センサネットワーク管理システムの設計

牧村 和慶[†]　間 博人[§]　斉藤 裕樹[‡]　戸辺 義人[‡]　徳田 英幸[§ ∮]

通信デバイスの小型化，軽量化，技術性能向上が進展し，無線通信機能を有する超小型センサの開発が進んでいる．近い将来，数十，数百といった数のセンサが街中や屋内などに設置され，相互にネットワークを構築する環境が考えられる．このようなセンサ同士がネットワークを組む，センサネットワークにおいて，センサノードの数が膨大になると，データ収集という計測技術のみならず，ノード自体の運用・管理面に工夫が必要となってくる．こうした背景から，我々はセンサネットワークを対象としてネットワーク管理システム SNAC を設計した．SNAC はノードの故障に伴うライフサイクル管理，実世界に配置される特殊性を考慮する．本稿では，実世界でセンサネットワークを活用するため管理システム SNAC (Sensor Networks Active Control) の概要とそのシステム構成について述べる．

Design of a Network Management System for Sensor Networks

Kazunori Makimura[†]　Hiroto Aida[§]　Hiroki Saito[‡]

Yoshito Tobe[‡]　Hideyuki Tokuda[§ ∮]

Development of tiny sensors equipped with wireless communication functionality is remarkable with the advancement of small, light, and high-performance wireless devices. In the near future, we will see tens or hundreds of these nodes deployed in town and inside buildings. When the number of these nodes grows, we need to consider the management aspect of the nodes as well as data processing. Based on such a background, we have designed SNAC, a sensor network management system. In SNAC, we have taken the life cycle management of the nodes and the peculiarity of deployment into the physical world into account. In this paper we describe the overview and the design of SNAC.

1. まえがき

近年，無線通信技術および半導体技術の発展に伴い，携帯電話に代表される無線通信デバイスが小型化，軽量化，省電力，高性能化が進んできている．特に小型化の進展は著しく，様々な「物」が無線通信機能を備える潜在的な力を生み出し，小型無線通信がユビキタスコンピューティングを支えるものとして期待される．一方，実世界から情報を収集するデバイスとしてセンサデバイスの小型化も進んでおり，センサ自体に無線通信機能を持たせることが可能となってきた[10]．このようなセンサを通信ノードとして動作させ，相互にマルチホップ転送を行うセンサネットワークの研究開発が進んでいる[3),6),12)]．UC Berkeley で開発された Mica Mote[6)]がテストプラットフォームとして商用化されたこともあり，現在，センサネットワークが「ネットワーク」の研究として，広がりを見せている．

センサネットワークは，環境モニタリング，構造劣化診断，農業プラント等への応用が考えられている．応用例として，環境モニタリングを取り上げると，2次元あるいは3次元の広範囲に渡る領域からセンサ情報を取得するため，数十，数百という数の膨大な数のセンサノードが実世界上に配置されることを想定しなければならない．そのとき，システム管理上いくつか考えるべき点がある．まず，こうしたセンサノードはバッテリ駆動であることが多いため，センサの交換作業を含めたライフサイクル管理が必要となる．次に，膨大な数のノードを一元的に管理することも考えなければならない．また，実世界上に配置されるという特質を利用すること考慮しなければならない．このようなセンサネットワーク特有の「ネットワーク管理」の枠組みとして，我々は SNAC (Sensor

† 東京電機大学 工学部 情報通信工学科
‡ 東京電機大学 工学部 情報メディア学科
§ 慶應義塾大学大学院 政策・メディア研究科
∮ 慶應義塾大学 環境情報学部
† Department of Information and Communication Engineering, Tokyo Denki University
‡ Department of Information Systems and Multimedia Design, Tokyo Denki University
§ Graduate School of Media and Governance, Keio University
∮ Faculty of Environmental Information, Keio University

Networks Active Control)[8] を提案し，設計した．SNAC は，1) 協調異状検出，2) ノード物理位置階層化管理，3) 階層化を利用した通信，4) 視覚化管理を特徴とする．本稿では，SNAC の概要，設計，Mica Mote を用いたプロトタイプについて述べる．

本稿の構成は以下の通りとする．第 2 章で実世界におけるセンサネットワークの課題について説明する．第 3 章，第 4 章で，各々，SNAC の設計指針，詳細設計を述べる．第5章で，Mica Mote 上の SNAC 実装を説明する．第 6 章で関連研究について触れ，最後に結論および今後の課題を述べる．

