様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　３０　年　　６　月　２３　日現在

機関番号：　２１６０２
研究種目：　基盤研究(C)（一般）
研究期間：　2015～2017
課題番号：　１５Ｋ００１３３

研究課題名（和文）形式的動作検証機能と自己修復機能を備えた無線センサネットワークの研究

研究課題名（英文）Wireless sensor network featuring a self-healing mechanism and a dedicated formal verification mechanism to verify its behavior

研究代表者

宮崎　敏明（Miyazaki, Toshiaki）

会津大学・コンピュータ理工学部・教授

研究者番号：７０４０４８９５

交付決定額（研究期間全体）：（直接経費）　　3,600,000円

研究成果の概要（和文）：無線を用いて各センサノードにプログラムをロードすることで、その動作を再定義可能な無線センサネットワーク: Reconfigurable Sensor Networkを実現した。センサノード動作を記述する専用言語Funclet+を設け、その記述から、形式検証ツールへの入力記述と、C言語記述を生成するトランスレータを開発した。形式検証ツールで生成された入力記述が動作検証された場合、生成されたC言語記述は正しく仕様を実現していることを保証できる。いくつかの典型的なセンサノードの動作を専用言語で記述し、形式検証ツールで検証した後、生成されたC言語記述が実機センサノードで正しく動作することを確認した。

研究成果の概要（英文）：We realized a wireless sensor network named Reconfigurable Sensor Network (RSN), in which behavior can be redefined by downloading a new program to sensor nodes remotely using wireless communication. To define RSN behavior, dedicated behavior description language Funclet+ was designed. A language translator was also developed. It generates an input description to a formal verification tool, and C description. Both generated descriptions represent the behavior described using Funclet+. Thus, by validating the generated input description using the formal verification tool, we can ensure that the generated C description properly represents the behavior described using Funclet+. We described some typical sensor node behaviors using Funclet+, and confirmed that the generated C descriptions can run on real sensor nodes properly after verifying the behaviors using the formal verification tool.

研究分野：　情報通信

キーワード：　センサネットワーク　形式検証　自己修復　プログラマブル　CSP　OTAP

１．研究開始当初の背景

無線センサネットワーク(WSN)は、敷設の容易性から、環境モニタリングなどに適し、普及が進んでいる。しかしながら、既存の WSN は、用途ごとにセンシング間隔やデータ収集法が固定であり、敷設後に、それらを変更するのは容易ではない。よって、同一エリアのセンシングであっても、他用途のセンシングのために、WSN 環境を共有することは難しく、用途に合わせ新たな設備の増設や既存設備の改修など、無駄な投資を強いられている。上記問題を解決する方法として、一度敷設した WSN のハードウエアを変更することなく、外部からその振る舞いをオンデマンドで再定義可能とし、新たな WSN を敷設することなく、用途に合った WSN 環境を短時間で提供可能とする Reconfigurable Sensor Network (RSN)の着想に至った。

２．研究の目的

外部より無線を用いて各センサノード(以下、単にノード)にプログラムをリモートダウンロードすることにより、その振る舞いを再定義可能な無線センサネットワーク：Reconfigurable Sensor Network(RSN)を実現する。各ノードは、ダメージを受けた近隣ノードの機能を自律的に肩代わりし、RSN の機能を継続させようとする自己修復動作も兼ね備える。また、プログラム記述言語は、ノード動作を正確に定義するのに十分な記述能力を持ち、かつ、記述したノード動作の無矛盾性をダウンロード前に、形式的検証手法によりチェックできる枠組みを持つ。

３．研究の方法

ノード動作を記述する専用言語 Funclet+を設け、その記述から、形式検証ツールへの入力記述と、C 言語記述を生成するトランスレータを開発した。図１に、システムの概要図を示す。Funclet+トランスレータは、センサノードの数や検証項目などを列挙したシステム構成定義ファイルを読み込み、Funclet+記述から CSP#記述を生成する。CSP#記述は、プロセス代数の一つである CSP (Communicating Sequential Process)を拡張したものであり、シンガポール国立大学で開発された形式検証ツール PAT の入力となる。Funclet+トランスレータは、CSP#記述と共に、各ノードに具体的に実装する C 言語記述を生成する。これら２つの記述は、共に同一の動作を表現している。よって、PAT によって CSP#記述された動作が検証された場合、C 言語記述の動作も検証されたと見なすことができ、安心して実機ノードに実装することができる。

