N2 に転送する。N2 は Middle Layer のクラスタヘッドであるが、Top Layer のリング参加ノードでもあるため、4.1.1 で述べた手順によって上位リングと同一リングの Successor を両方を保持する。ここで、N2 は上位レイヤーとしての自身のハッシュ値 (42) と 50 を比較し、上位レイヤーの Successor (N3) にリクエストを転送する。N3 のハッシュ値は 82 であるため、A50 の位置情報は N3 をクラスタヘッドとする Middle Layer 以下にあることが確定する。N3 は同様に値の比較、転送を繰り返すことで最終的に対象のノードにたどり着く。

4.3 エージェントの移動

エージェントの移動中に検索が行われた場合検索は失敗してしまう。したがって、エージェント移動前にエージェントの位置管理を行っているノードに対し、移動中であることを通知する。例えば、N15 で位置管理をされている A1 が N10 から N20 に移動行う場合を考える。A1 は移動前に N15 へ *UnderMigration* メッセージを送り、A1 の移動完了

^{*4} N5 より N1 の方が N3 に近いため、N3 の Successor に N1 が選ばれてリングが完成してしまうという事態を防ぐ。

表 1 CMSN の設定
Table 1 Ring configuration.

対象	階層	個数
クラスタ数	Top Layer	1
	Middle Layer	9
	Bottom Layer	36
リング参加ノード最大数	Top Layer	9
	Middle Layer	4
	Bottom Layer	2

表 2 検索に要する時間 [ms] の比較
Table 2 A comparison of the lookup time [ms]

手法	DHT 構築	検索 1 回当たり	標準偏差
CMSN	96,789	192	196
ランダムウォーク	—	2,046	1,861

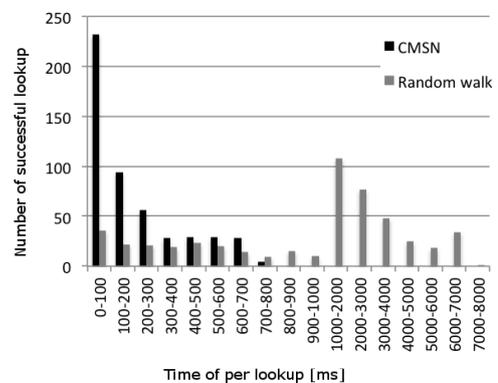


図 4 検索時間の度数分布
Fig. 4 Frequency distribution graph of lookup time.

後に N15 が管理する A1 の位置情報を更新する。また、現在の実装では仮に A1 の移動中に検索要求を受け取った場合は、A1 の移動終了を待ってから結果を返す。なお、A1 は自身の位置情報が N15 で管理されていることを、検索手順と同じ方法で知ることができる。

5. 評価

CMSN の有効性を確認するため、WSN 上をランダムに移動を繰り返すエージェントを検索する実験を行い、CMSN とランダムウォークそれぞれに要する、(1) 検索時間、(2) 検索成功率、(3) 消費電力を比較する。実験には TinyOS での消費電力測定を可能にした PowerTOSSIMZ[12] を利用する。シミュレーションは 12 × 12 のグリッド状ネットワークを想定する。リングの構築設定を表 1 に示す。

5.1 検索時間

表 2 にエージェント検索に要する時間の比較、図 4 に検索時間の度数分布を示す。CMSN による検索は多くが 100ms 以内、遅くとも 800ms 以内で完了する。一方、ランダムウォークは CMSN より遅い場合も多く、7s を超える

表 3 検索成功率の比較

Table 3 A comparison of the successful lookup rate

手法	検索回数	成功回数	成功率
CMSN	500	500	100%
ランダムウォーク	500	469	93.8%

表 4 バッテリー消費比較

Table 4 A comparison of the battery consumption

バッテリー容量: 21,600 [J]		
手法	DHT 構築	検索 1 回当たり
CMSN	5.75 (0.027%)	1.62 (0.008%)
ランダムウォーク	—	6.13 (0.028%)

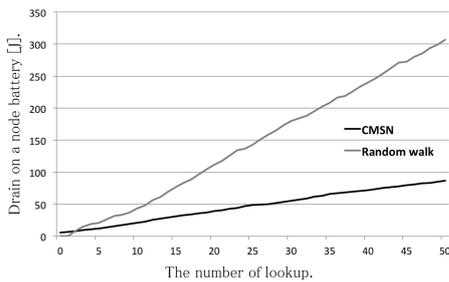


図 5 消費電力の比較

Fig. 5 A comparison of battery consumption

場合も確認できた。また、CMSN では DHT 構築に約 97s 要するが、これは運用開始時に一度だけの処理であり、数年間の運用を想定している WSN では実用上問題ない。

