6W-3

シナリオ記述による無線センサネットワークの機能割り当て法

小林広治† 宮崎敏明†

会津大学大学院コンピュータ理工学研究科†

1 はじめに

無線センサネットワーク(WSN)において、ユーザの要求や環境の変化に合わせて各センサノード(以下、単にノード)に適切な機能を割り当てることは、大規模な WSN を構築・運用する上で非常に重要である。過去、シナリオ記述から各ノードの機能割り当てを行う手法が提案されているが、実機実装において考慮しなければならない問題がいくつかある。本稿では、細かな動作まで定義可能なシナリオ記述を用いて、実用的な WSN 機能の更新が可能な手法を提案する。

2 シナリオ記述による機能割り当て

WSN はセンサデータを取得できる複数のノードで構成されるネットワーク[1]であり、山間奥地や災害現場等の環境観測に好適なため近年注目されている。複数のノードが取得したセンサデータを、無線ネットワークを通じて取得することにより、観測領域の物理現象をリアルタイムに把握することができる。従来の WSN では、一度プログラムを各ノードにインストールした後は、センシングやデータ通信等の機能は固定となるため、柔軟性に欠ける。ユーザ要求や環境変化に合わせて各ノードに適切な機能が動的に割り当てられることが望ましい。文献[2]では、各ノードへの機能の割り当て方を定義したシナリオを用いた動的な機能割り当て手法を提案しているが、具体的なセンサデータの扱いや機能定義のシナリオ記述に曖昧さが残り、改善の余地がある。

3 提案手法

3.1 機能割り当て法

文献[2]の手法では、各ノードに、自身のバッテリ残量や、位置情報、起動しているセンサ等の情報を「プロパティ」として保持させ、そのプロパティを定期的にブロードキャストし、近隣ノードに知らせる。これにより、各ノードは、近隣ノードの状態を互いに把握することができる。このプロパティ情報に基づき、ユーザが予め定義して配布したシナリオを用いて、各条件下でノードが自身に適切な機能を割り当てることができる。また、新たに定義したシナリオを再配布することで、WSN のカスタマイズを行うことが可能となる。

3.2 従来のシナリオ記述における問題点

WSN の振る舞いは、以下に示す 3 つの基本要素から構成される。

1. センサで取得すべき環境情報の種類
2. センシングとデータ送信のレート
3. 取得したデータの処理方法

文献[2]の機能割り当て方法では、要素 1 を指定することが可能であり、また、過去我々が提案した手法[3]では、要素 1 及び要素 2 を記述できる。しかしながら、いずれも要素 3、すなわち取得したデータの処理方法を具体的に記述できないという問題があった。

3.3 シナリオ記述手法の改善と実装

前述した問題を解決するために、本稿では、取得したデータの処理方法を記述できる様に新たにノードの従属関係とそのノードからセンサデータを取得する手順を記述できるように記法を拡張した。WSN で代表的なアプリケーションであるクラスタリング[4]を用いて、提案する機能割り当て法を説明する。クラスタリングとは、WSN 内の全ノードで幾つかのグループ（クラスタ）を形成し、各クラスタ内に設けたクラスタヘッドがクラスタ内の他のノードが取得したデータを集約することにより、各ノードが直接基地局にデータを送信するのに比べ、ネットワーク全体の消費電力を抑えるトポロジ制御を行うものである。図１に、クラスタリングの具体的手順を示す。クラスタ内のノードは、クラスタヘッド（以下、単にヘッド）とクラスタスレーブ（以下、単にスレーブ）の２つの機能から構成されており、クラスタ内のスレーブが取得したセンサデータをヘッドで集約し、基地局にそのデータを送信する。ここでは、全てのノード同士はルータを介して 1hop で通信可能であるとし、ネットワーク内に 3 つのクラスタを形成する。また、各クラスタ内にヘッドは１つのみ存在を許す。