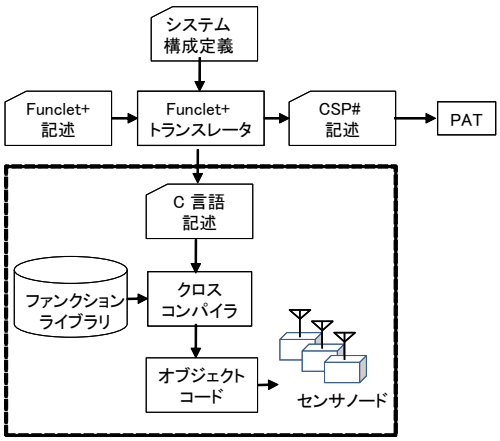

図 1．システム概要図

４．研究成果

図１に示したシステムを実際に試作した。ノードの動作を正確に規定するには、(A)自ノード・隣接ノードの状態把握およびその状態に依存するアクション、(B)センサ制御(ON/OFF, センシング周期など)、(C)データ処理(信号処理, データ収集, データ通信など)、の３項目が定義できなければならない。Funclet+言語の設計に際しては、上記を考慮した。また、Funclet+は、プロセス代数の一つである CSP に基づいている。従って、上述の Funclet+トランスレータは、Funclet+記述から CSP を一部拡張した CSP#記述を生成するが、双方の言語は、共に CSP を基本としているため、Funclet+トランスレータは、ほぼ一対一に記述を変換できる。生成された CSP#記述は、PAT を用いて検証される。一方、Funclet+トランスレータから生成された C 言語記述は、クロスコンパイラによりコンパイルされ、オブジェクトコードを得る。得られたオブジェクトコードは、図２に示す我々が過去に開発した実機センサノードと、専用プログラム開発環境を用いて、当該ノードへダウンロードされる。既開発のプログラム開発環境では、ノードの動作を、ユーザが C 言語を用いて直接記述する前提であるが、その入力 C 言語記述を、図１で示した Funclet+トランスレータが出力した C 言語記述をそのまま用いることにした。

構築した環境を用いて、いくつかの典型的なセンサノードの動作（アプリケーション）を Funclet+言語で記述し、その記述に誤りがないことを形式検証ツール PAT で検証できることを確認した。さらに、上述の Funclet+トランスレータから生成された C 言語記述をコンパイルして得られたオブジェクトコードを、実機ノードにダウンロードし、定義した仕様通りに正しく動作することも確認した。

図３に、Funclet+の記述例を示す。本記述例は、下記の動作を定義している。各ノードは、温度センサおよび光センサを搭載している。もし、当該ノードが直接通信できる他ノードが幾つか存在し、それらノードの少なくとも半分以上が光(Illuminance)をセンシン

グ（光センサを ON に）していたら、自身の光センサを OFF にし、温度センサを ON にする。そうでない場合は、逆に光センサを ON にし、温度センサを OFF にする。本動作が、各ノードで自律的かつ定期的に実行されることにより、環境の中に複数のノードが投入されると、光と温度をセンシングするノードは、それぞれ半数ずつとなるように、動的に調整が図られる。これは、あるノードが突然停止したり、新たなノードが追加されたりして、環境内のノード数が変化しても、それに応じて環境全体で光および温度をセンシングするノードの数が常にバランスされる様に各ノードが自律的に振る舞うことを意味し、当初目標であった自己修復機能を実現した一例でもある。

