7S-01

# C 言語で記述した無線センサネットワーク動作の形式検証法の提案

秋山直輝† 池田愛大† 宮崎敏明†
†会津大学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)は、温度や湿度といった環境の監視に適しており、また環境の変化に合わせてその動作を変更することも可能である。しかし、WSNを構成する各センサノードは自律的に動作し、しかも、相互に影響し合うため、WSNの振る舞いを、シミュレーションを用いて事前に動作保証することは困難である。

後藤ら[1]はセンサノードの動作記述に UML を導入し、UML 記述を、モデル検証ツール SPIN[2]の入力言語である Promela に変換し、その動作検証を試みた。Oleshchuk[3]は、センサノードの動作仕様を、直接 Promela を用いてモデリングし、SPIN で動作検証を行った。また、筆者らも、CSP[4]に基づく Fanclet+という独自言語を提案し、WSN の動作仕様の形式検証を試みた[5]。これら関連研究は、モデル検証・形式検証により、シミュレーションに頼らず、動作仕様の確認ができる利点はあるが、新たな仕様記述言語や環境を修得する必要があった。本稿では、C 言語で記述した無線センサノードで動作するプログラムを、そのまま SPIN を用いて検証する環境を提案する。

## 2. 提案環境

図１に、提案環境の概要を示す。ユーザによりC言語を用いて記述されたセンサノードの動作定義は、Promelaコードに変換され、SPINにより動作検証される。ここで、C言語記述自体は、センサノード単体の動作を定義しており、動作検証を行うためには、WSN全体の動作環境に関わる情報が別途必要となる。それら情報を構成ファイルとして別途用意する。Promelaコードを生成する際、変換器は、C言語記述に加え、当該構成ファイルを参照する。動作検証終了後、入力C言語記述は、そのままＣコンパイラを通され、オブジェクトコードが生成される。当該オブジェクトコードは、検証済みの新たなセンサノード動作を実現するものであり、安心してセンサノードに実装できる。

## 3. 形式検証及び実センサノードによる動作確認

センサノード動作定義プログラム２例を用いて、提案環境の動作を確認した。

### 3.1 クラスタヘッド選出プログラム

クラスタヘッド選出問題とは、複数のセンサノードの内、ある条件を満たす１つのセンサノードをクラスタヘッドとして選出するものである。各センサノードは、クラスタヘッド状態または一般ノード状態のいずれかの状態をとるものとし、周囲のセンサノードの状態を考慮して、自分の状態を決定する。

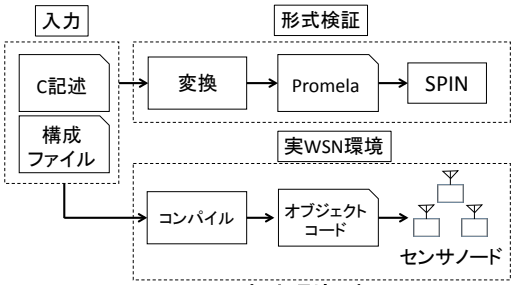


**図１．提案環境の概要**

図２に、本動作を実現する入力Ｃ言語記述の抜粋を示す。このプログラムは、各センサノードがクラスタヘッド状態になるべきかを判定する関数 HeadCheck()とデータを受信した際の処理を記述した関数 Receive()から成る。関数 HeadCheck()には、宛先センサノードの関数 Receive()に対して自状態を送信する記述が２４-２５行目にあり、宛先センサノード、送信する変数、宛先センサノードの呼び出す関数を指定できる。２４行目の関数 messageCompose2()は、送信する変数をメッセージ形式にパッキングするために使用される。本例では、自状態を表す変数 nodeState を、メッセージ形式_Tempvar に格納している。ここで、宛先センサノードは BROADCAST、すなわち自センサノードを除く、全てのセンサノードである。それを、２５行目の関数 sendChannelMessage()を用いて送信する。関数 HeadCheck()を定期的に呼び出すことにより、各センサノードは自状態を更新するとともに、それをブロードキャストし、隣接センサノード間で互いの状態を把握することになる。

ここでセンサノード $n$ は自状態を格納する変数 nodeState_$n$ | $n$={1,2,3, … ,$N$}を持ち、同変数は、図２の２行目で定義された変数 nodeState に、変換器がノード ID $n$ を”_”を付けて後部に連結し、Promela コード内に生成する。その値は、一般ノード状態であれば０、クラスタヘッド状態であれば１となる。クラスタヘッド選出プログラムでは、「全センサノードの内、1 つセンサノードのみがクラスタヘッドである。」ことから

$$\sum_{n=1}^{N} nodeState\_n == 1$$

が常に成立しなければならない。

上記の条件を線形時相論理 LTL で表現すると以下のようになる。いま、全センサノード数($N$)は $N$=4 とする。

ltl { <>[] (((nodeState_1 + nodeState_2 + nodeState_3 + nodeState_4) = = 1))

ここで、“<>[]”は、右側の数式が、常に成り立つ状態がいつかは来ると言う意味である。

本 LTL 式を、Promela コードと共に SPIN に入力し、反例がなければ、入力アルゴリズムが正しく動作することが保証される。実際に図２に示したクラスタヘッド選出プログラムは、上記の LTL 記述に対する反例はなく、またデッドロックがないことも検証された。もし、不具合が