前述したクラスタリングを実行するシナリオ記述は、図 2 の通りである。はじめに、タイマ割り込みによりセンサデータ（ここでは照度）を定期的にパケットとして送信するように”send_light()”関数を毎秒起動するように設定する(2 行目)。次に、1hop 内のヘッド機能が ON であるノードの数をカウント(3-7 行目)し、それが 3 つより少ないならば、自身のヘッド機能を ON とする。それ以外で、もし照度センサを持っているならば、スレーブ機能を ON とし、0.5 秒毎に照度サンプリングを実行する(8-17 行目)。18 行目以降は、ノードの従属関係に基づき、クラスタを形成するための記述である。自ノードがスレーブである時、1hop 内で最も距離の近いヘッドであるノードに従属する(18-28 行目)。反対に、自ノードがヘッドの時、自ノードに属しているノードをスレーブとして登録する(29-36 行目)。以上が、main 文となる。タイマ割り込みで呼ばれる”send_light()”関数は、自ノードの機能がヘッドかスレーブにより、照度データの扱いを定義している。照度データは取得または受信される度に、自ノードのバッファに自動的に保存され続ける。ここで、自身がヘッドである場合は、クラスタ内のスレーブから受信した照度データがバッファに保存される。それを”getSensorData()”関数によってバッファから取り出し、（最新値、平均値、最大値、最小値）のいずれかを”calculation()”関数の第一引数で指定して取得する。図 2 では、”RECENT”により最新値を取得している。スレーブ毎に取得した照度データの最新値の平均を算出し、基

Role Assignment Method for Wireless Sensor Network Based on Scenario Description

†Koji Kobayashi, †Toshiaki Miyazaki

†Graduate School of Computer Science and Engineering, The University of Aizu

地局である BS に送信する(39-50 行目)。一方、自身がスレーブである場合は、自らサンプリングした照度データの最新値を取り出し、それをクラスタ内のヘッドに送信する(51-57 行目)。以上が、今回記法を拡張し、取得したセンサデータ処理を詳細に記述できるようにした部分であり、従来法では記述が困難であったものである。

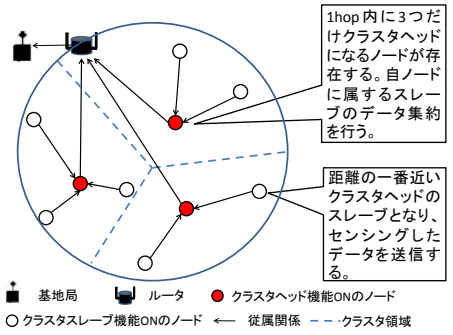

図 1: クラスタリング手順

4 検証

アットマークテクノ社製 Armadillo-420[5]をノードとし、それを 10 台と基地局およびルータを各 1 台用いて、図 1 の環境を構築し、動作検証を行った。本ノードは、Linux OS と WiFi モジュールを備えている。図 2 のシナリオ起動を毎秒行うと共に、各ノード同士のプロパティ交換を 1 秒から 2 秒の間でランダムに行った。同一環境で実験を 3 回行い、各ノードにヘッド機能またはスレーブ機能が安定して割り当てられるまで、その機能が反転した回数を調べた。結果は、ノード当たり平均 15.8 回であった。また、安定するまでにかかった時間は平均 46.6 秒であった。

5 おわりに

WSN の動作を規定するには、センサで取得する環境情報の種類、そのセンサのセンシングとデータ転送のレート、そして、取得したデータの処理方法という 3 要素が必要である。本稿では、従来困難であった取得したデータの処理方法を容易に記述可能とするシナリオ記法の拡張を行い、実機により実動作を確認した。今後は、WSN の動作をより簡便かつ詳細に定義できるように、シナリオ記法にさらなる改善を加える。

謝辞

本研究の一部は、総務省戦略的情報通信研究開発推進制度（SCOPE No.121802001）および科研費(No.23500095)の支援を受けて実施したものである。

