Asian Journal of Information Technology 10 (7): 296-305, 2011
ISSN: 1682-3915

Applying Domain Specific Modeling for Environmental Sensing Using Wireless Sensor Network

Mohammad Fajar, Kenji Hisazumi, Tsuneo Nakanishi and Akira Fukuda
Department of Advanced Information Technology, Ito Campus, Kyushu University,
744 Motooka, Nishi-Ku, 819-0395 Fukuoka, Japan

Abstract: In order to provide an easy way to develop Wireless Sensor Network (WSN) applications. Current proposals apply abstraction mechanisms of software engineering, to hide complexity and implementation details of WSN. However, most of them are focused on implementation issues and platform-dependent solution thus, resulting designs are difficult to be reused. Moreover, these proposals do not provide a way how to specify a network architecture for application and how to control task allocation for each node or a group of node. This study addresses these issues and proposes a new high-level of abstraction model based on Domain Specific Modeling (DSM) which enables developers to build WSN applications using logical and physical abstraction model, task allocation as well as automation via code generation. Evaluation results on a case study indicate that the use of proposed model is capable to increase productivity in development time about 6 times than manual approach. While evaluation of quality of generated codes in simulation environment shows the effectiveness of processing task in the case study, average calculation at intermediate nodes can reduce the cost of communication significantly.

Key words: Domain specific modeling, wireless sensor network, environmental sensing, task allocation, generation, Japan

INTRODUCTION

A Wireless Sensor Network (WSN) is a large-scale *ad-hoc* and multi-hop network, consisting of small devices such as sensor nodes, routers and sink node. These devices use limited resources including limited storage, communication capabilities as well as severely constrained power supplies and the networks often operate in harsh unattended environments. Due to these constraints, many researches have been done to address many aspects of wireless sensor networks designs. So far, most of them are focused on implementation issues of applications and platform-specific design which tend to be constructed one by one and possibly in a site-specific manner.

The lack of utilization of software engineering methodologies can causes a considerable amount of efforts to configure the system for each site and modification of the system to introduce additional features, the resulting design are difficult to be reused and costly (Fajar *et al*., 2010). To solve these problems, current proposals apply abstraction mechanisms of software engineering to hide the complexities and implementation details of WSN. For example, TinyOS and nesC (Gay *et al*., 2003) try to raise level of abstraction from low-level to high-level programming models, they were focused on abstracting hardware and allowing flexible control of nodes. Although, developers are still difficult to out of hardware details and resulting designs are too platform-dependent to be reused. More software engineering is based on an application-centric view, where developers only concentrate on application level of WSN thus, it providing flexibility in optimizing the application's performance. Cougar (Bonnet *et al*., 2000) and TinyDB (Madden *et al*., 2005) are examples work in this model. In addition, some proposals in this category use model-based development such as UML and MDD (Beckmann and Thoss, 2010).

These approaches offered more high level of abstraction model. However, they do not provide a clear way how to specify network architecture for application and do not provide a mechanism how to allocate a set of program instructions (task) to each node or a group of nodes including specify the behavior of nodes. As a

Corresponding Author: Mohammad Fajar, Department of Advanced Information Technology, Ito Campus, Kyushu University, 744 Motooka, Nishi-Ku, 819-0395 Fukuoka, Japan

distributed system, task allocation is important aspect to control which node performs which task. It could affect the reliability of WSN system performance and energy use. In order to satisfy both requirements in this study, researchers propose a new high-level of abstraction model based on DSM which enable developers to model logical and physical aspects of their environmental sensing applications. Logical abstraction model provides a mechanism to define tasks for WSN application and physical level model is used to specify a network architecture for the application (to satisfy the first requirement). While task allocation model supplies an easy way to allocate or distribute WSN tasks to each node or a group of nodes (to satisfy the second requirement). Finally, generator specifies a transformation mechanism from model to final target automatically.

This study presents domain specific models in details, demonstrate them in a case study and measure their contribution in development time as well as evaluate the quality of generated codes using simulation environment.

DOMAIN SPECIFIC MODELING

In software engineering, developers generally differentiate between modeling and coding activities. Models are used for designing systems, specifying required functionality and understanding them better while codes are written to implement the designs. As a software engineering methodology, DSM has two basic characteristics (Kelly and Tolvanen, 2008). First, raise the level of abstraction beyond programming by specifying the solution in a language that directly uses concepts and rules from a specific problem domain.