An Approach to Formal Verification for Wireless Sensor Network Behavior Specified Using C Language
†Naoki Akiyama, †Akihiro Ikeda, †Toshiaki Miyazaki
†Graduate School of Computer Science and Engineering, The University of Aizu

```
    //各センサノードで定義される変数
1:  int Head[N]; //N 個の隣接センサノードの状態を表す変数
2:  int nodeState; //自状態を格納する変数

    //自センサノードがクラスタヘッド状態になるべきかを判定
3: void HeadCheck(){
4: int i;
5: int counter;
        //自分がクラスタヘッド状態でない場合、
        //隣接セにクラスタヘッドが存在するか確認
6:      if(Head[SELFID] == 0){
7:          counter = 0;
8:          for(i = 0; i<N; i++){
9:              if(Head[i] != 0){
10:                 counter = 1;
11:             }
12:         }
            //隣接にクラスタヘッドいない場合、
            //自分がクラスタヘッド状態になる
13:         if(counter == 0){
14:             Head[SELFID] = 1;
15:         }
        //自分がクラスタヘッド状態で、かつ、
        //自分より小さい ID を持つセンサノードが
        //クラスタヘッド状態の場合、
        //自分を一般ノード状態にする
16:     } else {
17:         for(i = 0; i<SELFID; i++){
18:             if(Head[i] != 0){
19:                 Head[SELFID] = 0;
20:             }
21:         }
22:     }
23:     nodeState = Head[SELFID];
        //自分以外に自状態をブロードキャスト
24:     messageCompose2(&__Tempvar, BROADCAST,
        nodeState);
25:     sendChannelMessage("Receive", &__Tempvar);
26: }

    // データを受信した際の処理
27: void Receive(int from, int param){
        //送られてきた他センサノード状態を保存
28:     Head[from-1]=param;
29: }
```

**図２．クラスタヘッド選出を実現する入力 C 言語記述例.**

あれば、反例として、その状態が SPIN から出力されることになる。

**実センサノードを用いた確認**

図２のプログラムをコンパイルし、実センサノードに搭載して、その動作を確認した。センサノードの状態を可視化するために、当該センサノードがクラスタヘッド状態であれば、黄色の LED を、一般ノード状態であれば、赤色の LED を点灯するようにした。４台のセンサノードは、その起動順序にかかわらず、常に１つのセンサノードのみ、黄色の LED が点灯する、すなわち、常に１つのクラスタヘッドが選出されるように動作することを確認した。図３に、動作確認時のセンサノードのスナップショットを示す。

### 3.2 人感センサの反応を契機に、LED を点灯するプログラム

ここでは、人感(赤外線)センサが反応したことを契機に、人感センサが反応したセンサノードの LED を点灯させると同時に、隣接センサノードに信号を送り、その信号を受けたセンサノードも LED を点灯させるプログラムの動作を検証する。ここで、センサノード $n$ の LED の消灯・点灯を表す変数を nodeLED_n とし、消灯時は０、点灯時は１となるように制御する。

**図３．実センサノードによる確認。左上のセンサノードがクラスタヘッドになっている。**

本例では、人感センサが反応した後に、全てのノードの LED が点灯状態にならねばならない。この事実は、全センサノード数が２個の場合、LTL を用いて、下記のように表記できる。

ltl { ((nodeLED_1==1)||(nodeLED_2==1)) ->
<>[] ((nodeLED_1 + nodeLED_2) == 2)}

ここで“->”は因果関係を表し、”->”の左辺が条件、右辺が結果である。C 言語記述から変換した Promela コードと上記 LTL 記述を SPIN に与え、反例がないことを確認した。

**実センサノードを用いた確認**

その後、入力 C 言語記述をコンパイルし、実センサノードに当該プログラムを搭載し、あるセンサノードの人感センサが反応すると、仕様通りに、全てのセンサノードの LED が点灯すること確認した。

本例では、簡単のために、人感センサの反応時に LED の点灯動作を取り上げたが、侵入者の有無を確認するためにセンサのサンプリングレートを早くし、かつ、全搭載センサをアクティブにして周辺監視を強化するなどの動的動作を行わせるなどの実用的な応用が考えられる。

## 4. おわりに

本稿では、C 言語記述で与えられたセンサノードの動作仕様を、モデル検証ツール SPIN を用いて形式検証する環境を提案した。入力 C 言語は、SPIN の入力言語である Promela コードに自動変換される。よって、SPIN や Promela に精通していないユーザでも本提案環境を用いて、動作仕様の検証ができる。また、入力仕様が C 言語記述であるため、そのままコンパイルして得られたオブジェクトコードを実機センサノードに搭載することができる。

**謝辞** 本研究の一部は、科研費(15K00133)により実施した。

**参考文献**

[1] 後藤亮馬,和崎克, “UML-PROMELA 変換器を用いた ZigBeeIP/RPL プロトコルにおけるノード探索仕様の検証,” FIT2014, 13.1: 159-162. 2014.

[2] G. Holzmann, “The SPIN Model Checker: Primer and Reference Manual,” Addison-Wesley, 2003

[3] V. Oleshchuk,“Ad-hoc sensor networks: modeling, specification and verification. ,” 2nd IEEE Intertional Work on Intelligent Data Acquisition and Advanced Computing Systems Technology and Applications, pp. 76-79, Sept. 2003.

[4] C.A.R. Hoare, “Communicating Sequential Processes,” Communications of the ACM, vol. 21, no.8, pp. 666-677, August 1978, DOI:10.1145/359576.359585

[5] T. Miyazaki, N. Akiyama, “Formal approach to produce verified programs for wireless sensor nodes,” IEEE 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP) ,. pp. 1-6, 2016.