特別研究報告

題目

センサネットワークにおける
同期型センサ情報収集機構の実装と評価

指導教官

村田　正幸　教授

報告者

樫原　俊太郎

平成 16 年 2 月 19 日

大阪大学 基礎工学部 情報科学科

平成 15 年度 特別研究報告

センサネットワークにおける同期型センサ情報収集機構の実装と評価

樫原　俊太郎

内容梗概

数百，数千のセンサ端末を配置することにより環境の情報を収集する無線センサネットワークにおいては，センサ端末の有する電源容量が限られることから，電力効率のよいセンサ情報収集機構が必要不可欠である．大規模農場における温度変化の観測など，センサ端末の情報を定期的に収集するアプリケーションにおける，電力効率のよい情報収集のためには，センサ端末が同期したタイミングで情報を発信するのがよい．数百，数千のセンサを対象とした集中制御は現実的でなく，個々のセンサが自律的に情報発信のタイミングを決定することにより，同期を確立することが望ましい．我々の研究グループでは，蛍の発光やコオロギの鳴き声，ペースメーカー細胞などのような生物学的な相互同期にもとづく，パルス結合振動子モデルを採用することにより，電力効率のよい同期型センサ情報収集機構を提案している．しかしながら，その有効性は理想的な環境を想定したシミュレーションによって評価されており，実システムへの適用や，実用性については十分に検証されていない．そこで，本報告では，提案機構を市販の無線センサ端末 MOTE に実装し，実環境において実証実験した．また，そのため，シミュレーションでは考慮されていない電波の反射など環境の無線通信に与える影響や無線通信の不安定性といった問題を解決するための手法を提案した．実験の結果，本報告での改良により，同期型センサ情報収集機構が良好に動作し，効率的なセンサ情報収集が行えることを示した．

主な用語

センサネットワーク，情報収集，同期，パルス結合振動子，MOTE

目 次

図 目 次

表 目 次

1　はじめに

無線通信能力を備えたセンサ端末の低価格化により，多数のセンサ端末を配置したネットワークを構成することにより遠隔地や広域を監視，観測，制御する，無線センサネットワーク技術が多くの研究者，開発者の注目を集めている [1]．個々のセンサ端末は，温度，湿度，動きなど観測対象の状態や振る舞いを測定するためのセンサ，限られた計算能力，メモリ，無線送受信機を持ち，多くの場合，電池で動作する．センサ端末の獲得したセンサ情報は，無線通信により直接，間接的に基地局に集められ，基地局上で，あるいはインターネットを介して転送され，アプリケーションや利用者に提供される（図 1）．

環境内に配置されたセンサ端末は電池の入れ替えが困難なため，これらの端末からなるセンサネットワークによる長期間観測のためには，センサ情報の収集のための電力消費が低いセンサ情報収集機構が必要である．また，電力の枯渇によるセンサ端末の停止やセンサ端末の追加，移動に際しても手動による再調整を必要とせず，センサ情報の収集を継続できることが求められる．多数のセンサ端末が無作為に配置されることを考慮すると，集中型の制御は現実的ではない．センサネットワークにおける情報収集のための研究は [2-4] のように既になされており，様々な機構が提案されている．例えば LEACH [2] では，情報発信の消費電力は通信距離の 2 乗に比例すること，センサ端末から離れた場所に基地局が設置されることを考慮し，近接したセンサ端末でクラスタを構成し，クラスタごとにクラスタヘッドと呼ばれるセンサ端末が，クラスタ内のセンサ端末からセンサ情報を収集し，基地局に送信する機構を提案している．さらに，電力負担の大きいクラスタヘッドの役割をセンサ端末間で交代で担うことにより，センサ端末の電力消費を均一化し，センサネットワークの長寿命化を図る．しかしながら，いずれの提案においてもセンサネットワークのトポロジ全体の情報が必要になる，センサ端末の事前設定が必要であるなどの問題を有する．そのため，センサ端末数や観測領域などセンサネットワークの規模の拡大への拡張性，センサ端末の停止，移動，追加といった動的な変化への適応性に欠いている．

我々の研究グループでは，多様に変化するセンサネットワークにおいて，定期的なセンサ情報収集のための電力効率のよい同期型センサ情報収集機構を提案している [5]．この同期型センサ情報収集機構は，電源容量の限られた多数のセンサ端末の配置されたセンサネット

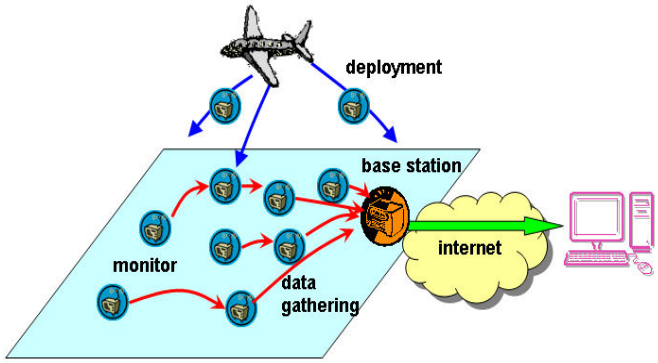

図 1: センサネットワーク

ワークから，望む周期で定期的にセンサ情報を収集するようなアプリケーションへの適用を対象としている．提案機構は，センサ端末数や領域の広さへの拡張性，センサ端末の故障の影響を受けない対障害性，センサ端末の停止，追加，移動に対する適応性，情報収集頻度に対する柔軟性を有し，完全に分散型である．電力消費をおさえるためには，センサ情報送受信の頻度を少なくすること，情報転送に関わらないセンサ端末が無線送受信機の電源を切ること，および送受信する情報量を削減することが効果的である．そのため，センサ端末が同期したタイミングでセンサネットワークの周縁部から順にセンサ情報を送信していく同期型のセンサ情報収集を行う．例えば，図 2 のようなセンサネットワークにおいては，センサ情報収集の周期にあわせて，まず，周辺部にあるセンサ端末 (図中，黒丸) が同じタイミングでセンサ情報を発信する．これを受けたより基地局に近いセンサ端末 (白丸) は，自らのセンサ情報と，受信したセンサ情報を集約して情報量を小さくし，互いに同期して，より基地局に近いセンサ端末 (灰色の丸) へとセンサ情報を伝える．なお，図中，四角は基地局を，また，破線で描かれたそれぞれの円はセンサ端末や基地局の発信する無線電波の伝播領域を表す．このような同期と情報集約を利用した情報収集により，センサネットワークの長寿命化を図ることができる．

観測対象となる領域へのセンサ端末の導入は計画的でなく，無作為のため，例えば，サーバを配置して，全てのセンサ端末を位置を管理し，センサ端末ごとの情報発信のタイミングを指示するといった集中型の制御は現実的でない，したがって，集中制御なしに個々のセン

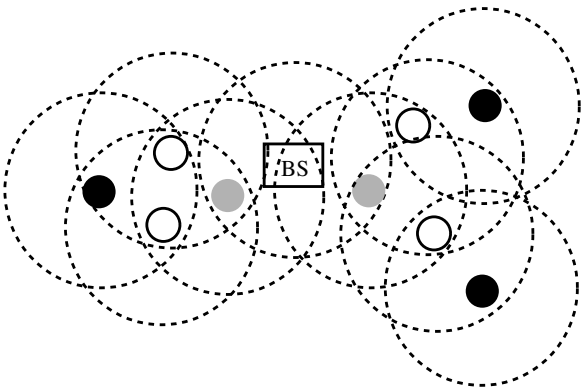

図 2: 同期型センサ情報収集

サ端末がセンサ情報を送信する周期やタイミングを決定することにより，センサネットワーク全体での同期を達成しなければならない．生物界においては個体間の相互作用によってこのような完全に分散された同期が達成されている．例えば，蛍は，互いに離れている時は各々の周期によって独立に発光している．しかしながら，群れになると，発光により近隣の蛍同士で互いに刺激を与え合い，周期を調整することで，群れが全体が同期して発光するようになる．このような生物界における同期機構はパルス結合振動子としてモデル化されている [6-8]．文献 [8] においては，パルス結合振動子をオブジェクトのクラスタリングに適用しており，類似したオブジェクト同士では強い刺激を与え，類似の少ないオブジェクトには負の刺激を与えることにより，似通ったオブジェクト同士のみが同期し，クラスタを形成するというクラスタリング手法を提案している．また，文献 [9] では，蛍の同期モデルにもとづき，それぞれのノードが自身のタイマの発火により，近隣のノードと互いに刺激を与え合い，タイマの周期を調整することで，結果として効率よくネットワーク全体が同期することを確かめている．

同期型センサ情報収集機構では，このパルス結合振動子モデルをセンサネットワークに適用し，センサ端末間の相互作用によって同期を確立し，効率のよいセンサ情報収集を実現している．しかしながら，提案手法の有効性は理想的な環境を想定したシミュレーションによって評価されており，電波の反射などが無線通信に与える影響や，パケットの衝突など

輻輳による送信遅延などが考慮されていない．本報告では，市販の無線センサ端末 MOTE [10] を用いた実システムへの実装，実証実験により，同期型センサ情報収集機構の実用性，有効性を検証するとともに，これらの問題を解決した．具体的には，受信回数や受信電波強度にもとづく不安定な通信のフィルタリング機能，輻輳によって遅延した刺激を無視する機能を提案した．

以降，2 章では我々の研究グループの提案する同期型センサ情報収集機構について述べる．3 章では，提案機構の実証実験に用いた無線センサ端末 MOTE および実装方法について述べる．4 章において，提案機構の有効性および実用性を評価し，さらに評価結果にもとづき実環境を考慮した提案機構の改良およびその有効性を評価する．最後に，5 章で本報告のまとめと今後の課題について述べる．

2　同期型センサ情報収集機構

2.1　センサネットワーク

センサーネットワークは，通信機能を有する小型のセンサ端末を多数配置し，それらによるネットワークを構成することで，刻々と変化する環境情報の取得，広域にわたる情報の同時把握や機器の制御を行うシステムである．センサの獲得した情報は，無線通信により基地局に集められ，基地局は収集したセンサ情報を集約し，利用者に提示する．センサネットワークは，大規模農場の日照や土壌，温度の観測や，交通量の調査，セキュリティ監視システムや，家電の制御，さらには火山や深海など危険な場所の観測など，広範囲な分野への応用が可能である．センサ端末の小型化，低価格化により，無線センサネットワーク技術には多くの研究者，開発者の注目が集められている [1].

