無線センサーネットワークにおける
プログラムの動的配備

末永 俊一郎

博士（情報学）

総合研究大学院大学
複合科学研究科
情報学専攻

平成 20 年度
(2008)

本論文は総合研究大学院大学複合科学研究科情報学専攻に
博士（情報学）授与の要件として提出した博士論文である。

審査委員：

吉岡 信和（主査）	
計 宇生	
山田 茂樹	
米田 友洋	
大須賀 昭彦	電気通信大学
本位田 真一	東京大学

（主査以外はアルファベット順）

Dynamic Deployment of Programs in Wireless Sensor Networks

Shunichiro Suenaga

DOCTOR OF
PHILOSOPHY

Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)

March, 2009

A dissertation submitted to
the Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Advisory Committee:

Nobukazu Yoshioka (Chair)	
Yusheng Ji	
Shigeki Yamada	
Tomohiro Yoneda	
Akihiko Osuga	The University of Electro-Communications
Shinichi Honiden	The University of Tokyo

(Alphabetical order of last name except chair)

内容梗概

無線センサネットワーク (Wireless Sensor Networks： WSN) は，センサデータを取得できる多数のノードが無線によって接続されるネットワークである．WSN は，ノードを物理的に敷設するだけでセンサデータの取得を可能にすることから，有線ネットワークを敷設することが困難であった場所における適用が期待されている．従来，WSN は特定のアプリケーションのために敷設し，そのアプリケーションだけのために運用されるのが一般的であった．WSN の実用化が進む将来においては，費用対効果の観点から，WSN が予め敷設された環境下で，複数のアプリケーションを運用する形態が現実的となる．

WSN のアプリケーションは，アプリケーションを実現できるプログラムを，WSN を構成するノードに配備することによって実現される．敷設された WSN で複数のアプリケーションを運用する場合，アプリケーションを追加する際にはプログラムを WSN に追加すること，アプリケーションを削除する際にはプログラムを WSN から削除すること，アプリケーションの動作を変更する場合には配備されたプログラムを変更する必要が生じる．こうした運用形態においては，アプリケーションを構成するプログラムを動的にノードに配備する手法が有効であると既存研究によって示されている．

本研究では，WSN 内部でアプリケーション固有の計測を行い，WSN 内部でアプリケーション固有の処理を行うアプリケーションを複数構成することを想定する．これらのアプリケーションは空間的な計測を必要とするため複数のプログラムで構成される．本論文では，こうしたアプリケーションを *LCA*(Locally Centralized Applications) と定義する．既存研究は，*LCA* を構成する際に以下の 2 つの課題を抱える．

課題 1:*LCA* のように WSN 内部で複数のプログラムが連携するアプリケーションを構築する際に，プログラマにアドホックな実装を強制し，その結果，*LCA* の継続的な構成を困難にする課題がある．WSN 内部でアプリケーション固有の処理を実施する *LCA* を継続的に構成するために，適したアーキテクチャが必要となる．

課題 2:*LCA* が稼動場所を変更する場合，既存研究では複数のプログラム間で通信を実施し，移動の同期を行う必要が生じる．WSN ではノード間の通信の際にパケットロスがあることが良く知られている．複数のプログラムが移動を行う際に通信に高く依存をした手法をとると，一部のプログラムが移動を開始できない可能性，移動に失敗する可能性が生じる．この結果，アプリケーションは，新たな稼動場所におきプログラム数の不足により処理を継続できない課題がある．

本研究ではこれらの課題を解決し，WSN の実利用を促進するために，*LCA* の継続実行を目標とし，以下 2 つの提案を行う．

提案 1:アプリケーション（*LCA*）を構成する際のアーキテクチャの提案
Master，*Slave-S*，*Slave-M* の 3 つのコンポーネント化されたプログラムから構成される

アーキテクチャを用いて LCA を構成する．
提案 2:複数プログラムの動的配備手法の提案
一つのプログラム ($Master$) が他のプログラム ($Slave\text{-}M$，$Slave\text{-}S$) を生成し，複数プログラムを動的に配備する手法を提案する．

本研究ではこれら提案に対する評価をシミュレーション環境を用いて実施した．提案するアーキテクチャによって既存手法より LCA の継続的な構成を行う際の信頼性を，約 15%〜50%向上した．また，提案する動的配備手法によって，動的配備の成功率を約 20%〜40%向上した．以上の結果から，提案手法は，複数のプログラムで構成されるアプリケーション (LCA) を既存の WSN 内で複数運用する際の有効な手段の一つとなることがわかった．

Abstract

Wireless Sensor Networks (WSN) are large-scale networks consisting of spatially distributed nodes with sensing capabilities to monitor conditions of the physical environment . They are deployed in environments that are difficult to access by traditional wired networks. In the past such WSNs have mostly been designed and deployed with specific applications in mind. Recent developments however show an increasing demand for WSNs to provide a shared network infrastructure for a multitude of different applications running in parallel.

Applications in WSNs are realized by deploying and executing distributed program modules on multiple nodes of the network. The simultaneous operation of multiple applications thus makes it necessary to enable dynamic adding, modifying and deleting of program modules throughout the network. Existing work so far has dealt with this issue by proposing dynamic deployment methods to effectively reprogram the nodes at run-time.

In our work we assumed the scenario of multiple applications running in a given WSN, each of which consist of multiple program modules running on specific nodes to conduct sensing and data processing tasks. We call such applications *LCA*s (Locally Centralized Applications) in this paper. Existing solutions suffer from two drawbacks considering the deployment and execution of *LCA*s.

Problem 1: Related work lacks a proper architecture for structuring *LCA*s, resulting in an ad-hoc development of *LCA*s and difficulties in their continued execution.

Problem 2: When a *LCA* needs to change its deployment location, existing work requires communication and synchronization between the program modules that need relocation. On the other hand, WSN are a highly unstable communication environment and thus entail two risks. For one, message loss can result in certain program modules to fail receiving the relocation instructions from the controlling process and thus will not relocate correctly. Furthermore, communication failures during the migration process itself can result in certain program modules to be lost.

To address the above issues and improve the practical development of *LCA*s we propose two novel approaches.

approach 1: An architecture for structuring *LCA*s.

We propose constructing a *LCA* by dividing responsibilities into three components, namely a *Master*, *Slave-S* and *Slave-M*.

approach 2: A dynamic deployment process for distributed program modules.

The *Master* component generates the other component programs (*Slave-S*, *Slave-M*) and controls their deployment dynamically.

We evaluated the efficiency of our proposal in a simulation environment. The results show that the architecture achieves a success rate of constructing LCAs 15% to 50% higher than in existing solutions. In addition, the dynamic deployment process reached deployment success rates 20% to 40% higher than in related work. We conclude that the proposed approaches are an effective way to operate multiple LCAs simultaneously in WSN environments.

目 次

図 目 次

表 目 次

第 1 章

序言

1.1 無線センサネットワーク (WSN) とは

無線センサネットワーク (Wireless Sensor Networks: WSN) は，センサを搭載したノードによって構成される無線ネットワークである．WSN は，無線によって “火災” や “物の存在有無” 等の現実世界の現象を把握することを可能にし，有線ネットワークを敷設することが困難であった場所や，敷設の容易性から実世界における適用が期待されている．WSN の代表的なアプリケーションとして，環境のモニタリング，対象物の検知と追跡，ヘルスモニタリング，構造物のモニタリング等がある [Kuorilehto 05][Culler 04].

WSN は，カルフォルニア大学バークレー校 (University of Calfornia, Berkeley: UCB) における SmartDust プロジェクト [Sma] により開発された超小型の MEMS 技術を起源とし，1999 年に Pister ら [Kahn 99] によって提唱された．SmartDust プロジェクトは 2001 年，NEST プロジェクト [NES]，WEBS プロジェクト [WEB]，CENS プロジェクト [CEN] を設立することにより，その役割を終えた．2002 年に，UCB で試作された WSN を構成するノード MICA[Hill 02] が，Crossbow Technology 社 [Xbo] から発売された．同時に，NEST プロジェクトが，MICA に搭載可能なオープンソースのオペレーティングシステム TinyOS[Hill 00] を公開した．この結果，WSN のアプリケーション開発環境が提供され，WSN の研究が広く開始された.

WSN はモバイルアドホックネットワーク (Mobile Adhoc NETwork: MANET) と良く似た技術であるが，センサを用いた計測処理を実施する以外に技術的な違いが大きく 2 つある [Tian 05]．ひとつの違いとして，WSN の電力は有限であることが挙げられ，もう一つの違いとして，WSN の計算資源が極小であることが挙げられる．WSN では，個々のノードが電池で稼動することを想定しており，WSN の最大かつ共通の課題は電力になる．省電力を達成するために，CPU やメモリ等の計算資源は極小に抑えられる．代表的な WSN

ノードとして MICAz[MIC] がある[*1]．図 1.1 に MICAz の外観を，表 1.1 に MICAz の仕様を示す．表 1.1 より，WSN におけるノード計算資源は極小であることがわかる．それに対して MANET では，個々のノードが電池で稼動する想定は WSN と共通しているが，WSN と異なり充電を実施することを前提としている．加えて，MANET は PC や PDA を想定し，計算資源に対する制約の厳しさは WSN に比して緩い．これらの技術的な違いに加えて，利用者の観点では，MANET における各ノードはノードを対等に見立て，システム利用者にノード間の 1 対 1 での通信を提供することを主眼にしているのに対し，WSN の場合，計測されたセンサデータが最終的にはシステム利用者のために特定のノードに集約され，N 対 1 通信を主眼にしている点が異なる．また，MANET は頻繁に移動を行うノードを想定し，ノードの移動を想定するのが常であるのに対し，WSN ではノードの移動は一般に想定しない．表 1.2 に，上で述べた MANET と WSN の違いを示す．

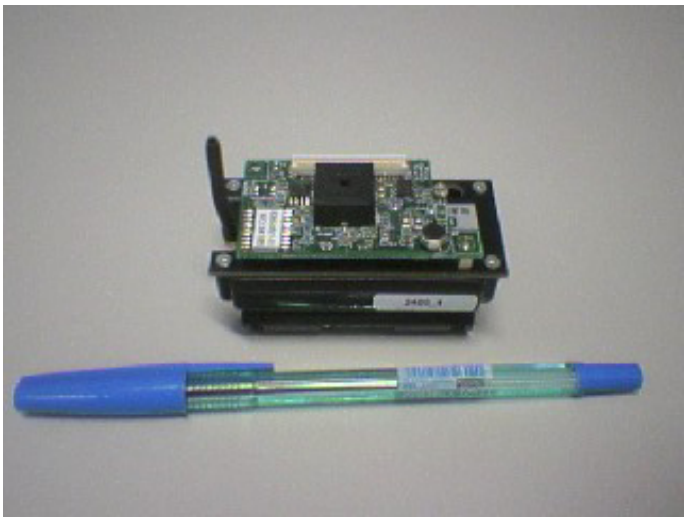

図 1.1: MICAz の外観

表 1.1: MICAz の仕様

プログラムメモリ	データメモリ	CPU	電源
128Kbyte	512Kbyte	7.4MHz	単三電池 2 本

図 1.2 に，WSN を用いたアプリケーションの一般的な構成を示す．WSN 内の各ノード

[*1] MICA の後継機であり，Zigbee での通信が可能．2008 年 7 月から Crossbow Technology 社は MICAz の後継機である IRIS の販売を開始している．

表 1.2: MANET との違い

項目	計算資源	電源	移動性	通信
WSN	極小：表 1.1 参照	単三電池 2 本等	一般的にはノードの移動を想定しない.	N 対 1
MANET	WSN に比し潤沢：PC，PDA，携帯等	充電可能	ノードは頻繁に移動する.	1 対 1

は，計測処理を実施し，無線ネットワークで取得したセンサデータをベースステーションに転送する．ベースステーションは，LAN に接続可能な PC であり，WSN の情報を収集する役割と，アプリケーションにセンサデータを提供する役割を担う．システム運用者は，ベースステーションを介して WSN の情報を収集し，取得されたデータを利用してアプリケーションを実現する．図 1.2 から，WSN の主要技術が以下のように抽出できる．

プログラミング言語・モデル

プログラマがアプリケーションの要求を満たす計測処理を記述するためのプログラミング言語，プログラミングを容易にするプログラミングモデル.

ミドルウェア

プログラマから，分散，組み込みの性質をもつ煩雑な WSN のプログラミングの負担を軽減するために，各ノードに搭載されるミドルウェア.

オペレーティングシステム

ハードウェアを隠蔽し，プログラミング可能にするための基本機能を持つオペレーティングシステム.

ハードウェア

省電力，極小な資源を実現する MEMS 技術を集約した無線通信可能なハードウェア.

ネットワーク

ノード間のルーティングや，トポロジの制御，通信時のセキュリティ確保等を実施する無線ネットワーク.

WSN における主要技術の動向・研究動向は表 1.3 に示すとおり整理できる．プログラミング言語・モデルは，研究初期は低レベルなプログラミング言語の提案やライブラリの提供

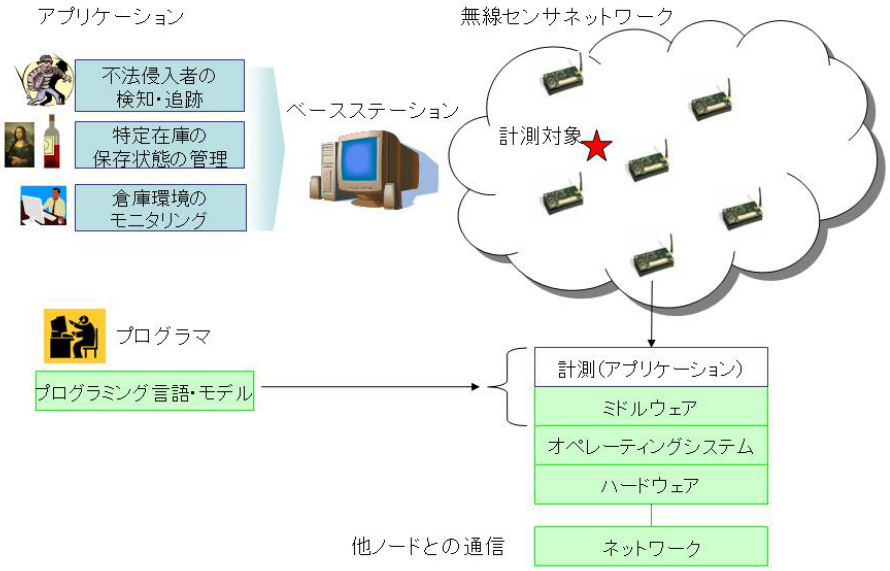

図 1.2: 一般的な WSN の構成と主要技術

が中心であったが，宣言的な記述を行うプログラミング言語や WSN 全体の振る舞いを集中的に記述するプログラミングモデルが提案されている．今後は，より記述・プログラミングの容易性が着目されると考えられる．プログラミング言語およびプログラミングモデルの詳細は 2.1.1 節で記述する．ミドルウェアは，WSN のアプリケーション開発で要求される処理と低レベルなオペレーティングシステムの乖離を埋める位置づけから，WSN 研究の中心となっている．初期は WSN の複雑性の隠蔽が主眼であったが，現在は動的に変化する環境への自動適応等が注目されており，今後もこの傾向は続くと考えられる．ミドルウェアの詳細は 2.1.3 節で記述する．オペレーティングシステム，ハードウェアは，TinyOS, MICAz がほとんど標準であった時代から他オペレーティングシステム，ハードウェアの登場による選択の時代へと移行すると考えられる．詳細はそれぞれ，2.1.2 節，2.1.4 節で記述する．ネットワークは，ルーティングプロトコルの研究が盛んであったが，セキュリティの研究が増加している．ネットワークに関する詳細は 2.1.5 で記述する．

表 1.3: WSN の主要技術と動向

項目	動向
プログラミング言語・モデル	2000 年頃は，低レベルのハードウェアの振る舞いを記述できるプログラミング言語の提案が主体であった．その後，開発効率を上昇させるためのライブラリの提供が中心となった．組み込みプログラミング，分散プログラミングの性質をもつ WSN のプログラミングの難しさは，最新の研究例においても課題認識されており，記述・プログラミングの容易性に着目した研究が主流となっている．詳細は 2.1.1 節で記述する．
ミドルウェア	2002 年頃からミドルウェアの研究は顕著に増加したが，初期はプログラマから WSN の複雑性を隠蔽することに主眼が置かれた．アプリケーションには必要とするデータの条件のみを記述させ，WSN 固有の複雑性（通信，電源管理等）はミドルウェアが隠蔽する研究例が多く見られた．その後，WSN 環境の変化に自動的に適応するような研究が増加し，ノードの追加削除やそれに伴うネットワークトポロジの変化，観測対象の移動など，環境の変化に対応して自動で WSN の構成を変更するような研究が主流となっている．詳細は 2.1.3 節で記述する．
オペレーティングシステム	オペレーティングシステムは TinyOS[Hill 00] が最も広く用いられてきた．TinyOS の最新版である TinyOS-2.1.0 は，MICAz, IRIS[IRI]，eyes[EYE]，TelosB[TEL]，shimmer[SHI] 等のハードウェアをサポートしている．TinyOS 以外の他オペレーティングシステムが登場し，Contiki[Dunkels 04b],[Dunkels 06]，MANTIS[Abrach 03]，SOS[Han 05] 等がある．詳細は 2.1.2 節で記述する．
ハードウェア	WSN を構成するノードが各種存在する．TinyOS が稼動する MICAz, IRIS，eyes，TelosB，shimmer 等や，JavaVM が稼動する SunSpot[SPT]，Sentilla[SEN] がある．詳細は 2.1.4 節で記述する．
ネットワーク	ルーティングプロトコルの研究が最も盛んであった．WSN のルーティングプロトコルは，省電力，耐障害性，変化するネットワーク環境，センシングデータの集約，QoS 等に特に着目する必要がある [Al-Karaki 04]．ルーティングプロトコルの提案数は非常に多く，アプリケーションに応じてプロトコルを選択するのが現実的といえる．ルーティングプロトコルの提案自体は減少傾向であり，現在はセキュリティ等に研究の中心が移行しつつある．詳細は 2.1.5 節で記述する．

1.2 研究の背景と目的

1.2.1 本研究の想定環境

従来，WSN は特定のアプリケーションを実現するために敷設されてきた．そのため，敷設された WSN は特定のアプリケーションに限定されたインフラとなっていた．WSN の実利用が行われる将来において，特定のアプリケーションに限定した運用形態は費用対効果の観点で非効率であり，WSN 上で複数のアプリケーションを運用することが現実的である．複数のアプリケーションを運用する際には，1.WSN の敷設後に新たなアプリケーションを追加すること，2. 不要なアプリケーションを削除すること，3. アプリケーションの動作を変更すること，3 つを実現する必要がある．本研究では，一般的な WSN と異なる以下の環境を想定し，既存の WSN を利用して複数のアプリケーションを運用することを想定する．

1. WSN は既に敷設されている
2. 複数のアプリケーションを既存の WSN で実現する
3. アプリケーションの追加・変更・削除を実現する

本研究では，図 1.3 に示すような，WSN が敷設された平置き倉庫[*2] において，図 1.4 に示すとおり，以下のアプリケーションを実現することを想定する．

特定在庫の保存状態の管理

在庫所有者のために，倉庫管理者が管理を要する高価な物品のために物品周辺で物品固有の計測を行う．なんらかの異常が発生した場合は WSN 内部で処理を実行する．例えば，温度異常がある場合は最寄の空調を調整したり，不法侵入者が存在する場合に追跡を実施する．

不法侵入者の検知と追跡

倉庫の特定のエリアで不法侵入者の検知を定期的に実施し，不法侵入者が検知された場合には，不法侵入者を取り囲むと同時に，ベースステーションに通知を行う．

倉庫環境のモニタリング

倉庫のノード全てにおいて，定期的な計測を実施する．火災や地震等の異常が発生

[*2] 屋根のある空間で，最小限の設備を所持し，製品やパレットを直接並べて保管する倉庫．レイアウトの変更もあることから，固定電源の敷設が難しい．

した場合は，WSN 内部で火災と取り囲み，WSN 内部の人間に通知を行い，地震が発生した場合は，倉庫の構造に変化がないか調査を実施する．

これらのアプリケーションは，WSN 内部で，アプリケーションの要求に応じた計測を実施し，計測結果に応じ，WSN 内部でアプリケーションで必要とされる処理を実施する．従来ベースステーションで実施されることが通常であったアプリケーション特有の処理を WSN 内部で実現することに特徴がある．本研究ではこうしたアプリケーションを *LCA*(Locally Centralized Application) と定義する．

図 1.3: 想定環境例-平置き倉庫

1.2.2 WSN のアーキテクチャ

WSN のアプリケーションを実現する際のアーキテクチャは，図 1.5 に示すとおり，大きくベースステーション集中型とネットワーク内処理型の 2 つが存在する．

1. ベースステーション集中型

各ノードが取得できるセンサデータを定期的にベースステーションに送り，ベースステーションで取得されたデータを解析することによってアプリケーションを実現する．

2. ネットワーク内処理型

各ノードにアプリケーション依存する処理を実行できるプログラム[*3] を配備し，ネッ

[*3] ベースステーション集中型にもプログラムは存在するがネットワーク内処理型に比べて単純なプログラムとなる．また，ベースステーション集中型ではアプリケーションの知識を必要としない．

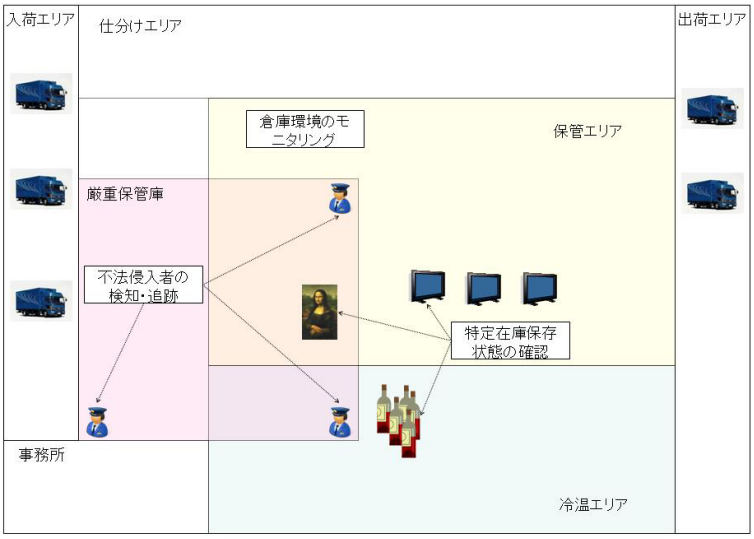

図 1.4: 想定環境とアプリケーション例

トワーク内部でアプリケーションの処理の一部（データの加工など）を実施する．ベースステーションとの通信は必要に応じて実施する．

ベースステーション集中型は単純なアーキテクチャであり，ベースステーションに収集されたデータを加工することによりアプリケーションを実現する．アプリケーションプログラムは PC で稼動するマシンで実行されることから，WSN 固有の性質を扱うプログラミング量が相対的に少なく済む長所がある．ただし，ベースステーション集中型には以下の 2 つの課題が存在する．

課題 1-1) 消費電力

ベースステーション付近のノードにトラフィックが集中し，これらのノードの電力が枯渇する恐れがある．例えば，MICAz で構成される WSN を想定し，各ノードが一秒間隔で 36byte[*4] のパケットを一分間送信すると仮定する．このとき，各ノードは 2.144J 消費する [Shnayder 04]．MICAz の許容容赦電力は 21600J であり，この仮定において，ノードの電力は約 7 日[*5] で枯渇する．例えば，一秒間に一回の間隔で平均温度を計測するようなアプリケーションは，この仮定に相当する．

課題 1-2) パケットロス

図 1.6 のように，WSN ではパケットロスがあることが知られている [Levis 03]，[tos].

[*4] TinyOS における標準的なパケットサイズ
[*5] 6.99=(21600/2.144)/60*24

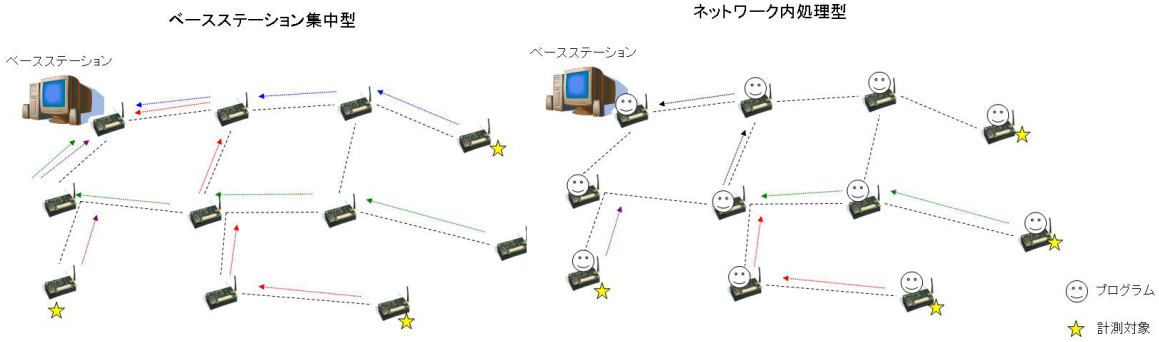

図 1.5: ベースステーション集中型とネットワーク内処理型

ベースステーションと距離が離れたノードや，複数ホップを要するノードが存在する場合，ノードからベースステーションにデータを送信・転送するため，データを複数回再送する必要性や，データを送信・転送できない可能性がある．ベースステーション集中型のように，データをベースステーションに頻繁に送信する必要が生じる場合には，通信の非効率さを誘発する結果となる．

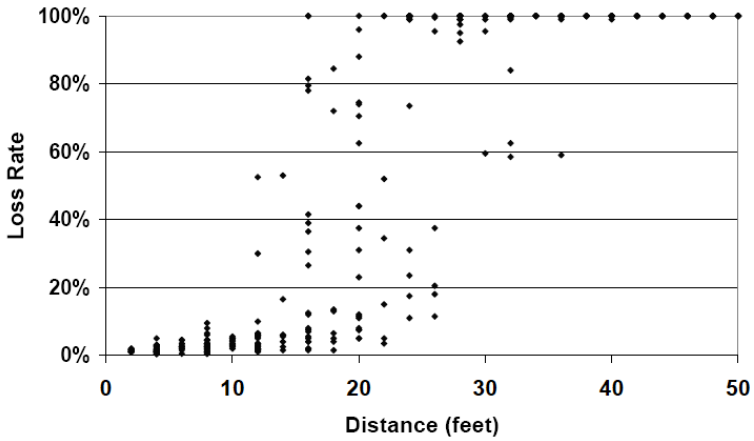

図 1.6: WSN におけるパケットロス（[Levis 03] より引用）：WSN ではノード間の距離に応じてパケットロスが発生することが知られている．

これらのベースステーション集中型の課題を補うため，ネットワーク内処理型のアーキテクチャが用いられる．ネットワーク内処理型は，ネットワーク内部でアプリケーションの機能を完全に実現したり，ネットーワーク内部でアプリケーションの機能を一部実現しベースステーションで残りの機能を実現するアーキテクチャである．ネットワーク内処理型の

代表例に，ネットワーク内部で計測されたデータを統計的に取りまとめベースステーションとの通信を削減する In-network Aggregation がある．In-network Aggregation の代表例に TAG[Madden 02]，CAG[Intanagonwiwat 03]，Krishnamachari ら [Krishnamachari 02] 等が挙げられる．TAG では，図 1.7 に示すように，ネットワーク内でデータを取りまとめることによって，統計処理（COUNT, MIN, HISTOGRAM, AVERAGE) によっては通信量を 80%程度削減することに成功している．ネットワーク内処理型は，ベースステーション集中型に比べて大幅に通信コストを削減することで，WSN の最大・共通の課題である電力問題に解を与え，WSN で一般的に用いられるアーキテクチャとなっている．ネットワーク内処理型は，ベースステーション集中型と異なり，アプリケーションの機能を実現するプログラムをノードに配備する必要があり，ベースステーション集中型に比べて配備されるプログラムの規模が大きくなる．その結果，ネットワーク内処理型のアーキテクチャを適用する場合，プログラマには WSN 固有の性質を理解したプログラミングが求められ，この付加を軽減するためにミドルウェアを適用するのが一般的である．

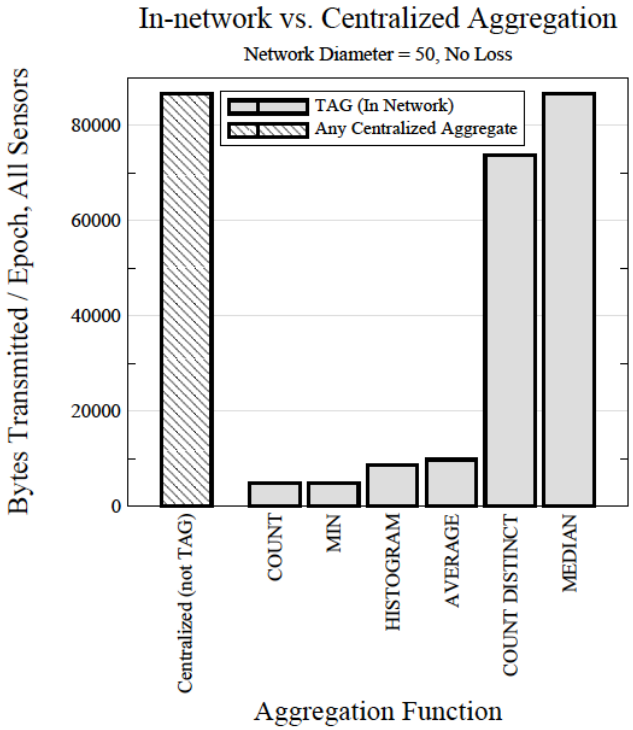

図 1.7: TAG による通信量の削減（[Madden 02] より引用）: 数 (COUNT)，最小値 (MIN)，度数分布 (HISTOGRAM)，平均 (AVERAGE) ではベースステーション処理型に比べて 80%程度の通信量の削減を実現している．

1.2.3 WSN のミドルウェアとリプログラミング

WSN のプログラミングは制約のあるリソース，動的に変化するネットワーク環境，配備されるノード数，取得可能なデータ等を考慮する必要があり，ハードウェアとネットワークに関わる低いレベルの処理を記述する必要が生じることから煩雑になる [Rubio 07]．その結果，ネットワーク内処理型のアーキテクチャを採用する際，ノードに配備されるプログラムを一から作成するのは非効率であり，ミドルウェアを用いたプログラマの負担軽減が一つの選択肢となる．WSN のミドルウェア研究は非常に多く存在するが，代表的なアプローチが 2 つある [Henricksen 06].