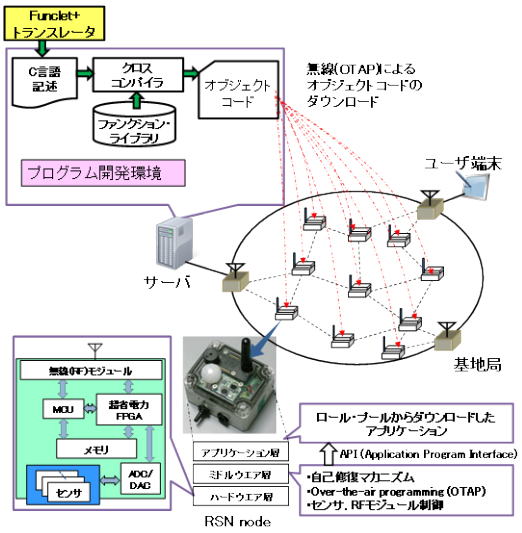

図 2. 既開発の実機センサノードとプログラム開発環境

```
 1. var heartBeatCnt
 2. var role // Array. 0=undefined, 1=Illuminance, 2=Temperature
 3. var illuminanceCnt
 4. var temperatureCnt
 5. var nextRole
 6. activate, // Initial state
 7.     HeartBeat!TIMER_CTRL 1, //Time wakes up every second
 8.     heartBeatCnt = _A_CHILD_0, // _A_CHILD_0is node_0 ID, i.e. =1
 9.     pass
10. InformRole?from par1,
11.     if(from != _A_PARENT_0){ // _A_PARENT_0 is not parent ID
12.         role[from] = par1
13.     },
14.     pass
15. HeartBeat?,
16.     if(heartBeatCnt == SELF){
17.         illuminanceCnt = 0, temperatureCnt = 0,
18.         for(i:_N_CHILD){ // for all chaild nodes
19.           if(_A_CHILD_0+i != SELF){ //if not myself
20.             if(role[_A_CHILD_0+i] == Illuminance){
21.               illuminanceCnt++
22.             },
23.             if(role[_A_CHILD_0+i] == Temperature){
24.               temperatureCnt++
25.             },
26.           },
27.         },
28.         if(illuminanceCnt >= temperatureCnt){
29.           nextRole = Temperature
30.         } else {
31.           nextRole = Illuminance
32.         },
33.         if(role[SELF] != nextRole){
34.           role[SELF] = nextRole,
35.           if(nextRole == Illuminance){
36.             _LIGHT!1, _TEMP!0 //on light, off temp
37.           } else {
38.             _TEMP!1, _LIGHT!0 //on temp, off light
39.           },
40.           InformRole!BROADCAST nextRole
41.         }
42.     },
43.     heartBeatCnt++,
44.     if(heartBeatCnt == _A_CHILD_LAST+1){
45.         heartBeatCnt = _A_CHILD_0
46.     },
47.     pass
```

図 3. Funclet+記述例

上記動作は、ハートビート・タイマ HeartBeat(図 3 の 15 行目)によって起動される。本例では、HeartBeat は、7 行目の記述によって、1 秒にセットされ、上記動作は毎秒起動されることを意味する。一方、変数 heartBeatCnt を用いることにより、同動作の開始タイミングはノード間でシリアライズされる。よって、ノードが 3 個であれば、本動作は、3 秒ごとに各ノードで起動されることになる。これにより、各ノードが、互いの状態を同時に確認し合うことにより、自分の保持する温度センサおよび光センサを絶え間なく ON/OFF させる、いわゆる発信状態に陥ることを防ぐことができる。また、図 3 の記述内で、_A_PARENT_0 は親ノードを表し、他の各ノード、すなわち子ノード（_A_PARENT_i| i=0,…,LAST）が、温度センサまたは光センサのどちらのセンサを ON にしているかの情報と、実際に当該センサがセンシングした値を集約する動作を行う。親ノードの動作は、別途 Funclet+で記述しているが、ここでは、紙面の制約から、親ノードの動作記述の提示は省略した。

図 3 の記述から Funclet+トランスレータを用いて、生成された CSP#記述を形式検証ツール PAT に入力し、形式検証を行うことができる。実際の形式検証は、アサーション(Assertion)として入力したノードの動作定義（仕様）に関する Yes/No の質問を PAT に問う形で行う。例えば、図 3 の例では、各ノードは、温度または光センサのどちらか一方のみを ON にしているので、「ノード総数は、温度センサを ON にしているノード数と光センサを ON にしているノード数の合計に等しい」はずである。本質問を、PAT に与えると、PAT は即座に Yes(Valid)と返答する。もし、No の場合は、PAT は、No となる場合（反例）を提示する。それを確認することにより、ユーザは、仕様の不備を認識し、入力記述を訂正できる。このように、形式検証では、一般的によく行われているシミュレーションを用いた動作確認とは異なり、一切の入力データや条件を与えることなく、入力仕様の正しさを検証できる。