2. センサネットワーク管理の課題

センサネットワークにおけるシステム管理という面から，次の 3 点を考慮する必要がある．

- 信頼性
- 柔軟性
- スケーラビリティ

信頼性として，センサネットワークがシステムとして正常に動作しているか否か特定する能力が要求される．柔軟性として，ノードの配置場所およびネットワークの構成に極力依存しないことが要求される．最後にノード数に依存しない管理システムが望まれる．

これらの点は，センサネットワークに限らず，いかなるネットワークシステムにも共通する課題であり，管理対象がセンサネットワークであることに起因する課題もある．センサネットワークで用いられるノードはバッテリ駆動であることが多く，しかも数が膨大であることが多く，「数の多さ」に対処することが必要となる．また，実世界に設置されることから，物理位置に依存した管理が必要となる．これはオフィス用の LAN とは対称的である．オフィス用 LAN においては，論理的にネットワークが構成されれば十分であり，ホストやルータの物理位置が意識されることは少ない．それに対して，センサネットワークにおいては，各ノードの位置が重要であるので，位置を加味した管理が必要とされる．仮想世界と物理世界のマッピングは重要となる．

一般的なネットワーク管理プロトコルとして SNMP（Simple Network Management Protocol）がある．SNMP は MIB と呼ばれる管理データベースで，マネージャがアクセスできるあらゆる種類の情報や統計的情報が MIB として定義されている．MIB に問い合わせることで，管理情報の応答を受ける．SNMP をセンサネットワークに適用した場合，1 対1の管理しか実装されていないために，上記した「数の多さ」「物理位置依存」に対処することが難しい．例えば，個別ノードではなく，位置の範囲で管理対象を指定することが不可能である．

3. SNAC 設計指針

本章では，第2章で述べたセンサネットワーク管理の課題を解決する，ネットワーク管理システム SNAC の設計指針を述べる．SNAC では，センサネットワークの特殊性を考慮した管理を実現することを目標とし，以下の 3 つを設計指針とする．

- システム故障時間の短縮
- 膨大数ノードの一括管理
- メンテナンス利便性の向上

3.1 システム故障における回復時間の短縮

センサノードの故障は許容するとしても，復旧に要する時間を短くしたい．そのため，センシングするデータ，電池残存量から実際にノードが故障する前に故障を検出する．また，故障したノード自身が故障を広告することは難しいので，周囲のノードの協調動作により故障を検出する．

3.2 膨大数ノードの一括管理

センサネットワークでは，既存の TCP/IP ネットワークのノードとは異なり，同一ドメイン内に数多くのセンサノードを現実世界に配置してネットワークを構築する．そのため，個々のノードよりも，ノードのまとまりを形成し，一括して管理する仕組みを設ける．その際に，ノードが置かれた物理的な位置を基準に階層的なグループを構築し，物理的位置に対応した領域全体に対して送信する，あるいは領域全体を代表するデータを引き出すことが可能となることを目指す．

3.3 メンテナンス利便性の向上

実世界に配置されたノードを意識することなく，管理 PC 上で管理を行うのが理想である．しかし，ノードの異常・故障が発生した際，交換・修理などを行うのは管理者である人である．そこでノードを実際に取り替える場合において，人が目的のノードが存在する場所まで行き，ノードを交換すること等を想定し，ノード自体を「指示器」として動作させ，視覚に訴える仕組みを提供することを目指す．

4. SNAC の設計

本章では, SNAC のシステムと設計の詳細を述べる. 特に, 階層的アドレス割当てと, それに伴う Areacast を詳述する.

4.1 システム概要

SNAC は, センサネットワーク全体を管理するノード NMN (Network Management Node) と, 管理される個々のセンサノード OSN (Operating Sensor Node) から成る. SNAC では,

- 協調異状検出
- 階層的アドレス割当て
- Areacast
- 視覚化

を実現する．以下では，上記４つの機能について述べる.

4.2 協調異状検出

センサノードが保持する電池電源残存量を監視し, 異状があれば, ノード自身が警報を発するシステムを想定する. このとき, まず, ノード電源が枯渇した状態では, 自ら異状を発信することが不可能となる点に注意しなければならない. そこで, 表1に示す通り, 電源残存量の「危険閾値」を定め, 電源残存量が危険閾値を下回れば危険状態として, 警報発信を可能とする.