センサ端末は，温度，湿度，光，磁気，加速度など，観測対象の状態を測定するセンサ，計算能力の限られた汎用 CPU やメモリ，無線送受信機を持ち，多くの場合，電池で動作する．センサ端末の環境への配置は無作為で，またセンサ端末数が多いことから，センサ端末の電池の管理，入れ替えは困難である．したがって，長期間にわたり情報収集を継続するための，電力消費が低いセンサ情報収集機構が必要とされている．また，電力の枯渇によるセンサ端末の停止，観測領域の拡大や変更のためのセンサ端末の動的な追加や移動の際にも，センサ端末やセンサ情報収集機構の再調整を必要とせず，センサ情報の収集を継続できることが求められる．

2.2　パルス結合振動子モデル

ペースメーカー細胞や蛍，コオロギなどに見られる同期メカニズムは，文献 [8] にもとづいた，パルス結合振動子モデルによって表現することができる [6]．本節ではパルス結合振動子モデルの概要を述べる．N 個の振動子 $\boldsymbol{O} = \{O_1, \cdots, O_N\}$ を考える．振動子 O_i は位相 $\phi_i \in [0,1]$ と状態 $x_i \in [0,1]$ を持つ．これらの関係は関数 f_i で表される．

$$x_i = f_i(\phi_i) \tag{1}$$

ただし，$f_i : [0,1] \to [0,1]$ は単調増加関数で，$f_i(0) = 0$，$f_i(1) = 1$ である．位相 ϕ_i は周期 T_i で 1 に遷移し，1 に到達すると 0 に戻る．ただし，$\frac{d\phi_i}{dt} = \frac{1}{T_i}$ である．状態 x_i が 1 に到達したとき振動子 O_i は発火し，状態 x_i は 0 に初期化され，位相 $\phi_i = f_i^{-1}(x_i)$ より，位相 ϕ_i も 0 に戻る．発火した振動子 O_i と対になった振動子 O_j は刺激を受け，状態 x_j は $\epsilon_i(\phi_j)$ だけ増加する．

$$x_j(t^+) = B(x_j(t) + \epsilon_i(\phi_j)) \tag{2}$$

ただし，関数 $B(x)$ は以下で与えられる．

$$B(x) = \begin{cases} x, & \text{if} \quad 0 \leq x \leq 1 \\ 0, & \text{if} \quad x < 0 \\ 1, & \text{if} \quad x > 1 \end{cases}$$

刺激を受けた振動子 O_j の位相 ϕ_j は $\phi_j = f_j^{-1}(x_j)$ で与えられる．刺激を受けた結果，状態 x_j が 1 に到達すると，振動子 O_j もまた発火する．このとき振動子 O_i と振動子 O_j は同期したと見なされる．

異なった周期や位相と状態の関数を持った振動子間においても任意の初期状態から同期することが証明されている [6]．また，全ての振動子が互いに刺激を与え合う関係になくとも，間接的な刺激の伝播により，振動子集合全体が同期に達する．

2.3 同期型センサ情報収集機構の概要

我々の研究グループの提案する同期型センサ情報収集機構では，基地局からの距離に応じたタイミングでセンサ端末が順次センサ情報を発信することにより，効率的にセンサ情報が収集される．

センサ端末の基地局からの距離をレベルと呼ぶ．図 2 において，基地局の信号を直接受信できる灰色の丸で示されたセンサ端末はレベル 1 ，これらレベル 1 のセンサ端末の信号を受信することのできる白丸のセンサ端末はレベル 2 となる．図中，破線の円はそれぞれ中心に位置するセンサ端末の発信する電波の到達範囲を示す．このように，センサ端末の無線送受信機の能力，あるいは電力消費をおさえるための無線通信の制御により，センサ端末の

通信可能範囲は限られる．したがって，同期型センサ情報収集機構では，センサ端末が他のセンサ情報の発信した情報を基地局へ中継する，マルチホップ通信が行われる．すなわち，それぞれのセンサ端末のレベルは基地局までのホップ数を表わしている．最もレベルの高い端末から順にセンサ情報を発信することによりセンサ情報が適切に基地局に伝えられるためには，センサ端末は自身より一つレベルの小さいセンサ端末の情報発信に先立ってセンサ情報を発信しなければならない．また，センサ端末が同期して情報を発信するため，MAC (Media Access Control) 層における電波の衝突が発生し，再送による通信の遅延が生じる可能性がある．そのため，例えば，時刻 t にセンサ情報を収集する場合，レベル 1 のセンサ端末は MAC 層の衝突を考慮した十分大きい時間 δ だけ早く，すなわち，時刻 $t-\delta$ にセンサ情報を発信しなければならない．さらに，レベル 2 のセンサ端末はレベル 1 のセンサ端末の情報発信に先立ち，時刻 $t-2\delta$ にセンサ情報を発信することになる．

同じレベルのセンサ端末が適切なタイミングでセンサ情報を発信するためには，互いに直接通信できないセンサ端末が同期を確立しなければならない．そこで，そのため，同期型センサ情報収集機構では，パルス結合振動子モデルを利用することにより同期を達成する．なお，基地局は近隣のセンサを同期させるために情報収集周期にしたがってビーコン信号を発する．N 個のセンサ端末を $\boldsymbol{S} = \{S_1, \cdots, S_N\}$ と表わす．センサ端末 S_i はレベル l_i に属する．初期状態では レベル l_i は無限大もしくは十分に大きな値である．センサ端末はタイマと状態 x_i を持つ．状態 s_i はタイマの位相 ϕ_i の単調増加関数 $f_i : [0,1] \to [0,1]$ で与えられる．例えば本報告では f_i として次式を用いる [6][8].

$$\forall i, f_i(\phi_i) = \frac{1}{b}\ln[1 + (e^b - 1)\phi_i] \tag{3}$$

$b > 0$ は同期の早さを左右するパラメータの一つであり，b が大きくなるにつれて，状態遷移が早くなり，より早く同期が達成される [6].

時刻 t にセンサ端末 S_i がセンサ端末 S_j の信号を受信したとする．センサ端末 l_j が l_i より小さい場合， S_i は刺激を受け，状態 x_i を以下のように変化させる．

$$x_i(t^+) = B(x_i(t) + \epsilon) \tag{4}$$

刺激により，状態 x_i が 1 になると，センサ端末 S_i の状態 x_i と位相 ϕ_i は 0 に戻る．

提案機構では，センサ端末 S_i のタイマは一つレベルの小さいセンサ端末 S_j ($l_j = l_i - 1$) の情報発信に同期する．したがって，先に述べたとおり，センサ端末 S_i は状態 x_i が 1 になるよりオフセット δ_i だけ早くセンサ情報を発信しなければならない．センサ端末 S_i のオフセット δ_i はセンサネットワーク全体で共通の値を用いる，センサ端末自身が MAC 層における輻輳の度合にもとづいて設定する，あるいは，一つレベルの小さいセンサ端末がセンサ情報の受信状態から適切なオフセットを定めてセンサ端末 S_i に通知する，などにより定められる．オフセット δ_i を考慮した位相 $\phi_i^{'}$ を次式で定義する．

$$\phi_i^{'} = p(\phi_i, \delta_i) = \begin{cases} \phi_i + \delta_i, & \text{if } \phi_i + \delta_i \leq 1 \\ \phi_i + \delta_i - 1, & \text{otherwise} \end{cases}$$

オフセットを考慮した状態 $x_i^{'}$ は $f_i(\phi_i^{'})$ により与えられる．したがって，$x_i^{'} = f_i(p(g_i(x_i(t^+)), \delta_i))$ となる．ここで， $g_i = f_i^{-1}$ である．状態 $x_i^{'}$ は状態 x_i よりオフセット δ_i だけ早く 1 に達するため，センサ端末 S_i は δ_i だけ早く情報を発信する．発信される情報には，センサ端末 S_i が自身のセンサにより獲得した情報を他のセンサ端末から受信した情報と集約したもの，レベル l_i が含まれる．なお，情報発信の消費電力は情報の大きさに比例するため，例えば複数の n ビットのセンサ情報を集約して，新たに n ビットのセンサ情報を生成し，これを送信するものとする [2]．なお，センサ端末 S_i は時刻 t に刺激を受けると，MAC 層での輻輳により遅れて届いた信号によって再度刺激を受けることを避けるため，時刻 t から $t + \delta_i$ まで，信号による刺激を受けない．

センサ端末 S_i のレベルは，そのセンサ端末の受け取ることができるセンサ情報のレベルの最も小さいものを l_j とした場合，$l_i = l_j + 1$ である．基地局の発信するビーコン信号のレベルは 0 である．新しいセンサ端末が，より基地局より遠いセンサ端末のメッセージを受け取った場合には，自身のレベルを誤って決定してしまう．しかしながら，時間が経つにしたがって，より基地局に近いセンサ端末のメッセージを受信し，自分のレベルを正しく知ることができる．

センサ端末 S_j が自身のレベルより一つ大きい場合 $l_j = l_i + 1$ には，センサ端末 S_i は，受信したセンサ情報を自身のセンサ情報と集約する．センサ端末は自身と同じかそれ以上のレベルのセンサ端末の信号には刺激を受けない．つまり，同じレベルにあるセンサ端末は，

互いに通信することなく，より低いレベルのセンサ端末からの刺激を受けることによって同期を達成する．

稼動中のセンサネットワークに新たに導入されたセンサ端末は，センサにより環境の観測を開始する．周囲のセンサ端末の情報発信を繰り返し受信することにより，レベルが適切に設定され，また，レベルに応じてタイマの同期が確立される．同期が確立されるまでに，新しいセンサ端末の情報発信をよりレベルの大きいセンサ端末が受信して刺激を受けることにより，同期のずれが伝播する可能性がある．しかしながら，新しいセンサ端末のタイマがよりレベルの小さいセンサ端末の情報発信との同期を確立し，適正なタイミングでセンサ情報を発信するようになると，再度全てのセンサ端末が正しい周期，タイミングで動作するようになる．