A. データベース型

WSN で計測可能なデータを中心に WSN を抽象化する．主な研究例に TinyDB [Madden 05][Yao 03]，Cougar [Yao 02]，SINA [Srisathapornphat 00]，DFuse [Kumar 03] ，TinyLIME [Curino 05]，DSWare [Li 03] 等がある．データベース型は，プログラマにアプリケーションが必要とするデータを記述させ，WSN の内部動作を意識することなくデータを取得することを可能にしている．詳細は 2.1.3 節で記述する．

B. イベントベース型

イベント駆動型の情報処理メカニズムを用いて WSN を抽象化する．アプリケーションは，把握したい現実世界の状態をイベントとして記述し，ミドルウェアに登録する．ミドルウェアは，定義されたイベントが WSN 内で観測された場合のみ，その結果をベースステーションやアプリケーションに通知する．主な研究例に，Enviro-Track [Abdelzaher 04]，Impala [Liu 03] COMiS [Kumar 05]，Mires [Souto 04] がある．詳細は 2.1.3 節で記述する．

本研究で想定する *LCA* はアプリケーションで必要とされる場所で計測を実施し，取得されたデータに応じて WSN 内部で処理を実施することを想定している．データベース型は，プログラマから WSN の内部動作を隠蔽するが，想定環境においては以下の課題を抱える．

課題 A-1) ネットワーク内部でのセンシング結果の取得

データベース型は，ベースステーションに結果を集めることを想定しているために，WSN 内部でアプリケーションで要求されるセンシング結果を取得することに限界が

ある．例えば，TinyDB では宣言的に記述されたクエリをプログラマが記述し，ベースステーションからクエリを各ノードに送信することによってデータを取得するが，データを集約する木構造はプログラマから隠蔽されて作成される．この結果，隣のノードであっても異なる親ノードに所属する 2 つのノードの平均温度はベースステーションに報告された後でなければ知ることができない．WSN 内部のどこで，どのように集約を実施するかの詳細はプログラマから隠蔽されており，ネットワーク内部でアプリケーションが要求するセンシング結果を取得できない．クエリの結果が常にベースステーションに返却され，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する課題，ベースステーションとの距離が離れたノード間の通信がパケットロスによって影響を受ける課題がある．前者の課題は，アプリケーションに依存するが，仮に 1 秒間隔で平均温度を取得すると，ノードの電力は約 7 日 (1.2.2) で枯渇する．後者の課題は，図 1.8 に示すとおり，9 ホップを要する WSN では，その通信信頼性が劣ることからベースステーションに結果を収集する手法は非効率であるといえる．

課題 A-2) WSN 内におけるアプリケーション固有の処理

本研究では，計測結果に応じ WSN 内部でアプリケーションに応じた処理を行うことを想定している (1.2.1 節参照)．データベース型は，ネットワーク内部での処理はミドルウェアが標準的にサポートするクエリによってのみ実施可能であり，アプリケーション依存する処理は，プログラマが個別に記述を行い，ノードに配備しておくことが必要であり，データベース型の対象外の領域になる．例えば，ある部屋で取得された平均温度が閾値を上回った場合，周辺の空調機器に対して温度を下げるような処理を行うことを考える．課題 A-1) を考慮せず[*6]，平均温度が取得できたとして，その際，空調を下げるような処理はプログラマが記述し，あらかじめ各ノードに配備を行う必要が生じる．アプリケーションが追加されるたびに，各ノードにプログラムを配備することは運用上支障を来たす．

イベントベース型は，把握したい現実世界の状態をイベントとして抽出することを可能にし，WSN 内部でイベントを把握することができる．ただし，データベース型と同様に WSN 内部でのアプリケーション依存の処理は個別に実装を行う必要があり，以下の課題を抱える．

[*6] WSN 内部で任意のセンシング結果を取得できると仮定する．

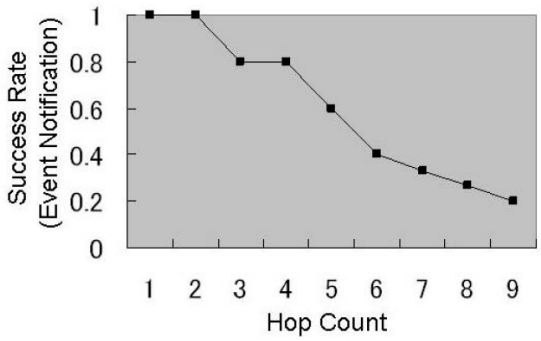

図 1.8: シミュレーション環境においてマルチホップ通信をさせた場合のイベント通知の成功率とホップ数の関連（[Suenaga 08] より引用）．グリッド状に配置したノード間隔が 10feet の場合の結果を示している．

課題 B-1) WSN 内におけるアプリケーション固有の処理

イベントベース型は，WSN 内部に配備されるプログラムによって特定のイベントを WSN 内部で検知することを可能にしている．EnviroTrack [Abdelzaher 04] は，Context Label と呼ばれるイベントの定義を記述することによって，計測対象の移動等のイベントを計測することを可能にしている．Context Lable には計測条件や，計測結果の集約方法等を記述することができる．Enviro Track では与えられた Context Label の情報を元に，Leader と Member によるグループを WSN 内部で形成し，複数のノードで計測対象を計測することを可能にしている．ただし,，本研究のように計測結果に応じて WSN 内部でアプリケーションに応じた処理を実施する場合 (1.2.1 節参照) には，アプリケーション依存の処理をプログラマが記述し，実行を行うノードに配備しておく必要が生じる．イベントベース型は，WSN 内部で発生するイベントを抽出することに特化しており，WSN 内部でイベント抽出以外のアプリケーション固有の処理を実施するための機構を提供していない．この結果，アプリケーションが追加されるたびに，各ノードにアプリケーション固有のプログラムを配備する必要が生じ，運用上支障を来たす結果となる．

以上のことから，*LCA* の実現にあたり，代表的なミドルウェアのアプローチには限界があり，*LCA* 固有の計測・処理を実行できる *LCA* 固有のプログラムをノードに配備することが課題になる．

本研究では，アプリケーションを段階的に実現することを想定しているため (1.2.1)，*LCA* を実現するプログラムを敷設された WSN 上に追加すること，不要な *LCA* を構成するプ

ログラムを WSN から削除すること，変更を要する *LCA* を構成するプログラムを変更することが必要になる．一般に，ノード上に配備されるプログラムを追加・削除・変更する際には，ノード，プログラミングボード，PC を物理的に接続し，ノードの ROM にプログラムを配備する必要がある．図 1.9 に，MICAz と MICAz のプログラミングボードである MIB520 の外観を示す．WSN を敷設後に新たな *LCA* を追加する必要が生じると，ノードを一度回収して *LCA* を実現するプログラムを焼きなおすことが必要になるが，このとき図 1.9 のように物理的な接続を実施していると運用に支障を来たす．この手間をなくすために，リプログラミング [Wang 06] が提案され，ベースステーションからノードにプログラムを無線経由で配備することを可能にし，プログラムの追加・変更・削除を実現している．リプログラミングは，アプリケーション固有の処理を実行するプログラムを記述し，そのプログラムをベースステーションから配備することを可能にし，代表的なミドルウェアのアプローチの課題 A-1，A-2，B-1 を解決することができる．

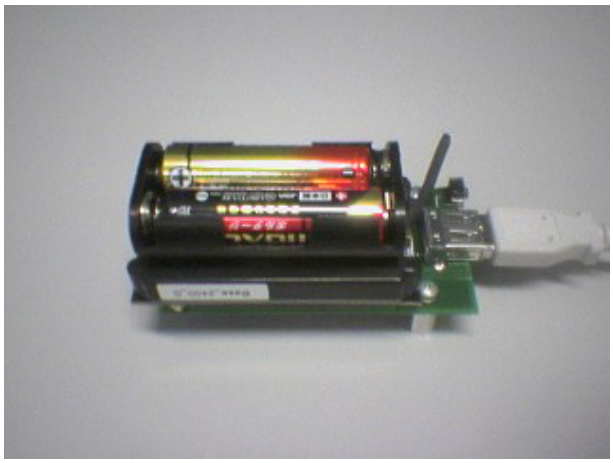

図 1.9: MICAz と MIB520 の接続例

以上のことから，本研究で想定するアプリケーション (*LCA*) を，本研究の想定する運用形態で実現する際の適合性は，表 1.4 のとおりまとめられる．なお，表 1.4 中における○，△，×は，それぞれ，可能，不向き，不可能を示す．データベース型，イベントベース型はアプリケーションの要求を満たさないために，本研究では不向きとなる．リプログラミングを用いた場合，*LCA* 任意の処理を実行するプログラムをノードに配備することが可能であり，プログラムを追加・削除・変更する運用形態をも満たす．

リプログラミングには大きく以下 2 つのアプローチが存在する．

表 1.4: *LCA* に対する代表的なミドルウェアのアプローチとリプログラミングの適合性

項目	*LCA* の要求	運用形態	適合性
データベース型	×:プログラマがクエリを記述することにより，結果をベースステーションで受信できるが，WSN 内部でセンシング結果を取得する用途に向かない．加えて，アプリケーション固有の処理を実施する場合には各ノードにプログラムを配備することが必要になる．	△：プログラマがクエリをその都度記述することでベースステーションで結果を取得することができる．ただし，ミドルウェアの記述力を上回るアプリケーション固有の処理（対象物の追跡等の NW 内部での処理等）を実施する場合，アプリケーション固有の処理を実行するプログラムの配備を要し，物理的な接続が必要になる．	×
イベントベース型	△：WSN 内部のイベントの検出は可能であるが，イベントに応じてアプリケーション固有の処理を WSN 内部で実施するために，プログラマが *LCA* 固有のプログラムを配備する必要がある．	×：イベントを解釈可能なミドルウェアを予め各ノードに配備しておく必要が生じる．アプリケーション固有の処理を実施する場合，アプリケーション固有の処理を実行するプログラムの配備を要し，物理的な接続が必要になる．	×
リプログラミング	○：*LCA* の処理を実行するプログラムを記述する必要は生じるが，WSN 内部に当該プログラムを送り込むことで可能．	○：ベースステーションから無線波を利用してプログラムを追加・削除・更新することが可能．	○

静的配備

常にベースステーションからノードにプログラムを配備する．

動的配備

ベースステーションからノードにプログラムを配備することができ，必要に応じて WSN 内部でプログラムを移動させて再配備することができる．

WSN におけるノードはプログラムメモリが限定されているため，複数のアプリケーションを実現する際，必要なノードにのみプログラムを配備する必要が生じる．全てのノードに全てのアプリケーションを実現できるプログラムを配備できない環境下で，アプリケーションの稼動場所が変更された場合，プログラムの配備場所を変更する必要が生じる．この際，静的な動的配備手法では，ベースステーションから常に配備を行う必要が生じベースステーション付近のノードの電力が枯渇する可能性がある．動的配備手法では WSN 内部に存在するプログラムを，WSN 内部で移動させることによりプログラムの再配備を実現することが可能であり，プログラムの移動距離を短くすることによって通信量を削減することができる[*7]．この結果，動的配備手法は静的配備手法よりも，全てのノードに全てのアプリケーションを実現できるプログラムを配備できない環境下で，アプリケーションの稼動場所が変更される場合に適する手法といえる．リプログラミングの詳細は 2.2 節，2.3 節で記述する．

1.2.4 本研究の目的

本研究で想定するアプリケーション (LCA) は，LCA 固有の計測を実施し，WSN 内部で LCA 固有の処理を行う．加えて，複数の LCA を実現し，個々の LCA を段階的に追加・削除・変更することを想定する．ノードのメモリ制約が厳しい WSN においては，全てのノードに複数の LCA を実現する全てのプログラムを配備することができない．さらに，LCA は WSN 内部で稼動場所を変更する可能性がある．この想定環境下では，WSN 内部で配備されたプログラムを WSN 内部で再配備する動的配備が有効となる．

LCA は，空間的な計測処理を必要とすることから，複数のプログラムで構成される．動的配備の既存研究は，WSN 内部で複数のプログラムが連携するアプリケーションを構築する際のアーキテクチャを提案しておらず，アプリケーションを構成するプログラムの機能

[*7] ベースステーションと旧配備場所を結ぶ線の垂直二等分線で WSN を区切った場合，ベースステーションに近い領域が静的配備が有利な領域，ベースステーションから遠い領域が動的配備が有利な領域となり，空間的な平均をとれば静的配備より動的配備が有利な領域が多くなる．詳細はで記述する．

分担が課題となる．また，複数プログラムで構成されるアプリケーションが，稼動場所を変更する際，全プログラムを動的に配備できない可能性があり，アプリケーションを継続して実行できない課題がある．これらの課題は，想定環境のように，既存の WSN を利用して，WSN 内部で計測・処理を実施するアプリケーションを実現することを難しくする．

以上のことから，本研究では，将来的な WSN のインフラと運用形態を想定し，既存の動的配備手法が抱える以下の課題を解決することを目的とする．

1. WSN 内部で計測・処理を実行するアプリケーションに適したアーキテクチャ
2. 複数のプログラムを動的配備する手法

1.3 まとめ

本研究では，以下の想定を行う．

1. 敷設済みの WSN が存在する
2. 複数のアプリケーションを一つの WSN を用いて運用する
3. アプリケーションは段階的に追加・削除・変更される
4. 個々のアプリケーションは複数のプログラムで実現される
5. 個々のアプリケーションは WSN 内部で固有の計測・処理を実行する
6. 個々のアプリケーションは稼動場所を変更する可能性がある

本想定に対し，ネットワーク内で処理を行う代表的なミドルウェアのアプローチであるデータベース型，イベントベース型の適用は不向きとなる（1.2.3 節）．その結果，リプログラミングを用いてアプリケーション固有の処理を実行できるプログラムを無線波経由で配備することが必要になる．リプログラミングには静的配備と動的配備の 2 つの手法があるが，上記想定においては動的配備が適する．ただし，既存の動的配備の研究例は，複数プログラムで構成されるアプリケーションのためのアーキテクチャ，複数プログラムの動的配備手法に課題があり，本研究で想定するアプリケーションの実現を難しくしている．本研究では，将来的な WSN の適用を見据え，これらの課題の解決を行うことを目的とする．

1.4 本論文の構成

本論文は 6 章から構成される．第 1 章は，本研究の背景や目的について述べた．第 2 章では，WSN の研究分野とその分野における動向と，本研究の関連研究を記述する．第 3 章で，本研究の想定環境と事例を取り上げ，本研究で想定するアプリケーションの要求分析を行い，分析された要求を満たす際に既存研究が抱える課題を特定する．第 4 章で，特定された課題に対する解決策として，動的配備を行う際のアーキテクチャと，動的配備手法を提案し，シミュレーション環境における評価結果を示す．第 5 章で，提案手法の有効性および限界について議論する．第 6 章で本研究をまとめ，本論文を結ぶ．

第 2 章

関連研究

2.1 WSN の研究分野

本章では，1.1 節で提示した WSN の研究分野のうち，プログラミング言語・モデル，ミドルウェア，オペレーティングシステム，ハードウェア，ネットワークにおける研究例を紹介する．

2.1.1 プログラミング言語・モデル

WSN のプログラミングに関する研究は，TinyOS[Hill 00] の研究が平行して実施され，NesC[Gay 03] が TinyOS 用のプログラミング言語として開発された．NesC は C 言語の拡張言語であり，コンポーネント指向のプログラミング言語である．コンポーネント間での連携はイベント（ハードウェア割り込み）とコマンド（他コンポーネントの呼び出し）を記述することで実現し，ハードウェアを操作する低レベルな処理を記述することができる．NesC は TinyOS 専用の言語であるが，C 言語や Java 言語によって記述するオペレーティングシステムやハードウェアも登場したことから，NesC，C，Java が標準的な WSN のプログラミング言語となっている．

NesC の登場によって TinyOS 上でのアプリケーション開発が可能になると，ハードウェアを操作する低レベルなプログラミングを実行することの難しさが指摘された．その結果，プログラマの負担を軽減し，WSN の詳細を隠蔽するプログラミングモデルの研究が開始された．プログラミングモデルには大きく以下のアプローチ [Rubio 07] がある．

ノードセントリックプログラミング

個々のノードまたは WSN 内部の複数のノードに着目し，アプリケーションの機能を実現する際の WSN の詳細をプログラマから隠蔽することを主眼とするアプローチ．

ライブラリの提供や，物理的な距離や属性を基に仮想的なグループを構築し，そのグループに対する操作 API を提供する手法が代表的である．

マクロプログラミング

WSN 全体を一つの計算資源と見立て，プログラマから WSN 固有の組み込みの難しさを隠蔽するだけでなく，分散の難しさも隠蔽するアプローチ．宣言的にセンサーネットワークの振る舞いを定義し，それをコンパイルしたプログラムが個々のノード上に展開され，総体として指定された振る舞いを行うような例がある．

ノードセントリックプログラムの代表的な研究例に，データ収集を行うアプリケーションプログラマにプログラミング言語，ライブラリ，コンパイラを提供する SNACK[Greenstein 04] や，データの収集と配布に関するライブラリを提供する sdlib[Chu 06]，複数のノードで地理的にグループを構成するための API を提供する Hood[Whitehouse 04]，Abstract Region[Welsh 04]，Spatial Programming[Borcea 04] 等が存在する．例えば，Hood は近隣ノードで構成されるグループ形成に対する API を提供する研究例であるが，図 2.1 に示すように WSN の非対称な通信を考慮したグループ形成を提案している．プログラマからグループ形成に対する詳細を隠蔽することでコード容量を 33%削減している．

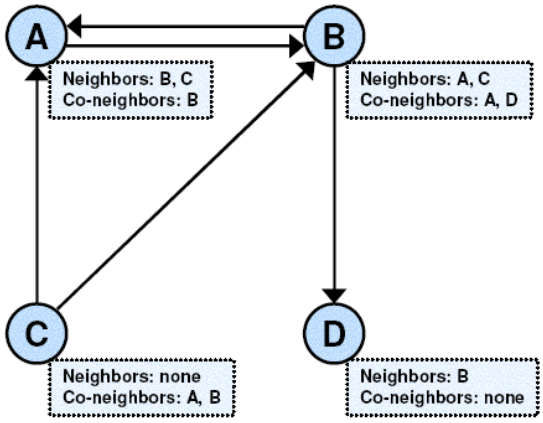

図 2.1: Hood によるグループ構成例（[Madden 02] より引用）：A は C からの通信を受信できるので，C を Neighbor として管理できるが，C は A からの通信を受信できないので A を Neighbor として管理できず，A に管理される Co-Neighbor となる

ノードセントリックプログラムは，WSN の特性のうち，組み込みプログラミングに対する難しさの隠蔽には成功しているが，分散プログラミングの難しさは完全に隠蔽されてい

ない．そこで，WSN をひとつの計算資源をみなすマクロプログラミングが登場した．マクロプログラミングの代表的な研究例に，ノード，近隣ノード，データのリモートアクセスを隠蔽し WSN 全般での振舞いを記述することを可能にした Kairos[Gummadi 05]，データの意味的なクエリ記述を可能にする Semantic Streams[Whitehouse 06]，関数型言語によりプログラムを記述する Regiment[Newton 07]，宣言的な記述によりプログラマが要求するデータを取得できる TinyDB[Madden 05] がある．例えば，Kairos は，図 2.2 に示すような中央集権的なマクロプログラミング手法を提供している．Kairos では，論理的に個々のノード ID を管理すること，隣接ノード（ワンホップ）のグループ構成，任意のノードからのデータ読みだし，という 3 つの原理を提供することでマクロプログラミングを実現している．

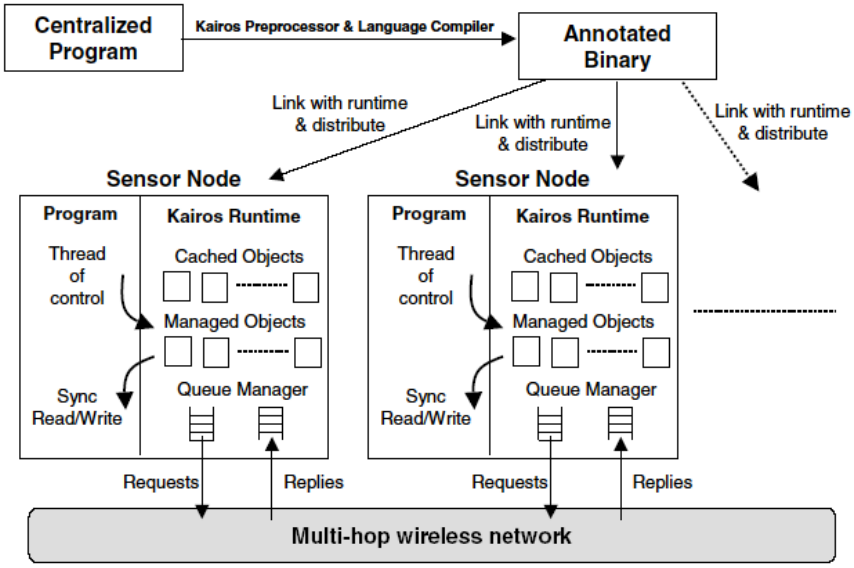

図 2.2: Kairos によるマクロプログラミング（[Gummadi 05] より引用）：Centralized Program により中央集権的な記述を行い，プリプロセッサがバイナリコードを作成し，作成されたコードが各ノードに存在する Runtime で稼動する．

2.1.2　オペレーティングシステム

TinyOS は超小型の WSN ノードを制御するために専用に開発されたオペレーティングシステムであり，極小な計算資源に対応している．TinyOS はコンポーネント指向のアーキテクチャを採用し，コンポーネント間の接続は，command（コンポーネントの呼び出

し），event（割り込み，ハードウェア割り込み）のハンドラの実装によって実現される．NesC は TinyOS 専用の言語であり，プログラマはアプリケーション独自の機能を NesC を用いて記述し，コンポーネントとしてまとめ，TinyOS で標準的に提供されるコンポーネントと接続することでアプリケーションを実現することができる．TinyOS におけるコンポーネントは Configuration ファイル，Module ファイル，Interface ファイルから構成される．それぞれのファイルの詳細を以下に示す．

Configuration ファイル　使用する他コンポーネントの定義を行う．また，他コンポーネントの Module ファイルで実装されているインタフェースの呼び出し（TinyOS では Wiring と呼ぶ）を行う．

Interface ファイル　各コンポーネントが所持するインタフェースの定義を記述する．Interface ファイルに command，evernt の記述を行うことで，他コンポーネントからの呼び出し (command)，他コンポーネントへの通知 (event) が可能となる．

Module ファイル　Interface ファイルで定義されたインタフェースの内部実装を記述する．表 2.1 に Module ファイルの記載例を示す．

TinyOS は WSN の研究において事実上の標準となっているが，TinyOS 以外のオペレーティングシステムが登場しつつある．Contiki[Dunkels 04b],[Dunkels 06] は，組み込みコンピュータ向けのオペレーティングシステムであったが，WSN への対応を行った．Contiki は，TinyOS が静的リンクであるのに対し，動的リンクに対応していることが特徴となる．MANTIS[Abrach 03] は，C 言語の API とマルチスレッドをサポートし，アプリケーション開発の容易性を主眼としている．SOS[Han 05] は，カーネル上に動的にモジュールをロードすることを可能にしている．またオペレーティングシステムとは性質が異なるが，仮想マシンも登場している．代表例に SunSpot 上で稼動する軽量な Java 仮想マシン Squark[SQW] がある．

表 2.1: TinyOS プログラム例

```
//Blink
//定期的に LED を点滅させる基本的なプログラム

module BlinkM {//モジュール名称の定義
        provides {//提供するインタフェースの定義
                interface StdControl;
        }
        uses {//使用する他コンポーネントのインタフェースの定義
                interface Timer;
                interface Leds;
        }
}

//実際の処理部
implementation {
        command result_t StdControl.init(){ //初期化
                call Leds.init();
                return SUCCESS;
        }
        command result_t StdControl.start(){ //起動時から一定間隔でタイマを開始
                return call Timer.start(TIMER_REPEAT, 1000);
        }
        command result_t StdControl.stop(){ //終了時にタイマを停止
                return call Timer.stop();
        }
        event result_t Timer.fired(){ //タイマが起動したら LED を光らせる
                call Leds.yellowToggle();
                return SUCCESS;
        }
}
```

2.1.3 ミドルウェア

WSN のミドルウェアには代表的なアプローチとしてデータベース型とイベントベース型がある [Henricksen 06] ことは，1.2.3 節で述べた．データベース型の代表例として，TinyDB[Madden 05][Yao 03]，Cougar[Yao 02] がある．TinyDB，Cougar の設計思想はほぼ同一であるが，TinyDB は WSN 内部における消費電力を抑えるような工夫がされている [Madden 02]．TinyDB の動作例を図 2.3 に示す．図 2.3 では，4 階の平均気温を 5 秒間隔で観測する，という簡単なクエリの動作例を示している．クエリは SQL を拡張した言語が用いられており，SELECT，FROM，WHERE，SAMPLE PERIOD 節から構成されている．ノード間では，予め，ベースステーションを頂点とする木構造のネットワークトポロジを構築している．クエリは頂点であるベースステーションノードから，木構造のノードトポロジを下って，親ノードから子ノードに転送される．センサーノードはこのクエリを受け取ると，SELECT 節の記述に従い，温度データを観測し，結果を，予めセンサーノードに設定されたメタデータと共にデータタプルに格納する．データタプルに格納されたデータが WHERE 節で指定された条件を満たせば，そのセンサーノードは観測結果を親ノードに報告する．図 2.3 の場合，floor というセンサーノードに予め設定されたメタデータが 4 と一致する場合，観測した温度データを親ノードに報告する．親ノードは，子ノードから報告されてきた観測結果と，自身が観測した結果を，集約し，集約結果をさらに親ノードに転送する．図 2.3 の場合，平均の集約関数が用いられている．最終的にベースステーションノードに結果が集められ，アプリケーションに対して 4 階の平均気温が返される．これら一連の動作は，SAMPLEPERIOD 節の記述に従い 5 秒毎に行われる．

SINA[Srisathapornphat 00] は，TinyDB，Cougar と同様にクエリの記述によってセンサデータの取得を可能にするが，内部機構が全く異なり WSN 内部でクラスタを構成している．このほか，動的にデータを集約するノードを変更することで，データ収集時の通信コストを削減する DFuse[Kumar 03]，近隣ノードのセンシングデータを共有メモリに保存する TinyLIME[Curino 05]，データストレージを提供する DSWare[Li 03] がある．

イベントベース型も同様に複数のミドルウェアが提案されてきた．EnviroTrack[Abdelzaher 04] では，抽出すべきイベントを Context Label に記述し，Context Labal に記述された内容に従い，対象イベントを計測すべきノードを動的に変更することができる．計測を行うノードは複数で構成することができ，1 つの Leader と複数の Member で構成されるグループで計測を実施する．

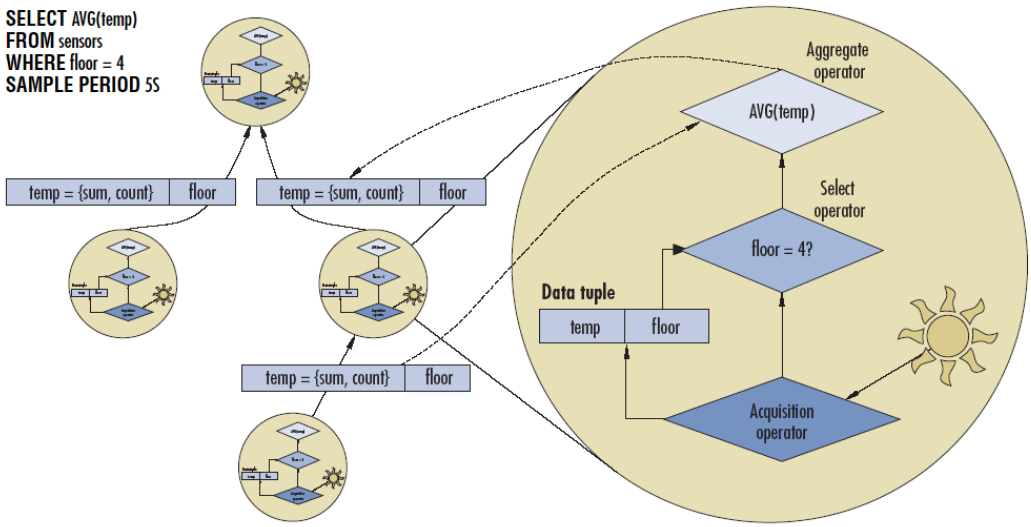

図 2.3: TinyDB の動作例（[Yao 03] より引用）

EnviroTrack は，移動する対象物の状態観測に着目したイベントベース型の WSN ミドルウェアである．プログラマは，Context Label と呼ばれるイベントの定義を記述する．Context Label には，計測したい対象物の条件や計測方法に加え，計測データの集約方法などを記述する．EnviroTrack では，与えられた Context Label を基に，対象イベントの計測を行うグループに参加するノード群を動的に決定する．グループに割り当てられたノードは Member となり，対象物を計測する．Member のうち，1 つのノードが Leader となり，他の Member が計測したデータを収集し，Context Label で定義された計測やデータの集約を実行する．Leader や Member などの役割は，計測対象物の移動に対応して，動的に別のノードに割り当て直す必要がある．EnviroTrack では，役割割り当てを短時間で行うために，Follower と呼ばれる，継続に携わる予備ノードの役割を導入している．Leader ノードから一定距離内に位置するノードは Follower となり，定期的に観測対象物を観測できるかどうかをチェックする．計測対象物を計測した場合，Member ノードとなり，本格的な計測を開始する．また，Leader ノードが計測対象物を計測できなくなった場合，Leader を辞任し，Member ノードの中から新たな Leader が選出される．1 つのイベントに 1 つのグループを割り当てることで，イベントの発生監視のために WSN 全体のノードに問い合わせる必要がなくなり，不必要なデータ計測，通信処理を削減することができる．EnviroTrack は，プログラマからグループ管理に対する詳細を隠蔽しているのが特徴となる．図 2.4 に EnviroTrack のアーキテクチャを示す．

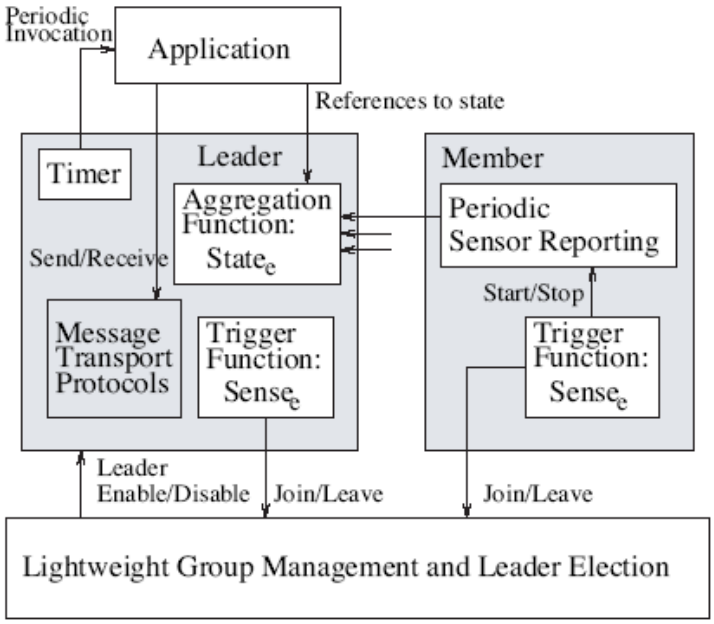

図 2.4: EnviroTrack のアーキテクチャ（[Yao 03] より引用）

EnviroTrack 以外のイベントベース型のミドルウェアとして，Impala [Liu 03] COMiS [Kumar 05]，Mires [Souto 04]，Yoneki ら [Yoneki 05] がある．Impala [Liu 03] は，タイマーを扱う Time Event，パケットの受信イベントを表す Packet Event，パケット送信確認を通知する Send Done Event，データの読み込み準備完了を表す Data Event，デバイスの故障を通知する Device Event という 5 つのイベントをプログラマに対して開放する．これらのイベントを利用しプログラミングを行うことによってアプリケーションが実現できる．COMiS[Kumar 05]，Mires [Souto 04] は publish/subscribe 機構を採用し，アプリケーションと WSN 間でのブローカの役割を果てしている．COMiS と Mires は設計思想がほぼ同一であり本質的な違いはない．COMiS では，イベントは，単純な観測データ値に対する条件か，他のイベントとの組み合せで表現される．アプリケーションは，複数のイベント定義を，観測したい対象地域の座標と共にミドルウェアに渡すことが必要になる．Mires も同等であるが，COMiS と異なり，イベントの組み合わせを統計関数 (合計，平均）で記述する点が異なる．

2.1.4　ハードウェア

Clossbow Technology から提供されている MICAz, IRIS は，現在最も標準的なものであり，Zigbee を用いてマルチホップ・アドホックネットワークを構築することができる．そのほか図 2.5 に示すとおり，eyes，TelosB，shimmer 等のハードウェアが登場している．SunSpot，Sentilla は JavaVM を搭載していることから Java によるプログラミングを可能にしている．表 2.2 に示すとおり[*1]，各ハードウェアのスペックは異なるが，SunSpot が表 2.2 に示す中では大きな計算資源を所持することがわかる．スペック以外の選択の観点としては，プログラミング言語，ライブラリの提供状況，開発環境，シミュレーション環境等がある．現在最も充実しているのは，TinyOS によってサポートされているハードウェア群 (MICAz，IRIS，TelosB 等）であり，プログラミング言語として NesC，TinyOS に含まれる標準ライブリ，TinyOS の Eclipse プラグイン，シミュレーション環境として TOSSIM[Levis 03] が提供されている．SunSpot は開発環境として NetBeans のプラグインを提供しているが，シミュレーション環境が存在しない．

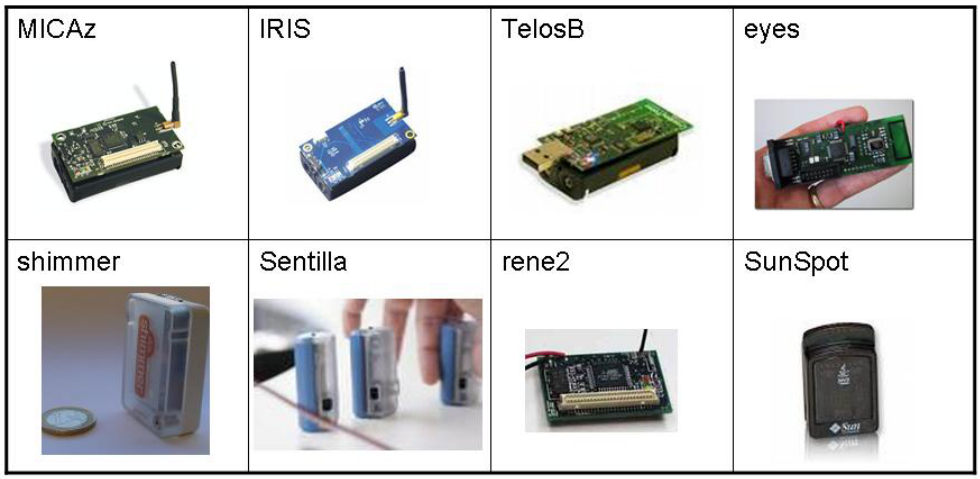

図 2.5: WSN を構成するハードウェア（ノード）の代表例

2.1.5　ネットワーク

WSN のネットワーク領域における研究の中心はルーティングプロトコルである．ルーティングプロトコルは以下の 3 つに分類できる [Al-Karaki 04].