表1. 電源残存量に応じたノードの電源状態

電源状態	電源残存量
正常	危険閾値以上
危険	危険閾値未満
異状	なし

次に, 電源残存量はあるものの, 無線送信能力が低下する場合に注意しなければならない. Mice Mote を例として考える. 図 1 は, 日本で使用される Mica Mote 1 を用いて, 電源電圧と送信到達距離を実測したものである. Mica Mote 1 で使用される 3V の電池に対し, 電源電圧が 1.5 V を下回ると送信が難しくなってくることがわかる. したがって, 危険閾値を決定する際には, ノード自体の動作持続可能性と共に, 他ノードへの送信機能も加味しなければならない. 他ノードへの送信到達可能性を考慮すると, ノード自体が異状警報を発するのではなく, 他ノードが通信の断絶を元に, 異状を検出することも可能となる. そこで, SNAC では他ノードによる異状検出も積極的に採用する.

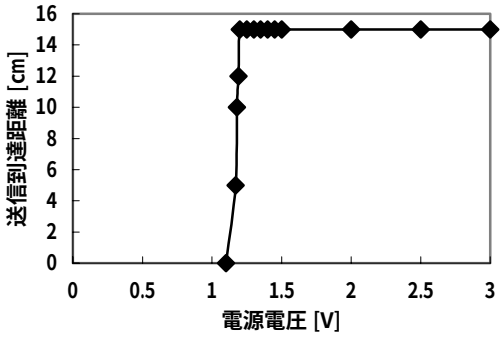

図 1. 電源電圧と送信到達距離

他ノードへの送信到達可能性を考慮すると, ノード自体が異状警報を発するのではなく, 他ノードが通信の断絶を元に, 異状を検出することも可能となる. そこで, SNAC では他ノードによる異状検出も積極的に採用する.

4.3 アドレス割当て

本節では, 領域で階層化されたアドレス割当てについて述べる. ノード群を実空間上のある領域をまとまりとして管理し, 管理する領域の大きさを可変にすることで, 管理のスケーラビリティを向上する. そのため, SNAC ではノードの配置位置を領域として, 階層的にアドレスを割り当てる. ノードへ動的にアドレスを付与することも SNAC の枠組みでは必要となるが, 本稿では触れないこととする.

図 2 は領域設定とアドレスの一例である.