センサ端末の長寿命化のためには，センサ端末が不要な機器や回路の電源を切る事が有効である [11-13]．同期を確立するまでは，他のセンサ端末からの刺激を受けるため，センサ端末は受信機の電源を入れておかなければならない．しかしながら，同期型センサ情報収集機構においては，いったん同期が確立され，情報発信のタイミングが確定した後は，位相 ϕ_i が 0 から $1-2\delta_i$ の間は送受信機ともに電源を切り，節電することができる．位相 ϕ_i が $1-2\delta_i$ に達すると，センサ端末は受信機の電源を入れ，レベルが一つ大きいセンサ端末からのセンサ情報を受信する．ただし，オフセット δ_i をそれらのセンサ端末に通知する，センサネットワーク全体で共通のオフセットを用いるなどが必要である．位相 ϕ_i が $1-\delta_i$ に達し，オフセットを考慮した状態 x' が 1 になると，センサ端末は送信機の電源を入れ，センサ情報を発信する．その後，位相 $\phi_i = 1$ において，レベルが一つ小さいセンサ端末の情報発信を受信し，同期を確立させた後，送受信機の電源を切る．ただし，位相 $\phi_i = 1$ においてレベルが一つ小さいセンサ端末からの情報発信を受信しなかった場合には，同期が失われたものと判断し，再度同期が確立されるまで受信機の電源を入れ続ける．このようにして，常に送受信機の電源を入れている場合と比較して，同期が確立している間は，少なくとも $2\delta_i$ だけ電力を節約する事ができる．したがって，提案機構を用いることにより，電力効率のよいセンサ情報収集が達成され，センサネットワークの長寿命化を図る事が出来る．

提案機構ではセンサ端末が故障した場合にも，継続してセンサネットワークからセンサ情

報を収集できる．センサ情報がレベルの確定前に故障した場合には，そのセンサ端末の発信するレベルは周囲のセンサ端末のレベルよりも大きいため，他に影響を与えることはない．ただし，レベルを適切に設定した後に故障した場合には，周囲のセンサ端末の同期を妨げるため，センサ情報の発信を停止しなければならない．

3　同期型センサ情報収集機構の実装

3.1　実装システムの概要

同期型センサ情報収集機構のために本報告で用いたシステムは複数のセンサ端末，一台の基地局，基地局を制御する一台のパーソナルコンピュータからなる．センサ端末は，同期型センサ情報収集機構にもとづき，周囲のセンサ端末もしくは基地局の発信する情報に刺激を受け，センサ情報を繰り返し適切なタイミングで送信する．基地局は，制御用コンピュータから RS-232C ポートを通じて送られてきた命令からセンサ情報収集周期を定め，周囲に定期的にビーコン信号を発信する．また，センサ端末からのセンサ情報を受信し，制御用コンピュータに転送する．制御用コンピュータは，RS-232C ポート を通じて基地局から転送されてきたセンサ情報を解析し，送信元センサ端末の識別子とレベル，センサ情報の発信時刻を記録する．また，RS-232C ポート を通じて基地局にセンサ情報収集の周期を変更する命令を送る．

3.2　無線センサ端末 MOTE

MOTE [10] は U. C. Berkeley 大学の NEST (Network Embedded Systems Technology) プロジェクトによって開発されたセンサネットワーク構築用のセンサ端末である．単三電池 2 本で駆動する名刺大の MICA2 と，ボタン電池で駆動する五百円玉大の MICA2DOT の 2 種類のセンサ端末がある (図 3)．センサ端末には専用のボードによってプログラムを書き込むことができる．表 1 は MOTE の主な仕様である．また，図 4 は MOTE の回路の概略である．

CPU として 7.4 MHz で駆動する汎用プロセッサ ATmega128 を備え，128 KB のプログラム領域と，4 KB の RAM，512 KB のフラッシュメモリを有する．またセンサや無線送受信機は 8 bit，帯域幅 4 KHz のバスを通じて制御する．MICA2DOT は温度センサを有し，MICA2 には，音，光，温度，加速度，磁気のセンサを有した基盤を接続する事ができる．無線 IC には CC1000 が用いられており，無線送受信機の送信電力は -20dBm ～ 10dBm の範囲で設定できる．MAC 層 プロトコルには CSMA/CA を使用しており，パケットの衝突

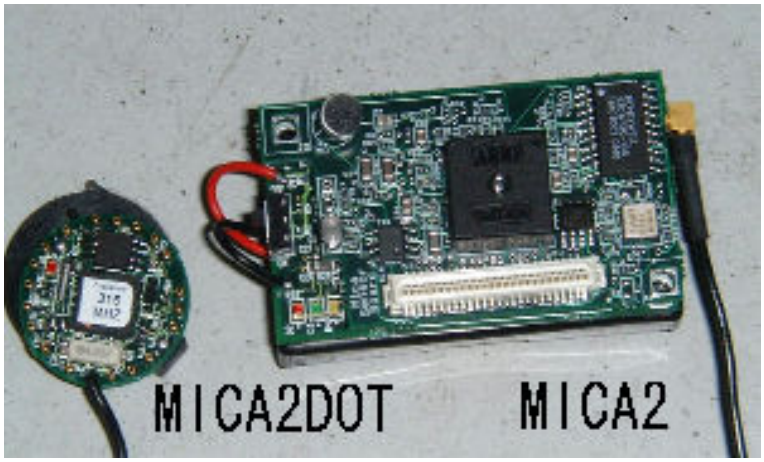

図 3: MOTE 端末

を避けるため，通信路が空いていることを確認してからランダムな長さの時間だけ待ち，パケットを送信する．MICA2 には赤，黄，緑の 3 つの LED が，MICA2DOT には赤の LED 1 つがついており，プログラムによって光らせる事ができる．

MOTE は U. C. Berkeley 大学でセンサ用に開発された TinyOS [14] によって制御される．TinyOS は Java や C 言語に似たプログラミング言語 NesC によって記述されている．プログラム開発のための無線通信，RS-232C 通信，送信電力制御，センサ情報取得などのライブラリやツールが提供されており，容易にプログラムの開発ができる．

3.3 MOTE へのセンサ情報収集機構の実装

MOTE は内部に 1 ミリ秒単位で動作する時計を持っている．例えば，本報告における実験では，プログラム内で，この時計の 0.1 秒ごとに位相 ϕ_i を 0.01 ずつ変位させることで，10 秒周期のタイマを実現している．タイマの変位後に，式 (3) により状態 x_i を求める．さらにオフセットを考慮した位相 ϕ_i' から，状態 x_i' を得る．状態 x_i が 1 に到達すると，位相 ϕ_i と状態 x_i を 0 に戻す．また，オフセットを考慮した状態 x_i' が 1 に到達すると，センサ情報を取得し，それまでに受信した他のセンサ端末からのセンサ情報とともにパケットを構成して発信した後，オフセットを考慮した位相 ϕ_i' と状態 x_i' を 0 に戻す．センサ端末の

表 1: MOTE の仕様

CPU	ATmega128 7.4MHz
プログラムメモリ	128KB
RAM	4KB
フラッシュメモリ	512KB
無線周波数/型式	315MHz/FSK
MAC プロトコル	CSMA/CA
電波到達距離	5 ～ 150m
センサ (MICA 型)	音，光，温度，加速度，磁気
センサ (DOT 型)	温度
電源	DC3V
電流 (sleep)	15μA
電流 (受信)	1.8mA
電流 (送信)	12mA

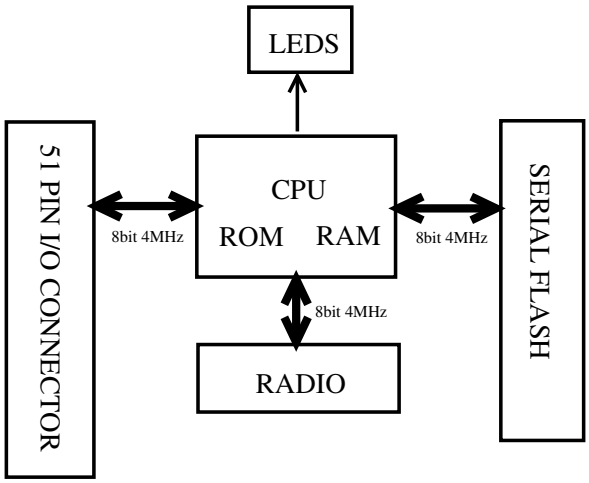

図 4: MOTE の回路構成

表 2: パケットのフォーマット

1 (bit)

センサ端末識別子	レベル	センサ情報	センサ情報取得時刻	オフセット	刺激	予備 1	予備 2	予備 3	予備 4	センサ端末識別子 1	レベル 1

8 (bit)

センサ情報 1	センサ情報取得時刻 1	・ ・ ・ ・ ・ ・	センサ端末識別子 20	レベル 20	センサ情報 20	センサ情報取得時刻 20

発信するパケットのフォーマットを表 2 に示す．パケットはパケットを送信したセンサ端末に関する 72 bit の情報と，動作検証用の予備の 40 bit，下位のレベルから受信した 20 個のセンサ情報 1120 bit の計 1232 bit で構成される．なお，我々の研究グループの提案する同期型センサ情報収集機構では，2.3 節で述べたとおり，情報送信の電力消費をおさえるため，複数のセンサ情報を効率よくとりまとめ，ひとつ分の大きさにして送信する．本報告では，センサ情報がどのセンサ端末を経由してどのように集められているかなど，詳細な動作検証を行うため，センサ端末は，周期内にレベルが一つ大きいセンサ端末から受信した全てのセンサ情報をメモリに蓄積し，センサ情報として発信する．基地局によるビーコン信号にも同じパケットフォーマットを用いる．ただし，レベルは 0 とする．