Second, generate final products in a chosen programming language or other form from these high level specifications. The generating final products involves software generator to produce target codes automatically (Czarnecki and Eisenecker, 2000; Greenfield *et al*., 2004). With these characteristics, DSM offers several fundamental benefits over general purposes modeling and manual approaches such as increased productivity, improved quality and share expert knowledge (Kelly and Tolvanen, 2008). Figure 1 shows the basic architecture of DSM.

The objective of the DSM solution in this research is to make environmental sensing application development possible for a fundamentally larger developers and domain experts: people having little or no experience in

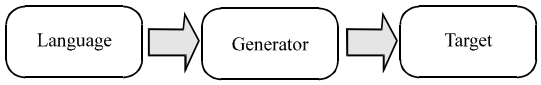

Fig. 1: Basic architecture of DSM

WSN programming especially nesC programming for TinyOS platform. The main idea is to get a situation where the developers could draw some pictures of the application, define some input parameters using dialog interface making relationship between components and then generate them to be executed in a target device. This quick and easy development approach is considered important for developers who wished to make some of their sensing application but did not have enough experience in WSN application development.

RELATED WORK

Different aspects of WSNs have been investigated in the literature including the use of software engineering approaches. In order to hide complexity and implementation detail of WSN system from developers, Sugihara and Gupta (2008) classified the abstraction level of programming for sensor networks into node-level, group-level and network-level abstraction.

Node-level are focused on abstracting hardware, allowing flexible control of nodes and some of them provide event-driven programming, this model is used by nesC (Gay *et al*., 2003), COMOS (Han *et al*., 2006) and TML (Newton *et al*., 2005). Group-level abstraction facilitates the collaboration among nodes so, developer can specify the behaviour of a group. Abstract regions (Welsh and Mainland, 2004) and SPIDEY (Mottola and Picco, 2006) are examples works in this model. In Network-level model, a sensor network is treated as a whole and is regarded as a single abstract machine, this model focusing on querying and processing system for extracting information from a WSN. Cougar (Bonnet *et al*., 2000), TinyDB (Madden *et al*., 2005) and Regiment (Newton and Welsh, 2004) provide this abstraction models.

However, there is a trade-off between them. Researchers can not define the behaviour of each node in the group-level abstraction or define group behaviour and network architecture in the network-level abstraction. In addition, some of them are focused on the implementation of platform-specific. Thus, resulting designs are too platform-dependent to be reused.

Furthermore, several studies have been done in order to reduce the cost of development, hiding the complexity and implementation detail of WSN, these works using

more high-level of abstraction than previous research. Sadilek (2007) proposed a language engineering approach to rapid prototyping of domain-specific languages based on Scheme and Eclipse EMF and present first experiences from the prototyping of a stream-oriented language for the description of earthquake detection algorithms. While, Losilla *et al*. (2007) proposed an architecture-centric approach to develop WSN applications.

The model presents a high level of abstraction using UML and a domain-specific language which allows designers to model their systems in a platform independent way thereby, obtaining more flexible and reusable designs. Similarly, Beckmann and Thoss (2010) presents an approach for Model Driven Software Development (MDSD) based on the data-centric OMG middleware standard Distributed Data Service (DDS) and it combines DDS features and MDSD for WSN systems.

However, the use of general modeling language such as UML in these researchers need to transform from model to model (general model to platform specific model) before transform it into a final target.

Researchers proposed DSM is to come up with a minimal sufficient representation of systems and this is the main reason for its 5-10 times productivity increases (Kelly and Tolvanen, 2008).

In addition, these current proposals are insufficient for the needs since, they do not provide a way how to specify a network architecture for application and how to control task allocation to each node or a group of nodes. Task allocation is important feature in a distributed sensor network where WSN application consists of a set of task to be assigned or distributed to each node and these nodes have limited resources as well as may have different capabilities.

Researchers need to control how to distribute the task for each node or a group of nodes to increase application performance the reliability system as well as efficiency of energy use.

In order to satisfy these requirements researchers proposes a domain specific model which offers some benefits: high level abstraction models provides an easy way to model environmental sensing applications including tasks of application and network architecture; easy to control tasks allocation to sensor nodes or a group of nodes and generate target platform (primitive code) automatically by generator.