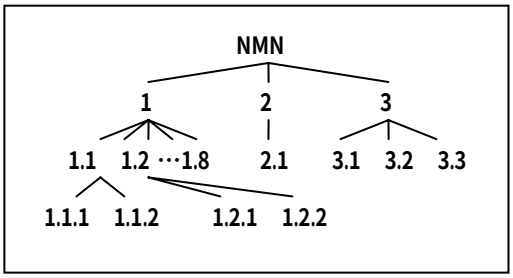

図 2. 領域階層化とアドレス

領域階層化とアドレス

アドレスは 8 ビット(1～255)で 1 階層の識別子とし, 可変数の階層を持たせて表現することとする. NMN が

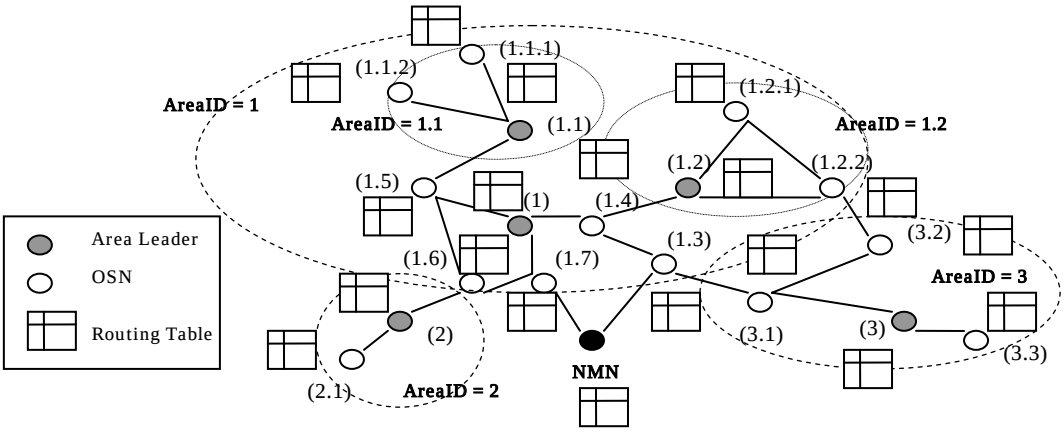

図 3. 領域とアドレス割当て

root となり, 上位の層から 1～255, 1.1～1.255, 255.1～255.255, 1.1.1～255.255.255, … とする. 各領域の代表を Area Leader と呼ぶ. 領域において, Area Leader 以外のノードから見て Area Leader は自ノードの PARENT とし, Area Leader から見ると領域内の他ノードは CHILD とする. PARENT は, CHILDのデータを集約, CHILD への経路を把握するノードである. 図 3 のノード(1.1)は(1)の CHILD であり, (1.1.1)と(1.1.2)の CHILD を有する PARENT である. また, (1.1)は(1.5)と兄弟関係, BROTHER であるとする.

4.4 Areacast

Areacast は, NMN と指定の領域との通信を行う機能であり, 1 対多の通信を実現する. 以下では, 目的, 動作概要および, Areacast を実現するためのルーティング方法とメッセージフォーマットについて述べる.

目的

Areacast は, ある領域内のすべてノードへの通信を目的とする. 例えば, ある特定の領域に配置されたノードすべてにクエリを送信したり, 領域内の集約された環境情報を取得したり, 広い範囲を管理することが考えられる. LAR[7]に見られるように, 位置情報をルーティングが提案されている. SNAC では, 位置情報を用いずに, ノードのアドレス割当てを工夫することで領域を指定した通信を実現する. 第 3 章で述べたように膨大数のノードが実世界で配置されることが考えられるため, トラフィックの抑制, 経路制御の効率化を図る.

動作概要

Areacast は NMN からある領域全体の通信を実現する. このとき領域は任意の階層を指定できるものとする. AODV[9], OLSR[1]の手法を基本として, リンクステート型のルーティングを行う. 各ノードはルーティングテーブルを保持する. ルーティングテーブルはフラッディングによる routing request (経路要求), routing replay (経路応答) パケットによって随時更新される. 宛先ノード, 宛先領域へパケットが到着すると送信元へその応答を返し, コネクションを確立させる. Areacast は, AODV ベースのルーティング手法を用いるが, アドレスとして宛先に領域名を記述する.

ルーティングとルーティングテーブル

各ノードは, 経路要求パケット, 経路応答パケットを受信した際, 最新の経路に更新する. パケットには自分のシーケンス番号を挿入し, 同じ送信先から同じパケットを受信した際にシーケンス番号を検査し, 最新のものであるか否かを判断する.

表 2, 表 3 は図 3 におけるノード(1), ノード(1.3)のルーティングテーブルである. ノード(1)は領域 1 の Area Leader であるため, ノード(1)領域内の 1 階層下の CHILD ノードへの経路をすべて保持する. 一方, ノード(1.3)は Area Leader ではないので, 領域 1 の他のすべてのノードへの経路を必ずしも保持する必要はない.

表 2. ノード(1)のルーティングテーブル

Destination Area/Node	Next Node
NMN	1.7
1.1	1.5
1.2	1.4
1.3	1.4
1.4	direct
1.5	direct
1.6	1.5
1.7	direct

表 3. ノード(1.3)のルーティングテーブル

Destination Area/Node	Next Node
NMN	direct
1	1.4
1.4	direct
3	3.1

ルーティングテーブルの基本原則として, Destination Area には PARENT ノードの Area Leader のエントリは必ず含めることとする. Area Leader は配置場所に応じて適切な場所へ配置する. 例えば, 膨大な数の CHILD ノードを持つ場合, 配置場所の中央へ配置する. CHILD ノードの経路は一定間隔に領域内に送信される経路確認(応答)パケットによって常に最新の経路に更新される. Area Leader は領域内のデフォルトルータの役目を負う. あるノードから領域内全体にパケットを送信する際, 各ノードのルーティングテーブルに基づき, 常に Area Leader に向けてパケットを転送する. 領域内全体へ送信するパケットを受信した際, 保持する経路に基づいて CHILD ノードへ向けて送信する. ただし, そのパケットを受信するまでに通った経路は送信パケットに付加され, 領域内の送信されていないノードへ向けて送信される. 図 5, 図 6 はある領域への Areacast を行った場合の, 各ノードの経路要求フローと経路応答フローを示す.