[*1] データシートとして公開されているハードウェアについてのみ記載した．

表 2.2: ハードウェア（ノード）

ハードウェア	プログラムメモリ	データメモリ	CPU	電源	OS
MICAz	128Kbyte	512Kbyte	7.4MHz	単三電池 2 本	TinyOS
IRIS	128Kbyte	512Kbyte	7.4MHz	単三電池 2 本	TinyOS
TelosB	60Kbyte	1Mbyte	8MHz	単三電池 2 本	TinyOS
SunSpot	512Kbyte	4Mbyte	180MHz	充電池	JavaVM
Sentilla	60Kbyte	1MKbyte	8MHz	単三電池 2 本	JavaVM

フラット型

各ノードが同様な機能を担い，ノード間通信を行う方式.

階層型

ノード間で親子関係を持つ等の階層構造を作成し，ノード間通信を行う方式.

位置情報型

ノードの物理的な位置をもとにノード間通信を行う方式.

フラット型の代表例に，Directed Diffusion [Intanagonwiwat 03] がある．Directed Diffusion は，sink ノードまでのデータ転送に対して動的な経路制御を提供する手法である．interest，exploratory，reinforce という 3 種類のメッセージを利用する．sink ノードが計測条件を記載した intetrest をフラッディングし，interest メッセージが転送される．各ノードは，interest を送信してきたノードを gradients として記憶する．interest に記載された計測結果を取得できたノードは，exploratory を gradients に対して送信する．gradients をたどることで最終的に sink ノードに到達するが，複数の経路が存在する可能性がある．そこで，reinforce を sink ノードから explpratory を送信してきたノードに対して送信することで一つの経路を確立する．この他に，フラット型のルーティングプロトコルとして，SPIN [Kulik 02]，Rumor Routing[Braginsky 02]，Shurgers ら [C.Schurgers 01] 等が挙げられる．

階層型の代表例に LEACH[Heinzelman 00] がある．LEACH は，WSN 内部でクラスタを構成し，確率的にクラスタヘッドを決定する手法を提案している．LEACH では，特定のノードがクラスタヘッドになるのではなく，全てのノードが均等にクラスタヘッドを担うように，ラウンドを導入し，ラウンド毎にクラスタヘッドを切り替えることで消費電力の

高いクラスタヘッドの役割を配分する工夫を行っている．この他に，階層型のルーティングプロトコルに，PEGASIS[Lindsey 02]，APTEEN[Manjeshwar 02]，Fang ら [Fang 03] 等がある．

位置情報型の代表例に GAF[Xu 01] がある．GAF は，各ノードで取得できる位置情報を用い，WSN を仮想的な格子状の空間に区切り，各格子内に存在するノードを活動状態，休止状態，探索状態で遷移させる．活動状態のノードがルーティングを実施し，探索状態のノードは一定時間後，活動状態に遷移する．休止状態のノードを作成し，各格子に一つの活動状態のノードを作成することにより，GAF は電力を節約する工夫を行っている．位置情報型の研究例は，各ノードが自身の位置情報を検知可能であるという前提を置くことが一般的である．位置情報を検知する Localization 技術は数多く提案されている [Mao 07]．位置情報型のルーティングプロトコルに，GPSR[Karp 00a]，GEAR[Yu 01]，GOAFR[Kuhn 03] 等がある．

本研究ではプログラムの動的配備の信頼性について提案するが，WSN のネットワークにおける研究を用いて通信の信頼性を向上する際，以下の手法が提案されている．

手法 1：複数経路での送信

ノード間の通信を複数経路で送信することによって冗長化する手法.

手法 2：動的なプロトコル変更

ルーティングプロトコルを動的に変更することにより，単一のルーティングプロトコルの欠点を他ルーティングプロトコルの特性によって補う手法.

手法 3：信頼性のあるトランスポートプロトコルの採用

インターネットで培われた TCP/IP 等の信頼性のあるトランスポートプロトコルを WSN に適用する手法.

手法 1 の代表的な研究例に，REAR[Hassanein 06]，MOR[Biagioni 04]，SWR[Tian 03]，ReInForM[Deb 03] がある．REAR は，Source（データ発生源）から Sink（データ消費者）の通信経路を中継ノードの電力考慮しながら 2 つ確保し，通信経路が切断された場合に冗長経路を利用する手法を提案している．MOR は，Routing Table を各ノードが所持し，Source となるノードが Sink へ近い隣接ノードにメッセージを送信することでルーティングを実現し，Ack を受信できない場合は他の隣接ノードにメッセージを送信する手法を提案している．SWR は，Source と Sink 間の経路を一つ確立し，経路が断絶した場合に中継ノードが代替ノードを検索する手法を提案している．ReInForM は，要求された信頼性を

達成するために複数経路を要求に従って確立し，一つのパケットを複数の経路で送信する手法を提案している．これらの手法は複数経路での送信を可能とするが，単一の経路で送信する場合に比べてメッセージ数が増加し消費電力が多くなることが課題となる．

手法 2 の代表的な研究例に，DRS[Ke 08]，Chameleon[Dunkels 07]，Mahjoub ら [Mahjoub 07] らがある．DRS は，Mesh，Tree のルーティングの切り替えをルールに基づいて動的に変更する手法，Chameleon は MAC プロトコルを動的に変更する手法，Majboub らは WSN 内の特定エリアへの Routing 時に，各ノードに planning をさせることによってルーティングを切り替える手法を提案している．これらの研究は動的なプロトコル変更を可能とするが，いつ，どのプロトコルを利用するべきか・組み合わせるべきかという問題は解決されていない．

手法 3 は，過去に Estrin ら [Estrin 99] によって WSN では適さないと指摘されていた．その理由は以下のとおりである．

1. WSN ノードのリソースの制約に適さない．

2. 膨大なノード数があり，設定時のブロードキャストが課題となる．

3. WSN はデータセントリックであり，グローバルなアドレスは不要である．

これらの指摘事項によって，TCP/IP の WSN ノードへの実装は非現実的であると考えられてきたが，Dunkel ら [Dunkels 04a] は TCP/IP プロトコルを MSP430(MOTE 相当のスペックのハードウェア) に TCP/IP スタックを実装できることを示し，Hui ら [Hui 08] は，TelosB 上に IPv6 スタックを実装できることを示した．これらの研究は，TCP/IP のように信頼性の高いトランスポートプロトコルが WSN 上で利用できる可能性を示唆している．ただし，これらの研究は TCP/IP スタックを WSN ノード上にダウンサイジングして搭載した段階であり，Estrin らの指摘事項に対する評価が今後の課題となる．

WSN のネットワーク領域における研究は，これまで述べてきたようにルーティングプロトコルを中心として行われてきたが，近年セキュリティが着目されている．WSN ではノードがサーバルーム等で稼動しないため，ノードの不正取得によって悪意のある攻撃を受けたり，リソースの不正使用が生じる可能性が指摘されたことによる．セキュリティの研究例は，攻撃者による不正なイベントを検知する研究や，盗聴・改竄を防ぐ研究があり，代表的な研究例に Yu ら [Yu 05]，Ye ら [Ye 07]，Li ら [Li 06]，TinySec[Karlof 04] 等がある．

2.2 リプログラミング概要

WSN では，ノード上に配備されるプログラムを変更する際，ノード，プログラミングボード，PC を物理的に接続し，ノードにプログラムを再配備する必要がある．ノードを一度回収してプログラムを配備することは運用に支障を来たすため，リプログラミング [Wang 06] が提案され，ベースステーションから無線波を利用してプログラムをノードに配備することを可能にした．リプログラミングには多くの研究例が存在するが，各研究を分類するための項目として以下がある．

1. 更新するプログラム
2. 対象範囲
3. シングル/マルチホップ
4. 階層的な更新
5. 静的配備・動的配備

1. 更新するプログラム

ほとんどのリプログラミングの研究例は，アプリケーションを実現するためのコンパイル済みのプログラムをベースステーションから WSN 内部に存在するノードに配備する．アプリケーションの変更時に，ノードに配備したプログラムを再配備する場合，変更の差分が小さい場合でもプログラムを再配備する必要が生じる．そこで，変更分だけを再配備する手法が提案されている．例えば，Incremental Network Programming[Jeong 04] は，変更分だけをスクリプト化してベースステーションから送信し，受信したノードがスクリプト化された変更分とノードに配備されているプログラムから最新のプログラムを自身に配備する手法を提案している．Reijers ら [Reijers 03] は，変更分をベースステーションから生成し，古いプログラムと最新のプログラム間で生じる関数・データに対するアドレスの変更をマイクロコントローラのインストラクションを用いて調整する手法を提案している．TinyCubes[Marrón 05b]，[Marrón 05a] は，アプリケーションを実現するプログラムを複数のモジュールに分割し，更新分のモジュールだけを送信する手法を提案している．

2. 対象範囲

ベースステーションから WSN のノードにプログラムを配備する際に，図 2.6 のように WSN 全体にプログラムを配備する手法と，個々のノードに配備するプログラムを指定できる研究

例に分類できる．前者は WSN で動作するアプリケーションが均一であることを想定しており，後者は一つの WSN 内で部分的に異なるアプリケーションを動作させることを前提としている．この差異は，プログラムを更新する際のアプローチに依存している．Máte[Levis 02] は，前者のアプローチであるが，更新するプログラムを分割し，ブロードキャストし，WSN 全体で一つのアプリケーションを動作させることを想定している．後者のアプローチとして，Deluge[Hui 04]，MNP[Wang 04] があるが，これらは advertise/request 機構を用いており，更新が必要なノードにのみプログラムを配備することを可能にしている．

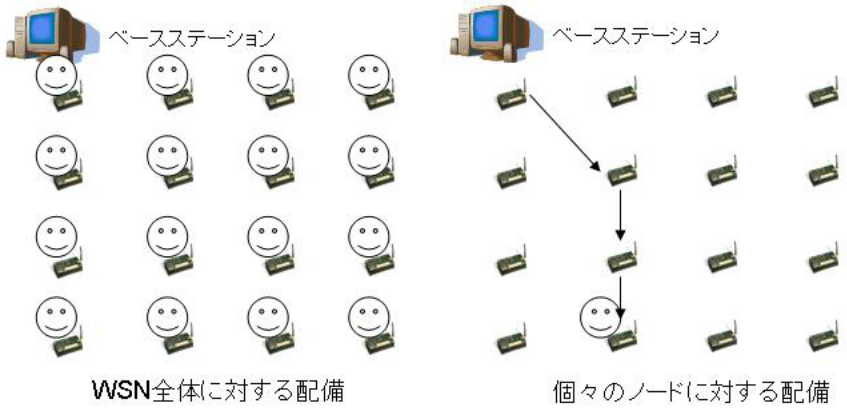

図 2.6: リプログラミングの対象範囲

3. シングル/マルチホップ

リプログラミングの研究初期や，複雑な仮定を想定した場合にはマルチホップに対応していない例が存在した．シングルホップ（通信範囲内のノードにのみプログラムを配備できる）にのみ対応した研究例に XNP[XNP]，Incremental Network Programming[Jeong 04]，Reijers ら [Reijers 03] がある．現在の研究動向は，時間経過とともに想定される WSN の空間的な規模が大きくなった結果，マルチホップが主流となっている．

4. 階層的な更新

プログラムの更新を行う際，更新されるプログラムや，更新情報をブロードキャストすることが必要になる．これは WSN 内部での通信コストの増大とコリジョンを引き起こす課題がある．FireCracker[Levis 04a]，Sprinkler[Naik 05] は，この課題を解決するため，図 2.7 のように，特定ノードに最新のプログラムを配備した後，古いプログラムが配備されている周辺ノードに特定ノードを経由して配備する手法を提案している．

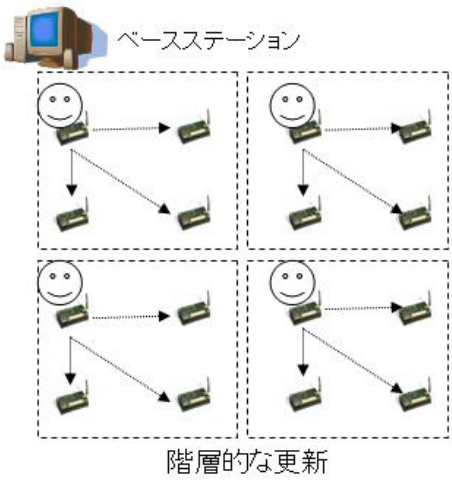

図 2.7: リプログラミングの階層的な更新

5. 静的配備・動的配備

1.2.3 節で，プログラムの配備手法に以下の 2 つのアプローチがあることを示した．

静的配備

常にベースステーションからノードにプログラムを配備する．

動的配備

ベースステーションからノードにプログラムを配備することができ，必要に応じて WSN 内部でプログラムを移動させて再配備することができる．

WSN 内部に存在しないプログラムをベースステーションから配備する場合，静的配備も動的配備もベースステーションからプログラムを配備することは同じである．WSN 内部に存在するプログラムの配備場所を変更する場合，静的配備・動的配備の違いが出る．静的配備は配備されたプログラムを消去し，ベースステーションから新たな配備先のノードにプログラムを配備する必要が生じる．動的配備では，配備されたプログラムを WSN 内部で移動させることによって再配備を行うことができる．WSN 内部に存在するプログラムの配備場所を変更することが多い場合，静的配備手法はベースステーション付近のノードに通信トラフィックが集中し，ベースステーション付近のノードの電力が枯渇する恐れがある[*2]．静的配備，動的配備の研究例の詳細は 2.3 節で述べるが，静

[*2] ベースステーションと旧配備場所を結ぶ線の垂直二等分線で WSN を区切った場合，ベースステーションに近い領域が静的配備が有利な領域，ベースステーションから遠い領域が動的配備が有利な領域となり，空間的な平均をとれば静的配備より動的配備が有利な領域が多くなる．詳細はで記述する．

的配備の代表例に Deluge[Hui 04]，Maté[Levis 02]，動的配備の代表例に Agilla[Fok 05b]，ActorNet[Kwon 06]，SensorWare[Boulis 03] が挙げられる．

表 2.3 に，本章で述べたリプログラミングの実現手法と関連研究の関連を示す．

2.3 静的配備と動的配備

2.3.1 静的配備

Deluge[Hui 04] におけるプログラムの更新は次のとおり行われる．各ノードは自身が所持するプログラムのバージョン情報を含む advertise メッセージを定期的にブロードキャストする．古いバージョンを持つノード (A) からの advertise メッセージを受信したノード (B) は，自身のバージョンを A に対して送信する．A は，B を含む周辺ノードの中から最新のバージョンを持つノードに対してプログラムを送信するように要求する．このように Deluge では単純な更新機構を適用しているが，性能向上のために大きく 3 つの工夫を行っている．

1. advertise メッセージの送信間隔の動的な変更により，ノード密集地域での通信集中を削減する
2. 安定したノードとの通信を動的に実行することによって，ノード間の非対称リンクへの対処を行う
3. ラウンドを導入することによる，空間範囲 (セル）内での複数ノードによるブロードキャスト集中を抑制する

図 2.8 に示すとおり，Deluge では送信するプログラム (Data Object) を，複数の Page に分割し，各 Page を送受信するパケットに分割することでデータ構造を管理している．Deluge では，プログラムの更新を行うノードの ID を指定して，ベースステーションから，ノード単位でプログラムを配備することができる．また各ノードで保持できるプログラム数は 2 つであるが，稼動できるプログラムの種類は一種類に限られる [del]．

ベースステーションからコンパイルされた全プログラムを送信すると，プログラムの容量が大きくなり，プログラムの通信・転送コストが増加し，ノードの電力を消費する課題がある．そこでプログラム容量を削減する方法が提案され，変更分のみ送信する手法と，ミドルウェアが解釈可能な軽量なスクリプトを送信する手法が提案された．

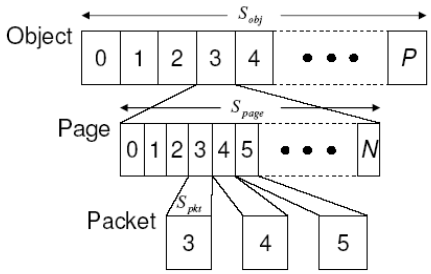

図 2.8: Deluge のデータ構造（[Hui 04] より引用）

Incremental Network Programming[Jeong 04] は前者の手法を適用し，Rsync[Rsy] アルゴリズムを適用して変更分を作成する．Rsnyc アルゴリズムは，プログラムを送信するノードが受信するノードからプログラムのブロック毎に作成されたチェックサムのリストを受信し，チェックサムのリストから最新のプログラムとの相違を計算し，変更分を作成する手法である．チェックサムのリストから変更分を作成するために，各センサノードの変更履歴の管理が求められ，加えて膨大な計算を必要とする．そこで，ベースステーションでこれらの処理を実施することを仮定している．

Maté[Levis 02] は，後者の手法を適用し，各ノードに VM を予め配備しておき，VM が解釈可能なスクリプトを送信することで軽量化を実現している．Maté の VM は，bytecode のインタプリタであり，各 bytecode は 1byte で表現される．この結果，100 行のスクリプトであっても 100byte で記述することを可能にし，コンパイルされたコードを送る方式では簡易なプログラムでも数 Kbyte を要したコード容量を，100byte 程度に低減することに成功している．なお，特定ノードのプログラムを更新できない課題があったが，Maté の拡張版である Trickle によって解決されている．

FireCracker[Levis 04a] は，リプログラミング時に WSN 内部で発生するバージョン管理情報によって生じる broadcast storm[Ni 99] を回避するために，階層的な更新を提案している．図 2.9 に示すとおり，FireCracker では，プログラムの送信元（ベースステーション）が WSN 内部の特定のノードを選択し，選択された特定のノードへプログラムを配備する（(a)，(b)）．送信元から特定ノードへプログラムを配備する際にプログラムを中継するノードも同様にプログラムをキャッシュし自身にプログラムを配備する．これにより，特定ノード上への経路上のノードには，特定ノードを含めて最新のプログラムが配備される．その後，これら最新のプログラムを持つノードがプログラムをブロードキャストすることによって WSN 全体にプログラムを配備する（(c)，(d)）．FireCracker は階層的な管

理を実現しているが，特定ノードの決定方法は提案範囲外となる．

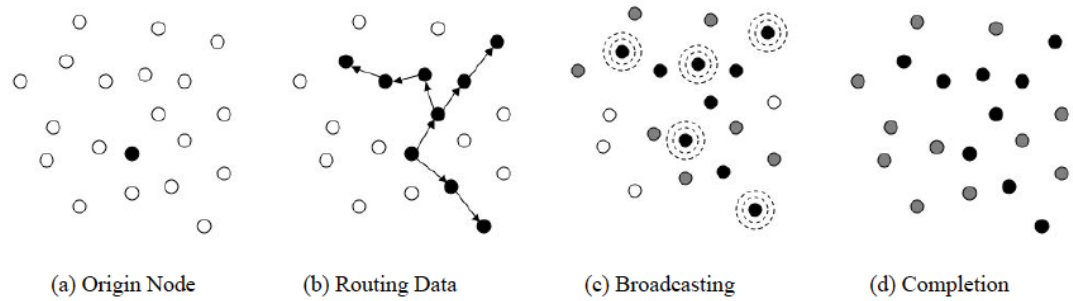

図 2.9: FireCracker の階層的な更新（[Levis 04a] より引用）

2.3.2 動的配備

Agilla[Fok 05b] は，Maté で提供された VM を拡張し，動的配備を実施可能にした研究例である．Agilla におけるプログラムは，VM で解釈可能な 1byte〜2byte の ISA(Instruction Set Architecture) で記述され，Maté と同様にプログラムのコード容量を削減できる．Agilla のモデルを図 2.10 に示す．Agilla は，メモリの制約が許す範囲で複数のプログラムを同時にノード上で稼動させること，プログラム間の通信を分散タプルスペース [Gelernter 85] を用いて記述すること，プログラムの配備場所の指定を行うこと，WSN 内部でプログラムを移動させることを実現している．その結果，プログラマは必要なノードのみに必要なプログラムを配備することや，WSN 内部でプログラム同士の通信を必要とするようなアプリケーションを実現することができる．Agilla は，WSN 内部の状況に応じ，動的にアプリケーションの動作を変更する用途を目的にしている [Fok 05a]．例えば，図 2.11 に示すとおり，火災を消防士に通知する例がある．1 のプログラムが火災を発見した場合，火災を取り囲み，2 のプログラムに通知を行う．2 のプログラムは消防士に火災を通知し，3 のプログラムが火災現場まで消防士を案内する．Agilla は，ソースコードが公開され，動的配備の研究で最も進んでいる研究のひとつであるが，アセンブラに近い ISA による記述，WSN の内部機構，プログラムの位置を熟知する必要がある．例えば，他のプログラムと通信を行う場合，通信を実施する分散タプルスペースの存在するノードの位置，通信相手のプログラムの位置を意識する必要性が生じる．また，各プログラムはスタックとヒープを持つが，スタックの順番やヒープに格納したデータ等を意識する必要が生じ，プログラミングの容易性という点で課題を抱えている．

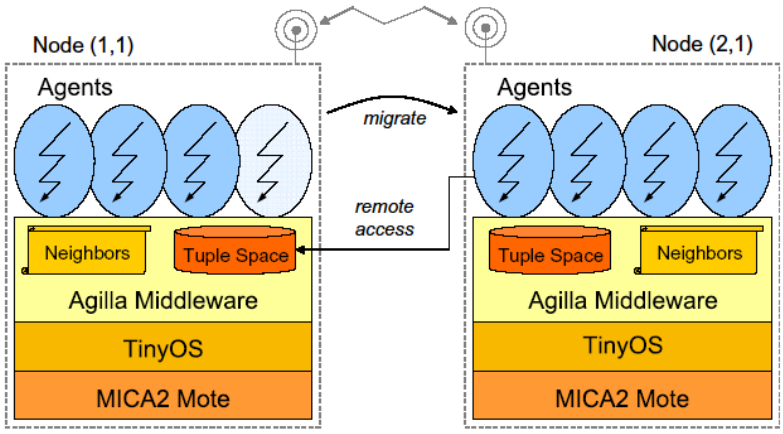

図 2.10: Agilla のモデル（[Fok 05b] より引用）

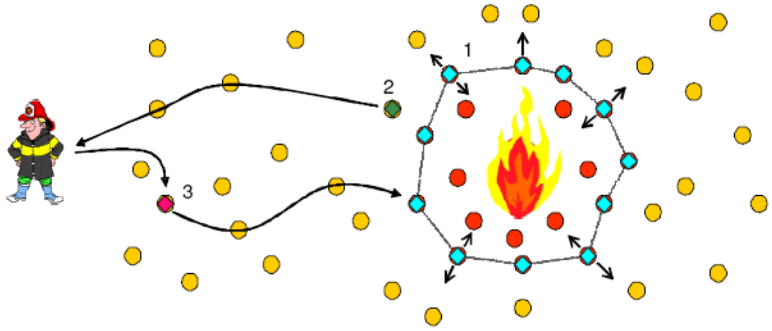

図 2.11: Agilla のアプリケーション例（[Fok 05a] より引用）

ActorNet[Kwon 06] は，actor モデル [Agha 97] を WSN に導入した研究例であり，各 actor は ActorNet で提供される Scheme インタプリタ上で動作する．ActorNet では，仮想メモリの提供，Scheme インタプリタと TinyOS 間の IO スケジューリングの同期，ガーベッジコレクションの提供という 3 つの工夫を行っている．ActorNet は MOTE で稼動することが可能であるが，内部実装に課題があり，通信モデルとしてブロードキャストのみに対応していることや，計測を行う actor を予めノードに配備する必要性，MOTE 上で稼動するミドルウェアが実質存在しないこと等 [ACT]，完成度は Agilla に劣り，コンセプトの提案が主体となっている．

SensorWare[Boulis 03] では，while,if,foreach 等の基本的な制御構造を記述できる独自のスクリプト言語でプログラムを記述できる．プログラムの動的配備をサポートし，複数の

プログラムを同一ノード上で稼動させることもできる．ただし，標準的なプラットフォームとして iPAQ を想定しており，1Mbyte 程度のプログラムメモリと 128Kbyte 程度のメモリが稼動環境として必要になる．WSN のノードとしては潤沢な計算資源（数百 MHz の CPU 及びメモリ）を必要とし，表 1.1 に示す一般的に想定される WSN の計算資源とは乖離がある．

2.4 リプログラミングの今後

リプログラミングの難しさは，極小な計算資源しか持たないノードに適合したリプログラミング機構を提案することである．そのため，リプログラミングミドルウェアのコード容量をノードに搭載可能とすること，リプログラミングのアルゴリズムをノードの計算資源相当にすること，リプログラミング時のメッセージ数を削減することが工夫の中心となってきた．

リプログラミングは TinyOS が登場した直後から研究が行われ，現在においても WSN 研究の中心テーマの一つであるが，今後は研究の方向性が変わると考えられる．2.1.1 節で述べたプログラミングモデルの変化は，リプログラミングの研究に最も影響を与える．2.1.1 節で述べたとおり，プログラミングモデルの主流は，ノードセントリックプログラミングからマクロプログラミングになると想定される．今後は，マクロプログラミングとリプログラミングの性質をあわせることによって，マクロプログラミングによって集中的に記述されたプログラムを，リプログラミングの機構を利用して各ノードに配備する研究等が行われると想定される．一般に，マクロプログラミングはプログラマから WSN の内部機構を隠蔽するので，プログラマはアプリケーション依存の処理に集中できる利点がある．ただし，アプリケーションのパフォーマンスに影響を与える内部機構に対する記述に向かない欠点を併せ持つ．そのため，マクロプログラミングとリプログラミングの融合を行う際には以下の要求を満たすことが必要になると考えられる．

標準的な言語仕様

WSN の内部機構を記述できる一方，プログラマから詳細を隠蔽するトレードオフを考慮し，リプログラミングに適した軽量な言語で WSN アプリケーションを記述できること．

内部機構まで分解可能なプリプロセッサ

現状のマクロプログラミング（Kairos[Gummadi 05]，Semantic

Streams[Whitehouse 06]，Regiment[Newton 07]）のプリプロセッサの能力は限定的であり，全てのノードに均一なプログラムを配備することしか出来ない．リプログラミング時にノード毎に異なるプログラムを配備することへの対応が必要である．それに加えて，内部機構についても各手法で提案されている記述言語にしか対応せず WSN のアプリケーションで必要とする内部機構の十分性に関して研究の余地がある．

WSN 内部でのプログラム間連携

WSN 内部に配備されたプログラム同士でアプリケーション機能を実現する際にプログラム間連携が必要となるが，既存のマクロプログラミング手法では近隣ノードのグループ構成レベルであり，プログラム間連携に対する記述能力と，連携メカニズムが必要となる．

リプログラミングの実現手法の充実

システム運用者からの要求に対応するために，表 2.3 に示す実現手法を網羅的にサポートするリプログラミング手法が必要となる．

2.5 WSN 以外の関連研究

WSN における動的配備は，プログラムを WSN 内部で移動させる観点でモバイルエージェント [Lange 99] と同様である．1990 年頃より，モバイルエージェントの研究は盛んに行われ，代表的な研究例に Telescript[White 99]，Voyager[Glass 99]，Bee-gent[Kawamura 99]，MOA[Milojičić 98]，Mole[Baumann 02] 等がある．本研究では，WSN 内部で特定のプログラムが複数のプログラムを抱えて移動し，必要に応じて特定のプログラムが他プログラムを生成して配備する手法を提案する．同様な手法はモバイルエージェントの研究においても提案されており，MobileSpace[Satoh 00]，石川ら [石川 05] により，モバイルエージェント間に階層関係を持たせ，階層関係を持つモバイルエージェント間でモバイルエージェント内部への移動を可能としている．MobileSpace は，階層化されたモバイルエージェントの構造を始めて提案し，親子関係を持つモバイルエージェントが移動，協調をネットワーク内部で実現するミドルウェアを提案した．階層構造を用いることで，モバイルエージェント間の協調を簡易化し，モバイルエージェントによる複雑な分散アプリケーションの実装手法を提案している．石川ら [石川 05] は，MobileSpace とコンセプトは同様であるが，

階層関係を持つモバイルエージェント間での協調の記述力を向上させ，Mobile Space より複雑な分散アプリケーションの実現を可能としている．

2.6 まとめ

本章では，WSN の研究分野とリプログラミングについて記述した．WSN の研究分野として，プログラミング言語・モデル，ミドルウェア，オペレーティングシステム，ハードウェア，ネットワークにおける研究動向を述べた．また，リプログラミングは，WSN の登場時から研究が開始され，複数の研究例があり多くの実現手法が存在する（表 2.3 参照）ことを示した．本論文では，リプログラミングの実現手法のうち，動的配備と静的配備に着目する．

表 2.3: リプログラミングの実現手法と関連研究

関連研究	更新するプログラム	対象範囲	シングル/マルチホップ	階層的な更新	静的配備･動的配備
Deluge[Hui 04]	全プログラム	ノード指定が可能	マルチホップ	不可能	静的配備
Seluge[Liu 08]	全プログラム	ノード指定が可能	マルチホップ	不可能	静的配備
MNP[Hui 04]	全プログラム	WSN 全体	マルチホップ	不可能	静的配備
XNP[XNP]	全プログラム	WSN 全体	シングルホップ	不可能	静的配備
Incremental[Jeong 04]	変更分	WSN 全体	シングルホップ	不可能	静的配備
Reijar ら [Reijers 03]	変更分	WSN 全体	シングルホップ	不可能	静的配備
TinyCubes[Marrón 05a]	変更分	ノード指定が可能	マルチホップ	不可能	静的配備
Trickle[Levis 04b]	全プログラム	ノード指定が可能	マルチホップ	不可能	静的配備
Maté[Levis 02]	全プログラム	WSN 全体	マルチホップ	不可能	静的配備
FireCracker[Levis 04a]	全プログラム	WSN 全体	マルチホップ	可能	静的配備
Sprinkler[Naik 05]	全プログラム	WSN 全体	マルチホップ	可能	静的配備
Agilla[Fok 05b]	変更分	ノード指定が可能	マルチホップ	不可能	動的配備
ActorNet[Kwon 06]	変更分	ノード指定が可能	マルチホップ	不可能	動的配備
SensorWare[Boulis 03]	変更分	ノード指定が可能	マルチホップ	不可能	動的配備

第 3 章

要求の分析と課題の特定

3.1 はじめに

WSN のアプリケーションは，個々のノードにプログラムを配備し，ノードを物理的に配置することによって実現される．WSN の既存研究の大半は，特定のアプリケーションのために WSN の敷設を実施することを想定している．この想定はアプリケーションの運用前に個々のノードにプログラムを配備し，ノードを配置して，アプリケーションを運用することを可能にした．しかし，今後の WSN の一つの運用形態として，WSN が事前に配備された環境下で，システム運用者の要求に応じて複数のアプリケーションを運用することが考えられる．この運用形態では，稼働中のアプリケーションを停止させずに，新規アプリケーションを構成するプログラムをノードに配備することや，配備したプログラムを変更することが要求される．また，WSN を構成するノードはプログラムメモリが非常に限られていることから，複数のアプリケーションを構成するプログラムを全ノードに配備することができず，個々のノードに必要なプログラムだけを配備することが必要になる．

予め敷設された WSN を利用して複数のアプリケーションを追加・変更しながら運用する際，ノードを回収してプログラムを追加・変更すると，アプリケーションの実行を妨げ，回収の手間も要する．そのため，無線を利用して特定のノードに配備されたプログラムを変更する手法としてリプログラミング [Wang 06] が提案された．2.3 章で述べたとおり，リプログラミングには静的配備，動的配備がありそれぞれの代表例に Deluge[Hui 04] と Agilla[Fok 05b] がある．リプログラミングを用いることで，ベースステーションから追加・変更されるプログラムを特定のノードに配備することができる．

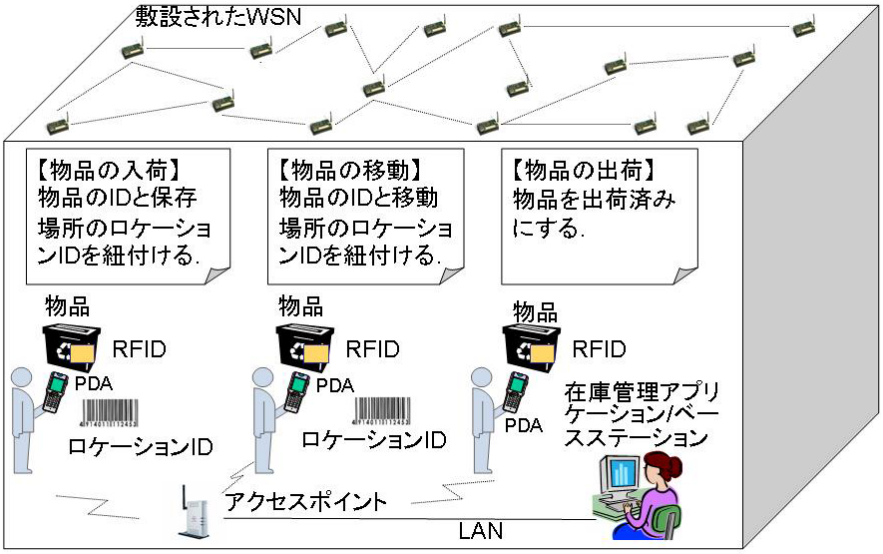

図 3.1: 想定環境

3.2 事例

本研究では，図 3.1 のように，WSN のインフラが事前に敷設された平置き倉庫を想定する．平置き倉庫とは，屋根のある空間で，最小限の設備を所持し，製品やパレットを直接並べて保管する倉庫である (図 1.3 参照)．想定する倉庫には入荷・出荷によって物品が倉庫に入ること，倉庫から出ることがある．加えて，他物品の出荷のためや，倉庫内移動のために物品が移動される場合があり，倉庫の状況に応じて物品の現在位置が変更される可能性がある．本研究では，図 3.1 に示すように，物品が移動されたのか，出荷されたのかといった在庫管理情報は，WSN とは別の在庫管理アプリケーションが把握していると仮定する．また，WSN を構成するノードにおいて，センサを利用することによって物品がノード周辺に存在するか，存在しないかという存在有無のみは検知できると仮定する．こうした在庫管理を実現できる既存システムとして，バーコードや RFID があり，流通業界において一般的に用いられている．