5.2 検索成功率

2.3 および 4.3 で述べたようにエージェントは移動するため、ランダムウォークによる検索が常に成功するわけではない。これを確認するため検索の成功率を比較する。ランダムウォークでは対象が見つかるまで隣接ノードにリクエストを転送し続ける。ただし、無限に転送することを避けるため、リクエストに TTL (Time To Live) 導入し、残り転送数が 0 になった時点で検索を終了し検索失敗とする。

表 3 に検索成功率の比較を示す。TTL はノード数 144 に対して十分だと考えられる 1,000 とする。CMSN が成功率 100%、ランダムウォークは 93.8% となった。つまり、ランダムウォークによる検索失敗はエージェントが存在しないことを意味しないが、CMSN での検索失敗はエージェントが存在しないことを意味する。これは、エージェントの協調で実現するアプリケーションにとって重要な性質である。

5.3 消費電力

表 4 および図 5 に消費電力の比較を示す。センサノードのバッテリー容量が 21,600J[12] であるため、CMSN は構築に約 0.027%、検索に約 0.008% バッテリーを消費し、ランダムウォークは検索に約 0.028% バッテリーを消費すること

がわかる。CMSN は DHT 構築に電力を消費するが、検索が数回行われた時点で CMSN の方が全体的な消費電力は低いことが分かる。また、7 回目の検索時には CMSN のバッテリー消費約 0.0728% に対し、ランダムウォークはその 2 倍以上の 0.1464% 消費することがわかる。

WSN は数年にわたる長期間での運用となるため、エージェントの移動による通信先変更が多くなると想定される。つまり、CMSN とランダムウォークの検索時における消費電力の差は、非常に大きくなると考えられる。したがって、CMSN におけるエージェント検索は消費電力を抑えることができるため有効であると考えられる。

5.4 関連研究

WSN では細かい転送制御が難しいため、フラッディングを行って目的の情報を保持するノードに対して検索要求を届けることが一般的である。その結果、膨大な経路制御パケットやデータパケットによる無駄な電力消費、複数の隣接ノードから同じデータを何度も受け取る imprecision や、地理的に近い場所で動作するノードが同じ情報を送信する overlap の問題が存在する。これらの問題に対し、SPIN[13] ではメタデータを利用し、周辺ノードがデータを要求しているか否かを事前に判定することで無駄なデータ送信を削除し省電力を実現している。また、Direct Diffusion[14] ではフラッディングによる検索要求/結果の転送履歴からノード間のリンク強度を調整することで徐々に特定のノードにだけ限定して転送を行い、冗長性を排除している。LEACH[15] やそれを改良した PEGASIS[16] では WSN にクラスタの概念を導入し、クラスタヘッドだけが情報を集約してベースステーションにデータを転送することにより上記の問題に対応している。これらの手法では省電力は実現できているものの、検索効率や速度に関しては考慮しておらず検索対象が移動することも考慮されていない。また、検索はベースステーションで行うことを想定しており、WSN 上のエージェントが任意の場所で検索することはできない。CMSN では、SPIN の 3-Stage Handshake を応用してリング構築を行い、DHT を利用することで送信先を限定し、消費電力を抑えている。

CSN [4] は WSN に DHT を導入した最初の研究であり、CMSN は CSN の影響を強く受けている。3.2 でも述べたように、CSN はリングの構築やデータ収集にベースステーションを利用することを前提にしているため、エージェントベースのミドルウェアには適していない。また、検索は常に最上位層のリングから行われるため、無駄な検索の転送が行われる可能性がある。CMSN は CSN を拡張し、任意のノードから検索や、エージェントの移動時の検索にも対応できる機構になっている。一方、モバイルエージェントベースのミドルウェアにおいて、エージェントの位置を管理する機構として MLDS[17] が存在する。MLDS では

予め決められた数のノードがクラスタを形成し、クラスタヘッドがその配下に存在するエージェントの位置情報を管理する。そして、各ノードに存在するエージェントを確認するため、 t 秒間隔で確認メッセージを交換しあうことでエージェントの移動に対応している。そのため、クラスタヘッドとノード間で常にメッセージをやりとりする必要がある。また、エージェントが移動してから、このメッセージが交換されるまでの間は古い位置情報を返信してしまう可能性がある。CMSN では、一度リングを形成した後はエージェント移動時にしか移動のためのメッセージ交換は行われない。また、移動中の検索要求はブロックするため、常に正しい位置情報を返すことができる。

6. まとめ

WSN では環境の変化、ネットワーク接続、分散運用などの問題に取り組む必要がある。また、ノードは電池で稼働するため、これらのアプリケーションは効率的な運用を行うことが求められる。これらの問題を隠蔽し、簡潔にアプリケーションを記述するためにミドルウェアが提案され、その中でモバイルエージェントベースのミドルウェアには効果的に通信相手の位置を検索する手段が必要になるが、既存のシステムでは実現されていない。