メッセージ

ある領域全体にパケットを送信する際の経路要求パケットフォーマットを図 7 に示す. 各ノードで同一の宛先アドレスのパケットを重複して受信することを防ぐために, 自分の ID をパケットの後尾に付加して転送する.

Type

かならず「1」を入り, Route Request メッセージであることを示す.

Hop count

送信元から何回転送されたかを表すフィールドである. 初期値は 0 とする.

Request ID

自分が最後に送信した Request メッセージで使用した ID に 1 加えた値とする.

Destination AreaID

目的の領域の Area Leader の ID を設定する.

Destination Sequence Number

その送信先のシーケンス番号を最後に知った番号を設定する. 経路表の更新と同時にシーケンス番号も更新する.

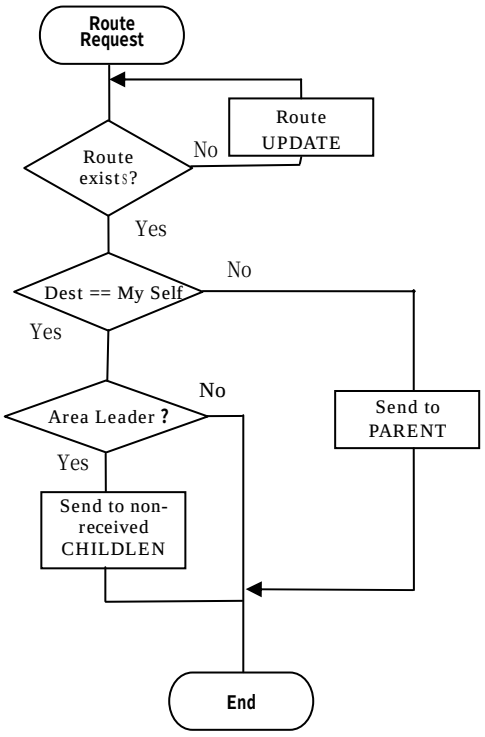

図 5. 経路要求の処理

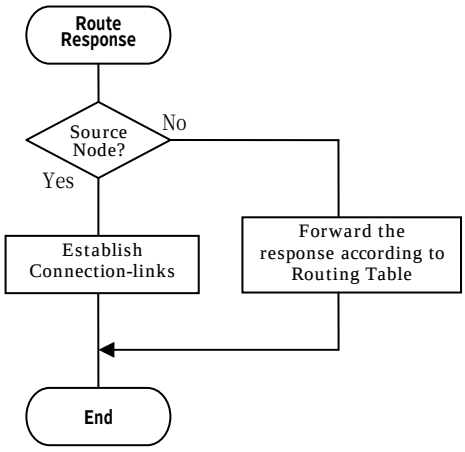

図 6. 経路応答の処理

図 7. 経路要求パケットフォーマット

Source ID

発信元のノード ID を設定する.

Source Sequence Number

自分自身のシーケンス番号を設定する. 応答して, 返ってくる際, 再送することも考えられるので, どのパケットが最新であるかを判断する.

Replay メッセージも Request メッセージとほぼ同様のフォーマットで, Request ID は設定せず, TTL を設定する. シーケンス番号は自分のシーケンス番号と確認し, 自分の番号のほうが小さい値であるとき, 書き換える. また, ホップ数は 0 に設定する.

Replay メッセージが送信元まで届くとコネクションが確立される. ここで初めて, 領域へ向けてパケットを送信する.

Areacast では, 領域に対して送信するだけでなく, 領域内のデータを Area Leader が集約することで, 領域の情報を NMN で一括して管理することを目指す. NMN まで経路を保持しているノードの場合, Area Leader を経由せず, 直接 NMN へ送信する. NMN への経路を保持していないノードは, 自分の領域の Area Leader を目指してパケットを転送する.

4.5 視覚化

OSN には, 管理用の指示器として LED が実装されると想定し, 実世界にノードが存在することを利用した管理を行う. 便宜上, 緑色, 赤色 2 種類の LED が存在するものとし, 2種類のコマンドを設計した.