本想定環境において，システム運用者である倉庫業者は，荷主からの要求に応じ，管理を要する高級な物品の周辺で，物品固有の計測と計測された状態に応じた処理を必要とする．例えば，高級ワインの周辺では，周辺温度を 15 ℃未満に保って保存を行い，15 ℃以上の温度が計測された場合，ベースステーションや PDA に通知し状況を報告し，WSN 内

部で空調を調整する．また，美術品の周辺には，夜間（光度が一定の閾値を下回った場合）に赤外線センサが反応した場合に不法侵入者とみなすプログラムを配備し，不法侵入者が検知された場合，侵入者を追跡するプログラムを WSN に散布し，緊急時のインフラとして WSN を利用する．

これらのアプリケーションは，物品の周辺で空間的な計測を実施するため，物品周辺の複数のノードに計測を行うプログラムを配備する必要がある．また，物品周辺での異常や平均値などの空間的な情報を判定するため，計測を行うプログラムと連携して，空間的な情報を判定し WSN 内部で要求に応じた処理を行うプログラムが必要になる．複数のアプリケーションをひとつの WSN を用いて実現する場合，取得されたデータを，その都度ベースステーションに送信すると，ベースステーション付近のノードの電力が枯渇する可能性があるため，WSN 内部で可能な限り処理を実行する．本論文では，こうしたアプリケーションを *LCA*(Locally Centralized Application) と定義する．

LCA は，物品が移動を行った場合，物品の新現在地周辺に移動を行って継続して実行できる必要がある．例えばワインがシステム運用者によって他の場所に移動された場合，ワイン周辺で計測を行うプログラムは配備場所を変更する必要がある．

3.3 動的配備の必要性

ある *LCA* の稼動場所の変更が必要になった場合，静的配備手法を用いると，*LCA* を構成するプログラムを配備先のノードから消去し，新配備先のノードにベースステーションから改めて配備することが必要になる．プログラムの配備場所の変更回数が多い環境下では，図 3.2 に示すとおり，ベースステーションからプログラムを配備する回数の増加によって，ベースステーション付近のノードがプログラムの転送によって電力を消費し，バッテリが枯渇する恐れがある．それに対して動的配備手法では，図 3.2 に示すとおり，ノード上のプログラムを旧配備場所から新配備場所に移動させ，WSN 内部で動的にプログラムを配備することができる．この手法は，ベースステーションからプログラムを配備する手法より，プログラムの平均移動距離が短く特定のノードの電力を消費しない長所があり，プログラムの配備場所の変更回数が多い環境下での運用に適している．

簡易化のためにグリッド上のネットワークを想定[*1] し，ネットワークのサイズを N, プログラムの新配備先の座標を (X,Y)，プログラムの旧配備場所の座標を (Xi,Yi) と置く．静的配備手法は常にベースステーションから配備を行うので，理想的なネットワークにおけ

[*1] 本想定によっても一般性は失われない．

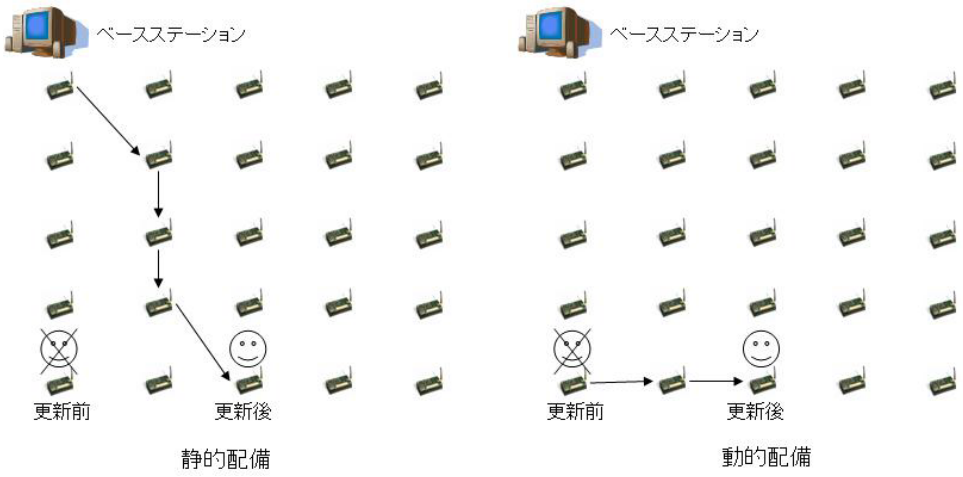

図 3.2: 動的配備と静的配備

る配備にかかるホップ数 Dist_S は 3.1 式で示される．なお，ベースステーションの座標を (1,1) とする．動的配備手法は，WSN 内部でプログラムを移動させることができるので，理想的なネットワークにおけるプログラムの移動に要するホップ数 Dist_D は 3.2 式で示される．仮に N が 10（10x10 の WSN) であると，Dist_S の平均は 6.15 となり，Dist_D の平均は 4.63 となる．ホップ数の平均距離は，中継ノードの数に比例するため，プログラムのための送信パケット数が同一であれば，中継ノードの数が少ないほうが送信パケットは少ない．以上のことから，WSN 内部でプログラムが稼動場所を変更する場合，動的配備に要する WSN 全体の通信量は，静的配備に要する WSN 全体の通信量よりも少ないことが言える[*2]．

$$Dist\_S = MAX(X-1, Y-1) \tag{3.1}$$

$$Dist\_D = MAX(ABS(X-Xi), ABS(Y-Yi)) \tag{3.2}$$

[*2] 直感的に解釈するためには，ベースステーションと旧配備場所を結ぶ線の垂直二等分線で WSN を区切り，ベースステーションに近い領域が静的配備が有利な領域，ベースステーションから遠い領域が動的配備が有利な領域と整理する．更新対象のプログラムの座標をベースステーションとの距離に応じて分類することで，静的配備より動的配備が有利な領域が多くなることがわかる．

3.4 要求の分析

本研究で想定する *LCA* は，物品の周辺で物品固有の計測や処理を行って，物品が移動した場合に，再度物品の周辺で計測や処理を実施できる必要がある．*LCA* 実現にあたる要求は次のとおり整理できる．

要求 A) 計測の実行

物品周辺の複数ノードで計測を実行できる．例えば，ワインの温度を監視するため，ワインの周辺に存在する複数のノードで温度を計測したり，複数のノードで異常な温度を監視することができる．

要求 B) 計測結果の判定と処理

複数のノードで計測された結果を判定し，要求に応じた処理を行う．例えば，ワインの周辺の複数ノードで計測されたデータを集計し平均温度を計算したり，複数ノードで計測された異常な温度データ数を計算し，ワインの周辺の温度が異常であることを判定できる．その後，WSN 内部で空調を変化させるプログラムに通知を行い，適正な温度に保つような指示を出すことができる．

要求 C) 情報の収集・交換

WSN 内外で情報交換や，*LCA* を構成するための情報を収集できる．例えば，ベースステーションでのみ管理されている物品が出荷されたのか，移動されたのかという情報は WSN 内部では判断できないため，WSN 内部のプログラムがベースステーションや WSN 外の計算資源との通信を実施し，情報の収集や交換を実施できる．

要求 D) 新現在地におけるプログラムの動的配備

LCA を構成するプログラムを新現在地周辺に移動させ，プログラムを配備して *LCA* を継続して構成できる．例えば，ワインが *LCA* が配備されていた場所から移動を行った場合，ワインを監視していた *LCA* は配備場所を変更し，新現在地に移動を行ってワインを継続して監視することができる．

3.5 課題

Agilla，ActorNet は MOTE[*3] で稼動するミドルウェアであり，WSN 内部でプログラムを動的に配備することを可能にするが，抽出された要求 A)〜D) を満たす場合，次の課題

[*3] MICAz とその前身である MICA2 の総称

を抱える．

課題 1:*LCA* の構成に適したアーキテクチャ

課題 2:複数プログラムの動的配備

課題 1: Agilla，ActorNet は，個々のプログラムに対して通信や動的配備の機能を提供するが，*LCA* のような複数のプログラムで空間的なアプリケーションを構成する際のアーキテクチャがなく，*LCA* を構築する際に複数のプログラムをアドホックに開発することになる．*LCA* を構成するプログラムを記述する際，以下 2 つの手法がある．

手法 1-1:全てのプログラムを同一にする

手法 1-2:個々のプログラムを記述する

手法 1-1 の欠点は，個々のプログラムのコードサイズが大きくなること，機能分担のためにプログラム間通信を実施することが挙げられる．プログラムのコードサイズが大きくなることにより，軽量に作成すべきプログラムのコード容量が増加して移動効率が悪くなる可能性や，利用しないメモリを確保する課題が生じる．機能分担のためのプログラム間通信は，通信が失敗した際に役割分担がうまくいかず *LCA* を構成できない可能性がある．以上のことから手法 1-1 は WSN に適する実装方針とはいえない．
手法 1-2 は個々のプログラムを記述することで，手法 1-1 の欠点を補うことができる．ただし，機能の異なるプログラムを個別に記述する必要が生じ，各プログラムに適した記述を行うことが必要になる．その結果，プログラマにアドホックな実装を強いることで以下の課題が生じる可能性がある．

- プログラムの実行に影響を及ぼす
- プログラムの動的配備に影響を及ぼす

前者の可能性として，プログラムの機能分担を最適化しないことによって，必要以上にプログラム間通信が必要になりプログラムが煩雑化する可能性や，冗長なプログラム間通信が，定期的な計測処理を一時停止する可能性と処理の不整合を生じる可能性が挙げられる．後者の可能性として，アドホックな実装によって軽量に作成すべきプログラムのコード容量が重くなってしまい，動的配備に所要する時間や配備の信頼性が低下する可能性がある．動的配備の既存研究では，複数のプログラムでアプリケーションを構成する際のアーキテクチャの提案がされておらず，手法 1-2 を選択することにより上記の課題を引き起こす．その結果，*LCA* の構成と *LCA* の継続的な構成に影響を与える可能性がある．以上のことか

ら，*LCA* の構成に適したアーキテクチャが必要となり，以下の課題が抽出される．
課題 1-(a):*LCA* の構成に適したアーキテクチャが必要

課題 2: Agilla では，プログラム間通信を特定のノードに存在する分散タプルスペース [Gelernter 85] を介して行い，ActorNet ではブロードキャストを用いて実行する．WSN に配備された複数のプログラムが配備場所の変更による移動を行う際，ベースステーションから複数のプログラムに移動を命令する手法と，一部のプログラムに移動を命令し，そのプログラムが他プログラムの移動を統率する手法が存在する．前者の手法は，ベースステーションからプログラムに対してプログラム数分の通信成功を必要とし，ベースステーションから距離が離れた場合には再送を要するなど管理が難しくなる課題がある[*4]．そこで後者の手法を取ることが WSN では適する．なお，本研究では既存研究における，他プログラムの移動を統率するプログラムをリーダと呼ぶ．

既存研究では，WSN 内部においてプログラムの位置管理を実施せず，ベースステーションでのみ位置管理を実施している[*5]．Agilla では，リーダが他プログラムに移動を命令する手法として，次の 2 つがある（図 3.3，図 3.4)．

手法 2-1:リーダーが各プログラムに通知　図 3.3 に示すように，リーダーが予め他プログラムの位置を取得し，移動時に他プログラムと通信を行い移動を通知する手法．

手法 2-2:リーダが特定ノードの分散タプルスペースに通知　図 3.4 に示すように，リーダーが，特定のノードに存在する分散タプルスペースに移動命令を書いておき，他プログラムは定期的に特定の分散タプルスペースを読み込み移動を検知する手法．この場合，他プログラムは移動命令の読み込みと結果の取得を定期的に繰り返す必要が生じる．

手法 2-2 は，他プログラムが移動の呼びかけを常時チェックする必要があり，呼びかけが通知されていない場合にもポーリングを行って電力を消費する課題がある．手法 2-1 はポーリングを必要とせず，手法 2-2 より適している．ただし，移動を行いながらお互いの位置を通知しあうことが必要である．WSN ではノード間の距離に依存してパケットロスが生じることは，1.2.2 節で述べた．プログラムの移動時に，通信に高く依存をしたアプ

[*4] 加えて，ある *LCA* が，WSN 内部で他 *LCA* からの通信を受けて配備を変更する場合等には他 *LCA* が全てのプログラムの位置を把握する必要が生じる．
[*5] 各ノードが WSN 全体の情報を所持すると通信数が増加する．また，整合性をとる必要があるため

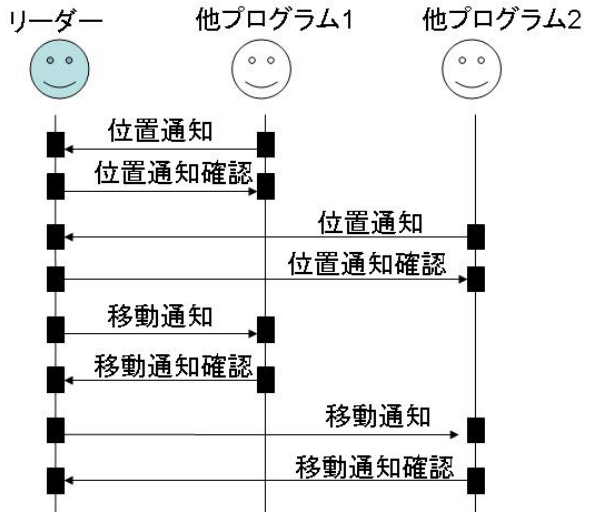

図 3.3: リーダが各プログラムに通知する手法

ローチをとると，次の課題によって，*LCA* を継続して構成できない可能性がある[*6] .

課題 2-(a):移動の呼びかけに失敗し，一部のプログラムが移動できない.

課題 2-(b):一部のプログラムが移動に失敗する.

なお，ActorNet もブロードキャストを用いてリーダが他プログラムに移動を通知し，移動時の通信手法は Agilla と異なるが，通信に高く依存をしたアプローチは共通であり，本質的な課題は変わらない.

3.6 まとめ

本章では，本研究で想定するアプリケーション *LCA* の要求の分析を行い，動的配備の既存研究が抱える課題を特定した．*LCA* を実現する際，以下 4 つの要求を満たすことが必要である.

要求 A) 計測の実行

要求 B) 計測結果の判定と処理

要求 C) 情報の収集・交換

要求 D) 新現在地におけるプログラムの動的配備

LCA を実現する際に，既存研究は，以下 2 つの課題を抱えている.

[*6] これを回避するために，プログラムの一部が移動できなかった場合に備えて，移動後にプログラム間で通信を実施したり，プログラムが存在しない場合に検索をしたり，プログラムが消失した場合に複製を行う対策があるが，通信が必ずしも信用できない WSN では状況を正確に把握することに限界がある.

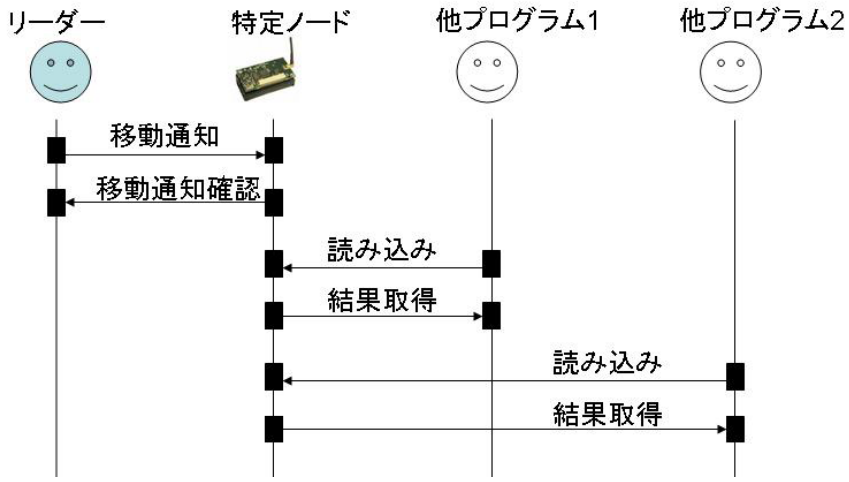

図 3.4: リーダが特定ノードの分散タプルスペースに通知する手法

課題 1:*LCA* の構成に適したアーキテクチャ

アーキテクチャの不在によってプログラマにアドホックな実装を強い，その結果，*LCA* を構成するプログラムの実行に影響を及ぼす可能性と，プログラムの動的配備に影響を及ぼす可能性があり，結果として *LCA* の（継続的な）構成の信頼性を低下させる．

課題 2:複数プログラムの動的配備手法

既存手法はプログラムの動的配備時に通信に高く依存したアプローチを取り，その結果，通信品質が高くない WSN においては以下の課題をかかえ，動的配備後に *LCA* を継続して構成できない可能性がある．

課題 2-(a) :移動の呼びかけに失敗し，一部のプログラムが移動できない．

課題 2-(b) :一部のプログラムが移動に失敗する．

第 4 章

アーキテクチャと複数プログラムの動的配備

本章では，3.4 節で述べた要求のうち，要求 A)，B)，C) を満たすための *LCA* を構成するためのアーキテクチャを 4.1.1 節で提案し，要求 D) を満たすための複数プログラムの動的配備手法を 4.1.2 節で提案する．

4.1 提案手法

4.1.1 アーキテクチャ

LCA が，物品周辺で計測・処理を実行する際の要求は A) 計測の実行，B) 計測結果の判定と処理，C) 情報の収集・交換となる．本論文では，要求 B)，A)，C) を専任する *Master*, *Slave-S*，*Slave-M* というコンポーネント化されたプログラムにより構成されるアーキテクチャを提案し，*LCA* を構成する．*Master*，*Slave-S*，*Slave-M* の機能を表 4.1 に示す．

Master，*Slave-S*，*Slave-M* でアーキテクチャを構成する理由を以下に示す．A) は，物品周辺のノードで計測を行うために，複数で物理的に異なるノードで動作する必要があり，一つのプログラムとしてコンポーネント化することが必要である．B) は，複数の A) のプログラムと通信を行って物品周辺での事象を把握するため，同様に一つのプログラムとしてコンポーネント化する．C) は，A) または B) に機能を負わせることが可能だが，対象物を追跡する場合や，通信失敗回数が多い場合など，移動を行って情報を収集する方が適する場合がある．A)，B) のどちらかに C) の機能を負わせると，A) は *LCA* 固有の計測を実施しているため，移動を行うと物品周辺での計測ができなくなる可能性がある．B) は A) からの報告を受け付けているため，移動を行うと A) と距離が離れて通信の劣化や通信の転送などが生じ WSN では効率が悪い．加えて，移動を考慮した場合，プログラムは軽量な方が効率が良い．そこで，C) も一つのプログラムとしてコンポーネント化する．

表 4.1: アーキテクチャ

プログラム	説明	要求
Master	*Slave-S*，*Slave-M* の動的な生成・配備を行い，ライフサイクルを管理する．*Slave-S* から報告される計測結果の判定を実施し，各 *LCA* の要求に応じた処理を行う．WSN 内外の通信を *Slave-M* に依頼する．*LCA* に一つだけ存在し，配備完了後は移動まで特定のノードで稼動する．	B
Slave-S	計測処理を実行し，異常の検知などの計測結果を *Master* に送信する．*LCA* に複数存在することができ，配備完了後は *Master* の移動まで特定のノードで稼動する．	A
Slave-M	*Master* からの命令によって，必要に応じて移動を行って情報収集や情報交換を行う.（例：アプリケーションとの通信を実行）*LCA* に一つ存在する．	C

表 4.1 に示すとおり，*Master* は，B) 以外に *Slave-S*，*Slave-M* の動的な生成・配備や，ライフサイクルの管理を行う（4.1.2 節で詳細を記述する）．*Slave-S* は物品周辺での計測処理を実施し，*Slave-M* は *Master* から依頼によって，情報収集と情報交換を実施する．

本アーキテクチャの動作例を以下および図 4.1，図 4.2，図 4.3 に示す．

1. ワインの周辺に各プログラムが配備される．*Slave-S* は 4 体，*Slave-M* は一体配備される．
2. 図 4.1 に示すように，*Slave-S* は定期的な計測を実施し，異常がある場合に *Master* に通知を行う．
3. 図 4.2 に示すように，計測対象が移動した場合，*Master* は *Slave-M* をベースステーションや PDA で稼動する在庫管理アプリケーションに派遣する．
4. *Master* は *Slave-M* 経由で在庫管理アプリケーションから新配備場所を取得する．
5. 図 4.3 に示すように，新配備場所に移動し，*LCA* を構成する．

Slave-S は，計測対象を計測するためのプログラムである．このプログラムは，物品の周辺で空間的なデータを取得するため *LCA* に複数個存在し，複数のノードに一体づつ配備される．*Slave-M* は，*Master* からの通知に応じて情報収集・交換を実行する．例えば，

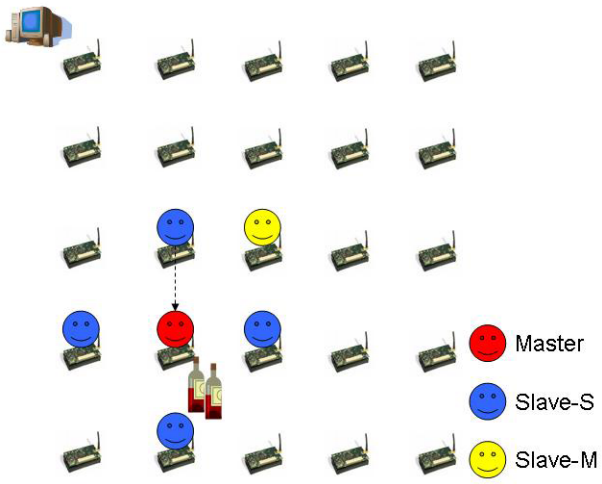

図 4.1: アーキテクチャの動作例 1：*Master*，*Slave-S*，*Slave-M* が配備され計測を開始する

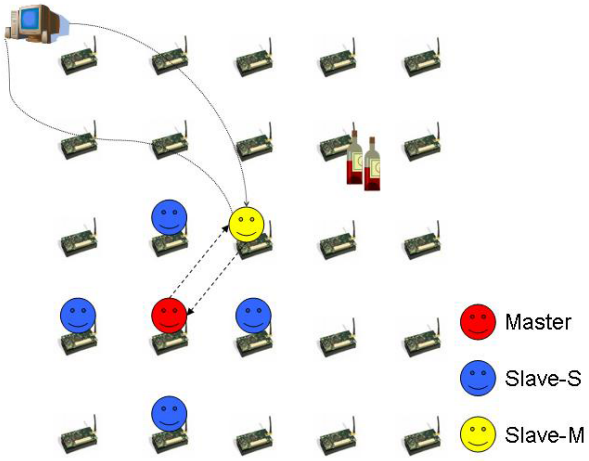

図 4.2: アーキテクチャの動作例 2：計測対象が移動を行った場合，*Master* は *Slave-M* をベースステーションに派遣し，Slave-M はベースステーションから物品の新現在を取得する．

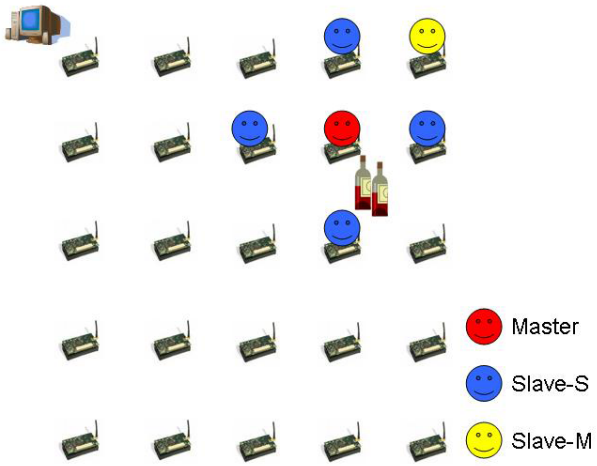

図 4.3: アーキテクチャの動作例 3：新配備場所に移動し *LCA* を再構成する．

在庫管理アプリケーション[*1] との通信を実施し，WSN 内部および WSN 外部で把握できない事象を報告・取得する．この例では，WSN 内部ではプログラム側で計測対象の存在有無は計測できるが，計測対象が移動したのか，出荷されたのか判断できない．在庫管理アプリケーションと通信を行って，計測対象の新現在地を取得する．出荷された場合には，*LCA* の活動を停止する結果を *Master* に通知する．*Master* は，*Slave-S* から報告された計測結果に合わせた判定を実施する．例えば，計測結果が異常であった場合に，在庫管理アプリケーションとの通信を *Slave-M* に依頼する．

4.1.2 複数プログラムの動的配備

LCA の稼動場所の変更によって，WSN 内部で複数プログラムを動的に配備する場合がある．既存手法を用いると，3.5 節で挙げた以下の課題 2-(a)，(b) によって *LCA* を継続して実行できない場合がある．

課題 2-(a) 移動の呼びかけに失敗し，一部のプログラムが移動できない．

課題 2-(b) 一部のプログラムが移動に失敗する．

提案手法では，これらの課題を解決するために，*LCA* を継続実行することに着眼した手法をとり，課題 2-(a)，(b) が起こる確率を減らす手法を提案する．

課題 2-(a) の確率を無くすため，特定のプログラムが，他のプログラムを生成する手法を提案する．この手法は，移動先で他のプログラムを生成し，配備することができるため，

[*1] ベースステーションや PDA で稼動を行っている在庫管理アプリケーション

他のプログラムに移動の呼びかけを行う必要がないが，移動前に配備したプログラムを消去する必要がある[*2]．課題 2-(b) は通信が 100%信頼できない WSN では完全に解決することはできないが，確率を下げることはできる．既存手法を用いてプログラム N 体を移動させる場合，個々のプログラムが移動に失敗する確率を P とすれば，N 体全てのプログラムが移動に成功する確率 (*Probability*) は以下の式で表現できる．なお，このときプログラムのコードサイズは同一と仮定する．

$$Probability = (1 - P)^N \tag{4.1}$$

提案手法では，この確率 (*Probability*) を向上させるために N を減らすアプローチをとり，*Master* が *Slave-S* と *Slave-M* のコードを全て保持する (つまり N が 1 になる)．以上のことから，提案手法では，課題 2-(a)，(b) が起こる確率を減らすことができる[*3]．

提案手法は以下の手順で複数プログラムの動的配備を行う．

1. *Master* に移動が通知される．
2. *Master* は配備した *Slave* を消去する命令をミドルウェアに通知する．
3. *Master* は移動を実施する．
4. *Master* は移動先で *Slave* を生成し配備する．

図 4.4 に (2)〜(4) に対応した例を示す．(2) *Master* は移動が必要になった際に，配備された *Slave* の消去をミドルウェアに通知する．*Slave* の消去はミドルウェアが実行する．(3) *Master* は *Slave* の消去確認を行わずに，次の配備場所に移動する．(4) *Master* が移動後，*Master* の配備先のノードのミドルウェアを介して *Slave* を生成し，*Slave* を周辺ノードに配備する．これを繰り返すことで，*LCA* を継続して実行することができる．

4.1.3 実装

表 4.4 に実装環境を示す．本研究では Agilla をベースとして，以下の機能を拡張した．

拡張機能 1　*Master* が *Slave* を生成する機能

拡張機能 2　*Master-Slave* 間のロケーション管理機能

[*2] 4.2.2 節で消去の成功率を評価し，評価環境において 100%を達成している．
[*3] 提案手法では，*Master* が移動に失敗すると *LCA* を継続して構成できない欠点があるが，既存手法でもリーダが一体存在することから，この欠点は既存手法と同じである．

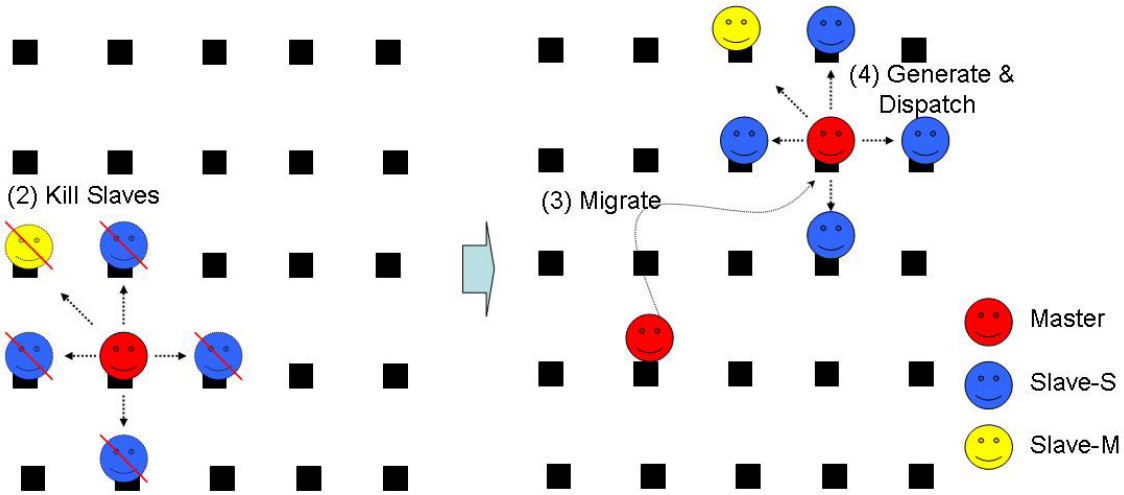

図 4.4: 複数プログラムの動的配備

拡張機能 3　*Slave* の消去機能

本研究で実装したミドルウェアを図 4.6 に示す．プログラムはミドルウェアに ISA(Instruction Set Architecture) により処理を依頼する．*Instruction Set Handler* は，実行される命令を振り分け，拡張された機能と Agilla で提供される機能を振り分ける．本研究で作成した ISA を表 4.2 に，本研究で実装したミドルウェアの主要コンポーネントを付録 D の表 D.1 に示す．

拡張機能 1: ミドルウェアは新たなプログラムが送られてきた場合，*Master* であるか否か，移動命令とプログラムの ISA から判別する．*Master* であった場合，*Code Manager* が *Master* のコードから *Slave* を生成し，*Master*，*Slave* の各コードをそれぞれのコードバッファに格納する．このとき，*Master* のコードは，実行コードバッファに格納され実行されるが，*Slave* のコードは *Slave Code* に格納された後，配備場所に送信される．*Slave* の配備場所は，*Master* コード中に記載された *Deployment Pattern* により決定され，*Deployment Pattern* に従って *LCA Manager* が *Slave-S*，*Slave-M* をそれぞれの配備場所に定められた数だけ送信する．*Slave-S*，*Slave-M* を受信したミドルウェアは，*Slave* のコードを実行コードバッファに格納し，*Slave-S*，*Slave-M* が実行される．

拡張機能 2: ミドルウェアは，*Slave* を送信し Ack を受け取った際に，*Slave* の送信先のノード ID と *Slave* のプログラム ID を *Location Table* に記録する．同様に *Slave* を受信した際には，*Master* の存在するノード ID と *Master* のプログラム ID を *Location Table* に記録する．*Location Table* に記録された情報を用いて *Master-Slave* 間の通信が可能になる．

表 4.2: 本研究で作成した ISA

ISA	説明
gsmove	*Master* の実行状態を保存して，*Master* のコードを移動させる．引数で配備パターン，移動先を指定する．
gwmove	*Master* の実行状態を保存せずに，*Master* のコードだけを移動させる．引数で配備パターン，移動先を指定する
killslave	本 ISA を実行する *Master* に配備された *Slave* を消去する．
masterloc	*Slave* が自身を配備した *Master* の位置をスタックに積む.（*Master* との通信が可能になる）
slavemloc	*Master* が自身の配備した *Slave-M* の位置をスタックに積む.（*Slave-M* との通信が可能になる）
startslaves	*Master* コード中で *Slave-S* の開始を宣言する．
endslaves	*Master* コード中で *Slave-S* の終了を宣言する．
startslavem	*Master* コード中で *Slave-M* の開始を宣言する．
endslavem	*Master* コード中で *Slave-M* の終了を宣言する．
gettime	時間とプログラム ID を取得（評価とデバッグのため）
spawnslaves	*Slave-S* を所定の場所に生成する．
spawnslavem	*Slave-M* を所定の場所に生成する．