送信電力を最も低い -20 dBm に設定した場合の MOTE の通信可能距離は 5~10 m であ

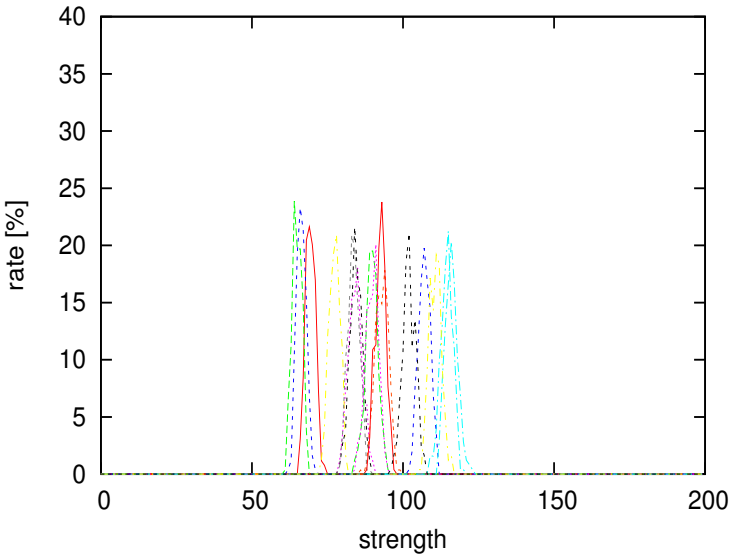

図 5: センサ端末ごとの受信電波強度の違い

る．4 ホップ程度のセンサネットワークでは，最大直径 80 m の空間が必要になり，また全てのセンサ端末の状況を把握しづらいため，受信した電波の強度が一定以下のものを無視することで，通信距離を擬似的に小さくすることにした．しかしながら，送信電力と電波の到達距離には，センサ端末による個体差や反射の影響などによるばらつきがある．図 5 は基地局から 1 m 離れた同じ場所で，異なるセンサ端末から送信電力 -20 dBm でそれぞれ 300 回信号を送った際の，基地局における受信電波強度を横軸に，その割合を縦軸にセンサ端末ごとに示した図である．なお，受信電波強度は値が小さいほど大きい．図より同じ送信電力でも，受信電波強度がセンサ端末間で 61 から 123 まで大きくばらついていることがわかる．また，同じセンサ端末でも受信電波強度は一定でない．また，図 6 のように，基地局を中心とした半径 1 m の円周上にセンサ端末を配置し，別のセンサ端末を角度 0 度，45 度，90 度，180 度と順次位置を変えて配置した場合の，受信電波強度の変化の違いを図 7 に示す．図 5 と同様，それぞれの設置場所について 300 回の電波発信を行った．図よりセンサ端末の配置場所によって影響を受けており，特に同じ位置に 2 つのセンサ端末が配置された場合の影響が大きいことがわかる．さらに図 8 は他に障害物のない屋上において，中央，高さ 1 m の壁際，およびその中間にセンサ端末を配置した場合の受信電波強度の違いを示してい

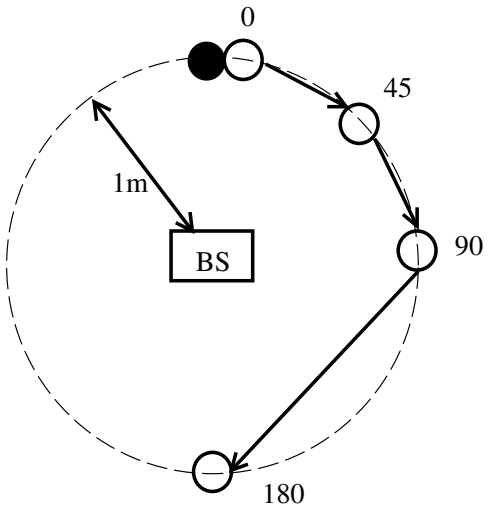

図 6: 干渉評価のためのセンサ端末配置

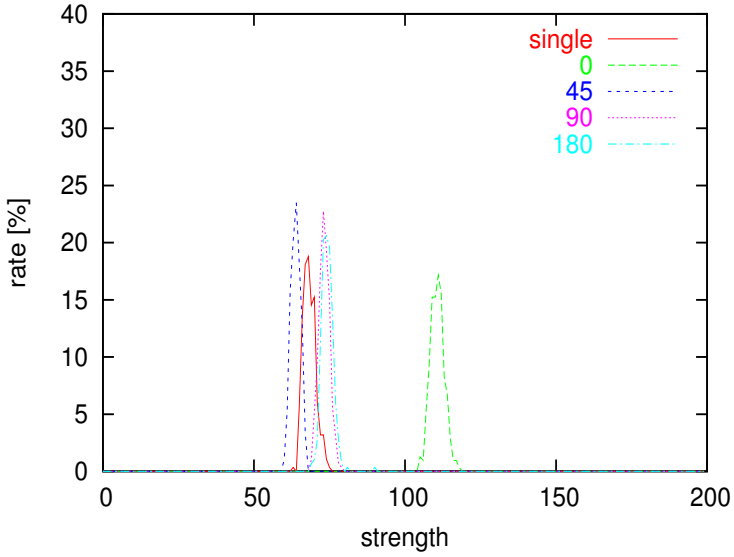

図 7: 他のセンサ端末による受信電波強度への影響

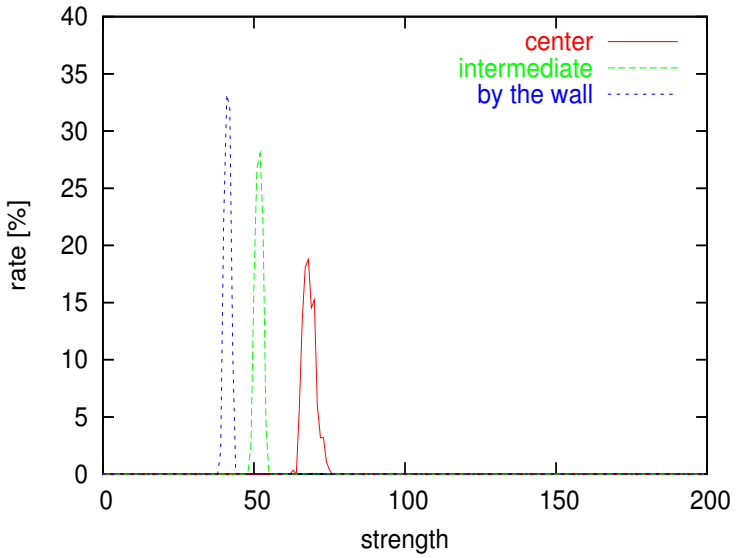

図 8: センサ端末の設置場所による受信電波強度の違い

る．センサ端末が壁際に近づくほど，壁による電波の反射により，受信電波強度が強くなっている． また，図 9 のように，屋内で基地局を壁際に固定し，基地局から 1 m の距離で壁との角度を 0 度，45 度，90 度と順次変えてセンサ端末を設置し，それぞれ 300 回の通信を行った場合の受信電波強度の観測結果を図 10 に示す．図に示されているように，同じ距離でも場所によって電波強度に差が出る．特に，基地局に対して 45 度の角度にセンサ端末を設置した結果においては，人がセンサ端末の間を通過したため一時的に電波強度が弱くなり，大きな変動がみられる．これらの予備実験の結果にもとづいて，受信電波強度が 120 以上の弱い電波を無視することとし，通信可能距離を 1.0~1.6 m 程度に制御した．

MOTE によるセンサ端末の電波受信時の具体的な動作は次のようになる．センサ端末は電波を受信すると，まず，受信電波強度を調べ，電波強度が 120 以上のものはこれを無視する．次に，受信したパケットから情報を発信したセンサ端末のレベルを調べ，自らのレベルより 2 以上小さい場合，レベルを調整するとともに刺激をうけ，状態をパケットに書かれた ϵ だけ変化させる．タイマの位相も同時に変化する．オフセットを考慮した刺激により状態が 1 になった場合にはセンサ情報を最小の送信電力で発信する．自らのレベルより 1 レベルの小さいセンサ端末からの受信であった場合には，刺激を受け，オフセットを考慮

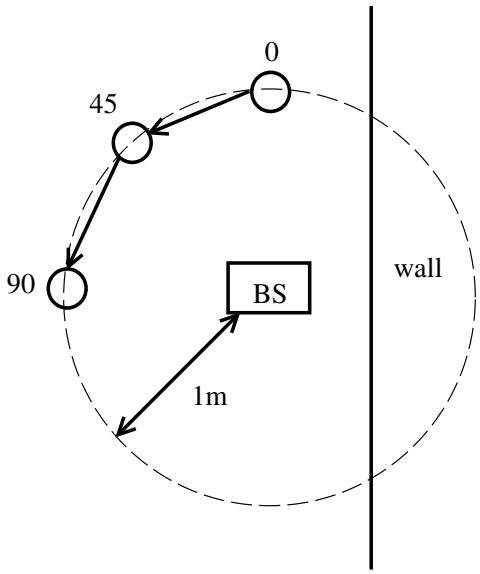

図 9: 屋内におけるセンサ端末設置場所

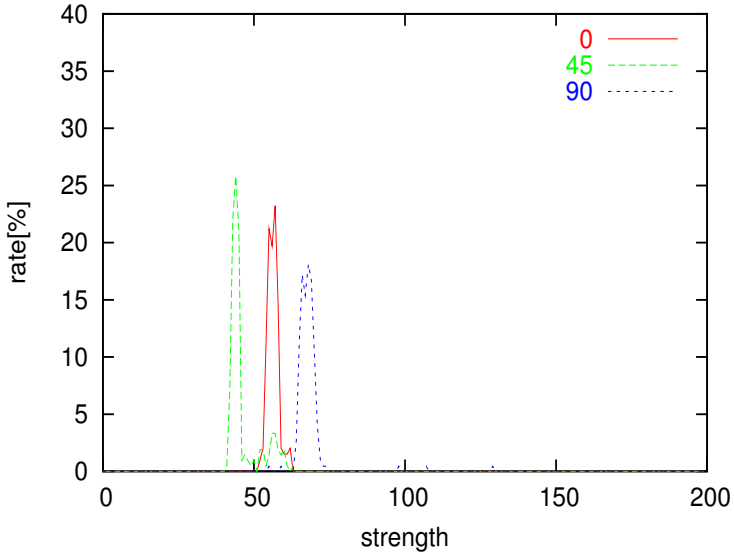

図 10: 屋内におけるセンサ端末設置場所による受信電波強度の違い

した状態が 1 になった場合にはセンサ情報を発信する．自らのレベルより 1 大きいセンサ端末からの電波であった場合には，受信時刻やセンサ情報などをメモリに最大 20 個格納する．センサ情報が 20 個を超えた場合は，21 個目以降のセンサ情報は棄却される．

センサ端末は，タイマの位相が 1 付近で同じレベルのセンサ端末から連続 10 回刺激を受けると，同期が確立されたと判断し，スリープモードに移行する．スリープモードでは，タイマの位相が 0 から $1 - 2\delta$ の間は，MOTE の機能を利用して，無線送受信機の電源を切る．