Show_route_to_node: 目的ノードまでの経路を肉眼で目視できるように, 最終ノードを赤色, 途中経路ノードを緑色に点滅させる(図 8) .

Show_alive_nodes: 協調異状検出機能による閾値に基づいて, ”正常”状態のノードのみ緑色に点滅し, ノード状態を知らせる.

図 9 と図 10 にこれらの擬似コードを示す.

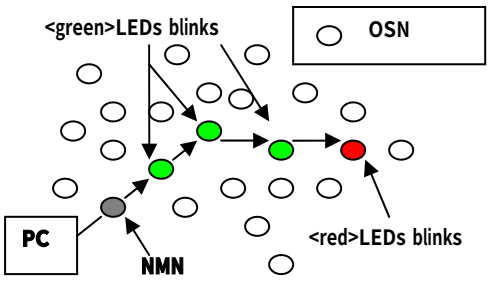

図 8. 視覚化による経路表示

```
//rcvpkt = receive packet;
//IDdst = destination ID; IDown = my local ID
void show_route_to_node(rcvpkt) {
   do{
      if(rcvpkt.IDdst != IDown) {
         Leds_blinks_green();
         forward by routing_table; break;}
      else if(rcvpkt.IDdst == IDown) {
         Leds_blinks_red(); break;}
      else error(rcvpkt, IDown); break;
      }while(TRUE);
}
```

図 9. pseudo-code: show_route_to_node()

```
//rcvpkt = receive packet;
//state = ALIVE or WEAKEN;
//link_table = linked nodes; buf = buffer;
void show_alive_nodes(rcvpkt) {
   do{
      if(buf == rcvpkt) break;
      else if(buf != rcvpkt){
         if(state == ALIVE) {
            Leds_blinks_green();
            multicast by link_table; break;}
         else if(state == WEAKEN) {
            Leds_blinks_red();
            multicast by link_table; break;}
         else error(rcvpkt, IDown); break;
      } buf = write_buffer(rcvpkt);
   }while(TRUE);
}
```

図 10. pseudo-code: show_alive_nodes()

5. 実装

MICA Mote を用い, 管理 PC 機能を Windows XP SP1 上に実装した. OSN のシステムを TinyOS[11]上に NesC 言語 [4]で実装する. 図 11 は SNAC のソフトウェア構成を示す. 図 11 において, ハッチングを施したブロックが新たに設計を行った部分である. NMN と OSN は Radio 機能で相互に通信を行う. Timer は MICA Mote に搭載されている LED の表示や Sense などのタイミングを制御する. Leds はイベントや Sense した情報に応じて赤･緑･黄色の三色 LED を点灯させる. Sense で検知する環境情報として, 動作電源電圧を監視する Voltage を実装する. Voltage 機能は, ノードのバッテリ残存量を検知し, その内部電圧情報を VoltageADC (図 12) によって数値化する. その数値を NMN の Listen プログラムによって問い合わせを行う. ProcessCMD は Voltage を検知した情報に応じて警告し, センサノードの環境情報を得る簡単なコマンド, push, pull 機能を備える.