図 4 は、図 3 の Funclet+記述から、Funclet+トランスレータを用いて得られた C 言語記述

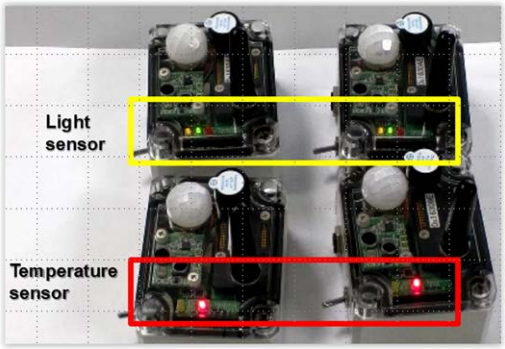

図 4. 図 3 で定義した動作を実機センサノードで実行している様子

を、図 2 で示したプログラム開発環境を用いて、4 個の実機ノードに実装し、実際に動作させたときのスナップショット写真である。実機ノードでは、光センサが ON の場合は黄色の LED が点灯し、温度センサが ON の場合は赤色の LED が点灯する。本例では、上段の２ノードが光センサを ON にし、下段の２ノードは温度センサを ON にしていることが分かる。このように、ON となっている温度センサと光センサの数は、ちょうど半数ずつになっている。実機ノードの数を増やした場合でも、図 3 で定義した仕様通り、温度センサと光センサの数をバランスするように、各ノードは、搭載センサの ON/OFF を自律制御することを確認した。

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計３件）

① Ikeda, N. Akiyama, T. Miyazaki, ″Formal Verification for Wireless Sensor Network in Consideration of Communication Errors,″ IEEE ICOIN2018, pp. 666-670, Chiang Mai, Thailand, Jan. 2018.
DOI: 10.1109/ICOIN.2018.8343202

② T. Miyazaki, N. Suematsu, D. Baba, P. Li, S. Guo, J. Kitamichi, T. Hayashi, and T. Tsukahara , ″Demand-Addressable Sensor Network: Toward Large-Scale Active Information Acquisition,″ IEEE Sensors Journal, vol. 16, no. 20, pp. 7421-7432, October 2016,
DOI: 10.1109/JSEN.2016.2575846

③ T. Miyazaki and N. Akiyama, ″Formal Approach to Produce Verified Programs for Wireless Sensor Nodes,″ IEEE 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP2016), Prague, July 2016.
DOI: 10.1109/CSNDSP.2016.7573994

〔学会発表〕（計５件）

① 秋山直輝, 池田愛大, 宮崎敏明,”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（An Approach to Formal Verification for Wireless Sensor Network Behavior Specified Using C Language),” 情報処理学会第 80 回全国大会(早稲田大学), 7S-01, March 2018.（学生奨励賞受賞）

② 池田愛大, 秋山直輝, 宮崎敏明,”無線センサネットワークにおけるデッドロック検出のための形式手法（Formal Approach to Detecting Deadlocks in Wireless Sensor Networks),” 情報処理学会第 79 回全国大会(名古屋大学), 1U-03, March 2017.

③ N. Akiyama, A. Ikeda and T. Miyazaki, ″Deadlock-free Behavior Definition for Wireless Sensor Nodes Using Formal Verification, ″ IEEE student session in 2016 Tohoku-Section Joint Convention of Institutes of Electrical and Information Engineers, Japan（平成 28 年度電気関係学会東北支部連合大会), 2A08, Sendai, Aug. 2016.

④ 鍋島俊, 宮崎敏明,”無線センサネットワークのための簡便な動作カスタマイズ環境（Easily Behavior Customization Environment for Wireless Sensor Networks),” 情報処理学会第 78 回全国大会(慶應大学), 1T-01, March 2016.（学生奨励賞受賞）

⑤ S. Nabeshima, and T. Miyazaki, ″Retargetable Scenario Compiler for Programmable Wireless Sensor Nodes, ″ IEEE student session in 2015 Tohoku-Section Joint Convention of Institutes of Electrical and Information Engineers, Japan（平成 27 年度電気関係学会東北支部連合大会), 2A03, Iwate, Aug. 2015.

〔産業財産権〕
○取得状況（計１件）
名称：センサネットワークシステム及びセンサネットワークシステムにおけるデータ取得方法
発明者：宮崎敏明
権利者：会津大学
種類：特許
番号：特許第５９４３４７６号
取得年月日：2016 年 6 月 3 日

国内外の別： 国内

〔その他〕
ホームページ
http://coll.u-aizu.ac.jp/

６．研究組織
(1)研究代表者
宮崎　敏明（MIYAZAKI, Toshiaki）
会津大学・コンピュータ理工学部・教授
研究者番号：70404895