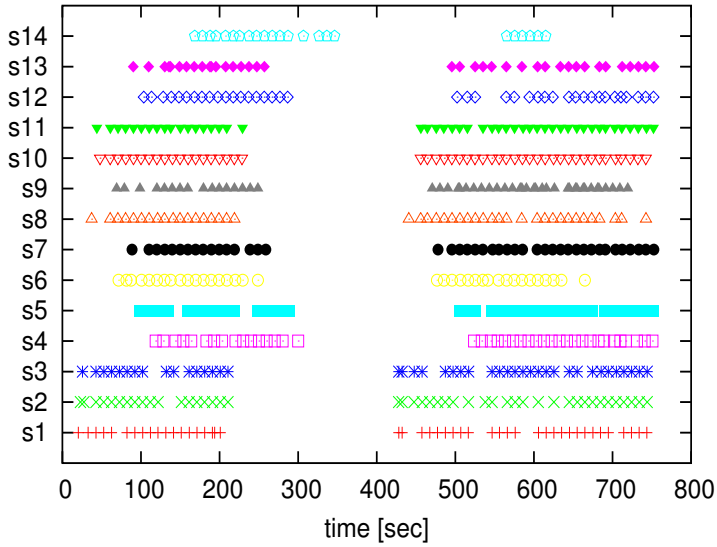

図 11: 送信電力を変えた場合のセンサ情報収集の様子

4　実装システムの実験評価と同期型センサ情報収集機構の改良

4.1　評価環境

提案手法では，無線通信は対称的であり，センサ端末はその通信範囲内のセンサ端末と相互に情報の送受信が可能であることを仮定している．しかしながら，実環境においては，壁や天井，障害物などで電波が反射，妨害されるため，通信が非対称となる場合がある．図 11 は，送信電力を最小にした状態で実験を行った後，同一環境で送信電力を大きくして実験を行った結果である．200 秒から 550 秒までのセンサ端末ごとに異なる空白は，送信電力の変更作業によるものである．送信電力が大きくなることにより，電波の到達範囲が広くなり，衝突が頻繁に起こるようになる．そのため，センサ端末 s6 では約 665 秒，センサ端末 s14 では約 619 秒からセンサ情報の発信が確認できなくなっている．そこで本報告では，同期型センサ情報収集機構の基本動作を検証するため，電波の反射の影響を避け，障害物が少なく人通りのない基礎工学部 G 棟の屋上（図 12）で実験を行った．

また，結果の考察を簡単にするため，図 13 に示すように，基地局（図中 BS）を中心とした同心円状に 16 台のセンサ端末を配置した．いずれのセンサ端末も温度センサを有し，

図 12: 実験環境

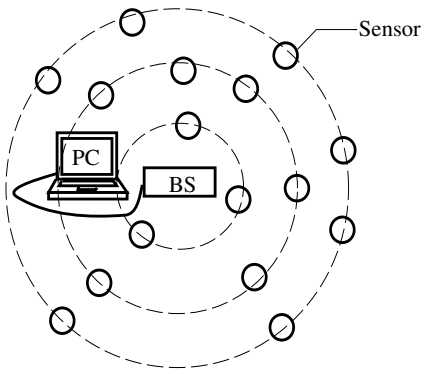

図 13: 実験システムにおけるセンサ端末配置

センサ情報送信のタイミングに観測を行う．3.3 章で述べたように情報の受信として認識する受信電波強度を調整することにより，基地局およびセンサ端末の情報発信の到達半径が 1.0∼1.6m になるようにした．円周の半径は 1.0m，2.0m，3.0m と 1.0m ごとに設定した．ただし，送信電力設定などが同じでも図 5 に示したようなセンサ端末の個体差，図 8 や図 10 に示した場所の影響，さらには図 7 に示した他の端末による影響のため，電波の到達距離が変化する．そのため，センサ端末の配置場所を円周の内側や外側に 10cm 程度ずらすなど，センサネットワークの構築に参加しやすいように一部調整を施した．

実験に使用しているラップトップコンピュータのバッテリの容量を考慮し，1 回の実験でできるだけ多くの回数の情報収集が行えるよう，センサ端末のタイマ周期は 10 秒とした．基地局は 10 秒ごとにビーコン信号を発信する．位相と状態の関数 f_i は全てのセンサ端末に共通とし，式 (3) で与えた．ただし，$b = 3.0$ とした [8]. 刺激 ϵ は 0.3 [6]，オフセット δ_i は全て 0.2 とした [5]．したがって，内側から n 番目の円周上のセンサ端末は，基地局のビーコン信号より $0.2n$ ，すなわち $2n$ 秒だけ早くセンサ情報を発信することになる．全てのセンサ端末がその位置に応じたタイミングで情報発信を行ったとき，センサネットワーク全体で同期がとれたとみなす．

4.2 実装システムによるセンサ情報収集の評価

提案機構を用いて実験を行った結果を図 14 に示す．図にはそれぞれのセンサ端末がセンサ情報を発信した時刻を示している．なお，見やすさのため，図 15 のように本節での実験は 6 台のセンサ端末で行った．センサ端末識別子の 10 の位はセンサ端末が内側から何番目の円周上に配置されているかを示し，1 の位はその円周上でセンサ端末を区別する番号である．時刻 122 秒でセンサネットワーク全体の同期が確立している．センサ端末 s11 ，センサ端末 s12 はレベル 1 でビーコン信号より約 2 秒早く，センサ端末 s21 ，センサ端末 s22 はレベル 2 で約 4 秒早く，さらに，センサ端末 s31 ，センサ端末 s32 はレベル 3 で約 6 秒早くセンサ情報を発信している．しかしながら，その後すぐに同期が失われている．これは，提案機構で考慮されていない，無線通信の衝突によるものである．例えば，レベル i のセンサ端末 S_i が，レベル j のセンサ端末 S_j と同期していたとする．したがってセンサ端末 S_j

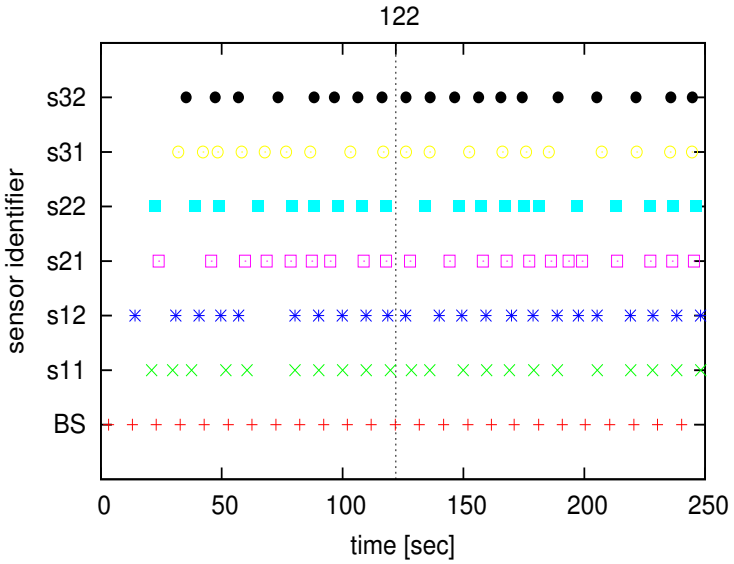

図 14: 提案機構におけるセンサ端末の情報発信のタイミング

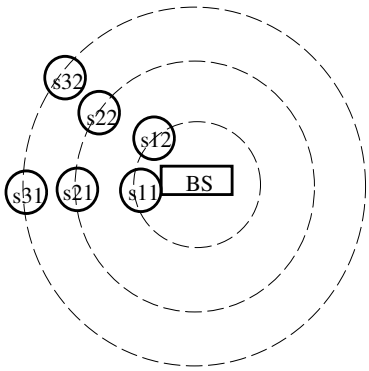

図 15: 提案機構の評価実験におけるセンサ端末配置

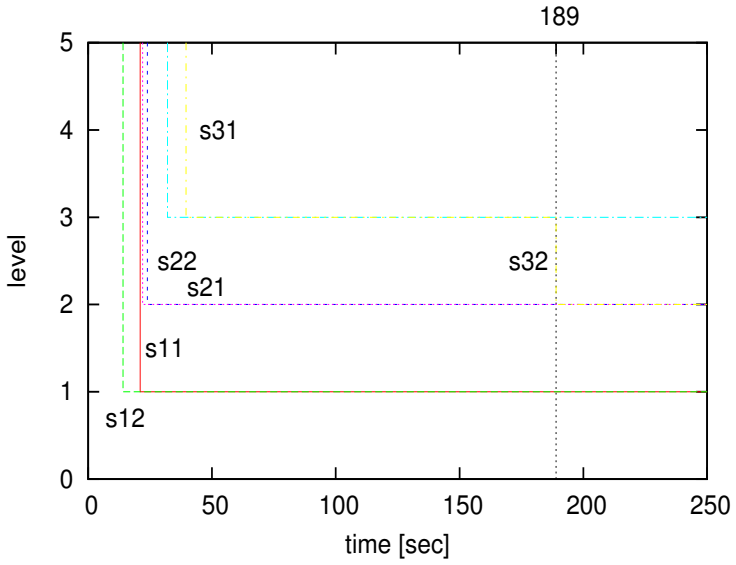

図 16: 提案機構におけるセンサ端末のレベルの変化の様子

からの信号は，センサ端末 S_i の状態が 1 のときに到達しなければならない．無線通信の衝突により，センサ端末 S_j からの信号が 0.01 だけ遅れて到達した場合，すでにセンサ端末 S_i の位相は 0.01 となっているため，刺激を受けて状態が変化してしまい，同期が崩れる．

センサ端末のレベル調整の様子を図 16 に示す．図よりセンサ端末の動作開始時はレベルが非常に大きな値に設定されているが，他のセンサ端末のセンサ情報を受信することにより，適切なレベルに調整されていることがわかる．しかしながら，時刻 189 秒でセンサ端末 s32 のレベルが 2 になっている．これは，提案手法では考慮されていない，電波の不安定さによるものである．本実験では図 5 - 10 に示した実験の結果から，受信電波強度に閾値を定めることにより，通信距離に制限を設けている．しかしながら，その制限は必ずしも完全なものでなく，電波状態によっては，あるセンサ端末の発した情報が非常に遠くまで届いてしまうことがある．センサ端末 s32 は，たまたま到達したレベル 1 のセンサ端末の信号により，レベルを 2 に変更している．周囲にはセンサ端末 s32 と同じかそれ以上のレベルのセンサ端末しかなく，また，レベル 1 のセンサ端末の信号は以降届かなかったため，センサ端末 s32 は刺激を受けることなく，同期を確立することができていない．さらに，センサ端末 s32 の発した電波はレベル 1 のセンサ端末に届かないため，センサ端末 s32 のセンサ