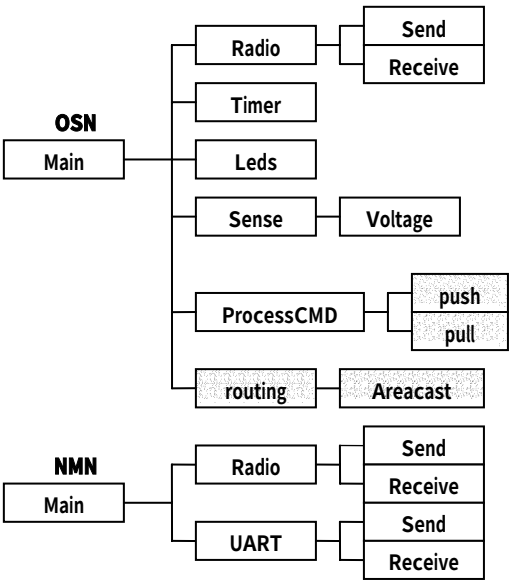

図 11. ソフトウェア構成図

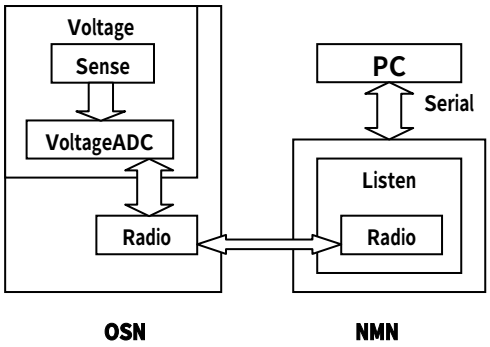

図 12. 動作電圧読み取り

例えば, push は, 目的のノードのバッテリ残存量の値を返す. pull は, バッテリ残存量の閾値を含むクエリを発し, バックグラウンドでシステムを起動させることで, その値より小さくなれば応答を返す. routing は, 領域管理の際のルーティング制御や, マルチホップ転送の機能を含む.

NesC プログラムを Mote に組み込み, NMN から JAVA によってセンサノード自身の情報やセンシングした情報などを取得する. また, 今回, 外部からノードを操作できるように管理 PC に HTTPD を構築し, WEB サーバ上の GUI で操作できるようにした [13]. 管理 PC 上のソフトウェア構成を図 13 に示す.

図 13 において, HIHORMain は管理 PC 上から発するコマンドと WEB からのコマンドを処理する. 図 14 と図 15 は TinyOS 上で実行される, NesC で記述されたバッテリ管理のプログラムの中心部分である.

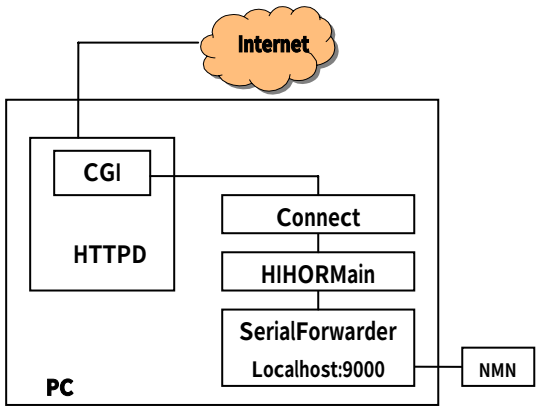

図 13. 管理 PC のソフトウェア構成

```
configuration Energy {
  provides {
    interface ADC as VoltageADC; }
}
implementation {
  components Voltage, EnergyM, IntToRfm, Main,
TimerC;
  //Energy
  Main.StdControl -> EnergyM;
  EnergyM.Timer ->TimerC.Timer[unique("Timer")];
  VoltageADC = Voltage;
  EnergyM.ADC -> Voltage.ADC;
  EnergyM.IntOutput -> IntToRfm;
//send
  Main.StdControl -> IntToRfm.StdControl;
}
```

図 14. バッテリ管理 NesC プログラム
コンフィグレーションファイル (Energy.nc)

```
module EnergyM {
  provides interface StdControl;
  uses {
    interface Timer; interface ADC; interface
IntOutput;}
}
implementation {
  event result_t Timer.fired() {
    return call ADC.getData(); //ADC の値を取得}
  event result_t ADC.dataReady(uint16_t data) {
    call IntOutput.output(data); //数値化し data に格
納
    return SUCCESS; }
```

図 15. バッテリ管理 NesC プログラム
モジュールファイル (EnergyM.nc)

HIHORMain は, パケット生成, SerialForwarder へのパケットの送信, SerialForwarder からのパケットの受信, GUI インタフェース等の機能を有する. 中継を実行する Connect は CGI からコマンドを受けて HIHORMain 上のプログラムを実行し, HIHORMain か

ら来るパケットを CGI に転送する機能を備える. SerialForwarder は, HIHORMain から受けたコマンドをシリアル通信によって実際に NMN に転送を行う. RS232 の IP ラッパと, パケットの転送などが主な機能である. 図 16 はブラウザ上からノード管理を行う GUI を示す.