DSM FOR ENVIRONMENTAL SENSING

The proposed model is used to develop environmental sensing applications, specify network architecture and control task allocation for each node or a group nodes. It supports multi-hop network, star, tree and mesh architecture, TinyOS platform as well as mica-family mote. It is also possible to employ the model in different platforms.

Overview of WSN-DSM: Researchers use a graphical language to raise level of abstraction and make application development as easy and natural as possible. The language provides graphical components, relationships and rules how to model environmental sensing applications. Figure 2 shows an instant of the model with notation and its symbol using problem domain such as sensing and node icon.

Generally, the model is separated into logical level model, physical level model and task allocation model. Logical level model represents a collection of tasks or a group of tasks and relationships among them. With this model, developer concentrates to define logical aspects of application in terms of tasks for application.

Physical level model reflects physical components and network architecture. Developers have flexibility in deciding nodes and a network architecture for each applications. Both of these models are linked by task allocation model. The task allocation model is used to control which nodes perform which tasks.

Each component in each level is connected by a relationship to describe their data flow. Horizontal relationships depict a connection between components at same level (from moving average to average calculation task) while vertical relationships show a connection between components at different level (from components in TG1 to components in Node group 1).

Logical level model: This model provides required components for modeling of environmental sensing application such as sensing tasks, filtering, sending/receiving and data collecting task as well as relationship components.

Each task consists of a set of program instructions which done by node. Some tasks which have regular contact and frequent intraction or mutual influence and work together to achieve a goal can be arranged into a group.

For modeling an application, this logical model can be seen as a rooted tree where data collection as a root, calculation tasks as their branches or intermediate nodes and sensing task as leaf nodes. Data flow design for sensing data is started from leaf node, through branch toward to root. It indicates the direction of relationship always leads to root node and is not