情報も収集できていない．本実験ではセンサ端末 s32 ともう一つのレベル 3 のセンサ端末 s31 は無線通信により影響を受けない距離だけ離れており，また，センサ端末 s32 に同期するレベル 4 のセンサ端末もないため，センサ端末 s32 が誤ったレベルを選んだことによる周囲への影響は観測されていない．しかしながら，センサ端末 s32 の周囲にレベル 3 のセンサ端末があった場合，そのセンサ端末は，センサ端末 s32 の不適切なタイミングのセンサ情報の発信に，刺激を受け，同期がくずれることになる．また，センサ端末 s32 の周囲にレベル 4 のセンサ端末があった場合，センサ端末 s32 がレベル 2 になることにより，レベル 4 のセンサ端末もレベル 3 となり，同じくセンサ情報が収集できなくなる，同期がくずれるなどの影響が生じる．

4.3 実環境を考慮した同期型センサ情報収集機構の改良

4.2 節で明らかにしたとおり，実環境では，無線通信の特性により，提案機構がうまく動作しない．本節では，同期型センサ情報収集の実現のため，提案機構に改良を加える．

実環境においてはパケットの衝突による再送や，衝突回避のため，無線通信の遅延が生じる．そこで本報告では，同期型センサ情報収集機構に新たな機能を追加する．センサ端末は，遅延した信号の刺激を受けないよう，状態が 1 から 0 に戻った後，しばらくの間，自らより 1 つ小さいレベルのセンサ端末からの信号を無視する．本報告における実験では，CSMA/CA におけるランダムな送信延期が最大 0.26 秒であったことや再送時間を考慮して，この期間を 0.6 秒とした．

また，図 8 や図 10 で示したように，実環境においては障害物によって電波が反射する．そのため，反射の存在しない理想的な環境では電波が届かないセンサ端末のセンサ情報を，反射の存在する環境では，受信してしまうセンサ端末があらわれる．しかしながら，このようなセンサ情報は常に届くとは限らず，通信は非対称になりやすい．偶発的な刺激を受けたセンサ端末は，レベルやタイマを誤って設定するため，センサ情報を基地局まで届けることができなくなる．そこで本報告では，受信頻度や受信電波強度でのフィルタリングにより，不安定な通信の刺激を受けないようにした．具体的には，受信頻度によるフィルタリングでは，センサ端末の状態が 3 回 1 になる間に 2 回以上受信したセンサ情報からのみ刺激を受

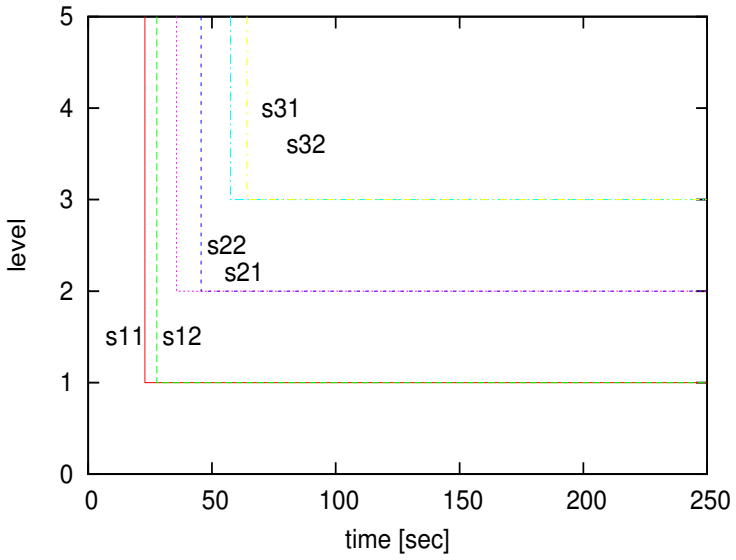

図 17: 改良機構におけるセンサ端末のレベルの変化の様子

けることとする．また，受信電波強度によるフィルタリングでは，受信電波強度が 100 以上の弱い信号からの刺激を受けないこととした．これらの閾値は同期にかかる時間と通信の不安定さによって決定される．したがって，レベルが 2 以上小さいセンサ端末からのセンサ情報を受信した場合，まず，受信電波強度が 100 以上のものを除外する．次に受信したセンサ情報のレベルを記録する．すでに同じレベルが記録されていた場合には，刺激を受ける．なお，記録したレベルはセンサ端末の状態が 3 回 1 になると初期化される．

4.4 改良システムの実験評価

4.4.1 基本動作の検証

改良手法を実装したシステムにおける実験結果を図 17 および図 18 に示す．図 17 にセンサ端末のレベルの変化の様子を示す．2 回以上の刺激を受けることによりレベルを変化させるため，フィルタ追加前と比較して，レベルの調整に時間がかかっているが，正しくレベルを認識している．また，図 18 より，時刻 78 秒でセンサ端末 s11，s12 はビーコン信号より，約 2 秒早く，センサ端末 s21，s22 は約 4 秒早く，センサ端末 s31，s32 は約 6 秒早くセン

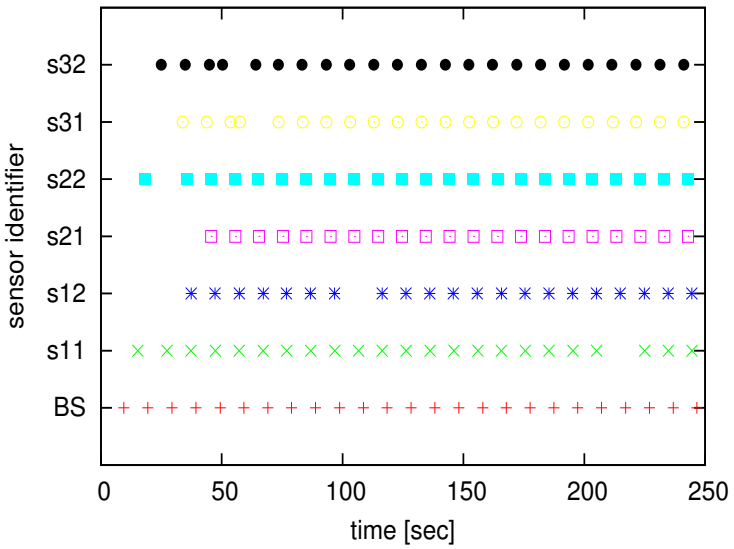

図 18: 改良機構におけるセンサ端末の情報発信のタイミング

サ情報を発信しており，センサネットワーク全体が同期している．無線通信の輻輳によって遅延したセンサ情報を無視するとともに，安定した通信でのセンサ情報によってのみ刺激を受けることにより，同期が保たれている．

4.4.2 センサ端末の動的な追加と削除

図 19 にセンサ端末 s11 から s13 がレベル 1，センサ端末 s21 から s24 がレベル 2，センサ端末 s31 から s34 がレベル 3，センサ端末 s41 から s43 がレベル 4 の状態でセンサネットワーク全体の同期が確立された後，時刻 845 秒にセンサ端末 s13，855 秒にセンサ端末 s12，865 秒にセンサ端末 s11 を削除した時の様子を示す．レベル 1 のセンサ端末が取り除かれることにより，基地局からのビーコン信号が届かなくなるが，全体の同期は維持されている．

一方，図 20 にセンサ端末 s14 がレベル 1，センサ端末 s21 から s23 がレベル 2，センサ端末 s31 から s33 がレベル 3，センサ端末 s41 がレベル 4 の状態でセンサネットワーク全体が同期した後に，時刻 300 秒にセンサ端末 s11 から s13 の 3 台のセンサ端末を一番内側の円周上に追加した結果を示す．新たに導入されたセンサ端末のレベルは，受信頻度による