参考文献

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless Sensor Network:A Survey,” Comp. Networks J, vol.38, no. 4, pp. 393-422, March 2002
[2] C. Frank, and K. Romer, “Algorithms for Generic Role Assinment in Wireless Sensor Networks,” Proceedings of ACM Conference on Embedded Network Sensor Systems (SenSys’05), pp. 230-242, San Diego, CA, USA, November 02-04, 2005
[3]K. Kobayashi, and T. Miyazaki, "A Dynamic Role Assignment for Wireless Sensor Nodes," IEEE student session in 2013 Tohoku-Section Joint Convention of Institutes of Electrical and Information Engineers, Japan, p. 1B11, August. 2013
[4] T. J. Kwon, M. Gerla, “Efficient Flooding with Passive Clustering (PC) in Ad Hoc Networks,” Proceedings of the IEEE, vol.91, pp. 1210-1220, Aug 2003
[5] Armadillo-420, Atmark Techno, http://armadillo.atmark-techno.com

```
1. main{  //1 秒毎に起動
    /* 照度センサ値の送信間隔を設定 */
    // 1 秒毎にパケットを送信する処理を行う関数を呼ぶ
2.  signal(send_light, 1);
    /* 近隣ノードのヘッドの数をカウントする処理 */
3.  snlist = listUp(1, ”cluster_head”); //1hop 内のヘッドリスト
4.  i = 0;
5.  foreach sn snlist {
6.    i++;
7.  }
    /* 自ノードの機能発火条件処理 */
8.  If(i < 3){ //ネットワーク内のヘッドが 3 つより少ないか
9.    setProperty(cluster_head_state, ”ON”); //ヘッド機能 ON
10.   setProperty(cluster_slave_state, ”OFF”); //スレーブ機能 OFF
11.   setProperty(light_state,”OFF”); //照度センサ機能 OFF
12. }else if(hasSensor(light)){ //照度センサをもっているか
13.   setProperty(cluster_head_state, ”OFF”); //ヘッド機能 OFF
14.   setProperty(cluster_slave_state, ”ON”); //スレーブ機能 ON
      //サンプリングレート 0.5 秒間隔
15.   setProperty(light_sampling_rate, 0.5);
16.   setProperty(light_state, ”ON”); //照度センサ機能 ON
17. }
     /* クラスタの決定処理 */
    //スレーブ機能 ON の時、距離の最も近いヘッドに属す
18. if(getProperty(cluster_slave_state) == “ON”){
19.   near_dist = MAX;
20.   snlist = listUp(1, “cluster_head”); //1hop 内のヘッドリスト
21.   foreach sn snlist{
        //自ノードと近隣ノードの距離
22.     dist = get_distance(self_node, sn.name);
23.     if(dist < near_dist){ //ノード間距離をチェック
24.     near_dist = dist;
25.     setProperty(cluster_head_node, sn.name); //本ヘッドに属す
26.     }
27.   }
28. }
    //ヘッド機能 ON の時、自ノードに属したスレーブを登録する
29. else If(getProperty(cluster_head_state) == “ON”){
30.   num = 0; //クラスタ内のスレーブ番号
31.   snlist = listUp(1, “my_cluster_slave”);
32.   foreach sn snlist{
        //自ノードに属しているスレーブの数だけ登録
33.     setPropertyAt(cluster_slave_node, num, sn.name);
34.     num++;
35.   }
36. }
37.} //main 終わり
   // 照度値を送信する割り込み関数
38.send_light(){
   /* 自身のヘッド機能がＯＮならば、自身に属するスレーブの照度の平均を計算し、BS に送信する */
39. If(getProperty(cluster_head_state) == “ON”){
40.   i = 0;
41.   average = 0;
      //自身に属するスレーブ取得
42.   snlist = listUp(0, cluster_slave);
43.   foreach sn snlist{
        //スレーブの照度値をバッファから取得
44.     getSensorData(sn.name, light, &data);
45.     average += calculation(RECENT, data); //最新値を取得
46.     i++;
47.   }
48.   average = average / i; //平均値の計算
     //BS 宛ての wp ポートへ照度の平均値を送信
49.   send(“BS”, wp, light, average);
50. }
    /* 自身のスレーブ起動がＯＮならば、自身が属しているヘッドに照度値を送信する*/
51. else if(getProperty(cluster_slave_status) == “ON”){
52.   snlist = listUp(0, cluster_head); //自身が属するヘッドを取得
53.   foreach sn snlist{
        //自身の照度値をバッファから取得
54.     getsensorData(self, light, &data);
      //クラスタヘッド宛ての wp ポートへ照度の最新値を送信
55.     send(sn.name, wp, light, calculation(RECENT, data));
56.   }
57. }
58.}
```

図 2: クラスタリングのシナリオ記述