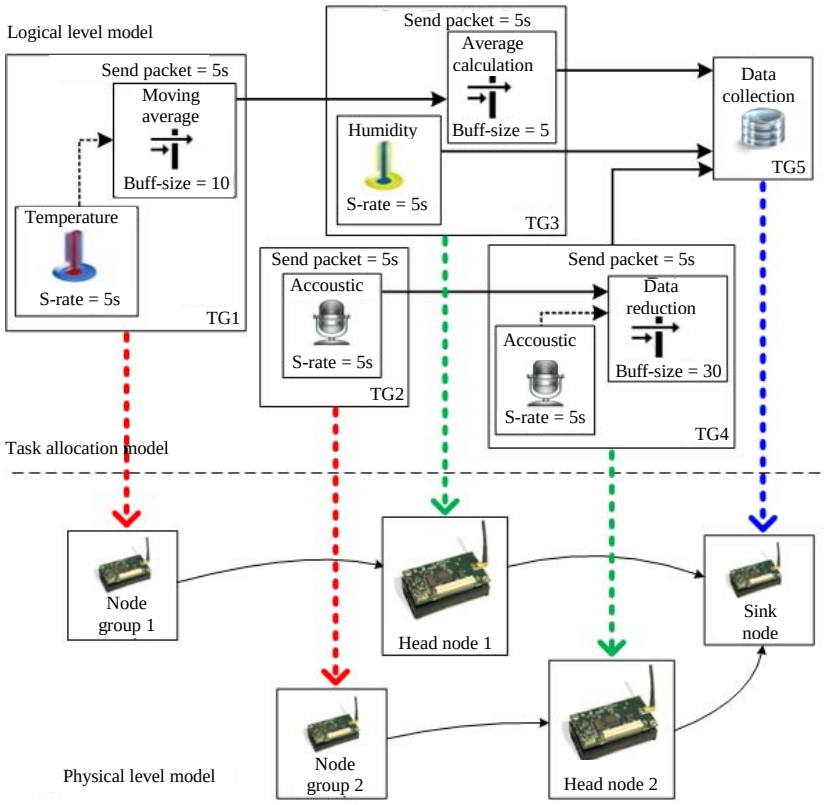

Fig. 2: An overview of WSN-DSM for environmental sensing

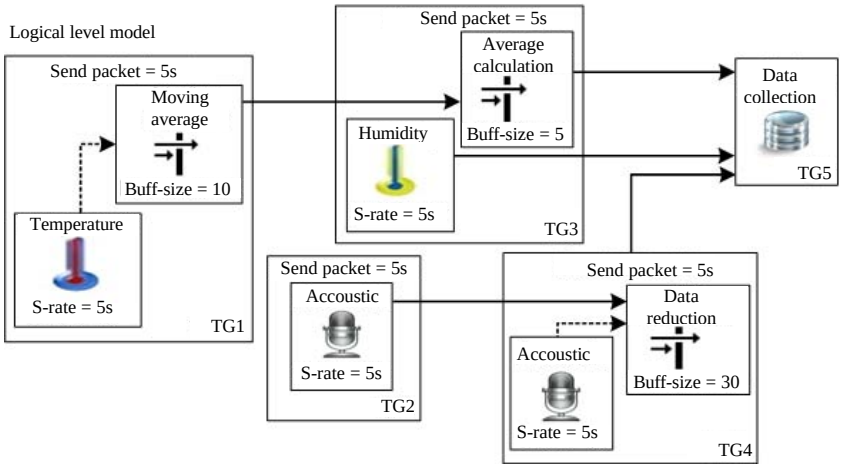

Fig. 3: Logical level model

allowed to make a relationship with opposite direction (from data collection to sensing task). Modeling activity involves: define a group and its sending period; put one or more tasks to the group and decide their parameters (sensing task and its sampling rate); connect each task to form their data flow. To add more groups, repeat first step. For making a connection between two tasks, researchers havetwo types of relationship. Intra and inter relationship. Intra-relationship is used to link tasks at a same group while inter-relationship to connect tasks at different group. For example in Fig. 3 researchers define temperature sensing task and its sampling rate = 5 sec, moving average task with buffer size = 10 elements and then connect both of them using intra-relationship (dotted line) to form their data flow. This relationship describes that after temperature data is sensed, the data stored into

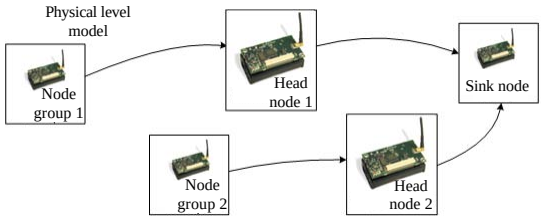

Fig. 4: Physical level mode with tree architecture

moving average's buffer and then moving average task calculate all of data in the buffer and sent its result to average calculation task, a relation between moving average and average calculation is specified by using inter-relationship (solid line). Finally, all of data are collected by data collection task. This scheme is similar for TG2, TG3 and TG4 in Fig. 3. In transformation process, generator translates intra-relationship as a relation at the same node and uses a buffer as their media (moving average's buffer). While inter-relationship is mapped as a relationship at different nodes and uses sending and receiving task to connect them. For sensing symbols, generator translates them into codes for sensing activity (read function), moving average symbol into codes of moving average algorithm.

Physical level model: This level provides required components to model network architecture of application such as sensor nodes, intermediate nodes, sink node and relationship components. These nodes are organized into node groups. Each group consists of one or more nodes which have similar behaviour and often they have same resources capabilities. Relationships at this level are used to form network architecture. For example if the nodes form tree architecture, root node of the tree should be sink node, leaf nodes as sensor nodes and branches as intermediate nodes. The flow of sensing data is started from leaf, through branch toward in other words sensor nodes send data to intermediate node and ends at sink node.

The behaviours of each node are decided by tasks at logical level (will be performed in allocation model). Modeling work at this level involves: define a node group, put one or more sensor nodes to the group if we need intermediate nodes then researchers should define a new group and put one or more intermediate nodes to the group. Repeat first or third step to add more groups. Fourth step define a group and put a sink node to the group. Finally, fifth step connect each node to form network architecture. Figure 4 shows an example of physical components which formed in tree architecture. The model consist of two sensor node groups: Node group-1 and Node group-2, two intermediate nodes: head node 1 and 2 as well as one sink node. Generator translates this model as a tree with sink node as its root (Node ID = 0).

Task allocation model: Task allocation is an instrument that can help us in identifying task scopes and parameters in dividing the entire work into parts and portions and assigning these loads to different nodes so, they can be properly controlled during the whole task performance. The flexibility and easiness to control task allocation could affect on application performance and reliability of WSN including energy use. This model provides required components for making relationship between logical components and physical model components. It uses vertical relationships (colored dotted-line) with some rules if a group of task is leaf node which contained sensing task then, this task can be allocated to sensor node, a sensor node group or intermediate nodes while data collection task should be allocated to sink node.