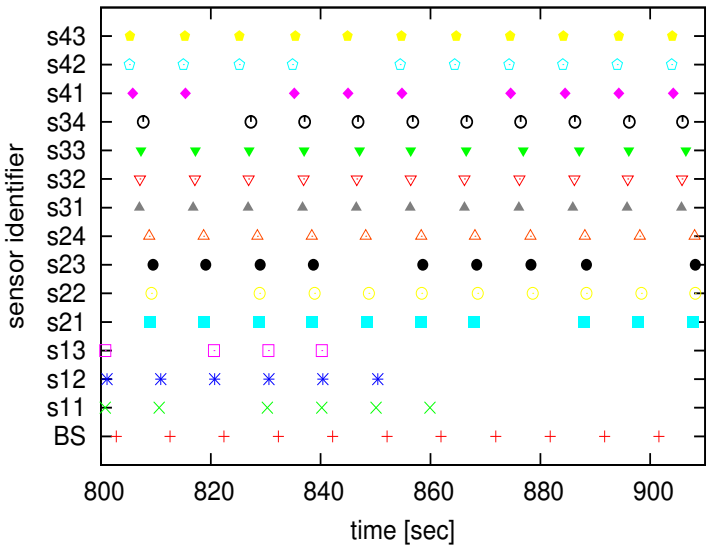

図 19: センサ端末除去後のセンサ端末の情報発信のタイミング

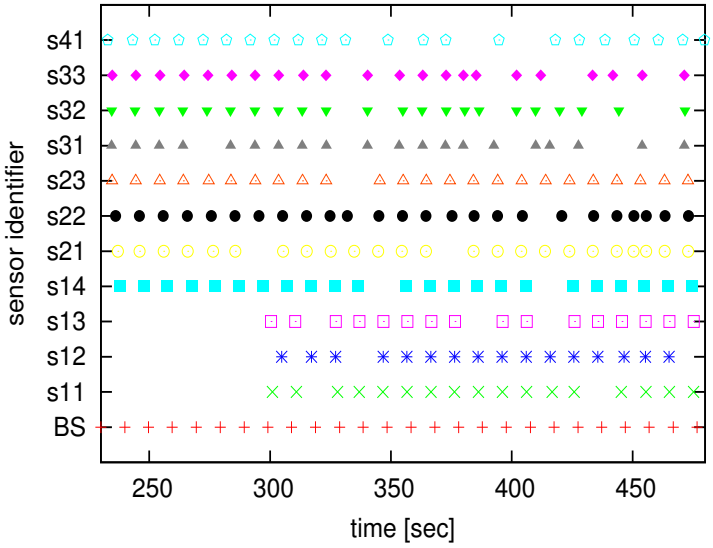

図 20: センサ端末追加後のセンサ端末の情報発信のタイミング

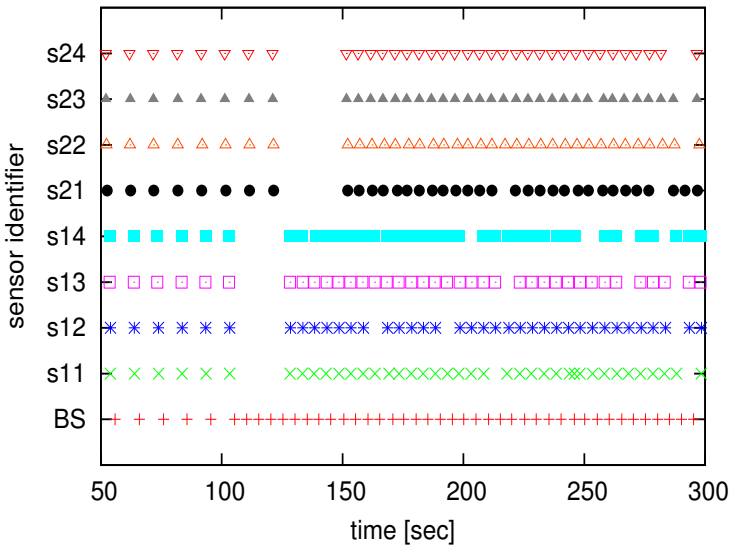

図 21: 情報収集頻度の変更

フィルタのために 2 度のビーコン信号によって適切にレベル 1 に調整される．しかしながら，タイマの同期までには時間がかかるため，周囲のレベル 2 のセンサ端末は新しいセンサ端末による誤ったタイミングの刺激を受ける．刺激により，レベル 2 のセンサ端末の同期が乱れ，さらにより大きいレベルのセンサ端末へ誤った刺激が伝播していく．その結果，センサネットワーク全体の同期が崩れる．センサ端末 s11 から s13 がビーコン信号を繰り返し受信し，タイマの同期が確立されると，センサ端末 s21 から s41 は次第に適切なタイミングで刺激を受けるようになり，時刻 470 秒でセンサネットワーク全体の同期が再度確立されている．

4.4.3 センサ情報収集頻度の変更

センサ情報収集頻度に対する柔軟性を検証するための実験を行った．センサ端末 s11 から s14 がレベル 1，センサ端末 s21 から s24 がレベル 2 で全体が同期した後，ビーコン信号の発信周期を半分にした．実験結果を図 21 に示す．図より，いったんセンサ情報の収集が途切れるが，新たな周期でのセンサ端末間の刺激により，時刻 155 秒において倍の頻度でのセンサ情報収集が達成されていることが分かる．

ただし，文献 [5] で述べられているように，周期変更に対応するためにはセンサ端末の同期機構に変更を加える必要がある．センサ端末 S_i のセンサ情報発信とセンサ端末 S_j のタイマが同期しているものとする．したがって，レベル $l_i = l_j - 1$ である．ビーコン信号の周期変更により，センサ端末 S_i がこれまでの半分の周期でセンサ情報を発信するようになったとする．センサ端末 S_i がセンサ情報を発信した時，これを受信したセンサ端末 S_j の位相 ϕ_j は 0.5 から 1.0 に変位する．センサ端末 S_j は位相 $1 - \delta_j$ にセンサ情報を発信するため，オフセット δ_j が 0.5 より小さいと，刺激による位相の変位によりセンサ情報を発信できない．一方，オフセット δ_j が 0.5 より大きい場合には，情報発信のタイミングが 0.5 遅れてしまう．そこで，情報収集の周期変更後，複数回同期が確認されたときに，センサ端末 S_j はオフセットを $1 - \phi_j - \delta_j$ に変更する．実験では 4 回としている．周期が半分になった場合には，オフセットは $0.5 + \delta_j$ となり，センサ端末 S_j はセンサ端末 S_i の情報発信より δ_j だけ早くセンサ情報を発信するようになるため，新しい周期でのセンサ情報収集が達成される．

5　おわりに

本報告では，我々の研究グループの提案する同期型センサ情報収集機構の実環境における有効性，実用性を検証するため，市販の無線センサ端末 MOTE への実装と実証実験を行った．その結果，提案機構では考慮されていなかった無線通信の輻輳による信号受信の遅れや，通信の不安定さのため，同期が容易に乱れることを明らかにした，本報告ではそれらの問題を解決するため，信号受信頻度と受信電波強度によるフィルタリング機構を提案し，実験により，実環境において，同期型のセンサ情報収集が行えることを示した．本報告の機構は拡張性，適応性，柔軟性，対障害性にすぐれ，センサネットワークより効率よく定期的にセンサ情報を収集することができる．

本報告における実験は，電波の反射の比較的少ない場所で行われており，障害物が多く，電波環境の大きく変化する，屋内での実験評価は行っていない．電波の干渉，輻輳の発生しやすい環境では，不安定な無線通信や遅延により，本報告で提案した改良機構を用いても，良好なセンサ情報収集が達成できない可能性がある．そこで，そのような環境での実験を行い，同期型センサ情報収集機構のさらなる改良を行いたい．また，本報告では，センサ端末の追加や除去への適応性，センサ情報収集頻度への変化への柔軟性については実験により確認されたが，拡張性，対故障性や電力効率のよさについては検証されていない．今後はこれらについても実験評価を行いたい．

謝辞

報告を終えるにあたり，御指導，御教授を頂いた大阪大学 サイバーメディアセンター 先端ネットワーク環境研究部門の村田正幸教授に深く感謝致します．また，本報告において終始直接御指導頂いた大阪大学 情報科学研究科 情報ネットワーク学専攻マルチメディアネットワーク講座 (宮原研究室) の若宮直紀助教授に深く感謝致します．

並びに，日頃から適切な助言を頂いた宮原秀夫教授，大阪府立看護大学の菅野雅嗣助教授，大阪大学サイバーメディアセンターの馬場健一助教授，今瀬研究室の大崎博之助教授，村田研究室の長谷川剛助教授，大阪市立大学の阿多信吾講師，大阪大学経済学部の荒川伸一助手に心から感謝致します．

最後に，本報告において御協力頂いた村田研究室および宮原研究室の皆様方に心からお礼申し上げます．

付録

MoteMsg.h

パケットのフォーマット.

```
enum {
  BUFFER_SIZE = 20
};

typedef struct dataTmp
{
  uint16_t sourceMoteID;
  uint8_t  levelNum;
  uint16_t data;
  uint16_t dataTime;
} dataTmp;

struct MoteMsg
{
  uint16_t sourceMoteID;
  uint8_t  levelNum;
  uint16_t data;
  uint16_t dataTime;
  uint8_t delta;
  uint8_t epsilon;
  uint8_t etc1;
  uint8_t etc2;
  uint8_t etc3;
  uint16_t etc4;
  struct dataTmp receivedData[BUFFER_SIZE];
};

enum {
  AM_MOTEMSG = 10
};
```