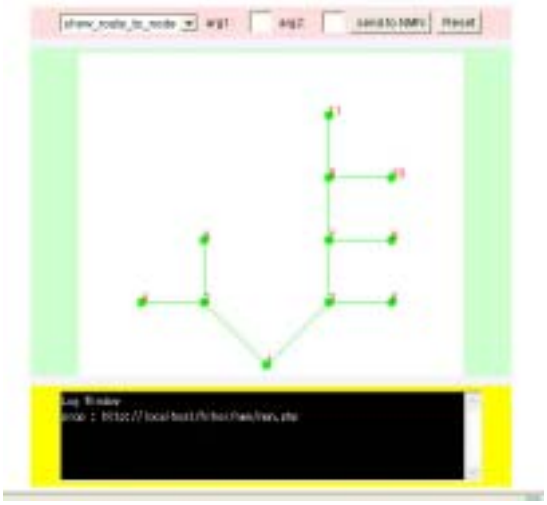

図 16. WEB ブラウザ上の GUI

6. 関連研究

センサネットワークにおける階層的なノード管理手法として, R-tree[5]を用いて不均一分布するノードをノード密度に応じて動的に管理単位を変更する手法が提案されている [2]. この手法は対象とするノードの近隣のノードを探索する問合せに適している. これに対して, SNAC で採用するノード管理は管理ノードからある領域全体へのメッセージの送信, 領域で集約したデータの管理ノードへの送信を効率よく実行することを目指したものである.

アドホックネットワークの位置情報を利用したルーティング手法として LAR [7]が提案されている. SNAC では厳密な位置情報よりも, ノードの群管理に重点を置く手法を採用する.

7. むすび

本稿では, センサネットワーク管理システムとして SNAC を提案し, SNAC の設計について述べた. SNAC では, 通信ノードがセンサノードである特異性を考慮して, 全体アーキテクチャを考えた. 現在, Mica Mote を用いた小規模のプロトタイプで実証を行っている. 今後, スケーラビリティの検証を念頭に, 数十単位のノード数での環境で評価を行う予定である.

謝辞

本研究は, 独立行政法人通信総合研究所モバイルネットワークグループとの共同研究で実施されています. また, 本研究開始に先立ち, 早稲田大学理工学部中島研究室藤波香織氏には, Mica Mote NesC プログラミングのご指導をいただきました. ここに感謝いたします.

参考文献

1) Clausen, T., Jacquet, P., Laouiti, A., and Project Hipercom, INRIA: Optimized Link State Routing Protocol (OLSR), *RFC 3626*, (Oct. 2003).
2) Demirbas, M. and Ferhatosmanoglu, H.: Peer-to-Peer spatial queries in sensor networks, *Proc. of IEEE P2P 2003*, pp.32-39, (Sep. 2003).
3) Estrin, D., Govindan, R., Heidemann, J. S., and Kumar, S.: Next century challenges: Scalable coordination in sensor networks, *Proc. of ACM MOBICOM*, pp. 263-270, (Aug. 1999).
4) Gay, D., Levis, P., Behren, R. von, Welsh, M., Brewer, E., and Culler, D.: The nesC Language: A holistic approach to networked embedded systems, *Proc. of Programming Language Design and Implementation (PLDI) 2003,* (June 2003).
5) Guttman, A.: A dynamic index structure for spatial searching, *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, pp. 47 – 57, (1984).
6) Hill, J. and Culler, D.: A wireless embedded sensor architecture for system-level optimization, *UC Berkeley Technical Report*, (2002).
7) Ko, Y-B. and Vaidya, N. H.: Location-aided routing (LAR) in mobile ad hoc networks, *Proc. of ACM MOBICOM*, pp. 66 – 75, (1998).
8) Makimura, K., Saito, H., and Tobe, Y.: Network management for distributed sensors, *ACM SIGCOMM 2003 poster*, (Aug. 2003).
9) Perkins, C.E. and Royer, E.M.: Ad-hoc on-demand distance vector routing (AODV), *Proc. of IEEE 2nd Workshop on Mobile Computing Systems and Applications*, pp. 90-100, (1999).
10) Pister, K. S. J.: Smart Dust - Hardware limits to wireless sensor networks, *IEEE Int. Conf. on Distributed Computing Systems*, Keynote Address, (2003).
11) WEBS: Wireless Embedded Systems, TinyOS . http://webs.cs.berkeley.edu/.
12) WINS: Wireless Integrated Network Sensors, http://www.janet.ucla.edu/WINS/.
13) 澤義和, 牧村和慶, 戸辺義人, 絹川博之: センサネットワークノード Web 管理システム, 第 66 回情報処理学会全国大会発表予定, (2004.3).