なお，*Master-Slave* 間の通信は Agilla で提供されている分散タプルスペースを介して行われる．

拡張機能 3: プログラムの移動が必要になった場合，*Master* は *Slave* を消去する命令をミドルウェアに下す．*KillManager* は，*Location Table* の情報をもとに消去対象となる *Slave* の存在するノードに消去命令を送信し，命令を受信したミドルウェアは，指定された *Master* の ID を親にもつ *Slave* の消去を行う．

Master，*Slave* のプログラミングは，Agilla で提案されている ISA と，本研究で拡張した ISA(表 4.2) を用いて記述し，表 4.3 のように *Master* の内部に *Slave* の処理を記述する形を取る．*Slave-S* の処理は，startslaves，endslaves という ISA の間に記述する．同様に，*Slave-M* の処理は，startslavem，endslavem の間に記述する．

なお，*Slave* の配備場所を決定する *Deployment Pattern*(配備パターン) は，ミドルウェアの中に保存された *Slave* の空間的な配備情報である．本研究で使用した配備パターンを図 4.5 に示す．配備パターン 1 は，*Slave-S* を *Master* の隣接ノードのうち，二次元平面で表現した場合の上下左右の合計 4 つの隣接ノードに一体づつ配備し，*Slave-M* を *Master* の隣接ノードのうち，二次元平面で表現した場合の右上の隣接ノードに一体配備する．配備

表 4.3: プログラムの構造

// Master Code ————-
(*Master* の処理の記述)
// Slave-S Code ————
startslaves //*Slave-S* の開始
(*Slave-S* の処理の記述)
endslaves //*Slave-S* の終了
// Slave-M Code ————
startslavem //*Slave-M* の開始
(*Slave-M* の処理の記述)
endslavem //*Slave-M* の終了

表 4.4: 実装環境

項目	説明
OS	TinyOS 1.2.2[Hill 00]
開発言語	NesC 1.2.7[Gay 03]
ランタイム	Agilla 3.0.2[Agi]
開発環境	Windows XP Professional SP2, Cygwin 1.5.23

パターン 2 は，*Slave-S*，*Slave-M* をそれぞれ一体づつ，*Master* の隣接ノードのうち二次元平面で表現した場合の上下の合計 2 つのノードに配備する．配備パターン 3 は，*Slave-S*，*Slave-M* をそれぞれ一体づつ，*Master* の 2 ホップを要する隣接ノードのうち二次元平面で表現した場合の上下の合計 2 つのノード配備する．図 4.5 に示す配備パターンは *Master*，*Slave-S*，*Slave-M* を異なるノードに配備しているが，ひとつのノードに複数のコンポーネントを配備することは可能である．配備パターンの充実や，WSN の変化する状況（ノードの残バッテリ，他ノードとのネットワークリンクの状況等）に合わせた配備パターンの取得は，本研究の提案範囲外である．

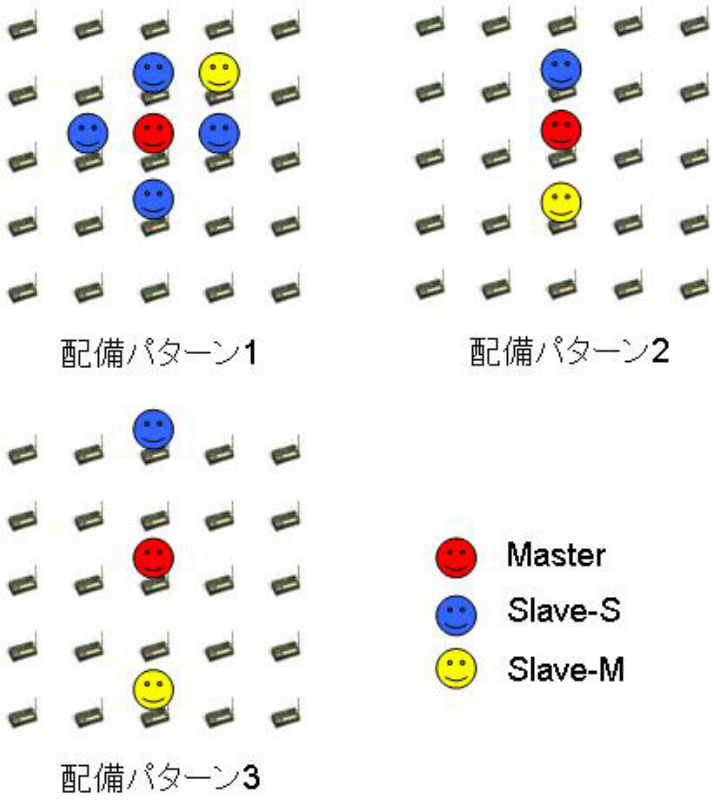

図 4.5: Deployment patterns1-3

4.2 評価

4.2.1 シミュレーション環境

本論文では，TOSSIM[Levis 03] において Lossy モデル[*4] を使用して評価を行った．なお，評価時のルーティングプロトコルとして Geographic Routing を用い，ノード間の通信は単純な Greedy Forwarding[Karp 00b] で行われる．評価における共通の計測条件を表 4.5 に示す．なお，隣接ノードのみ通信可能としているのは，効率的にマルチホップ通信を評価するためであり，提案手法の制約ではない．

本論文の評価で用いる 5feet，8feet，10feet はグリッド上に配置されるノード間の距離を示し，それぞれ約 1.5m，2.4m，3.0m に相当する．TOSSIM の Lossy モデルは，ノード間の距離に応じたノード間リンクのビットエラー率を記述したファイルをツールを用いて

[*4] TOSSIM では，TinyOS の実装言語である NesC で記述されたプログラムを稼動させることができる．また，Lossy モデルを用いることでノード間の非対称リンクとパケットロスをシミュレーションできる．

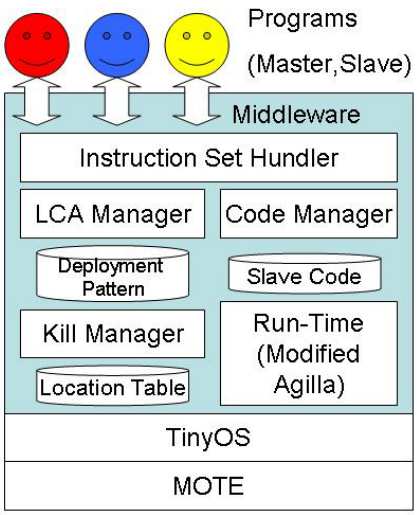

図 4.6: ミドルウェア

生成し，このファイルをシミュレーション時に読み込むことによって，実測値と同等なパケットロスを再現する（[Levis 03]）．表 4.5 に，生成されたファイルにおける，全隣接ノード間（対角線を含む）の平均ビットエラー率と全隣接ノード間（対角線を含む）の非対称リンクの割合を示す[*5]．

表 4.5: 共通の計測条件

項目	説明
トポロジ	グリッド (8 × 8)
Lossy モデル	5feet,8feet,10feet
平均ビットエラー率	0.18%(5feet),0.29%(8feet),0.96%(10feet)
非対称リンクの割合	63.86%(5feet),77.14%(8feet),75.24%(10feet)
ノード間通信	対角線を含む隣接ノードのみ可
通信速度	40Kbps
メッセージサイズ	36byte(Active Message)

[*5] パケットロス率や，パケットエラー率の詳細，算出式については文献 [Levis 03] [tos] を参照のこと．なお，本論文中では，ノード間の距離が増加した場合に ”パケットロスが多い ”と記述しているが，文献 [Levis 03][tos] において，ノード間の距離が増加した場合にパケットロスが多くなることが示されているため，妥当な表現である．

4.2.2　アーキテクチャの評価

LCA を実現する場合，*Master* が移動を実施しながら *Slave* を配備して，*LCA* を構成することが必要になる．そこで，提案手法を用いて以下のシナリオを評価した．（具体的なコードは付録 A を参照のこと）[*6] ．を用いて評価を行った．

1. *Master* が配備場所に移動する．
2. *Master* が *Slave-S* を隣接ノードに 1 体づつ合計 4 体配備する．
3. *Master* が *Slave-M* を隣接ノードに 1 体配備する．
4. *Master* の命令に応じて *Slave-M* がベースステーションに行く．
5. *Slave-M* がベースステーションに報告を行う．
6. ベースステーションより新しい場所を取得し移動を行う

この測定を 6 回繰り返すことを 1 回とし，5feet，8feet，10feet で各 15 回づつ実施した結果を，それぞれ図 4.7，4.8，4.9 に示す．なお，本計測では，*LCA* の移動間隔として 2 ホップ移動させた．成功率は (1) から (6) までの全てに成功した確率を示す，2 回移動する成功率は，各環境でそれぞれ，100%，93.3%，73.3%であった．移動回数が増えるに従って，パケットロスが多い環境では，移動を伴う *LCA* 構成の成功率が顕著に低下した．5 回移動する成功率は，各環境でそれぞれ，86.6%，60.0%，33.3%であった．本評価により，2 回の移動を伴い，初期配備も含めて合計三ヶ所で計測を実施する *LCA* は，5feet，8feet の環境において 90%以上の確率で継続して運用できることがわかる．ただし，ノード間隔が広がりパケットロスが多い環境における *LCA* の継続運用や，稼動場所の変更の多い *LCA* の継続運用は今後の課題となる．

Agilla の手法を用いて，複数のプログラムが移動を実施して *LCA* を連続して構成する場合の 5feet，8feet，10feet における評価を，図 4.7，4.8，4.9 に示す (15 回の測定)．既存手法より，提案手法のほうが *LCA* を継続運用する際に有効な手法であることがわかる．

提案手法の欠点として，以下が考えられる．

欠点 (a):*Master* の移動失敗により *LCA* が構成できない

欠点 (b):*Slave* の配備失敗により *LCA* を構成できない

[*6] 本評価では，図 4.5 に示す配備パターンの 1 を利用している．なお，付録 A に示すコード容量は 92byte である．

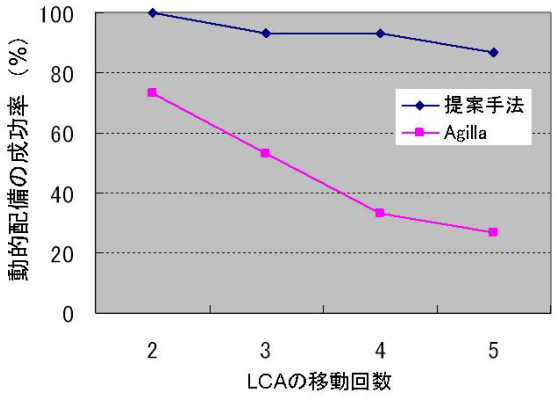

図 4.7: アーキテクチャの評価 (5feet)

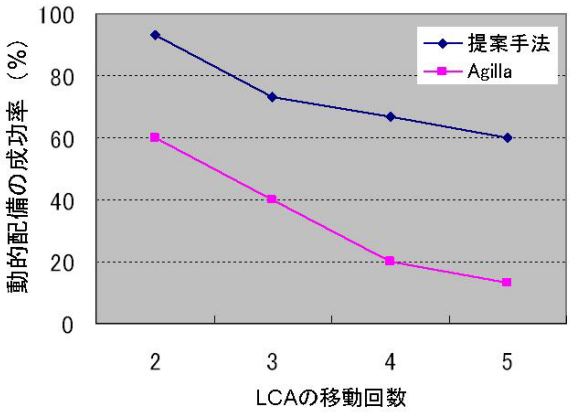

図 4.8: アーキテクチャの評価 (8feet)

欠点 (c):*Slave* の消去失敗により *Slave* が残存する

Slave-S の消去に失敗した場合，*Slave-S* は，*Master* の旧存在場所の分散タプルスペースに継続して計測結果を通知する．そのため，利用者のいないデータを送受信するために *Slave-S* が存在するノードと *Master* の旧存在場所のノードの電力を無駄に消費し続ける[*7]．

これらの欠点を評価するため，付録 A に示すコードをシミュレーション環境において，2 ホップ移動させる評価を各環境で 60 回実施した．表 4.6 に評価結果を示す．

欠点 (a): *Master* の移動失敗率は，*Master* が移動に失敗した確率を示す．各シミュレーション環境で，5.0%，8.3%，18.3%計測された．パケットロスが多い環境での *Master* の

[*7] *Slave-S* のコードは高々十数 byte〜数十 byte であり，メモリに対する制約よりも，電力を消費し続ける方が課題と考えられる．本評価における *Slave-S* のコード容量は 20byte であり，メモリの空き容量 (60Kbyte) に対する割合は 0.03%となる．

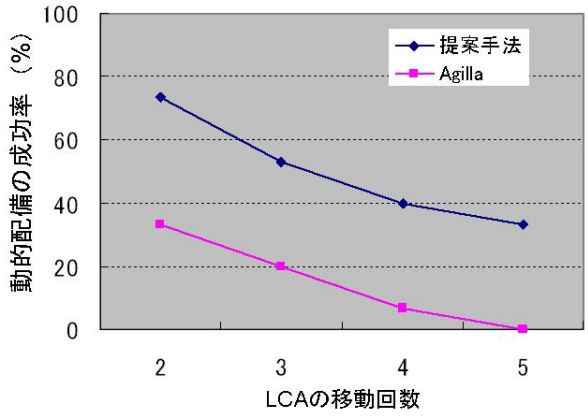

図 4.9: アーキテクチャの評価 (10feet)

移動失敗は今後の課題となる．

欠点 (b)，(c): 本ミドルウェアでは，配備の失敗や，消去の失敗時に再送[*8] を行うことで対応している．再送無-配備成功率は，再送を実施しなかった場合に *Slave-S* を 4 体，*Slave-M* を 1 体配備することに成功した確率である．再送を実施しない場合，各環境では，80.0%，54.9%，19.0%しか配備に成功しない．再送無-KILL 成功率は，再送を行わない場合に配備した *Slave* を全て消去できる確率である．各環境で，90.9%，68.6%，52.4%しか消去できない．再送を行うことで，評価環境では配備成功率，KILL 成功率ともに 100%を達成している．以上の結果から，提案するアーキテクチャは *LCA* を構成する際のアーキテクチャとして妥当と判断できる．

表 4.6: 欠点の評価

シミュレーション環境	5feet	8feet	10feet
Master の移動失敗率	5.0%	8.3%	18.3%
再送無-配備成功率	80.0%	54.9%	19.0%
再送有-配備成功率	100%	100%	100%
再送無-KILL 成功率	90.9%	68.6%	52.4%
再送有-KILL 成功率	100%	100%	100%

[*8] 本評価では，*Slave* の配備失敗時の再送回数は 3 回，消去失敗時の再送回数は 2 回として評価を実施した．再送条件は提案手法，Agilla ともに同等であり，詳細は 4.2.5 節で述べる．

4.2.3 複数プログラムの動的配備の評価

提案手法，Agilla の手法で，3 つのプログラムを一定の距離を保ちながら移動させ，複数のプログラムを動的に配備する際の成功率を計測した．以下を 4 回繰り返すことを 1 回の計測として，15 回の計測を実施し，各手法によって 3 つのプログラム（図 4.5 における配備パターン 2，3 ）が移動しながら自身を何回配備できるかを評価した．

・1 つのプログラムに移動を通知する．

・3 つのプログラムが隣接ノードに移動を行い自身を配備する

Agilla の手法では各プログラムのコード容量は，73byte，45byte，45byte[*9] である．提案手法では，*Slave-S,M* を 1 体づつ準備し 3 体の動的配備を実現した．*Master* のコード容量は 60byte であり，*Slave-S,M* のコード容量は 12byte である．

計測結果を表 4.7，表 4.8 に示す．表 4.7 はプログラム間の距離が 1 ホップ[*10] の時の結果であり，表 4.8 はプログラム間の距離が 2 ホップ[*11] の時の結果である．平均配備可能数は，Agilla の手法では，3 つのプログラムのどれか一つが移動に失敗したり，移動の呼びかけの通信が失敗して取り残されるまで 3 つのプログラムが全て配備された回数の平均である．提案手法の場合は，*Master* が移動に失敗するか，*Slave*2 体のうち 1 体でも配備に失敗[*12] するまで配備できた回数の平均である．成功率は，4 回移動を成功させ，かつ 3 つのプログラムが全て配備される確率である．

表 4.7 は，プログラム間の間隔が 1 ホップ (配備パターン 2) の時の計測結果である．提案手法では，10feet の Lossy 環境において，成功率が 66.7%であるのに対し，Agilla の手法では，40.0%であった．平均配備可能数は，それぞれ 3.60 と 2.66 であった．また，パケットロスが少ない 5feet の Lossy 環境において，提案手法が 100%の確率で配備ができるのに対して，Agilla の手法では 80.0%に留まった．

表 4.8 は，移動を行うプログラム間の間隔が 2 ホップ (配備パターン 3) の時の計測結果である．提案手法では，10feet の Lossy 環境でも，成功率が 60.0%であるのに対して，Agilla の手法では，13.3%であった．平均配備可能数は，3.2 と，1.2 であった．また，パケット

[*9] Agilla の手法では，お互いの位置を把握するためのコードが煩雑になり，提案手法よりコード容量が増える．

[*10] 一つのプログラムを中心として，隣接ノードのうち 2 つのノードにプログラムが存在している．図 4.5 に示す配備パターンの 2 を利用．

[*11] 一つのプログラムを中心として，2 ホップ隣のノードのうち 2 つのノードにプログラムが存在している．図 4.5 に示す配備パターンの 3 を利用．

[*12] 提案手法ではある場所で *Slave* の配備に失敗しても，次の配備場所で *Slave* の配備に成功できる可能性があるが，この場合は失敗としている．

ロスが少ない 5feet の Lossy 環境において，提案手法が 100%の確率で配備ができるのに対して，Agilla の手法では 53.3%に留まった．以上の結果から，提案手法の動的配備手法は Agilla で動的配備を実現するよりも有効であると判断できる．

表 4.7: 複数プログラムの動的配備（1 ホップ）-配備パターン 2

プログラム	5feet	8feet	10feet
Agilla-平均配備可能数	3.46	3.00	2.66
Agilla-成功率	80.0%	60.0%	40.0%
提案手法-平均配備可能数	4.00	3.73	3.60
提案手法-成功率	100%	80.0%	66.7%

表 4.8: 複数プログラムの動的配備 (2 ホップ）-配備パターン 3

プログラム	5feet	8feet	10feet
Agilla-平均配備可能数	2.86	2.06	1.20
Agilla-成功率	53.3%	26.7%	13.3%
提案手法-平均配備可能数	4.00	3.40	3.20
提案手法-成功率	100%	73.3%	60.0%

4.2.4　オーバヘッドの評価

提案手法は *Master* が移動先で *Slave* を配備するため，トラフィックが *Master* の移動先のノードに集中し，動的配備時間のオーバヘッドが生じる．そこで，本節で**動的配備時間**と動的配備に必要な**トラフィック**の計測結果を示す．

動的配備時間：動的配備時間を，プログラムが動的配備を開始する時間（*T_Start*）から，全てのプログラムが動的配備を終了して実行可能となる時間（*T_End*）の差分と定義する．図 4.11，4.10 に，それぞれ提案手法，Agill の動的配備時のシーケンス図を示す．なお，図 4.11，4.10 では，Agilla における他プログラム，提案手法における *Slave* は簡単のため各一体のみ示す．提案手法の動的配備時間は，*Master* が配備した *Slave* を消去する時間（T_Start）から，*Master* によって配備された全ての *Slave* が実行可能となる時間（T_End）との差分に相当する．Agilla の動的配備時間は，リーダーが他プログラムに移動を命令す

る時間（T_Start）から，全てのプログラムが移動に完了し，全てのプログラムが実行可能となる時間（T_End）との差分に相当する．

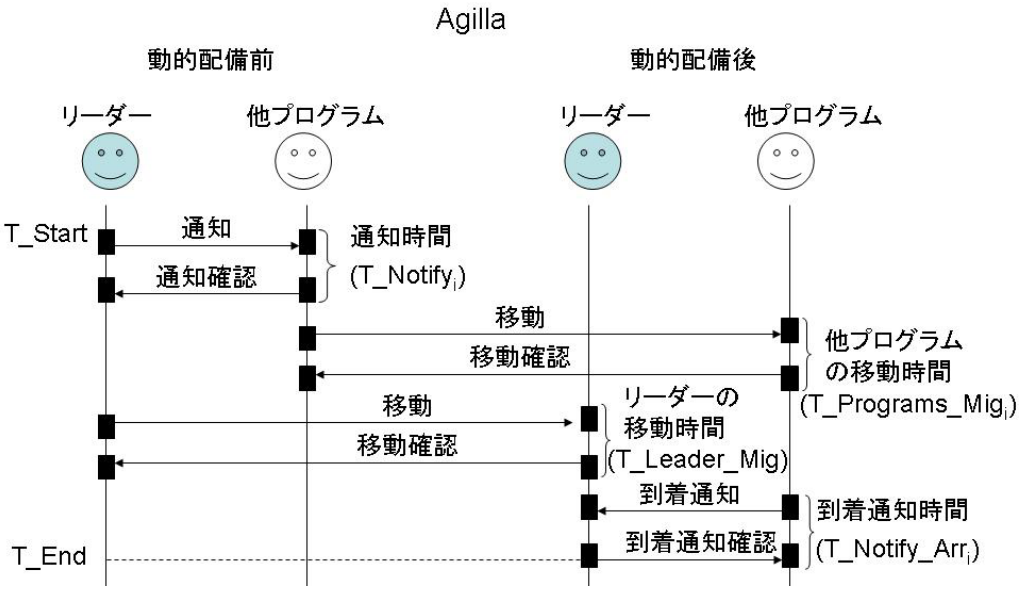

図 4.10: Agilla の動的配備時のシーケンス図

表 4.9 に 5feet，8feet，10feet における，各パターン (図 4.5) で 1 ホップ *LCA* を移動させた際の平均動的配備時間（計測回数 15 回）を示す[*13]．表 4.9 より，以下 2 つの傾向がわかる．

傾向 1:提案手法の動的配備時間は，パケットロスが少なく (5feet，8feet)，プログラム数が少ない環境（パターン 2，3）では Agilla と同程度の動的配備時間となる．

傾向 2:提案手法の動的配備時間は，パケットロスが多く (8feet，10feet)，プログラム数が大きな環境 (パターン 1) では，Agilla に比べて 2 倍〜3 倍と長くなる．

トラフィック：各シミュレーション環境における，提案手法と Agilla の動的配備時のパケット数を評価した結果を，図 4.12，図 4.13，図 4.14，図 4.15 に示す．なお，図 4.12 はパケットロスがない理想的なネットワーク環境における評価結果を示している．図 4.12，図 4.13，図 4.14，図 4.15 は，配備パターン 2 で動的配備を実施する場合の WSN 内部での全送信パケットと，動的配備の移動距離を示し，15 回の計測の平均値である．提案手法，

[*13] 動的配備に失敗した場合は除く，なお配備パターン 1 のコード容量は提案手法の *Master* が 92byte(*Slave-S*：20byte，*Slave-M*：31byte)，Agilla のリーダーは 139byte，他プログラムは 45byte である．配備パターン 4，5 は，提案手法の *Master* が 60byte(*Slave-S*：12byte,*Slave-M*：12byte)，Agilla のリーダーは 73byte，他プログラムは 45byte である．

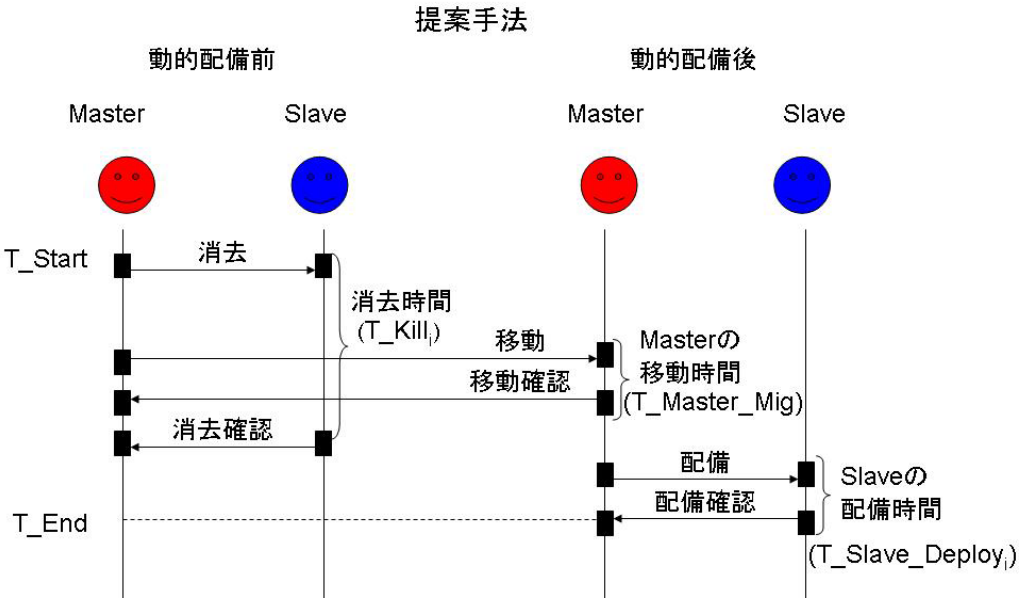

図 4.11: 提案手法の動的配備時のシーケンス図

Agilla のコード容量を表 4.10 に示す.

図 4.11 に示すとおり，提案手法のパケットの合計数は，各ノードで送信される消去・消去確認，移動・移動確認，配備・配備確認に要する送信パケット数の和に相当し，Agilla のパケットの合計数は，各ノードで送信される通知・通知確認，移動・移動確認（リーダ，他プログラム），到着通知・到着通知確認に要する送信パケット数の和に相当する．図 4.12，図 4.13，図 4.14，図 4.15 より，提案手法は Agilla に比し，動的配備時のトラフィックを半減できることが分かる．

4.2.5　補則：再送回数について

提案手法，Agilla ともにシミュレーション環境において再送を実施した．提案手法では，図 4.11 における，消去・消去確認を 2 回，*Master* の移動を 3 回，*Slave* の配備・配備確認を 3 回再送している．Agilla では，図 4.10 における，通知・通知確認を 3 回，全プログラムの移動を 3 回，*Slave* の到着・到着通知確認を 3 回再送している．提案手法，Agilla ともに再送回数は同等である．

表 4.9: 平均動的配備時間

パターン	手法	5feet	8feet	10feet
1	Agilla	3.78(sec)	4.47(sec)	7.94(sec)
1	提案手法	4.33(sec)	11.51(sec)	21.19(sec)
2	Agilla	1.35(sec)	1.50(sec)	1.76(sec)
2	提案手法	1.07(sec)	2.28(sec)	3.48(sec)
3	Agilla	1.40(sec)	1.83(sec)	2.15(sec)
3	提案手法	1.32(sec)	2.62(sec)	4.01(sec)

表 4.10: パターン 2 のコード容量

手法	リーダ/*Master*	他プログラム/*Slave*
Agilla	73(byte)	45(byte)
提案手法	60(byte)	12(byte)

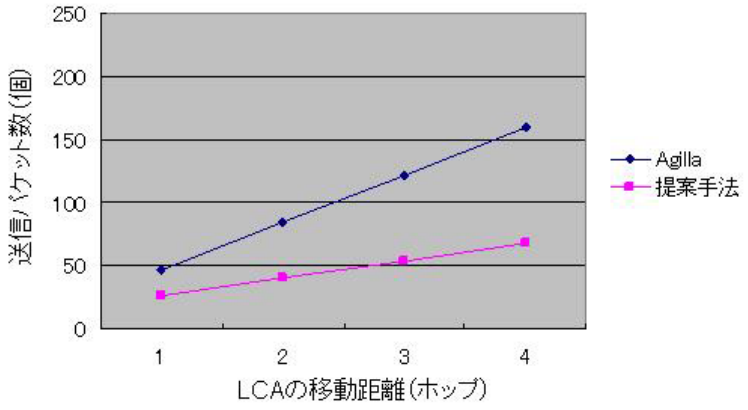

図 4.12: 動的配備時のパケット数 (パケットロスなし)

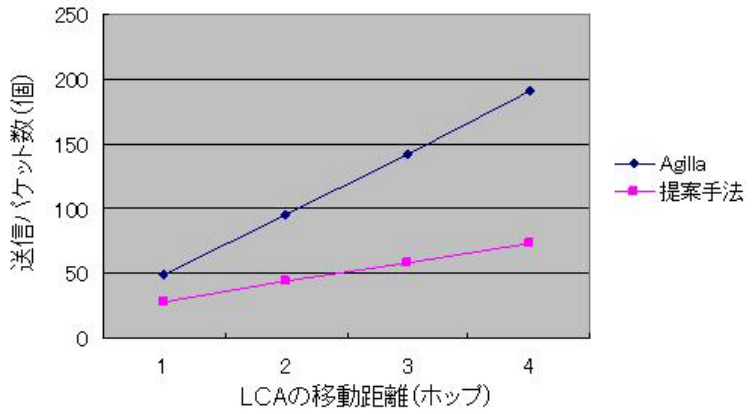

図 4.13: 動的配備時のパケット数 (5feet)

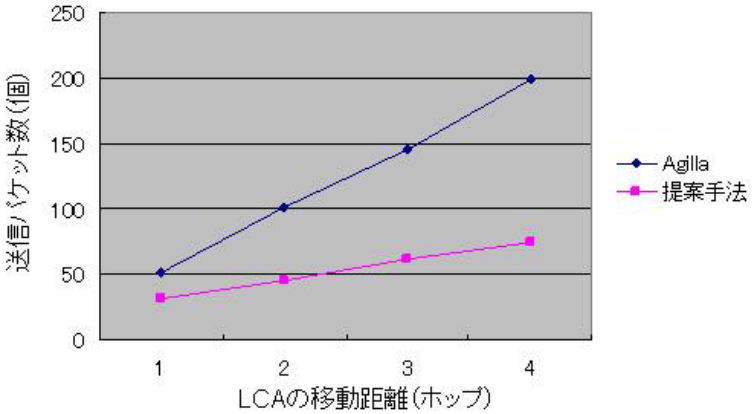

図 4.14: 動的配備時のパケット数 (8feet)

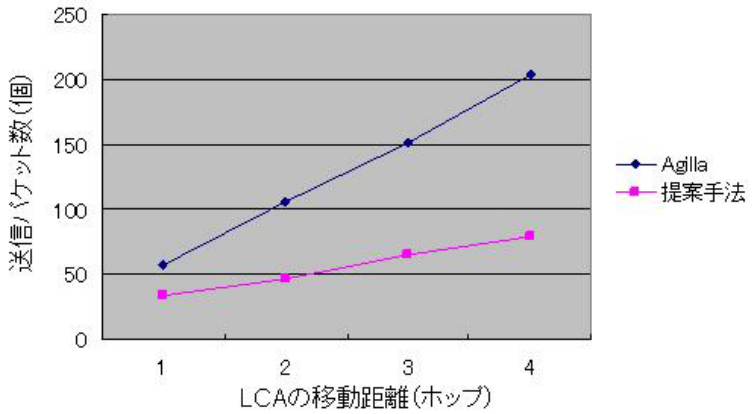

図 4.15: 動的配備時のパケット数 (10feet)

4.3 まとめ

本章では，*LCA* を構成する際の手法として，以下を実現するミドルウェアを構築した．
1) コンポーネント化されたプログラムによって構成されるアーキテクチャ
2) 複数プログラムの動的配備手法
シミュレーション環境において，評価を実施した結果，1) 提案アーキテクチャは既存手法よりも，約 15%〜50%*LCA* の継続構成の成功率を向上させることができた．2) 提案する動的配備手法は，既存手法よりも約 20%〜40%動的配備の成功率を向上させることができた．

提案手法は *Master* が *Slave* を所持し，配備先で *Slave* を配備する．また，配備前に *Slave* を消去する．この手法は，以下の欠点が生じる可能性がある．
3)*Master* が *Slave* を配備するのに時間がかかり，動的配備に時間がかかる
4)*Master* が *Slave* を抱えて移動をするため，トラフィックが増大する
5)*Slave* の消去失敗や *Slave* の配備失敗が生じる
これらについて評価を実施した結果，3) シミュレーション環境において提案手法の動的配備時間は既存手法の動的配備時間に比べ，最大で約三倍となった．動的配備時間については，5.2 で議論する．4)WSN 全体でのトラフィックは提案手法は Agilla に比べて半減できることが計測された．5)*Slave* の消去と，*Slave* の配備をシミュレーション環境においては 100%の確率で実施することができた．なお，この際に再送を行うことによって，*Slave* の消去と *Slave* の配備を実現しているが，その再送回数は既存手法と同等である．

本章では提案アーキテクチャによるアプリケーション構成の有効性と，提案する複数プログラムの動的配備の有効性を示した．提案手法は，複数のプログラムで構成されるアプリケーションをひとつの WSN 内で複数運用する際の有効な手段の一つとなると考えられる．

第 5 章

議論

本章では，本研究で提案した手法について，その有効性や限界について議論を行う．5.1 節では，提案手法が既存手法に比べて動的配備の成功率を向上できた理由を述べる．5.2 節では，*Master* が *Slave* を抱えることによって生じる提案手法のオーバヘッドの影響を動的配備時間とトラフィックの観点から議論する．5.3 節，5.4 節では，提案手法と既存手法の比較を消費電力とメモリの観点から行い，5.5 節で提案手法の限界をまとめる．5.6 節，5.7 節では，提案手法の最適化の余地を考察し，5.8 節，5.9 節で，提案手法の対象とするアプリケーションと有効な範囲を議論する．5.10 節において，提案手法と関連研究の比較を行う．

5.1 動的配備

5.1.1 成功率向上の理由

提案する動的配備手法は Agilla に比較して動的配備の成功率を約 20%〜40%向上できることを，4.2.3 節で示した．本節では，この理由について考察を加える．

提案手法と Agilla はモバイルエージェントのランタイムを所持し，プログラムの動的配備時にモバイルエージェントで用いられる smove(強い移動)，wmove(弱い移動) を用いて移動を行う．図 5.1 に，smove，wmove の詳細なプロトコルを示す．図 5.1 に示すとおり，smove ではプログラムコード (Code) の他，State，Heap，Stack，Rxn およびそれらの Ack を送受信する．wmove では Code と State のみである．また，動的配備時に Agilla では図 4.10 に示すとおり，リーダと他プログラム間で分散タプルスペースを介して移動通知と到着確認通知を実施し，提案手法は図 4.11 に示すとおり消去を実施する．前者を Tuple，後者を KILL とした際の通信プロトコルを図 5.2 に示す．図 5.1，図 5.2 中の Code，State，Heap，Stack，Rxn，Ack，TupleRequest，TupleResponse，KillMsg，KillAck の

説明とパケット数を表 5.1 に示す.