本研究では CSN を拡張し、WSN において効果的にエージェントを検索する機構を実現し、検索時間、検索成功率、消費電力を比較することで有効性を示した。今後の課題として、現在はクラスタヘッドを固定しているため、そのノードだけ多くの電力を消費してしまう。そこで、LEACH などで導入されているクラスタヘッドのローテーションや、定期的にノードをスリープさせることによって、システム全体としての消費電力削減を実現する必要がある。

参考文献

- [1] Madden, S. R., Franklin, M. J., Hellerstein, J. M. and Hong, W.: TinyDB: An acquisitional query processing system for sensor networks, *ACM Transactions on Database Systems*, Vol. 30, No. 1, pp. 122–173 (2005).
- [2] Abdelzaher, T. F., Blum, B. M., Cao, Q., Chen, Y., Evans, D., George, J., George, S., Gu, L., He, T., Krishnamurthy, S., Luo, L., Son, S. H., Stankovic, J. A., Stoleru, R. and Wood, A. D.: EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks, *Proceedings of the 24th International Conference on Distributed Computing System*, pp. 582–589 (2004).
- [3] Chien-Liang, F., Roman, G.-C. and Lu, C.: Agilla: A Mobile Agent Middleware for Self-Adaptive Wireless Sensor Networks, *ACM Transactions on Autonomous and Adaptive System*, Vol. 4, No. 3, pp. 1–26 (2009).
- [4] Ali, M. and Uzmi, Z. A.: CSN: A network protocol for serving dynamic queries in large-scale wireless sensor networks, *Proceeding of the Second Annual Conference on Communication Networks and Services Research*, pp. 165–174 (2004).
- [5] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F. and Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications, *ACM SIGCOMM Computer Communication Review*, Vol. 31, No. 4, pp. 149–160 (2001).
- [6] Gelernter, D.: Generative Communication in Linda, *ACM Transactions on Programming Languages and Systems*, Vol. 7, No. 1, pp. 80–112 (1985).
- [7] Butun, I., Morgera, S. D. and Sankar, R.: A Survey of Intrusion Detection Systems in Wireless Sensor Networks, *IEEE Communications Surveys and Tutorials*, Vol. 16, No. 1, pp. 266–282 (online), DOI: 10.1109/SURV.2013.050113.00191 (2014).
- [8] Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S.: A Scalable Content-addressable Network, *SIGCOMM Comput. Commun. Rev.*, Vol. 31, No. 4, pp. 161–172 (online), DOI: 10.1145/964723.383072 (2001).
- [9] Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D. and Kubiawicz, J. D.: Tapestry: A Resilient Global-scale Overlay for Service Deployment, *IEEE Journal on Selected Areas in Communications*, Vol. 22, pp. 41–53 (2004).
- [10] Hill, J., Szewczyk, R., Woo, A., Hollar, S. and Culler, David andr Pister, K.: System architecture directions for networked sensors, *ACM SIGARCH Computer Architecture News*, Vol. 28, No. 5, pp. 93–104 (2000).
- [11] Gay, D., Levis, P., Behren, R., Welsh, M., Brewer, E. and Culler, D.: The nesC language: A holistic approach to networked embedded systems, *ACM SIGPLAN Notices*, Vol. 38, No. 5, pp. 1–11 (2003).
- [12] Perla, E., Catháin, A., Carbajo, R. S., Huggard, M. and Goldrick, C. M.: PowerTOSSIM z: realistic energy modelling for wireless sensor network environments, *Proceedings of the 3rd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and networks*, pp. 35–42 (2008).
- [13] Heinzelman, W. R., Kulik, J. and Balakrishnan, H.: Adaptive Protocols for Information Dissemination in Wireless Sensor Networks, *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom '99*, New York, NY, USA, ACM, pp. 174–185 (online), DOI: 10.1145/313451.313529 (1999).
- [14] Intanagonwivat, C., Govindan, R., Estrin, D., Heidemann, J. and Silva, F.: Directed Diffusion for Wireless Sensor Networking, *IEEE/ACM Trans. Netw.*, Vol. 11, No. 1, pp. 2–16 (online), DOI: 10.1109/TNET.2002.808417 (2003).
- [15] Heinzelman, W. B., Chandrakasan, A. P. and Balakrishnan, H.: An Application-specific Protocol Architecture for Wireless Microsensor Networks, *Trans. Wireless. Comm.*, Vol. 1, No. 4, pp. 660–670 (online), DOI: 10.1109/TWC.2002.804190 (2002).
- [16] Lindsey, S., Raghavendra, C. and Sivalingam, K. M.: Data Gathering Algorithms in Sensor Networks Using Energy Metrics, *IEEE Trans. Parallel Distrib. Syst.*, Vol. 13, No. 9, pp. 924–935 (online), DOI: 10.1109/TPDS.2002.1036066 (2002).
- [17] Bhattacharya, S., Fok, C.-L., Lu, C. and Roman, G.-C.: MLDS: A flexible location directory service for tiered sensor networks, *Computer Communications*, Vol. 31, No. 6, pp. 1160–1172 (2008).