synchM.nc

センサ端末のプログラム.

```
includes MoteMsg;

module synch
{
  provides interface StdControl;
  uses {
    command result_t PowerEnable();
    command result_t PowerDisable();
    interface StdControl as TimerControl;
    interface Timer as Timer0;
```

```
    interface Leds;
    interface StdControl as SensorControl;
    interface ADC;
    interface StdControl as CommControl;
    interface SendMsg as DataMsg;
    interface ReceiveMsg;
    interface SysTime;
    interface CC1000Control;
  }
}
implementation
{
  TOS_Msg msg[2];
  uint8_t currentMsg;

  uint32_t beforeTime;
  uint32_t *currentTime;
  uint16_t cTime;

  double delta;
  double epsilon;
  double b;
  double state;
  double state2;
  double phase;
  double phase2;
  double ignore;

  uint8_t etc1;
  uint8_t etc2;
  uint8_t etc3;
  uint16_t etc4;

  uint8_t sleepCount;
  uint8_t senseFlag;
  uint8_t freqFlag;
  uint8_t ignoreFlag;
  uint8_t stimulatedLv;
  uint8_t myLevel;
  uint8_t bufferedNum;
  uint16_t radioStrength;
  uint16_t strongRS;
  struct dataTmp bufferedData[BUFFER_SIZE];

  /*for mica2dot*/
  /*uint16_t getTime(){
    uint16_t retTime;

    call SysTime.get(currentTime);
    if (*currentTime > beforeTime){
      retTime = (*currentTime/400000) + cTime;
    } else {
```

```
    cTime += 10737;
    retTime = (*currentTime/400000) + cTime;
  }
  beforeTime = *currentTime;
  return retTime;
}*/

/*for mica2*/
uint16_t getTime(){
  uint16_t retTime;

  call SysTime.get(currentTime);
  if (*currentTime > beforeTime){
    retTime = (*currentTime/737280) + cTime;
  } else {
    cTime += 5825;
    retTime = (*currentTime/737280) + cTime;
  }
  beforeTime = *currentTime;
  return retTime;
}

void pInit(){
  currentMsg = 0;
  b = 3;
  phase = 0;
  state = 0;
  phase2 = 0;
  state2 = 0;
  ignore = 0;
  epsilon = 0.3;
  delta = 0.2;
  sleepCount = 0;
  senseFlag = 0;
  freqFlag = 0;
  ignoreFlag = 0;
  myLevel = 255;
  bufferedNum = 0;
  stimulatedLv = myLevel;
  radioStrength = 120;
  strongRS = 100;
  etc1 = 0;
  etc2 = 0;
  etc3 = 0;
  etc4 = 0;
}

command result_t StdControl.init() {
  call Leds.init();
  call Leds.yellowOff(); call Leds.redOff(); call Leds.greenOff();

  call SensorControl.init();
```

```
  call CommControl.init();
  call TimerControl.init();
  call SysTime.init();

  call PowerDisable();

  call CC1000Control.SetRFPower(1);

  pInit();

  cTime = 0;
  call SysTime.get(currentTime);
  beforeTime = *currentTime;

  return SUCCESS;
}

command result_t StdControl.start() {
  call SensorControl.start();
  call CommControl.start();
  call CC1000Control.SetRFPower(1);
  call TimerControl.start();
  call Timer0.start(TIMER_REPEAT, 100);
  return SUCCESS;
}

command result_t StdControl.stop() {
  call SensorControl.stop();

  call CommControl.stop();

  call Timer0.stop();
  return SUCCESS;
}

void msgClear(){
  uint8_t j;
  for (j=0;j<TOSH_DATA_LENGTH;j++) msg[currentMsg].data[j] = 0;
}

event result_t ADC.dataReady(uint16_t data) {
  uint8_t i;

  struct MoteMsg *pack = (struct MoteMsg *)msg[currentMsg].data;

  pack->data = data;
  pack->levelNum = myLevel;
  pack->sourceMoteID = TOS_LOCAL_ADDRESS;
  pack->dataTime = getTime();
  pack->delta = (uint8_t)(delta*100);
  pack->epsilon = (uint8_t)(epsilon*100);
  pack->etc1 = etc1;
```

```
  pack->etc2 = etc2;
  pack->etc3 = etc3;
  pack->etc4 = etc4;
  for(i=0;i < BUFFER_SIZE;i++){
    pack->receivedData[i].Num = i+1;
  }
  for(i=0;i < bufferedNum;i++){
    pack->receivedData[i].sourceMoteID = bufferedData[i].sourceMoteID;
    pack->receivedData[i].levelNum = bufferedData[i].levelNum;
    pack->receivedData[i].data = bufferedData[i].data;
    pack->receivedData[i].dataTime = bufferedData[i].dataTime;
  }

  if (call DataMsg.send(TOS_BCAST_ADDR, sizeof(struct MoteMsg),
                        &msg[currentMsg]))
  {
    call Leds.yellowToggle();
  }

  phase2 = 0;
  state2 = 0;
  bufferedNum = 0;
  senseFlag = 0;
  freqFlag = 0;

  return SUCCESS;
}

event result_t DataMsg.sendDone(TOS_MsgPtr sent, result_t success) {
  return SUCCESS;
}

double getState(double phi){
  return log(1+(exp(b)-1)*phi)/b;
}

double getPhase(double x){
  return (exp(b*x)-1)/(exp(b)-1);
}

double stimulus(double x,double phi, double e, uint8_t f){
  double ret;
  if (f == 1 && phi < 0.06){
    ret = x;
    sleepCount++;
  } else {
    if (f == 1 && phi > 0.94) sleepCount++;
      ret = x + e;
      if (ret > 0.9999999) ret = 1;
    }
  return ret;
}
```

```
double addDelta(double phi, double sig){
  double ret;
  ret = phi + sig;
  if (ret > 1) ret -= 1;
  return ret;
}

void changeState2(double phi){
  state2 = getState(phi);
  if (state2 > 0.9999999){
    msgClear();
    call ADC.getData();
  }
}

void sleepmode(double d){
  uint8_t span;

  call Timer0.stop();
  span = (uint8_t)(1 -2*d);
  call PowerEnable();
  call Timer0.start(TIMER_ONE_SHOT, span*100);
  call PowerDisable();
  call Timer0.start(TIMER_REPEAT, 100);
  span = (uint8_t)(1 - d);
  phase = span;
  state = getState(phase);
}

void changeState(double phi){
  double p;
  state = getState(phi);
  p = phi;
  if (p <= delta) p = p + 1;
  if (p > (ignore + delta)) ignoreFlag = 0;
  if (state > 0.9999999){
    senseFlag++;
    if (senseFlag > 2) {
      senseFlag = 0;
      stimulatedLv = myLevel;
    }
    phase = 0;
    state = 0;
    if (sleepCount >= 10) {
      sleepCount = 0;
      sleepmode(delta);
    }
  }
}

event result_t Timer0.fired() {
```

```
  if (phase > 0.9999999) phase = 1;
  changeState(phase);
  if (phase2 > 0.9999999) phase2 = 1;
  changeState2(phase2);
  phase += 0.01;
  phase2 += 0.01;
  return SUCCESS;
}

uint16_t timeSynch(uint16_t baseT,uint16_t curT,uint16_t pasT){
  uint16_t tmp;
  if (curT > pasT){
    tmp = curT-pasT;
  } else tmp = 0;
  if (baseT > tmp){
    tmp = baseT - tmp;
  } else tmp = 0;
  return tmp;
}

event TOS_MsgPtr ReceiveMsg.receive(TOS_MsgPtr pmsg) {
  uint8_t i;
  uint8_t allowStim;
  uint8_t stm;
  uint16_t rTime;

  if (pmsg->strength < radioStrength){
    struct MoteMsg *rmsg = (struct MoteMsg *)pmsg->data;
    if (rmsg->levelNum < myLevel){
      if (rmsg->levelNum == myLevel-1||
         (rmsg->levelNum < myLevel-1&&pmsg->strength < strongRS)){
        call Leds.redToggle();
          if (ignoreFlag == 0 ||
             (ignoreFlag == 1 &&
             (phase <= ignore ||phase >= addDelta(ignore,delta)))){
            if (freqFlag >= 4) {
              delta = (1 - phase +delta);
              freqFlag = 0;
            }
            if (rmsg->levelNum < myLevel-1){
              if (stimulatedLv > rmsg->levelNum){
                stimulatedLv = rmsg->levelNum;
                senseFlag = 0;
                allowStim = 0;
              } else if (stimulatedLv == rmsg->levelNum) {
                allowStim = 1;
              } else {
                allowStim = 0;
              }
              stm = 0;
            } else {
              allowStim = 1;
```

```
            stm = 1;
          }
        if (allowStim == 1){
          freqFlag++;
          etc4 = pmsg->strength;
          myLevel = rmsg->levelNum+1;
          state = getState(phase);
          etc2 = (uint8_t)(phase*100);
          state = stimulus(state,phase,epsilon,stm);
          ignore = state;
          ignoreFlag = 1;
          etc1 = (uint8_t)(rmsg->sourceMoteID);
          if (phase <= 0.94 && phase > 0.06) {
            sleepCount = 0;
          }
          phase = getPhase(state);
          etc3 = (uint8_t)(phase*100);
          phase2 = addDelta(phase,delta);
          changeState(phase);
          changeState2(phase2);
        }
      }
    }
  } else if (rmsg->levelNum == (myLevel+1)){
    call Leds.greenToggle();
    if (bufferedNum < BUFFER_SIZE){
      bufferedData[bufferedNum].sourceMoteID = rmsg->sourceMoteID;
      bufferedData[bufferedNum].levelNum = rmsg->levelNum;
      bufferedData[bufferedNum].data = rmsg->data;
      rTime = getTime();
      bufferedData[bufferedNum].dataTime = rTime;
      bufferedNum++;
      for (i=0;i < BUFFER_SIZE&&bufferedNum
           <BUFFER_SIZE&&rmsg->receivedData[i].sourceMoteID!=0;i++){
        bufferedData[bufferedNum].sourceMoteID
          = rmsg->receivedData[i].sourceMoteID;
        bufferedData[bufferedNum].levelNum = rmsg->receivedData[i].levelNum;
        bufferedData[bufferedNum].data = rmsg->receivedData[i].data;
        bufferedData[bufferedNum].dataTime
          = timeSynch(rTime,rmsg->dataTime,rmsg->receivedData[i].dataTime);
        bufferedNum++;
      }
    }
  } else {
    //call Leds.redToggle();
  }
} else if (myLevel == 255) {
  etc4 = pmsg->strength;
}
return pmsg;
}
```

```
}
```

synch.nc

synchM.nc 内の関数と無線通信やタイマなどのライブラリの対応関係を記述したプログラム.

```
includes MoteMsg;

configuration synch {}
implementation
{
  components Main, synchM, TimerC, LedsC, Temp, \
  GenericComm as Comm, SysTimeC, CC1000ControlM, HPLPowerManagementM;

  Main.StdControl -> synchM;

  synchM.TimerControl -> TimerC.StdControl;
  synchM.Timer0 -> TimerC.Timer[unique("Timer")];
  synchM.PowerEnable -> HPLPowerManagementM.Enable;
  synchM.PowerDisable -> HPLPowerManagementM.Disable;
  synchM.Leds -> LedsC;
  synchM.SensorControl -> Temp;
  synchM.ADC -> Temp;
  synchM.CommControl -> Comm;
  synchM.ReceiveMsg -> Comm.ReceiveMsg[AM_MOTEMSG];
  synchM.DataMsg -> Comm.SendMsg[AM_MOTEMSG];
  synchM.SysTime -> SysTimeC;
  synchM.CC1000Control -> CC1000ControlM;
}
```

参考文献

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks (Elsevier) journal*, vol. 38, pp. 393–422, Mar. 2002.

[2] W. R. Heinzelman, A. Chandraksan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the IEEE Wireless Communication and Networking Conference*, pp. 3005–3014, Jan. 2000.

[3] S.Lindsey, C. Raghavendra, and K. Sivalingam, "Data gathering in sensor networks using the energy*delay metric," in *Proceedings of the 15th International Parallel & Distributed Processing Symposium (IPDPS-01)*, pp. 2001–2008, Apr. 2001.

[4] K. Dasgupta, K. Kalpakis, and P. Namjoshi, "An efficient clustering-based heuristic for data gathering and aggregation in sensor networks," in *Proceegings of the IEEE Wireless Communication and Networking Conference (WCNC)*, pp. 16–20, Mar. 2003.

[5] N. Wakamiya and M. Murata, "Scalable and robust scheme for data fusion in sensor networks," in *Proceedings of International Workshop on Biologically Inspired Approaches to Advanced Information Technology (Bio-ADIT)*, pp. 112–127, Jan. 2004.

[6] R. E. Mirollo and S. H. Strogatz, "Synchronization of pulse-coupled biological oscillators," *Society for Industrial and Applied Mathematics Journal on Applied Mathematics*, vol. 50, pp. 1645–1662, Dec. 1990.

[7] X. Guardiola, A. Diaz-Guilera, M. Llas, and C. Perez, "Synchronization, diversity, and topology of networks of integrate and fire oscillators," *The America Physical Society Physical Review E*, vol. 62, pp. 5565–5569, Oct. 2000.

[8] M. B. H. Rhouma and H. Frigui, "Self-organization of pulse-coupled oscillators with application to clustering," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 23, pp. 180–195, Feb. 2001.

[9] I. Wokoma, I. Liabotis, O. Prnjat, L. Sacks, and I. Marshall, "A weakly coupled adaptive gossip protocol for application level active networks," in *Proceedings of IEEE 3rd International Workshop on Policies for Distributed System and Networks - Policy 2002*, pp. 244–247, June 2002.

[10] "MOTE." available at `URL: <http://www.xbow.com/Products/Wireless_Sensor_Networks.htm>`.

[11] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," *IEEE Design & Test of Computer*, vol. 18, pp. 62–74, Mar. 2001.

[12] J. Aslam, Q. Li, and D. Rus, "Three power-aware routing algorithms for sensor networks," *Wireless Communications and Mobile Computing*, vol. 3, pp. 187–208, Mar. 2003.

[13] C. Schurgers, V. Tsiatsis, S. Ganeriwal and M. Srivastava, "Topology management for sensor networks: Exploiting latency and density," in *Proceedings of Symposium on Mobile Ad Hoc Networking & Computing (MoviHoc'02)*, pp. 135–145, June 2002.

[14] "TinyOS." available at `URL: <http://webs.cs.berkeley.edu/tos/>`.