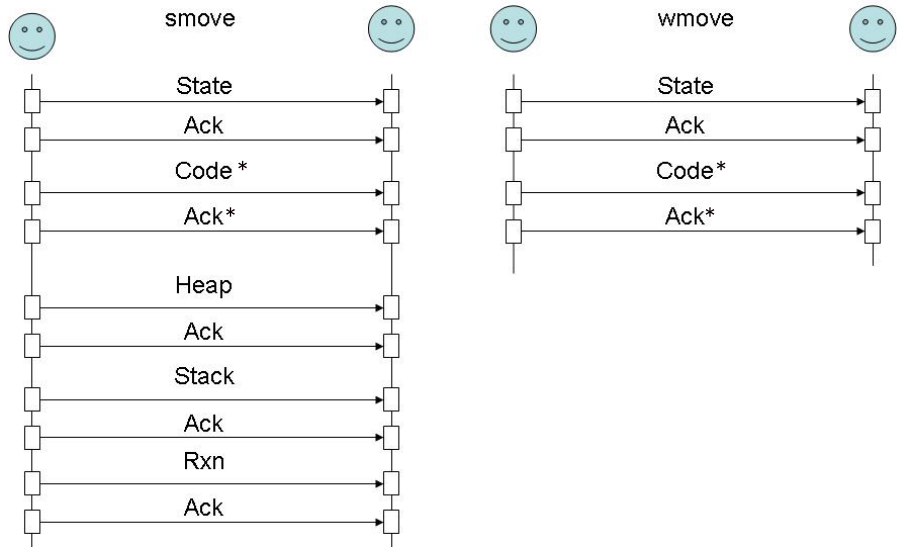

図 5.1: smove と wmove のプロトコル

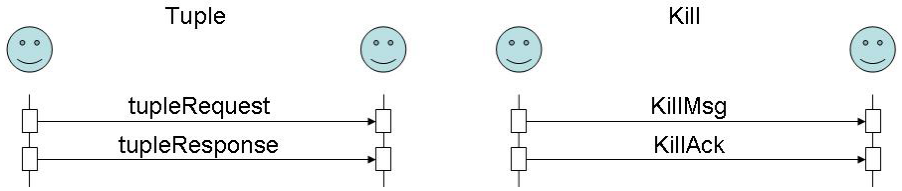

図 5.2: タプル通信と Kill のプロトコル

LCA を構成して，動的配備を実施する場合，Agilla ではプログラム同士の位置管理を実施し，お互いの位置を記憶したまま移動する必要が生じる．それに加えて，位置管理を実施する前に，特定の分散タプルスペースでお互いの位置を通知しあう初期化処理が必要となる．この初期化処理を毎回実施することは無意味であるため，プログラムカウンタを所持してこの処理をスキップする必要が生じる．これらの理由によって，Agilla では，全プログラムが実行状態を維持する必要が生じ，N 個のプログラムで動的配備する際に N 個のプログラム全てが smove を必要とする．図 5.3 に，Agilla の動的配備時のシーケンスと smove の関係を示す．提案手法では *Master* のみが smove で移動し，*Slave* は *Master* の移動先で配備パターンにしたがって生成・配備される．*Slave* は動的配備前に消去されており，実行状態を維持しないため，wmove で *Master* の移動先から *Slave* の新配備先に送信

表 5.1: smove と wmove のパケットとパケット数

パケット	説明	パケット数
Code	プログラムコードを 26byte 単位で区切り，プログラムのコード（ISA) を格納する．26byte 以上では複数個になる．	複数
State	目的ノードの ID，送信ノードの ID，プログラム ID，プログラムカウンタ，コードサイズ，Rxn 数，Heap 数等を格納.	1
Heap	目的ノードの ID，送信ノードの ID，プログラムのヒープに格納されたデータを格納.	1
Stack	目的ノードの ID，送信ノードの ID，プログラムのスタックに積まれたデータを格納.	1
Rxn	目的ノードの ID，送信ノードの ID，Rxn（コールバック関数）を格納.	1
Ack	Code, State, Heap, Stack, Rxn の送信ノードに中継ノードから送られる受信確認．目的ノードの ID，送信ノードの ID を格納.	1
TupleRequest	目的ノードの ID，送信ノードの ID，目的ノードへ送信するタプルデータ等を格納.	1
TupleResponse	目的ノードの ID，送信ノードの ID，目的ノードから取得されたタプルデータを格納.	1
KillMsg	目的ノードの ID，送信ノードの ID，Kill 対象となる *Slave* のプログラム ID，対象 *Slave* に紐付けられた *Master* のプログラム ID を格納.	1
KillAck	中継ノードから KillMsg 送信ノードに対して送信される受信確認.	1

される．以上のことから，提案手法は Agilla に比べて，N-1 個分の State，Heap，Stack，Rxn およびそれらの Ack のトラフィックを削減することができる．図 5.4 に，提案手法の動的配備時のシーケンスと smove，wmove の関係を示す．

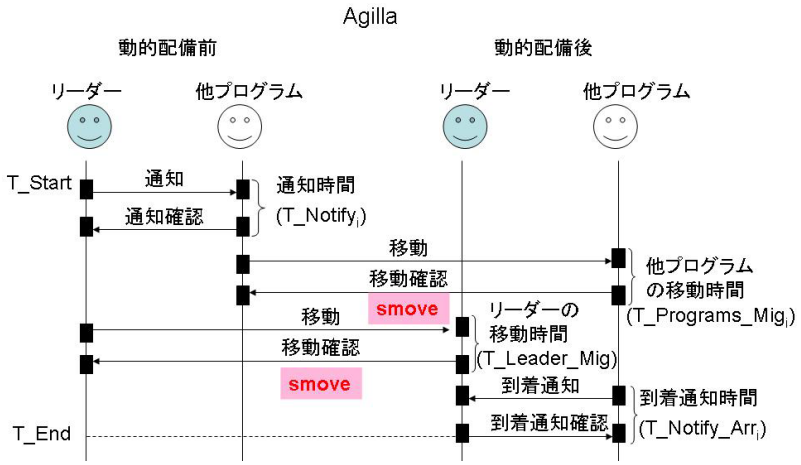

図 5.3: Agilla の動的配備時のシーケンスと smove

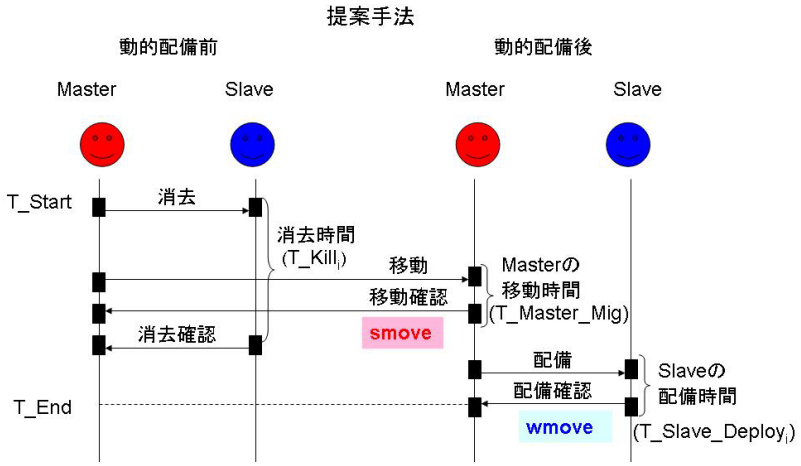

図 5.4: 提案手法の動的配備時のシーケンスと smove，wmove

図 5.1，図 5.3 および図 5.4 より，理想的なネットワークにおける動的配備時の Agilla，提案手法の送信パケット数は，式 5.1，5.2 で表せる．

$$Trafiic\_A \quad = \quad 2(N-1)(TupleRequest + TupleResponse)$$

$$
\begin{aligned}
&+ \quad M((State\_L + Stack\_L + Heap\_L + Rxn\_L + 4 * Ack) \\
&+ \quad (CodeSize\_L/26)(Code\_L + Ack) \\
&+ \quad (N-1)((State\_O + Stack\_O + Heap\_O + Rxn\_O + 4 * Ack) \\
&+ \quad (CodeSize\_O/26)(Code\_O + Ack))) \qquad (5.1)
\end{aligned}
$$

$$
\begin{aligned}
Trafiic\_P \quad &= \quad (N-1)(KillMsg + KillAck) \\
&+ \quad M((State\_M + Stack\_M + Heap\_M + Rxn\_M + 4 * Ack) \\
&+ \quad (CodeSize\_M/26)(Code\_M + Ack)) \\
&+ \quad (N-1)(State\_S + Ack) + (CodeSize\_S/26)(Code\_S + Ack) \qquad (5.2)
\end{aligned}
$$

なお，式 5.1，5.2 における M は動的配備の移動距離，Nはプログラム数を示し，添え字_L，_O，_M，_S はそれぞれ，リーダ，他プログラム，*Master*，*Slave* を示す．表 5.1 に示すとおり，Code(プログラムコード) 以外は 1byte である．プログラム数 (N) を 3，4，5 とした場合の理想的なネットワークにおける Agilla，提案手法のトラフィックを図 5.5，図 5.6，図 5.7 に示す[*1]．なお，提案手法のコード容量は *Master* が 60byte，*Slave-S* が 12byte，*Slave-M* が 12byte であり，図 5.5，5.6，5.7 で共通している．Agilla のコード容量は，図 5.5 ではリーダが 73byte，他プログラムが 45byte，図 5.5 ではリーダが 95byte，他プログラムが 45byte，図 5.5 ではリーダが 117byte，他プログラムが 45byte となる．提案手法のコード容量がプログラム数に依存せず一定となり，Agilla はプログラム数によってコード容量が変わる．コード容量の違いについては，5.4 節で詳細を述べる．図 5.5，図 5.6，図 5.7 の範囲において，提案手法は動的配備時のトラフィックを Agilla に比して最低 43%〜最大 69%削減することができることがわかる．この結果，提案手法の動的配備の成功率は，送信パケット数が多い既存手法に比べて高い値を達成している[*2]．

5.1.2　既存手法のライブラリ化による影響

5.1.1 節の式 5.1，5.2 で示される提案手法，既存手法のトラフィックの差異は，大きく以下の 2 つに分類できる．

[*1] パケットロスがある環境でのトラフィックは図 4.13，図 4.14，図 4.15 を参照のこと．
[*2] 提案手法，既存手法ともに動的配備時に再送を実施するが，再送回数と再送間隔は同一である．

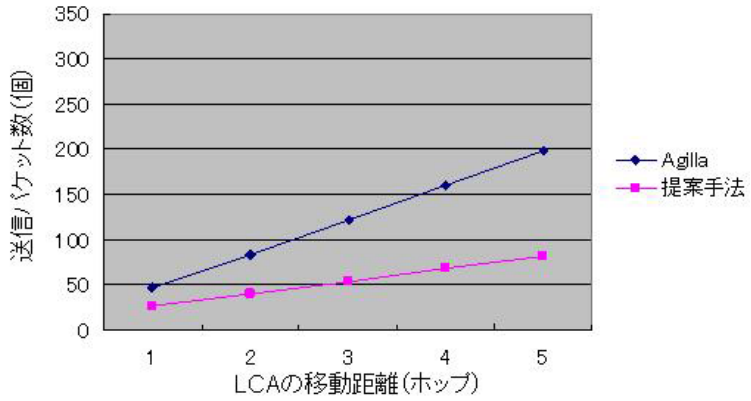

図 5.5: 動的配備時のパケット数-プログラム数 3 (パケットロスなし)

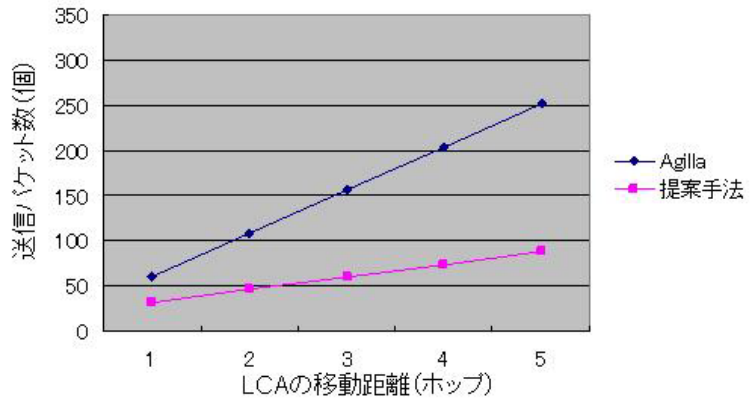

図 5.6: 動的配備時のパケット数-プログラム数 4 (パケットロスなし)

実行状態の維持　既存手法では実行状態を N（プログラム数）だけ必要とし，提案手法に比べて N-1 個分の Heap，Stack，Rxn およびその Ack のトラフィックが増加する．

コード容量　既存手法では，プログラム同士の位置管理を行うためのコードが提案手法に比べて増加する．提案手法では *Master* が *Slave* を保持しているので，位置管理は不要となる．

Agilla では位置管理を実施するためのコード[*3] が必要となる．リーダ，他プログラムで必要となる位置管理のコードをライブラリ化することによって 1byte で記述できたと仮定

[*3] 付録 B に Agilla の他プログラムのコードを示す.

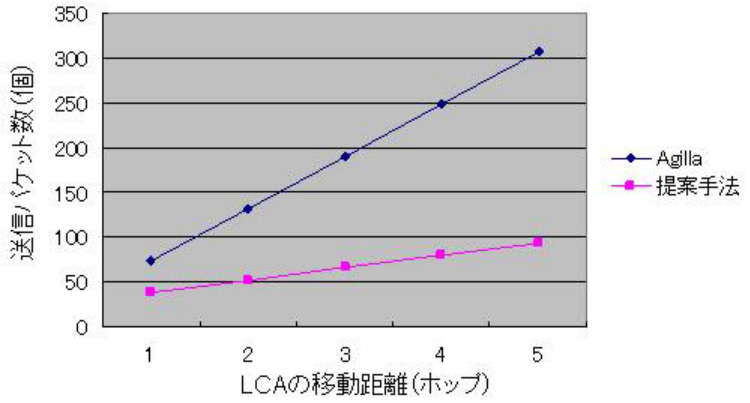

図 5.7: 動的配備時のパケット数-プログラム数 5 (パケットロスなし)

する[*4]．図 5.8 に，図 5.5 で示すトラフィックを，Agilla でコードをライブラリ化することによって削減できた場合のトラフィックを示す．図 5.8 より，以下の 2 つがわかる．

1. Agilla の位置管理のコードを限界まで削減できたと仮定しても，提案手法の優位性は変わらない．
2. Agilla では，位置管理のためのコード容量の増加よりも，実行状態の維持の方がトラフィックの増加に影響を与えている．

Agilla における実行状態の維持は，Agilla が WSN 内部でプログラム同士の位置管理を実施していないことに起因している．実行状態の維持を不要とするためには，なんらかの位置管理機構を実装する必要があり，リーダが動的に他プログラムの位置を把握し，移動通知を実施できること，他プログラムが動的にリーダの位置を把握できることが求められる．これらの要求を満たす実装手法は複数あると考えられるが，提案手法は有効な実装手法の一つと考察できる．

5.1.3 Lossy 環境の影響

4.2.3 節では，表 4.7，表 4.8 において，配備パターン 1 および 2 の 5feet，8feet，10feet の動的配備の平均成功回数を示した．図 5.9 に，1feet，3feet，13feet，15feet の Lossy 環境での評価を加えた提案手法，Agilla の動的配備の成功率を示す．1feet，3feet，13feet，

[*4] 1byte で表現することは，動的配備されるプログラムの座標を静的に記述することが求められ，プログラムが移動する場合は不可能となる．

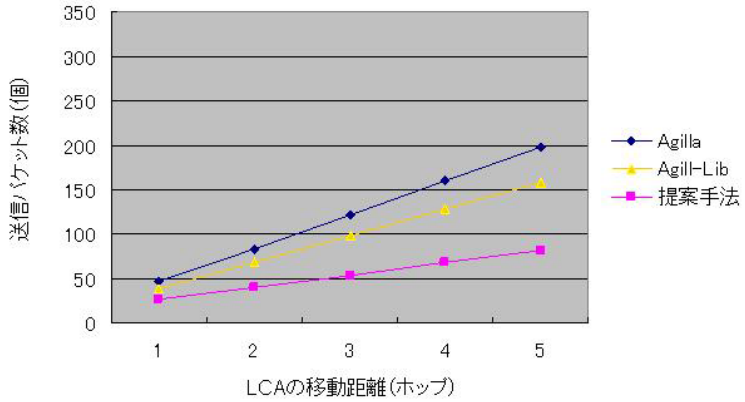

図 5.8: ライブラリ化の影響

15feet での評価では，表 4.7，表 4.8 と同様に各環境で 15 回の計測結果の平均を取得した．なお，1 回の計測あたり 4 回の動的配備を実施しており，図 5.9 において動的配備の平均成功回数が 4 であれば 15 回の計測全て 4 回動的配備できたことを意味する．

図 5.9 より，以下の 2 つがわかる．

1．パケットロスの少ない環境 (1-3feet) では提案手法，Agilla の平均成功回数は 3.5 以上になる．その結果，その差は 0.5 以下となり，提案手法と Agilla の動的配備の成功回数に顕著な差は存在しない．

2．提案手法は既存手法に比べ，計測を実施した全環境で平均成功回数が上回る．

1 より，将来的にパケットロスの少ない環境が実現された場合（信頼性の高い通信プロトコルの登場など），提案手法の有効性が薄れることが考察できる．2 は，5.1.1 節で記述したとおり，提案手法のトラフィックの削減によって成功回数が向上していると考察できる．なお，図 5.9 から，15feet では提案手法の平均成功回数も 4.0 の半分である 2.0 を下回ることがわかる．3 回移動する LCA を想定した場合，提案手法の有効範囲は 10feet までとなる．なお，20feet の環境で測定を実施した結果，提案手法，既存手法ともに現状の再送回数では有意な結果を取得することが不可能であった．

5.2 提案手法のオーバヘッド

本節では，4.2.4 節において評価された**動的配備時間**，**トラフィック**について論じる．

動的配備時間：提案手法の動的配備時間 ($Time\_P$)，Agilla($Time\_A$) の動的配備時間は以下の式で表現できる（図 4.11，4.10 参照）．なお，5.3,5.4 式における N は全プログラム

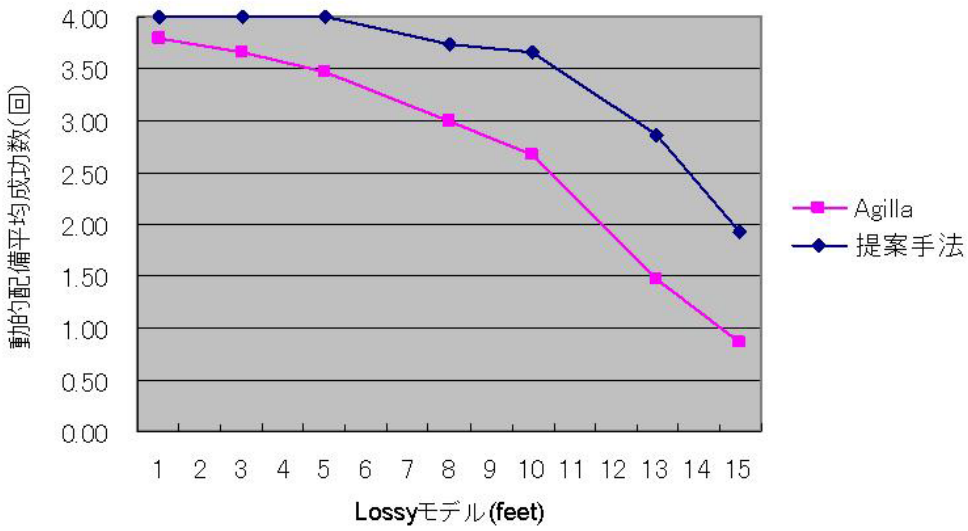

図 5.9: 動的配備の成功率と Lossy 環境の関係

数を示す[*5]．

$$Time\_P = MAX(\sum_{i=1}^{N-1} T\_Kill_i, T\_Master\_Mig + \sum_{i=1}^{N-1} T\_Slave\_Deploy_i) \quad (5.3)$$

$$Time\_A = MAX((\sum_{i=1}^{N-1} T\_Notify_i + T\_Leader\_Mig), MAX_{1<i<N-1}(\sum_{j=1}^{i} T\_Notify_j + T\_Programs\_Mig_i + T\_Notify\_Arr_i)) \quad (5.4)$$

4.2.4 節では，平均動的配備時間に以下の傾向があった．

傾向 1:提案手法の動的配備時間は，パケットロスが少なく (5feet，8feet)，プログラム数が少ない環境（パターン 2，3）では Agilla と同程度の動的配備時間となる．

傾向 2:提案手法の動的配備時間は，パケットロスが多く (8feet，10feet)，プログラム数が大きな環境 (パターン 1) では，Agilla に比べて 2 倍〜3 倍と長くなる．

[*5] 5.3 式に示すように，提案手法は消去確認を行わずに *Master* が移動を行い，*Slave* を移動先で配備する．5.4 式に示すように，Agilla のリーダーは全ての他プログラムに順番に通知を行ってから移動を行う．Agilla の他プログラムは通知を待ってから移動と到着通知を実施する．

傾向 1 については，N が少なく，パケットロスが少ない環境では，*Slave* の配備時間を要さず，全ての *Slave* を配備する時間を意味する 5.3 式の第二引数の第二項 ($\sum_{i=1}^{N-1} T\_Slave\_Deploy_i$) の影響が小さく，提案手法の動的配備時間が短くなると考えられる．

傾向 2 については，N が多く，パケットロスが多い環境では，*Slave* の再送が *Master* の存在するノードで集中・連続して起こることにより，上に示した 5.3 式の第二引数の第二項の影響が大きく，動的配備時間が長くなる．Agilla の場合は，5.4 式に示すとおり，再送が起こってもプログラムが存在するノードが別であり，再送は複数のノードで分散して生じるのでリーダーと他プログラムの移動は大きなオーバヘッドとはならない．

提案手法の動的配備時間は，最悪の場合 Agilla の動的配備時間の約三倍かかる (表 4.9 における配備パターン 1，10feet) が，本研究では倉庫を想定し，計測対象の物品は通常は移動を行わず，物品移動時の動的配備速度に対する要求を想定していない．例えば，二日間倉庫に存在する物品が，配備パターン 1，10feet 環境で二回移動を行ったとしても，動的配備時間は約 0.025% [*6] となり，無視できる範囲となる．

なお，表 4.9 では配備パターン 1〜3 までの動的配備時間を示したが，提案手法において *Slave* 数を増加した際の動的配備時間を図 5.11 に示す（計測回数 15 回の平均値）．図 5.11 には，配備パターン 1〜3 に加えて，図 5.10 に示す配備パターン 4〜6 を追加した結果を示す．図 5.10 に示すとおり，配備パターン 4 は合計 8 個の *Slave*(*Slave-S*：7，*Slave-M*：１) を所持し，配備パターン 5 は合計 12 個の *Slave*(*Slave-S*：11，*Slave-M*：１) を所持し，配備パターン 6 は合計 14 個の *Slave*(*Slave-S*：13，*Slave-M*：１) を所持する．

図 5.11 より，以下の 2 つがわかる．

1．配備パターン 4，5，6 の範囲では，*Slave* 数を増加しても動的配備時間が頭打ちになる現象が計測されない．

2．*Slave* 数を増加すると動的配備時間が増加し，配備パターン 6(*Slave* 数 14) では 120 秒程度かかる．

1 については，現状の再送回数と再送間隔を用いると，配備パターン 1〜7 までの動的配備を実現でき，再送回数と再送間隔に従って，時間が増加していると考察できる．時間的な限界よりも，再送回数を減少することにより，動的配備に失敗する回数が増加し，このトレードオフをとる必要が生じると考察できる．

2 については，適用するアプリケーションによって限界となることが考えられる．*Slave* を

[*6] 100(21.19*2)/2(24*60*60)

14 個程度必要とするような，空間的な大きな規模で計測や処理を実施する *LCA* を想定する．この *LCA* が，動的配備時間中（120 秒）の計測が無視できないような計測や処理を実施する場合は，提案手法は不向きとなる．*Slave* 数と適合するアプリケーションの整理は今後の課題となる．

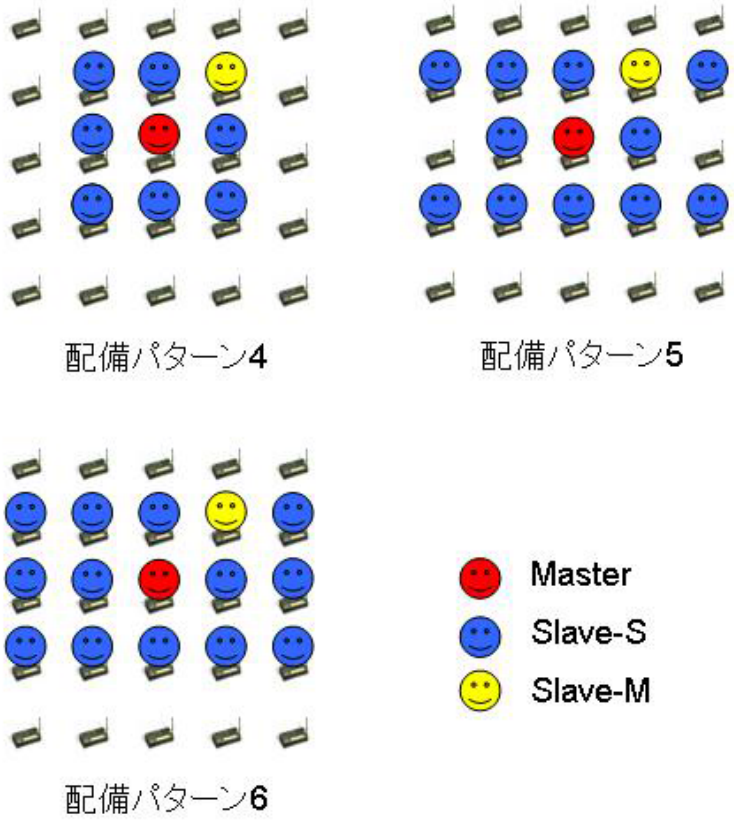

図 5.10: Deployment patterns 4-6

トラフィック：図 4.12，図 4.13，図 4.14，図 4.15，図 5.5，図 5.6，図 5.7 より，提案手法は Agilla に比較して動的配備時のトラフィックを半減できることがわかった．また，両手法の動的配備のトラフィックの詳細も式 5.1，5.2 で示した．なお，図 4.12，図 4.13，図 4.14，図 4.15 には，配備パターン 2 でのトラフィックを示したが，*Slave* 数を増加させて配備パターン 1 で実施した場合との比較結果を図 5.12 に示す．図 5.12 には 10feet の環境と，パケットロスがない環境における両手法のトラフィックを示すが，10feet の環境では 15 回の平均値を示す．図 5.12 より，*Slave*（プログラム）数が増加し，*LCA* の移動距離が増加するに従って，提案手法のトラフィックと既存手法のトラフィックの差が大きくなり，提案手法は効率良くプログラムを動的配備できることがわかる．

図 5.11: 提案手法の平均動的配備時間と Slave 数

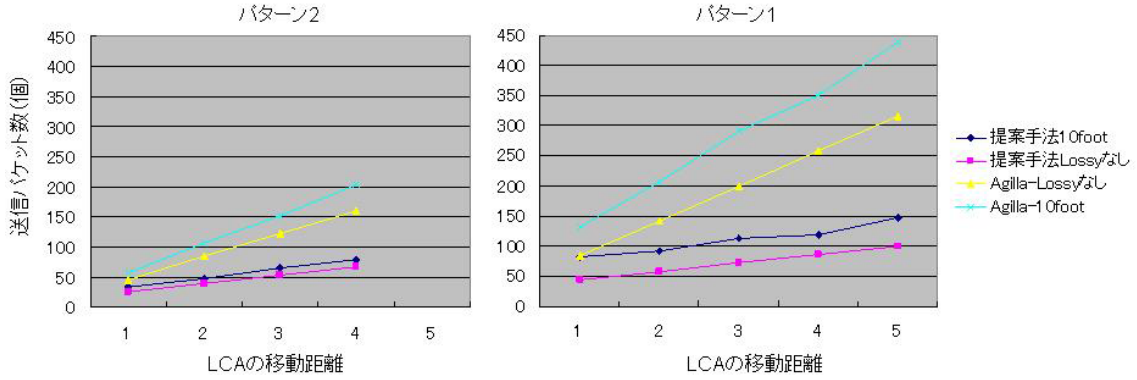

図 5.12: 配備パターン 1，2 のトラフィック

提案手法は *Slave* が *Master* の存在するノードから配備されるために一部のノードに負荷がかかる．表 5.2 に，理想的なネットワークにおける各パターンにおける動的配備時の最大トラフィックをもつノードの送信パケット数を示す[*7]．表 5.2 は，配備パターンによっては，提案手法は一部にトラフィックが集中する制約が存在することを示唆する．ただし，本制約は，本研究の想定環境である倉庫において，物品がある程度均質に分散することを考慮すると，*Master* も均質に分散するために，特定のノードに *Slave* を配備する負荷が集中することにはならず，WSN 内部のトラフィックを Agilla に比して半減していることから，想定環境における課題とならないと考察できる．

[*7] Agilla における配備パターン 4，5，6 の配備はシミュレーション環境の制限によって実施することが不可能であった．

表 5.2: 最大トラフィックをもつノードの送信パケット数

パターン	手法	パケット数
1	Agilla	30
1	提案手法	22
2	Agilla	14
2	提案手法	18
3	Agilla	14
3	提案手法	18
4	Agilla	NA
4	提案手法	32
5	Agilla	NA
5	提案手法	48
6	Agilla	NA
6	提案手法	56

5.3 消費電力

提案手法と Agilla の差異は動的配備のみであり，その他は差異はない．評価で用いたシミュレーション環境である TOSSIM の拡張である PowerTOSSIM[Shnayder 04] は，各ノードがシミュレーション中に消費した総消費電力を計測することができる．PowerTOSSIM において，提案ミドルウェア，Agilla を 10 分間動作させた時[*8] の消費電力を表 5.3 に示す．なお，表 5.3 は 10 ノードの平均値を示している．表 5.3 より，提案手法と Agilla は同一の消費電力となることがわかる．本結果と，5.1.1 節でのべた動的配備時のトラフィックの差異から提案手法は Agilla に比して，動的配備を考慮した場合に消費電力を減少することができると考察できる．

表 5.3: 消費電力

ミドルウェア	消費電力 (mJ)
Agilla	20084.74
提案手法	20084.74

なお，提案手法，Agilla ともにミドルウェアの Active/Sleep の制御は実施しておらず，

[*8] この時，動的配備は実施しない．

常に Active となっている．ミドルウェア稼動時の Active/Sleep の制御は TinyOS で標準的にサポートされているが，動的配備されるプログラムの処理内容と整合を取らない限り，プログラムの処理を停止してしまう可能性があり，今後の課題となる．

5.4 メモリに対する提案手法の妥当性

提案手法は *Master* が *Slave* を抱えて動的配備を実行する．そのため，Agilla に比してメモリを消費する可能性がある．提案手法，Agilla が *LCA* を構成する際の，両手法における全プログラムの合計コード容量 (*T_CodeSize_P*，*T_CodeSize_A*) は以下の式で表現できる（算出根拠は付録 C 参照）．

$$\begin{aligned} T\_CodeSize\_P &= Master\_Process + 2 * Slave\_CodeSize \\ &+ 6 + (N-1) * Slave\_CodeSize \end{aligned} \tag{5.5}$$

$$\begin{aligned} T\_CodeSize\_A &= Master\_Process + 22 * (N-1) \\ &+ (N-1) * (Slave\_CodeSize + 34) \end{aligned} \tag{5.6}$$

なお，式 5.5,5.6 における *Master_Process*, *Slave_CodeSize* は *Master* の処理部，*Slave* のコード容量を，*N* は全プログラム数を示す．図 5.13，5.14 に提案手法，Agilla によって記述されるプログラムの合計コード容量を示す．図 5.13 では，*Master_Process* を 30byte, *Slave_CodeSize* を 100byte とし，*Slave* のコード容量が *Master* より大きな場合の合計コード容量とプログラム数の関係を示す．同様に，図 5.14 では，*Master_Process* を 50byte, *Slave_CodeSize* を 30byte とし，*Slave* のコード容量が *Master* より小さい場合の合計コード容量とプログラム数の関係を示す．なお，Agilla のリーダーはプログラム数によってコード容量が可変となるため，図 5.13，図 5.14 に *Master* のコード容量（固定）と共に示す．図 5.13，図 5.14 より，*Slave* のコード容量が小さく，プログラム数が多い場合，提案手法の合計コード容量は Agilla に比べ小さいことがわかる．*Slave* のコード容量が大きく，プログラム数が少ない場合に，提案手法の合計コード容量は Agilla に比べ大きくなる．

図 5.13 において，*Slave*，他プログラム数が 2 の場合（合計三つのプログラムで *LCA* を構成する場合），*Master* のコード容量は 236byte となり，リーダーのコード容量は 74byte となる．この差（162byte) は，メモリの空き領域 (60Kbyte) の 0.27%に相当し，メモリに対して大きな制約とならない．加えて，この差は，図 5.14 のような *Slave* が小さなアプリ

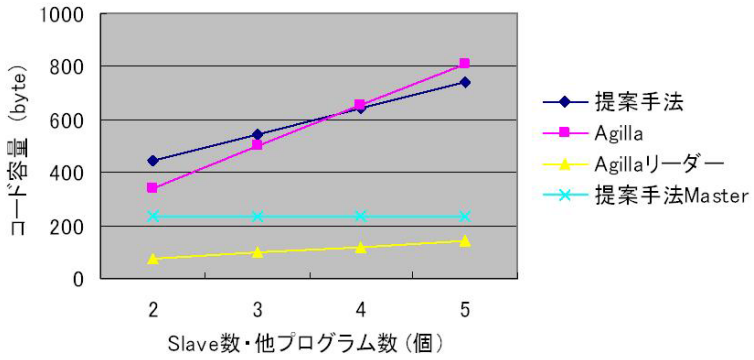

図 5.13: 全プログラムの合計コード容量 1

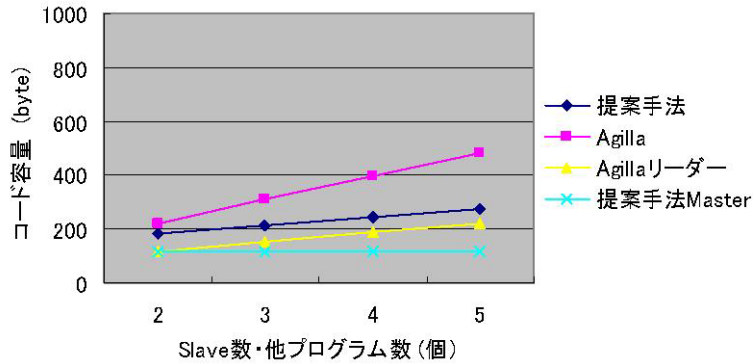

図 5.14: 全プログラムの合計コード容量 2

ケーションによって相殺される可能性がある．以上のことより，*Master* が *Slave* を所持する提案手法は妥当といえる．

5.5 提案手法の限界

提案手法には，以下の限界がある．

限界 1 ノードが動く場合

限界 2 電池が切れた場合

限界 3 ROM に対するリプログラミング

限界 4 メモリが溢れる場合

限界 5 動的配備時間中の計測

限界 6 *Slave* の実行状態の維持

限界 1: 提案ミドルウェアでは Geographic Routing を用いノード間のルーティングを実施している．そのため，ノードが動くとノードのアドレスが変更し，ノードの移動後にルーティングを実施することができない．この結果，*Master* が動的配備を実施し *Slave* を配備した後，*Slave* を配備したノードが移動してしまった場合，*Master* は *Slave* を Kill することができない限界がある．同様に *Master* が配備されたノードが移動をすることにより，当該 *Master* によって配備された *Slave-S* が計測結果を *Master* に報告できない限界が生じる．ノードが動くことに対する提案手法の限界はルーティングプロトコルに依存しており，Geographicrc Routing でないルーティングプロトコルを実装することで本限界を解消することができる．

限界 2: 提案ミドルウェアは ROM に書き込まれるが，動的配備されるプログラムはミドルウェアによって静的に確保されたメモリ領域 (RAM) に格納される．電池切れにより電源が OFF になることにより，メモリ領域に格納されていたプログラムが消失し，電池を入れ替えたとしてもプログラムを再度配備することが必要になる．この限界はミドルウェアで確保されたメモリ領域に対してプログラムを配備するミドルウェアの共通の欠点になる．配備されるプログラムを ROM へ書き込みを行う Deluge[Hui 04] や MNP[Wang 04] には本限界は存在しないが，ROM へ書き込みを行う場合には電力を消費することが知られている [Shnayder 04]．電力を消費して本限界を解消するか，電力を節約し本限界を受け入れるかのトレードオフを考慮する必要がある．

限界 3:提案ミドルウェアはミドルウェアによって確保されたメモリ領域 (RAM) にのみ，動的配備されるプログラムを格納可能であり，ROM に書き込まれているミドルウェア自体のリプログラミングを実施することができない．その結果，ミドルウェアに変更を要するような場合[*9] には，既存研究によって提案されているミドルウェア (Deluge[Hui 04] や MNP[Wang 04]) を併用する必要が生じる．

限界 4:本ミドルウェアのコード容量は 66.3Kbyte である[*10]．MOTE には，128Kbyte のプログラムメモリがあるため，本ミドルウェアは約 60Kbyte を *LCA* を構成するプログラムの領域のために利用することができる．本ミドルウェアは *LCA* を構成するプログラ

[*9] 動的配備されるプログラムに新規 ISA を追加するような場合や，ミドルウェアのパラメータの変更を実施する場合が相当する
[*10] Agilla のミドルウェアのコード容量は 57.9Kbyte である．

ムの領域が溢れてしまった場合，*LCA* を構成することができず，提案範囲外となる．

溢れる可能性については次のとおり考察できる．例えば，付録 A に示す *Master*[*11] が図 4.5 における配備パターン 1 で，ある特定ノード及びその隣接ノードに 1 体づつ存在する場合，特定ノードに存在する *Master*，*Slave-S,M* のコード容量の合計値は 203byte となる[*12]．従って，付録 A に示す規模のコード容量を持つ *Master* は，ある地域に密集しても各ノードに約 300 個存在することができる[*13]．本評価で用いた最大の空間領域を持つ 10feet（3m）四方の領域において 300 個程度の計測対象はプログラムメモリの制約上は実現できる[*14]．より大きなコード容量をもつ *Master*,*Slave* を配備する場合は，メモリが溢れる可能性があり，溢れた場合の対策が必要になる．

限界 5:提案手法は動的配備中には *Master* が移動を行い，*Master* が移動後に *Slave* を配備することから，動的配備中の計測や処理が不可能である．図 5.11 に示すとおり，*Slave* 数が 14 個の環境下 (10feet) では動的配備に約 120 秒かかり，この間の計測や処理が実施できない．動的配備が必要な環境下では，コード容量や動的配備にかかる通信 (State, Heap, Stack, Rxn および Ack) を減少し最適化することでトラフィックを削減し，動的配備時間を削減する工夫ができると考えられるが，動的配備時間を 0 にすることは不可能であり提案手法の限界となる．なお，メモリ領域が十分大きい等の動的配備の必要性がない環境下で，全てのノードにアプリケーションを実現するプログラムを配備することが可能であれば，本限界は考慮する必要がない．

限界 6:本ミドルウェアはモバイルエージェントのランタイムである Agilla を拡張しているため，プログラムは実行状態を保ったまま移動を行うことができる．提案手法では，*Master* は実行状態を保って移動はできるが，*Slave* は，*Master* の移動のたびに破棄され，移動後生成されるために，実行状態や内部変数を維持することはできない．ただし，本論文で提案する，*Slave-S* は計測処理を繰り返し，*Slave-M* は *Master* からの通知を待つプログラムであり，*Master* 移動の前後で実行状態や内部変数を維持する必要がないため，*LCA* 構成時の限界とはならない．

[*11] 92byte
[*12] 203=92+4*20+31:*Master*：92byte，*Slave-S*：20byte,*Slave-M*：31byte
[*13] 296 = 60*1000/203
[*14] データメモリ，バッテリ等の計算資源に対するプログラム間の競合解消は提案範囲外となる．これらは，TinyOS に依存する．これは既存研究（Agilla，Deluge）も同じである．

5.6 Master の冗長化について

提案手法では，*Master* が消失した場合 *LCA* の構成が不可能になる[*15]．この欠点を補うため，予めオリジナルの他にコピーを作成し，*Master* が消失した場合にコピーによって処理を継続する手法が考えられる．

著者らは，動的配備されるプログラムのロバストな振る舞いを実現するミドルウェア Ragilla(Robust Agilla)[Platon 08] を実装し，プログラムのコピー作成，ノードの残電力に応じたプログラムの移動を実機上 (MICAz) で示した．Ragilla では，プログラムのコピーを指定した範囲に，指定した数だけ作成することを可能にしており，オリジナルがなんらかの原因（例：ハードウェア障害によるノードの活動停止）によって消失した場合，コピーのうち最もプログラムＩＤが若いプログラムがオリジナルの役割を引継ぎ，所定の場所にコピーを作成する．

例えば，コピー数を 2 個，コピーの作成先を WSN 全体（本例では 4 × 4 とする）とした場合の動作例を図 5.15 および以下に示す．

1. ハードウェア障害によってオリジナルが稼動しているノードの活動が停止する．
2. ミドルウェアがハードウェアの故障を検知し，コピーの中から次のオリジナル候補を選定する．
3. ミドルウェアがオリジナル候補を指定された範囲内にコピーする．

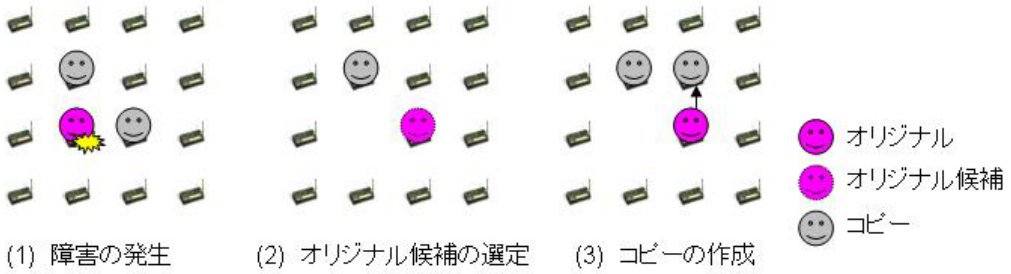

図 5.15: Ragilla の動作例

Ragilla で，実装した NesC コンポーネントを本研究で作成したミドルウェアに実装することにより，提案手法における *Master* の冗長化が可能になる．ただし，*Master* を冗長化

[*15] Agilla でもリーダが消失した場合，*LCA* の構成が不可能となり，この欠点は提案手法，既存手法ともに共通である．

することによって以下の弊害が生じる可能性がある．

1．管理情報の不整合による無駄なコピーの増加

Ragilla では，WSN 内部のノードが，自身のノードに存在するプログラム ID を定期的にブロードキャストし，各ノードが WSN 内部に存在するプログラム ID を個別に管理している．その結果，ノード間の通信が不安定な環境下では，各ノードで管理されている情報が一致しない可能性が生じる．例えば，オリジナルが消失していない場合でも，ノード間通信が不安定な場合はオリジナルが消失したと判断し，結果としてコピーが複数できてしまう可能性がある．

2．通信コストの増加

オリジナルとコピーの情報管理は，ノード間で分散管理するか，特定のノードで集中管理をするかに依らず必要となる．この結果，情報管理を実施するために通信コストが増加する．

3．*Master-Slave* 間の情報管理（整合性の確保・コストの増加）

Master（その 1) が *Slave* を配備後に消失し，あらたな *Master*（その 2）に処理を引き継いだ際に，*Master*(その 1）に紐づいていた *Slave* を *Master*(その 2) に再度紐付ける必要が生じる．この際，1 および 2 と同様に整合性を確保する必要が生じると同時に，通信コストが増加する．整合性が確保できない場合，*Master*(その 2) が動的配備を実施した場合に残存 *Slave* が残り，これを繰り返すことで残存 *Slave* 数が増加する可能性がある．

以上のことから，*Master* の冗長化を実施することは可能であるが，冗長化をすることによる恩恵と，冗長化をすることによる不都合のトレードオフを考慮する必要が生じ，この問題の解決は今後の課題となる．

5.7 提案手法の最適化の余地

提案手法および Agilla は WSN 内部でプログラムを移動させることによって動的配備を行うことができる．静的配備と動的配備の違いについては，3.3 節で述べたが，提案手法はまだ最適化の余地がある．図 5.16 に示すとおり，動的配備される *Master* が WSN 内部で動的配備する際の距離を Dis_1 とし，ベースステーションから再配備する際の距離を Dis_2 とする．

Dis_1 が Dis_2 より短い場合，WSN 内部でプログラムを動的配備したほうが，中継ノードの数を抑えることができるため，WSN 全体での消費電力を節約できる．一方，Dis_1 が Dis_2 より長い場合は，ベースステーションからプログラムを再配備したほうが，中継ノー

ドの数を抑えることができるため，WSN 全体での消費電力を節約できる．厳密には，前者における動的配備を指示するために *Slave-M* が *Master* に次の配備先を連絡する通信コストと，後者における *Master* を一度消去する通信コストの比較が必要である[*16] が，提案手法には最適化を行う余地がある．

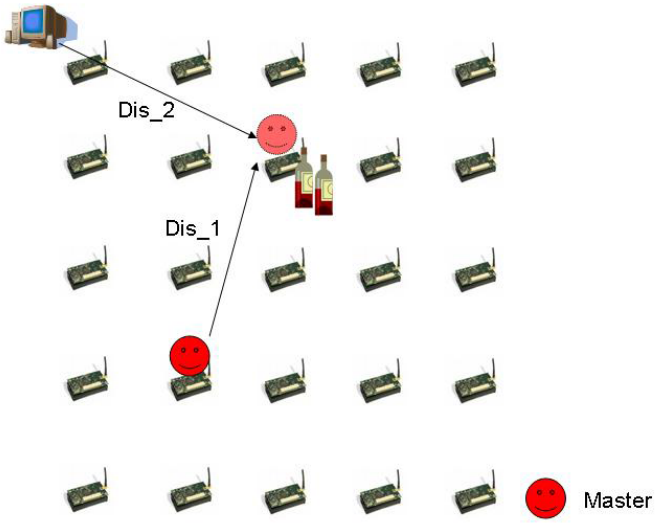

図 5.16: 動的配備時の距離と最適化

5.8 対象とするアプリケーション

本論文では，*LCA* の構成を対象とし，ノードに *Slave-S* を配備して，周辺環境の計測を実施し，必要に応じてアプリケーション独自の処理を行うことを目的にしている．そのため，本論文で事例として用いた物品の周辺の計測や，ある限られた範囲における異常の監視などのアプリケーションが対象領域となる．

本論文では，物品の現在位置をベースステーションや PDA で稼動する在庫管理アプリケーションが管理していると仮定しているが，*Slave-M* が，物品が移動した後に新移動先を取得することができれば，*Master* は在庫管理アプリケーションとの通信を行わずに移動する手法を実現できる[*17]．加えて，他プログラム，他 *Master* からの通知によって，*Master* が移動を行うことも可能であり，在庫管理アプリケーションの仮定は提案手法の制約とはならない．

[*16] *Master* が配備場所を変更するために *Slave-M* をベースステーションに派遣しているのは同等である．
[*17] *Slave-M* が物品の新現在位置を検知するための効率的なアルゴリズムが必要になる．現在の実装ではランダム移動を行うことになる．

なお，*Slave-M* は移動に適した軽量なプログラムであり，3.2 節で事例としてあげた侵入者を追跡するプログラムとしても転用できる．この場合，*Slave-S* に美術品の周辺で侵入者を検知させ，*Master* が侵入者と判定した場合に，*Slave-M* に侵入者を追跡させるモデルが実現できる．追跡を行う際に，*Slave-M* は，*Master*，*Slave-S* と離れて，侵入者を追跡し，*Master* による動的配備を必要としない．この際，*Slave-M* は *Master* との通信を実施せずに，独立したプログラムとして処理を実施する．本モデルの実現にあたっては，*Slave-M* が侵入者を逃さず追跡するアルゴリズムを開発する必要があり，*Slave-M* が全てのノードに自身を複製して，WSN 全体で侵入者を追跡するアルゴリズムが適すると考えられる．

現在の提案手法の信頼性では，動的配備の回数が多いアプリケーションへの適用は限界がある．例えば，オフィスに多くの人が存在して，人毎にアプリケーションを構成したい場合は，現状の動的配備の成功率ではアプリケーションの要求を満たさないと考えられる．複数のベースステーションを敷設して LAN と WSN で構成されるネットワークを作成することや，信頼性の高い無線の通信プロトコルと併用するアプローチが考えられる．また，5.5 節の限界 4 で述べたように，提案手法は動的配備中の計測が不可能であり，厳密な定期計測を行うアプリケーションを想定する場合，提案手法は不向きとなる．こうしたアプリケーションを実現する場合，全てのノードに定期計測を実施できるプログラムを予め配備しておくことが必要になる．

5.9 提案手法の有効範囲

提案手法は，WSN が以下の性質を満たす場合に有効となる．

1. 複数のアプリケーションを 1 つの WSN 上で実現する．
2. 全てのアプリケーションを構成するプログラムを全ノードに配備することができない．
3. 個々のアプリケーションは複数のプログラムで構成される．
4. アプリケーションが稼動場所を変更する可能性がある．

(1) は，WSN がアプリケーションの実現前に敷設されている環境を意味し，今後 WSN の適用が広がるにつれて，当該運用形態は一般的になると考えられる．
(2) の性質は，メモリ制約に依存する．2.1.4 節に示すとおり，現在の WSN を構成するノードはメモリが潤沢でないため本性質は妥当であるが，今後メモリが潤沢なハードウェアが登場することによって，提案手法の有効性が薄れる可能性がある．

(3) は，個々のアプリケーションの空間的な広がりや，冗長性に依存し，WSN のアプリケーションがスケーラブル，ロバストになるにつれて一般的になると考えられる．
(4) の性質は，対象とするアプリケーションに依存する．WSN で捕捉する物理現象が移動する場合や WSN 環境が変化する場合に本性質を満たし，前者には，火災・侵入者の包囲や，移動する管理対象物の監視があり，後者にはノードの電池切れや，ネットワーク状態の変化がある．
以上のことから，提案手法はメモリ資源が貧弱なノードにおいて，性質 (1)，(3)，(4) を満たす場合に有効であるといえる．

5.10 関連研究との比較

WSN ノードに配備されたプログラムを無線波で更新し，ノードにプログラムを動的に配備する研究に，Agilla[Fok 05b]，ActorNet[Kwon 06]，SensorWare[Boulis 03] がある．Agilla は，本研究の実装のベースとしたが，複数プログラムの動的配備を行うためにプログラム同士で通信を実施し，各プログラムが個別に移動を行う必要が生じる．また，複数プログラムでアプリケーションを構成する際のアーキテクチャが存在しない．その結果，*LCA* の継続的な構成を行う際の信頼性が低下する課題がある．4 章で示したとおり，提案手法に比べて *LCA* の継続的な構成を行う際の信頼性が劣る．
ActorNet は，actor モデル [Agha 97] を WSN に取り入れた研究例であり，プログラムが移動せずに周辺ノードのセンサデータを取得するモデルを提案している．ただし，本モデルを実現するためには，予め計測条件を所持した measure actor を各ノードに配備することが必要となり，提案手法で解決している複数プログラムの動的配備は提案範囲外となっている．SensorWare は，Agilla や ActorNet と異なり，MOTE に比較して資源が潤沢な iPAQ で稼動し，本研究とは想定環境がことなる．また，SensorWare は複数プログラムの動的配備は提案範囲外となっている．

WSN ノードに配備されたプログラムを無線波で更新し，ノードにプログラムを静的に配備する研究に Deluge[Hui 04]，Mate[Levis 02]，FireCracker[Levis 04a] がある．Deluge は，プログラムの更新を行うノードの ID を指定して，ベースステーションから，各ノードにプログラムを配備することができる．ROM に書き込みを要する容量の大きなプログラムを変更できるが，表 5.4 に示すとおり，プログラム容量が大きくなる課題がある[*18] ．

[*18] 表 5.4 では，動的配備されるプログラムとして，もっとも基本的なプログラムである Blink を示す．Blink では定期的にノードの LED を点滅させることができる．

それに加えて，ベースステーション周辺の特定のノードが，プログラムの送受信によって集中して電力を消費する課題がある．また各ノードで稼動できるプログラムの種類は一種類であり，複数のプログラムを段階的に更新・追加する用途に向かない．Maté は，Agilla のベースとなったバーチャルマシンであり，軽量なプログラムでノード上のプログラムを更新することができる．プログラムを 1byte の ISA で記述することで，従来は Kbyte 単位であったプログラムのコード容量を，十数 byte〜数十 byte まで軽量化することに成功した．ただし，Deluge と同様に各ノードで稼動させられるプログラムは一種類であり，複数のプログラムを段階的に更新できない課題がある．加えて，特定ノードのプログラムを更新できない課題がある．FireCracker[Levis 04a] は，ある特定のノードにプログラムを送り込み，周辺ノードのプログラムを特定ノードから更新する手法を提案しているが，提案手法のように異なる機能のプログラムを周辺ノードに配備することができない．

表 5.4: 動的配備されるプログラムのコード容量 (Deluge，提案手法，Agilla)

ミドルウェア	プログラムのコード容量 (Blink)	ミドルウェアのコード容量
Deluge	20.4Kbyte	22.9Kbyte
提案手法	0.023Kbyte	66.3Kbyte
Agilla	0.023Kbyte	57.9Kbyte

WSN 以外のネットワークでは，Mobile Space[Satoh 00]，石川ら [石川 05] が階層型のモバイルエージェントを提案し，提案手法と同様に複数のモバイルエージェント（プログラム）を一緒に移動させることを実現している．これらの研究は，モバイルエージェント間での協調を目的とし，モバイルエージェント間に階層関係を持たせ，一時的・永続的な強調のために複数のモバイルエージェントが一緒に移動をすることを提案している．それに対して，提案手法は，動的配備の信頼性を向上するために，*Master* と *Slave* を抱えて移動することを提案しており，目的が異なる．また，Mobile Space や石川らは，計算資源が潤沢な環境と，帯域が広く安定したネットワークを想定しているのに対し，提案手法は計算資源が極小であり，通信が不安定な WSN を想定している．本研究とは，想定環境と，アプリケーションの継続的な実行を目的とした点が異なる．

5.11 まとめ

本章では，提案手法の有効な範囲や限界について論じた．提案する動的配備手法の成功率の高さは，既存手法に比べて動的配備時のトラフィックを削減することにより達成されていることを示した．また，トラフィックが増加する原因を，既存手法におけるコード容量の増加分と実行状態の維持分に分離し，既存ミドルウェアをライブラリ化することによってコード容量を削減できたと仮定しても，提案手法の有効性は変わらないことを示した．提案手法は電池切れやノードの移動に対応できない等の 6 つの限界が存在するが，提案手法は WSN が以下の性質を満たす際に有効となり，今後の WSN の運用形態に適していると考えられる．

1. 複数のアプリケーションを 1 つの WSN 上で実現する．
2. 全てのアプリケーションを構成するプログラムを全ノードに配備することができない．
3. 個々のアプリケーションは複数のプログラムで構成される．
4. アプリケーションが稼動場所を変更する可能性がある．

第 6 章
結言

本章では，本研究を通しての成果をまとめ，将来研究の展望について述べ，本論文の結びとする．

6.1 結論

本研究では，敷設された WSN を利用して，複数のアプリケーションを追加・変更・削除しながら運用を行う将来的な WSN の利用形態を想定した．本研究では，想定環境において，複数プログラムで構成されるアプリケーション（*LCA*）の構成や，継続的な構成の信頼性を向上させることを目的とした．

1 章では，想定環境においてリプログラミングを用いることが適することを示した．WSN のアプリケーション開発で用いられる典型的なアーキテクチャとしてベースステーション集中型とネットワーク内処理型の 2 つが存在し，ネットワーク内処理型が一般的に用いられることを述べた．また，ネットワーク内処理型のアーキテクチャを実現する場合には，開発効率を向上させるためにミドルウェアを適用することが一般的であることを示した．WSN 内部でアプリケーション固有の処理を実現する *LCA* を構成する場合，WSN における代表的なミドルウェアアプローチ（データベース型，イベントベース型）の適用は課題を抱え，リプログラミングを用いることが適当であることを示した．

2 章では，関連研究について述べた．一般的な WSN の研究分野（プログラミング言語・モデル，ミドルウェア，オペレーティングシステム，ハードウェア，ネットワーク）について動向を述べた．関連研究としてリプログラミングに言及し，リプログラミングの概要と，リプログラミングの種別（静的配備，動的配備等）について詳細を述べた．

3 章では，想定環境における要求の分析と課題の特定を行った．想定環境で，*LCA* で構成する場合，*LCA* を構成するプログラムには 4 つの要求（計測の実行，計測結果の判定，情報の収集・交換，プログラムの動的配備）が存在すると分析された．既存手法を用いて，

これらの要求を満たす場合，2 つの課題が存在することを示した．一つ目の課題として，*LCA* を構成する際のアーキテクチャが存在しないことによって，*LCA* の（継続的な）構成の信頼性を低下させる課題が特定された．二つ目の課題として，既存手法はプログラムの動的配備時に通信に高く依存したアプローチを取り，その結果，動的配備に失敗し，動的配備後に *LCA* を継続して構成できない課題が特定された．

4 章では，提案手法の説明と評価を行った．提案手法は 2 つあり，1 つは *LCA* を構成する際のアーキテクチャの提案，もう 1 つは *LCA* を構成するプログラムの動的配備手法の提案である．前者については，*Master*，*Slave-S*，*Slave-M* で構成されるアーキテクチャを提案し，*LCA* を継続して構成する信頼性を既存手法に比べ，約 15%〜50%向上できた．後者については，*Master* が他 *Slave* を生成して配備する手法を提案し，動的配備の成功率を既存手法に比べ，約 20%〜40%向上できた．また，提案手法は *Master* が *Slave* を抱えて，配備先で *Slave* を生成するため，*Master* の配備先のノードにおけるトラフィックの増加や，動的配備にかかる時間が増加する可能性が考えられるため，評価を実施した．評価の結果，提案手法の WSN 全体でのトラフィックは既存手法の半分となることが示された．一方で，提案手法の動的配備時間は *Slave* の数が増加した場合に，既存手法の数倍となることが示された．

5 章では，提案手法の有効な範囲と限界について言及した．提案手法は，既存手法に比して動的配備時のトラフィックを半減することによって *LCA* の継続的な構成を行う際の信頼性を向上していることが示された．動的配備時間の差異は，プログラム (*Slave*) 数を増加することにより顕著になり，*Slave* 数が 15 体になると，提案手法の動的配備時間は 120 秒となった．*Slave* 数が多い LCA の構成を行い，動的配備時間中の計測が無視できないような計測や処理を実施する場合，提案手法は不向きとなる．提案手法には，(1) ノードが動く場合，(2) 電池が切れた場合，(3)ROM に対するリプログラミング，(4) メモリが溢れる場合，(5) 動的配備時間中の計測，(6)*Slave* の実行状態の維持，の 6 つの限界が存在することを示した．また，提案手法の有効な範囲として，WSN が以下の性質を満たす際に有効であることが示された．

1. 複数のアプリケーションを 1 つの WSN 上で実現する．
2. 全てのアプリケーションを構成するプログラムを全ノードに配備することができない．
3. 個々のアプリケーションは複数のプログラムで構成される．
4. アプリケーションが稼動場所を変更する可能性がある．

6.2 課題と今後の展望

提案手法をより実用的にするためには，以下に示す 3 つの課題が存在する．以下の課題の説明と解決のアプローチを示し，本論文を終える．

1. 配備パターンの動的な取得
2. プログラムの位置管理機構の導入
3. 複雑なプログラム間連携

1．配備パターンの動的な取得
本研究では，動的配備時に *Master* が *Slave* の配備場所を決定するために配備パターンを導入した．配備パターンは静的にミドルウェアの中で定義され，*Master* の動的配備時に *Master* を受信したノードによって配備パターンに従って *Slave* が配備される．WSN では，ノードの残電力やノード間のリンクが変化することが知られている．そのため，配備パターンを静的に定義することによって，残電力が少ないノードに *Slave* を配備する可能性や，通信成功確率が少ないノードに *Slave* を配備する可能性があり，この結果，*LCA* の構成を困難にする課題が生じる．提案手法をより現実の WSN に適する手法にするために，*Master* を受信したノードがノードの状況や周辺ノードの状況に応じて配備パターンを決定し，*Slave* を適したノードに配備することが必要と考えられる．これを実現するために，各ノードが予め周辺ノードの残電力や，周辺ノードとのリンク状況を把握する必要が生じる．著者らは，[Platon 08] において，実機（MICAz）を用いて，動的配備されたプログラムがノードの残電力を考慮しながら移動することや，ハードウェア障害時にグループを構成するプログラムが自律的にグループを再構成することを示した．配備パターンの状況に合わせた変更は，これらの研究資産を用いることで開発可能である．
2．プログラムの位置管理機構の導入
シミュレーション環境では見られなかったが，*Slave* は *Master* によって生成・Kill されるために Kill に失敗した場合に *Slave* が残存する可能性がある．提案手法はノードの移動に対応していないが，仮にノードが移動することを考慮すると，*Slave* の残存が課題となると考えられる．この場合，WSN 内部に存在する不要なプログラムを排除することが必要になり，そのためのプログラムの位置管理機構が必要となる．ベースステーションで全てのプログラムの位置の管理を実施することは，通信コストが高くなり，非効率であるため，WSN 内部でプログラムの位置管理を実施する必要がある．Bhattacharya ら [Bhattacharya 08],

著者ら [Suenaga 08] の研究を組み合わせることで，残存 *Slave* の把握が可能になり，提案手法の有効性が高まると考えられる．

3. 複雑なプログラム間連携

提案手法は，*Master*，*Slave* 間での連携を可能にするが，*Slave* 間および *Master* 間での連携は対象としていない．本研究で提案した *Slave-S*，*Slave-M* は基本的な *Slave* であり，今後他の機能を所持した *Slave* を作成することで，より複雑なアプリケーション構成が可能であり，*Slave* 間連携が必要になる可能性がある．同様に，異なる *LCA* 間で連携を行う場合には *Master* 間での連携が必要になる．これらを実現する手法として，(2) の位置管理機構や，*Master*，*Slave* が提供する機能を検索できる機構の導入がある．また，提案ミドルウェアは *Master*，*Slave* の一階層にしか対応していないが，多階層に対応し，必要に応じて *Master*・*Slave* の一部を切り出して処理を実施する拡張を行うことで，複雑な連携を実現できると考えられる．

謝辞

本研究は，多くの方々のご指導とご助力のもとに遂行されました．以下に特にお世話になった方々のお名前を記して感謝の意を表します．

まず何より，指導教員である東京大学，国立情報学研究所の本位田真一教授，総合研究大学院大学，国立情報学研究所の吉岡信和准教授に感謝致します．本位田先生には，ご多忙にも関わらず，私のために多くのお時間を割いて頂き，研究指導をして頂きました．また研究指導に留まらず，社会人および人間としてご指導頂き，多くのことを学ぶことが出来ました，大変感謝しております．吉岡先生には，ご多忙中の中，研究の相談に乗っていただき，論文の訂正や研究の方向性など，きめ細やかな指導をして頂きました．大変感謝しております．こうして本論文を完成することができたのも，ひとえに，本位田先生，吉岡先生のご指導によるものです．ありがとうございました．

本博士論文の審査委員をご快諾頂きました，電気通信大学の大須賀昭彦教授，国立情報学研究所の計宇生准教授，山田茂樹教授，米田友洋教授に深く感謝いたします．大須賀先生にはエージェント技術の観点から，的確かつ有意義なコメントを頂きました．計先生にはセンサとネットワークに関する知見を元に多くのコメントを頂きました．山田先生にはネットワークの見識から鋭いご指摘を頂きました．米田先生には信頼性に関する観点から，参考になる多くのコメントを頂くことができました．大須賀先生，計先生，山田先生，米田先生のご指摘は，本論文を完成させるために大いに参考になり，研究の質を高めることができました．審査委員の皆様，ありがとうございました．

本位田研究室のメンバには，公私にわたり大変お世話になりました．国立情報学研究所の石川冬樹助教，早稲田大学の鄭顕志助教には，常に的確な助言を頂くことができ，研究を進めることが出来ました．ありがとうございました.（株）三菱総合研究所の松崎和賢博士,（株）シリウステクノロジーのエリック・プラトン博士の両氏には，共同研究やディスカッションを通して多くのコメントや知見を頂くことが出来ました．ありがとうございました．センサネットワークのプロジェクト (XAC) メンバーには，多くの議論や共同作業などで大変お世話になりました．XAC メンバーの皆様，ありがとうございました．個々に

お名前を記すことが出来なかった本位田研究室の皆様にも大変お世話になりました．ありがとうございました．
また，私を会社から派遣してくださいました日本ユニシス (株）の保科剛氏，羽田昭弘氏，新堀聡氏に深く感謝いたします．
　冒頭にも述べましたとおり，本研究は多くの方々のご助力によってなすことができました．ここに記述させて頂いた方々はその一部であり，お名前を記すことが出来なかった多くの皆様に感謝いたします．最後に，研究生活を支えてくれた家族に感謝致します．

参考文献

[Abdelzaher 04] Abdelzaher, T., Blum, B., Cao, Q., Chen, Y., Evans, D., George, J., George, S., Gu, L., He, T., Krishnamurthy, S., Luo, L., Son, S., Stankovic, J., Stoleru, R., and Wood, A.: EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks, in *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pp. 582–589, Washington, DC, USA (2004), IEEE Computer Society

[Abrach 03] Abrach, H., Bhatti, S., Carlson, J., Dai, H., Rose, J., Sheth, A., Shucker, B., Deng, J., and Han, R.: MANTIS: system support for multimodAl NeTworks of in-situ sensors, in *WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pp. 50–59, New York, NY, USA (2003), ACM

[ACT] YoungMin Kwon's Homepage`http://osl.cs.uiuc.edu/~ykwon4/#software`

[Agha 97] Agha, G., Mason, I. A., Smith, S. F., and Talcott, C. L.: A Foundation for Actor Computation, *J. Funct. Program.*, Vol. 7, No. 1, pp. 1–72 (1997)

[Agi] Pre-Packaged Versions of Agilla`http://mobilab.wustl.edu/projects/agilla/download/pre3/index.html`

[Al-Karaki 04] Al-Karaki, J. and Kamal, A.: Routing techniques in wireless sensor networks: a survey, *Wireless Communications, IEEE*, Vol. 11, No. 6, pp. 6–28 (2004)

[Baumann 02] Baumann, J., Rothermel, H. K., Strasser, M., and Theilmann, W.: MOLE: a mobile agent system, *Softw. Pract. Exper.*, Vol. 32, No. 6, pp. 575–603 (2002)

[Bhattacharya 08] 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)

[Biagioni 04] Biagioni, E. and Chen, S. H.: A Reliability Layer for Ad-Hoc Wireless Sensor Network Routing, in *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9*, p. 90300, Washington, DC, USA (2004), IEEE Computer Society

[Borcea 04] Borcea, C., Intanagonwiwat, C., Kang, P., Kremer, U., and Iftode, L.: Spatial Programming Using Smart Messages: Design and Implementation, in *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pp. 690–699, Washington, DC, USA (2004), IEEE Computer Society

[Boulis 03] Boulis, A., Han, C.-C., and Srivastava, M. B.: Design and implementation of a framework for efficient and programmable sensor networks, in *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pp. 187–200, New York, NY, USA (2003), ACM Press

[Braginsky 02] Braginsky, D. and Estrin, D.: Rumor routing algorthim for sensor networks, in *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pp. 22–31, New York, NY, USA (2002), ACM

[CEN] CENS:`http://research.cens.ucla.edu/`

[Chu 06] Chu, D., Lin, K., Linares, A., Nguyen, G., and Hellerstein, J. M.: Sdlib: a sensor network data and communications library for rapid and robust application development, in *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pp. 432–440, New York, NY, USA (2006), ACM

[C.Schurgers 01] C.Schurgers, and Srivastava, M.: Energy efficient routing in wireless sensor networks, in *MILCOM '2001: Military Communications Conference*, pp. 357–361, New York, NY, USA (2001), ACM

[Culler 04] Culler, D. E., Estrin, D., and Srivastava, M. B.: Guest Editors' Introduction: Overview of Sensor Networks, *IEEE Computer*, Vol. 37, No. 8, pp. 41–49 (2004)

[Curino 05] Curino, C., Giani, M., Giorgetta, M., Giusti, A., Murphy, A. L., and Picco, G. P.: TinyLIME: Bridging Mobile and Sensor Networks through Middleware, in *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pp. 61–72, Washington, DC, USA (2005), IEEE Computer Society

[Deb 03] Deb, B., Bhatnagar, S., and Nath, B.: ReInForM: Reliable Information Forwarding Using Multiple Paths in Sensor Networks, in *LCN '03: Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks*, p. 406, Washington, DC, USA (2003), IEEE Computer Society

[del] Deluge 2.0 - TinyOS Network Programming`http://www.cs.berkeley.edu/~jwhui/deluge/deluge-manual.pdf`

[Dunkels 04a] Dunkels, A., Alonso, J., and Voigt, T.: Making TCP/IP Viable for Wireless Sensor Networks (2004)

[Dunkels 04b] Dunkels, A., Gronvall, B., and Voigt, T.: Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors, in *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, pp. 455–462, Washington, DC, USA (2004), IEEE Computer Society

[Dunkels 06] Dunkels, A., Finne, N., Eriksson, J., and Voigt, T.: Run-time dynamic linking for reprogramming wireless sensor networks, in *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pp. 15–28, New York, NY, USA (2006), ACM

[Dunkels 07] Dunkels, A., Österlind, F., and He, Z.: An adaptive communication architecture for wireless sensor networks, in *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pp. 335–349, New York, NY, USA (2007), ACM

[Estrin 99] Estrin, D., Govindan, R., Heidemann, J., and Kumar, S.: Next century challenges: scalable coordination in sensor networks, in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 263–270, New York, NY, USA (1999), ACM

[EYE] eyes:`http://www.eyes.eu.org/sensnet.htm`

[Fang 03] Fang, Q., Zhao, F., and Guibas, L. J.: Lightweight sensing and communication protocols for target enumeration and aggregation, in *MobiHoc*, pp. 165–176 (2003)

[Fok 05a] Fok, C.-L., Roman, G.-C., and Lu, C.: Mobile agent middleware for sensor networks: an application case study, in *IPSN*, pp. 382–387 (2005)

[Fok 05b] Fok, C.-L., Roman, G.-C., and Lu, C.: Rapid Development and Flexible Deployment of Adaptive Wireless Sensor Network Applications, in *ICDCS '05: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pp. 653–662, Washington, DC, USA (2005), IEEE Computer Society

[Gay 03] Gay, D., Levis, P., Behren, von J. R., Welsh, M., Brewer, E. A., and Culler, D. E.: The nesC language: A holistic approach to networked embedded systems, in *PLDI*, pp. 1–11 (2003)

[Gelernter 85] Gelernter, D.: Generative communication in Linda, *ACM Trans. Program. Lang. Syst.*, Vol. 7, No. 1, pp. 80–112 (1985)

[Glass 99] Glass, G.: ObjectSpace Voyager Core package technical overview, pp. 611–627 (1999)

[Greenstein 04] Greenstein, B., Kohler, E., and Estrin, D.: A sensor network application construction kit (SNACK), in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 69–80, New York, NY, USA (2004), ACM

[Gummadi 05] Gummadi, R., Gnawali, O., and Govindan, R.: Macro-programming Wireless Sensor Networks Using *airos*, in *DCOSS*, pp. 126–140 (2005)

[Han 05] Han, C.-C., Kumar, R., Shea, R., Kohler, E., and Srivastava, M.: A dynamic operating system for sensor nodes, in *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pp. 163–176, New York, NY, USA (2005), ACM

[Hassanein 06] Hassanein, H. and Luo, J.: Reliable Energy Aware Routing In Wireless Sensor Networks, in *DSSNS '06: Proceedings of the Second IEEE Workshop on Dependability and Security in Sensor Networks and Systems*, pp. 54–64, Washington, DC, USA (2006), IEEE Computer Society

[Heinzelman 00] Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H.: Energy-Efficient Communication Protocol for Wireless Microsensor Networks, in *HICSS* (2000)

[Henricksen 06] Henricksen, K. and Robinson, R.: A survey of middleware for sensor networks: state-of-the-art and future directions, in *MidSens '06: Proceedings of the international workshop on Middleware for sensor networks*, pp. 60–65, New York, NY, USA (2006), ACM

[Hill 00] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K.: System architecture directions for networked sensors, *SIGPLAN Not.*, Vol. 35, No. 11, pp. 93–104 (2000)

[Hill 02] Hill, J. and Culler, D.: A wireless embedded sensor architecture for system-level optimization., *UC Berkeley Technical Report, 2002* (2002)

[Hui 04] Hui, J. W. and Culler, D.: The dynamic behavior of a data dissemination protocol for network programming at scale, in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 81–94, New York, NY, USA (2004), ACM Press

[Hui 08] Hui, J. W. and Culler, D. E.: IP is dead, long live IP for wireless sensor networks, in *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 15–28, New York, NY, USA (2008), ACM

[Intanagonwiwat 03] Intanagonwiwat, 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 (2003)

[IRI] IRIS:`http://www.xbow.com/Products/productdetails.aspx?sid=264`

[Jeong 04] Jeong, J. and Culler, D.: Incremental Network Programming for Wireless Sensors, in *SECON* (2004)

[Kahn 99] Kahn, J. M., Katz, R. H., and Pister, K. S. J.: Next Century Challenges: Mobile Networking for "Smart Dust", in *MOBICOM*, pp. 271–278 (1999)

[Karlof 04] Karlof, C., Sastry, N., and Wagner, D.: TinySec: a link layer security architecture for wireless sensor networks, in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 162–175, New York, NY, USA (2004), ACM

[Karp 00a] Karp, B. and Kung, H. T.: GPSR: greedy perimeter stateless routing for wireless networks, in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 243–254, New York, NY, USA (2000), ACM

[Karp 00b] Karp, B. and Kung, H. T.: GPSR: greedy perimeter stateless routing for wireless networks, in *Mobile Computing and Networking*, pp. 243–254 (2000)

[Kawamura 99] Kawamura, T., Hasegawa, T., Ohsuga, A., and Honiden, S.: Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems, in *APSEC '99: Proceedings of the Sixth Asia Pacific Software Engineering Conference*, p. 260, Washington, DC, USA (1999), IEEE Computer Society

[Ke 08] Ke, W. and Little, T.: Dynamic Routing Selection for Wireless Sensor Networks, *Network Computing and Applications, 2008. NCA '08. Seventh IEEE International Symposium on*, pp. 136–143 (2008)

[Krishnamachari 02] Krishnamachari, B., Estrin, D., and Wicker, S. B.: The Impact of Data Aggregation in Wireless Sensor Networks, in *ICDCS Workshops*, pp. 575–578 (2002)

[Kuhn 03] Kuhn, F., Wattenhofer, R., and Zollinger, A.: Worst-Case optimal and average-case efficient geometric ad-hoc routing, in *MobiHoc*, pp. 267–278 (2003)

[Kulik 02] Kulik, J., Heinzelman, W. R., and Balakrishnan, H.: Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks, *Wireless Networks*, Vol. 8, No. 2-3, pp. 169–185 (2002)

[Kumar 03] Kumar, R., Wolenetz, M., Agarwalla, B., Shin, J., Hutto, P., Paul, A., and Ramachandran, U.: DFuse: a framework for distributed data fusion, in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 114–125, New York, NY, USA (2003), ACM

[Kumar 05] Kumar, A. V. U. P., V, A. M. R., and Janakiram, D.: Distributed collaboration for event detection in wireless sensor networks, in *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pp. 1–8, New York, NY, USA (2005), ACM

[Kuorilehto 05] Kuorilehto, M., Hannikainen, M., and Hamalainen, T. D.: A survey of application distribution in wireless sensor networks, *EURASIP J. Wirel. Commun. Netw.*, Vol. 5, No. 5, pp. 774–788 (2005)

[Kwon 06] Kwon, Y., Sundresh, S., Mechitov, K., and Agha, G.: ActorNet: an actor platform for wireless sensor networks, in *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pp. 1297–1300, New York, NY, USA (2006), ACM Press

[Lange 99] Lange, D. B. and Oshima, M.: Seven good reasons for mobile agents, *Commun. ACM*, Vol. 42, No. 3, pp. 88–89 (1999)

[Levis 02] Levis, P. and Culler, D.: Maté: a tiny virtual machine for sensor networks, in *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pp. 85–95, New York, NY, USA (2002), ACM Press

[Levis 03] Levis, P., Lee, N., Welsh, M., and Culler, D.: TOSSIM: accurate and scalable simulation of entire tinyOS applications, in *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 126–137, New York, NY, USA (2003), ACM Press

[Levis 04a] Levis, P. and Culler, D.: The firecracker protocol, in *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop*, p. 3, New York, NY, USA (2004), ACM

[Levis 04b] Levis, P., Patel, N., Culler, D., and Shenker, S.: Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks, in *NSDI'04: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation*, pp. 2–2, Berkeley, CA, USA (2004), USENIX Association

[Li 03] Li, S., Son, S. H., and Stankovic, J. A.: Event Detection Services Using Data Service Middleware in Distributed Sensor Networks, in *IPSN*, pp. 502–517 (2003)

[Li 06] Li, F. and Wu, J.: A probabilistic voting-based filtering scheme in wireless sensor networks, in *IWCMC '06: Proceedings of the 2006 international conference on Wireless communications and mobile computing*, pp. 27–32, New York, NY, USA (2006), ACM

[Lindsey 02] Lindsey, S. and Raghavendra, C. S.: PEGASIS: Power-efficient gathering in sensor information systems, in *Aerospace Conference Proceedings*, pp. 1125–1130 (2002)

[Liu 03] Liu, T. and Martonosi, M.: Impala: a middleware system for managing autonomic, parallel sensor systems, in *PPOPP*, pp. 107–118 (2003)

[Liu 08] Liu, A., Oh, Y.-H., and Ning, P.: Secure and DoS-Resistant Code Dissemination in Wireless Sensor Networks Using Seluge, in *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, pp. 561–562, Washington, DC, USA (2008), IEEE Computer Society

[Madden 02] Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W.: TAG: a Tiny AGgregation service for ad-hoc sensor networks, *SIGOPS Oper. Syst. Rev.*, Vol. 36, No. SI, pp. 131–146 (2002)

[Madden 05] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W.: TinyDB: an acquisitional query processing system for sensor networks, *ACM Trans. Database Syst.*, Vol. 30, No. 1, pp. 122–173 (2005)

[Mahjoub 07] Mahjoub, D. and El-Rewini, H.: Adaptive Constraint-Based Multi-Objective Routing for Wireless Sensor Networks, *Pervasive Services, IEEE International Conference on*, pp. 72–75 (2007)

[Manjeshwar 02] Manjeshwar, A. and Agrawal, D. P.: APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks, in *IPDPS* (2002)

[Mao 07] Mao, G., Fidan, B., and Anderson, B. D.: Wireless sensor network localization techniques, in *Computer Networks, Volume 51, Issue 10, 11* (2007)

[Marrón 05a] Marrón, P. J., Lachenmann, A., Minder, D., Gauger, M., Saukh, O., and Rothermel, K.: Management and configuration issues for sensor networks, *Int. Journal of Network Management*, Vol. 15, No. 4, pp. 235–253 (2005)

[Marrón 05b] Marrón, P. J., Lachenmann, A., Minder, D., Hähner, J., Sauter, R., and Rothermel, K.: TinyCubus: A Flexible and Adaptive Framework for Sensor Networks, in *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN 2005)*, pp. 278–289 (2005)

[MIC] Crossbow Mote:`http://www.xbow.jp/zigbee-smartdust.html`

[Milojičić 98] Milojičić, D. S., LaForge, W., and Chauhan, D.: Mobile objects and agents (MOA), in *COOTS'98: Proceedings of the 4th conference on USENIX Conference on Object-Oriented Technologies and Systems*, pp. 13–13, Berkeley, CA, USA (1998), USENIX Association

[Naik 05] Naik, V., Arora, A., Sinha, P., and Zhang, H.: Sprinkler: A Reliable and Energy Efficient Data Dissemination Service for Wireless Embedded Devices, in *RTSS*, pp. 277–286 (2005)

[NES] NEST Project:`http://nest.cs.berkeley.edu/nest-index.html`

[Newton 07] Newton, R., Morrisett, G., and Welsh, M.: The regiment macroprogramming system, in *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pp. 489–498, New York, NY, USA (2007), ACM

[Ni 99] Ni, S.-Y., Tseng, Y.-C., Chen, Y.-S., and Sheu, J.-P.: The broadcast storm problem in a mobile ad hoc network, in *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 151–162, New York, NY, USA (1999), ACM

[Platon 08] Platon, E., Suenaga, S., Yoshioka, N., and Honiden, S.: Transparent Applications Lofetime Management in Wireless Sensor Networks (2008)

[Reijers 03] Reijers, N. and Langendoen, K.: Efficient code distribution in wireless sensor networks, in *Wireless Sensor Networks and Applications*, pp. 60–67 (2003)

[Rsy] Efficient Algorithms for Sorting and Synchronization`http://www.samba.org/~tridge/phd_thesis.pdf`

[Rubio 07] Rubio, B., Diaz, M., and Troya, J. M.: Programming Approaches and Challenges for Wireless Sensor Networks, in *ICSNC '07: Proceedings of the Second International Conference on Systems and Networks Communications*, p. 36, Washington, DC, USA (2007), IEEE Computer Society

[Satoh 00] Satoh, I.: MobileSpaces: A Framework for Building Adaptive Distributed Applications Using a Hierarchical Mobile Agent System, in *ICDCS '00: Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, pp. 161–168, Washington, DC, USA (2000), IEEE Computer Society

[SEN] SENTILLA:`http://www.sentilla.com/index.html`

[SHI] Shimmer:`http://shimmer-research.com/wordpress/?page_id=20`

[Shnayder 04] Shnayder, V., Hempstead, M., Chen, rong B., Allen, G. W., and Welsh, M.: Simulating the power consumption of large-scale sensor network applications, in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 188–200, New York, NY, USA (2004), ACM

[Sma] SmartDusthttp://mobilab.wustl.edu/projects/agilla/download/pre3/index.html

[Souto 04] Souto, E., Germano Guimar a., Vasconcelos, G., Vieira, M., Rosa, N., and Ferraz, C.: A message-oriented middleware for sensor networks, in *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pp. 127–134, New York, NY, USA (2004), ACM

[SPT] SunSpot:http://www.sunspotworld.com

[SQW] Squark:https://squawk.dev.java.net

[Srisathapornphat 00] Srisathapornphat, C., Jaikaeo, C., and Shen, C.-C.: Sensor Information Networking Architecture, in *ICPP Workshops*, pp. 23–30 (2000)

[Suenaga 08] Suenaga, S. and Honiden, S.: Name-based location service for mobile agents in wireless sensor networks, in *MOBILWARE*, p. 3 (2008)

[TEL] TelosB:http://www.xbow.com/Products/productdetails.aspx?sid=252

[Tian 03] Tian, D. and Georganas, N.: Energy efficient routing with guaranteed delivery in wireless sensor networks, *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, Vol. 3, pp. 1923–1929 vol.3 (2003)

[Tian 05] Tian, H. and Shen, H.: An Optimal Coverage Scheme for Wireless Sensor Network, in *ICN (1)*, pp. 722–730 (2005)

[tos] TOSSIM: A Simulator for TinyOS Networkshttp://www.cs.berkeley.edu/~pal/pubs/nido.pdf

[Wang 04] Wang, L.: MNP: multihop network reprogramming service for sensor networks, in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 285–286, New York, NY, USA (2004), ACM Press

[Wang 06] Wang, Q., Zhu, Y., and Cheng, L.: Reprogramming wireless sensor networks: challenges and approaches, *IEEE Network*, Vol. 20, No. 3, pp. 48–55 (2006)

[WEB] Berkeley WEBS:http://local.cs.berkeley.edu/webs/

[Welsh 04] Welsh, M. and Mainland, G.: Programming Sensor Networks Using Abstract Regions., in *NSDI*, pp. 29–42 (2004)

[White 99] White, J. E.: Telescript technology: mobile agents, pp. 460–493 (1999)

[Whitehouse 04] Whitehouse, K., Sharp, C., Brewer, E., and Culler, D.: Hood: a neighborhood abstraction for sensor networks, in *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pp. 99–110, New York, NY, USA (2004), ACM Press

[Whitehouse 06] Whitehouse, K., Zhao, F., and Liu, J.: Semantic Streams: A Framework for Composable Semantic Interpretation of Sensor Data, in *EWSN*, pp. 5–20 (2006)

[Xbo] Crossbow Technology:`http://www.xbow.com/`

[XNP] Mote In-Network Programming User Reference`http://www.tinyos.net/tinyos-1.x/doc/Xnp.pdf`

[Xu 01] Xu, Y., Heidemann, J., and Estrin, D.: Geography-informed energy conservation for Ad Hoc routing, in *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 70–84, New York, NY, USA (2001), ACM

[Yao 02] Yao, Y. and Gehrke, J.: The Cougar Approach to In-Network Query Processing in Sensor Networks (2002)

[Yao 03] Yao, Y. and Gehrke, J.: Query Processing in Sensor Networks, in *CIDR* (2003)

[Ye 07] Ye, F., Yang, H., and Liu, Z.: Catching "Moles" in Sensor Networks, in *ICDCS '07: Proceedings of the 27th International Conference on Distributed Computing Systems*, p. 69, Washington, DC, USA (2007), IEEE Computer Society

[Yoneki 05] Yoneki, E. and Bacon, J.: Unified Semantics for Event Correlation over Time and Space in Hybrid Network Environments, in *OTM Conferences (1)*, pp. 366–384 (2005)

[Yu 01] Yu, Y., Govindan, R., and Estrin, D.: Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks, in *UCLA Computer Science Department Technical Report* (2001)

[Yu 05] Yu, Z. and Guan, Y.: A dynamic en-route scheme for filtering false data injection in wireless sensor networks, in *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pp. 294–295, New York, NY, USA (2005), ACM

[石川 05] 石川 冬樹, 吉岡 信和, 田原 康之, 本位田 真一：階層構造制御に注目したモバイルエージェントフレームワークとそのマルチメディア応用, 「ソフトウェアエージェントとその応用」特集号, pp. 1402–1417, 電子情報通信学会 (2005)

研究業績

主著

査読付き学会誌論文

1. 末永 俊一郎，吉岡 信和，本位田 真一：**無線センサネットワークにおける複数プログラムの動的配備**, 情報処理学会論文誌，Vol. 50, No. 1,pp.14-30, 2009.

査読付き国際会議等発表

1. Shunichiro Suenaga, Nobukazu Yoshioka and Shinichi Honiden: **Generative Dynamic Deployment of Multiple Components in Wireless Sensor Networks** , In *Proceedings of the 6th International Conference on Wireless On-demand Network Systems and Services (WONS2009)*, pp.197-254, 2009.

2. Shunichiro Suenaga and Shinichi Honiden: **Name-based Location Service for Mobile Agents in Wireless Sensor Networks**, In *Proceedings of the 1st International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications (MOBILWARE 2008)*, CDROM, 6pages, 2008.

3. Shunichiro Suenaga and Shinichi Honiden: **Constructing Locally Centralized Applications by Mobile Agents in Wireless Sensor Networks**, In *Proceedings of the 2nd International Workshop on Agent Technology for Sensor Networks (ATSN 2008) located at the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pp.79-82, 2008.

4. Shunichiro Suenaga and Shinichi Honiden: **Name-based Location Service for Mobile Agents in Wireless Sensor Networks**, In *Proceedings of the 1st International Workshop on Agent Technology for Sensor Networks (ATSN 2007) located*

at the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), pp.45-46, 2007.

査読付き国内学会発表

1. 末永 俊一郎，本位田 真一: **無線センサネットワークにおけるモバイルエージェント間のメッセージ配信**, 合同エージェントワークショップ&シンポジウム 2007 (JAWS2007) 論文集，2007.

2. 末永 俊一郎，本位田 真一: **無線センサネットワークにおけるモバイルエージェントの直接通信**, マルチメディア，分散，協調とモバイルシンポジウム 2007 (DICOMO2007) 論文集，2007.

3. 末永 俊一郎，松崎 和賢，本位田 真一: **無線センサネットワークにおけるプログラミングモデル - プログラミングサポートとプログラミングアブストラクション**, 合同エージェントワークショップ&シンポジウム 2006 (JAWS2006) 論文集，2006.

主著以外の業績

査読付き国際会議等発表

1. Eric Platon, Shunichiro Suenaga,Nobukazu Yoshioka and Shinichi Honiden: **Transparent Applications Lifetime Management in Wireless Sensor Networks**, In *Demo Track of the 10th International Conference on Ubiquitous Computing (UbiComp 2008)*, 2008.

査読付き国内学会等発表

1. Christian Sommer, Shunichiro Suenaga, Nobukazu Yoshioka and Shinichi Honiden: **Novel Applications in Ubiquitous Computing**, Joint Agent Workshops and Symposium 2007 (JAWS2007)，2007.

2. 松崎 和賢，末永 俊一郎，本位田 真一: **無線センサネットワークにおけるプログラミングモデル - マクロプログラミングとトラディショナルアプローチ**, 合同エージェントワークショップ&シンポジウム 2006 (JAWS2006) 論文集，2006.

付 録 A

サンプルコード

```
// Master Code ----------------------
//リアクション登録（新現在地が報告された場合発動）
BEGIN       pushn ido
            pusht  LOCATION
            pushc 2
            pushc RXN1
            regrxn
//リアクション登録（ID が取得できない場合発動）
            pushn rfi
            pushc 1
            pushc RXN2
            regrxn
//アイドル
IDLE      （アイドル中の任意の処理を記述）
            pushc IDLE
            jumps
//リアクションの発動（新現在地が報告された）
RXN1        pop
            setvar 0
            pushc IDO
            endrxn
//新現在地へ移動
IDO         killslave   //Slave の削除
            getvar 0
            gsmove 1   //  配備パターンの指定
            pushc IDLE
            jumps
//リアクションの発動（Slave-S が ID を取得できない）
RXN2        pushc RFI
            endrxn
//Slave-M を派遣する
RFI         pushn gob
            pushc 1
            slavemloc
            rout
            pushc IDLE
            jumps
// Slave-S Code ----------------------
            startslaves
```

```
//ID が取得できない場合の擬似処理
            pushn rfi
            pushc 1
            masterloc
            rout
//定期的な観測を実施
SENSE       pushc 16
            sleep
            pushc TEMPERATURE
            sense
            pushc THRESHOLD
            cgt
            rjupmc EME
            pushc SENSE
            jumps
//異常時の処理
EME         (温度異常時の任意の処理を記述)
            endslaves
// Slave-M Code ----------------------
            startslavem
//リアクションの登録
BEGM        pushn gob
            pushc 1
            pushcl CLN
            regrxn
//Master から指示があるまで待機
WAIT        wait
//リアクションの発動
CLN         masterloc
            setvar 0
            pushcl CCC
            endrxn
//ベースステーションに報告
CCC         pushloc 1 1
            smove
            pushn loc
            getvar 0
            pushc 2
            out
            endslavem
```

付 録 B

Agilla-他プログラムのコード

```
//初期化処理（リーダと位置情報を交換）
INIT        gettime
            pushn bbb
            loc
            pushc 2
            pushloc 1 2
            rout
//コールバック関数の定義（リーダからの通知により移動）
BEGIN       pushn ggg
            pusht LOCATION
            pushc 2
            pushc RXN
            regrxn
//定期的なセンシングの実施
SENSE       pushc 2
            putled
            pushc 16
            sleep
            pushc SENSE
            jumps
//コールバック関数の処理（リーダの位置を記憶）
RXN         pop
            setvar 0
            clear
            pushc IDO
            endrxn
//移動処理（移動後，センシングを実施）
IDO         getvar 0
            dec
            smove
            clear
            gettime
            pushn bbb
            loc
            pushc 2
            getvar 0
            rout
            pushc SENSE
            jumps
```

付 録 C

コード容量の算出根拠

提案手法の *Master* のコード容量 (*Master_CodeSize*)，Agilla のリーダーのコード容量 (*Leader_CodeSize*) は以下の式 C.1,C.2 で記述できる．

$$\begin{aligned} Master\_CodeSize &= Master\_Process \\ &+ SlaveS\_CodeSize \\ &+ SlaveM\_CodeSize + 6 \end{aligned} \tag{C.1}$$

$$\begin{aligned} Leader\_CodeSize &= Leader\_Process \\ &+ 22 * (N - 1) \end{aligned} \tag{C.2}$$

式 C.1 における *Master_Process* は *Master* の処理部分のコード容量，*SlaveS_CodeSize*, *SlaveM_CodeSize* は，*Slave-S*，*Slave-M* のコード容量，6 は提案手法独自の ISA による追加分[*1] を示す．式 C.2 における *Leader_Process* はリーダーの処理部分のコード容量，N は全プログラム数，22 はリーダーと他プログラム間の通信に要するコード容量を示す．

動的配備される全プログラムを考慮し，提案手法，Agilla のプログラムのコード容量の合計 (*T_CodeSize_P*，*T_CodeSize_A*) は，以下の式 C.3,C.4 で表現できる．

$$\begin{aligned} T\_CodeSize\_P &= Master\_CodeSize \\ &+ (N - 2) * SlaveS\_CodeSize \\ &+ SlaveM\_CodeSize \end{aligned} \tag{C.3}$$

$$\begin{aligned} T\_CodeSize\_A &= Leader\_CodeSize + (N - 1) \\ &* Programs\_CodeSize \end{aligned} \tag{C.4}$$

[*1] *Slave* の消去，*Slave* の宣言

式 C.4 における *Programs_CodeSize* はリーダーを除く他プログラムのコード容量を示す．簡易化のため，*Slave-S* と *Slave-M* のコード容量を同一と仮定し，*SlaveS_CodeSize*, *SlaveM_CodeSize* を *Slave_CodeSize* と定義する．Agilla の他プログラムに *Slave* と同様な動作を行わせる場合，34byte のリーダーとの位置交換と到着通知確認を行うための処理が必要になる．Agilla の他プログラムと *Slave* の処理を同等とすると，式 C.3，C.4 は以下の式 C.5,C.6 に変換できる．

$$\begin{aligned} T\_CodeSize\_P &= Master\_CodeSize + (N-1) \\ &* \quad Slave\_CodeSize \end{aligned} \tag{C.5}$$

$$\begin{aligned} T\_CodeSize\_A &= Leader\_CodeSize + (N-1) \\ &* \quad (Slave\_CodeSize + 34) \end{aligned} \tag{C.6}$$

式 C.1，C.2 における *Master_Process*，*Leader_Process* は同等な処理を実施することから式 C.2 における *Leder_Process* を *Master_Process* で置き換え，式 C.1，C.2 を式 C.5，C.6 に代入すると，式 5.5，5.6(5.4 節）が導出される．

付 録 D

ミドルウェアの主要コンポーネント

表 D.1: 主要コンポーネント

名称	説明
AgentManager	動的配備されるプログラムの起動・終了を動的配備を実施するコンポーネント (AgentSender, AgentReceiver）と連携して実施する．起動されたプログラムを Engine に渡し，プログラムの実行を行う．
AgentSender	動的配備されるプログラムの送信処理を実行する．受信側処理である AgentReceiver からの Ack を待って Code，State，Heap，Stack，Rxn を連続して送信し，プログラムの動的配備を実行する．
AgentReceiver	動的配備されるプログラムの受信処理を実行する．送信側処理である AgentSender によって送信される Code，State，Heap，Stack，Rxn を受信し，Ack を送信する．
ExecutionEngine	プログラムに記述されている ISA を実行する．ISA を CodeMgr を介して一つずつ読み込み，各 ISA に対応したコンポーネントに処理を依頼し，当該コンポーネントからの処理結果を待ち，次の ISA を読み込む．これを繰り返すことで，プログラムの実行を行う．
CodeMgr	動的配備されるプログラムのコードを格納する．*Master*，*Slave* は異なる領域に格納する．Execution Engine からの要求に応じてプログラムの ISA を渡す．
TupleSpace	プログラム間通信を実現する分散タプルスペースを提供する．タプルスペースに対するオペレーション (read/write) を提供する．
GroupMgr	*Master*，*Slave* の親子関係を管理し，配備パターンに従って *Slave* の配備先を決定する．*Master*，*Slave* のプログラム ID と配備されたノード ID を管理する．
KillMgr	*Master* からの要求に従って，GroupMgr と連携して *Master* が配備した *Slave* を消去し，CodeMgr によって確保されていたメモリ領域を開放する．