If a group of task contains a task for intermediate node (ex. average calculation task) then the group should be allocated to intermediate nodes. Modeling work at this level involves: select a vertical relationship, put the relationship on a group of task at logical level and draw the relationship toward a node or a group of nodes at physical level. Repeat first steps to allocate another task until all of tasks have been distributed to nodes.

Task allocation model provides some rules to avoid task allocation errors such as the direction of relationship always from task components at logical level toward node components at physical level (from top to bottom), some tasks are prepared only for certain nodes for example average calculation task in this research the task is space domain where it calculates data from two or more sensor nodes. The type of such task should be allocated to intermediate nodes not for sensor nodes.

In a complex WSN comprising of many nodes with various capabilities, proper allocation of tasks to nodes is an important aspect and the results of simulation should be evaluated and revised if necessary before deploy the application in a real system. As an example in Fig. 5 researchers use vertical relationship to allocate three tasks in TG1 (sensing temperature, moving average and send task) to Node group 1, this mean all of nodes in the Node group 1 perform these tasks. The sequence of execution of tasks is determined by intra-relationship in TG1.

Each node in TG1 performs temperature sensing, puts temperature data into moving average's buffer then performs moving average task to calculate all of data in the buffer and finally, performs sending task to send

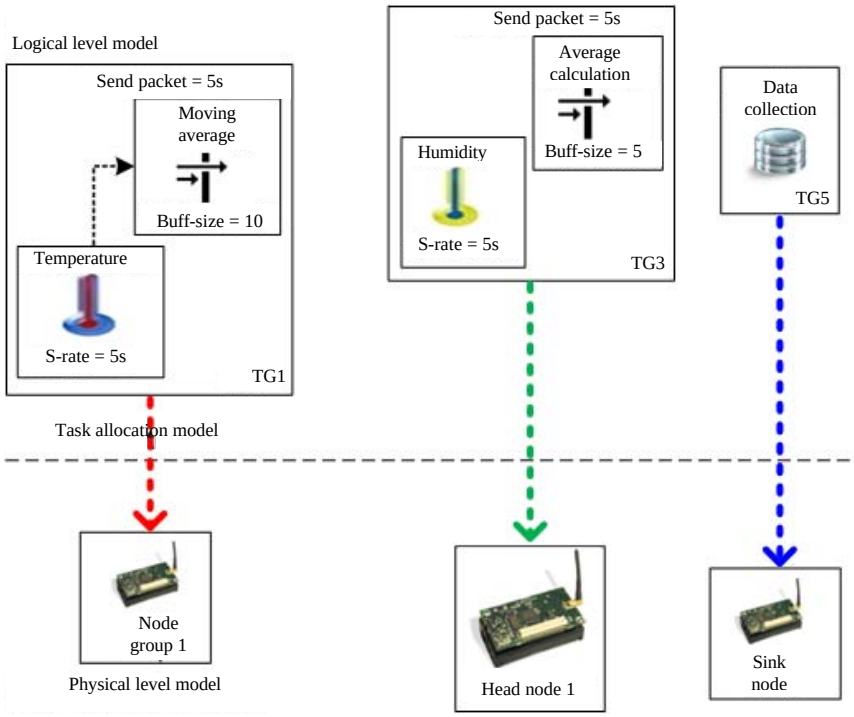

Fig. 5: Vertical relationships for task allocation model

calculation result. In transformation process when a task is allocated to a node (temperature sensing), generator transforms it as an activity to perform sensing for the node, read and decide its sampling rate from model and add sending task to the node. If a task is allocated to a group of node then, generator translates it as activity for all of sensor nodes in the group and completes them with a sending task. All of these activities are translated into appropriate codes of platform target.

Execution timing of the task: The execution of task in WSN applications can be triggered either by time, event or certain conditions. For a simple environmental sensing application might sample a few sensors every 20 min and send it to sink node while fire detection application might transmit the data if temperature is >45°C. This execution behaviour is specified in DSM model. Timing explicitly define node control-flow behavior for example sampling rate in sensing task is used to determine the number of sample per second while sending time is used by node to specify delivery period to send data to network. In processing task, this timing is specified implicitly in algorithm for example a calculation process is started when a buffer is full or reached a certain condition. For example in Fig. 2 sensing task is done every 5 sec. Each time temperature has been sampled and stored into moving average's buffer. Sensor node checks the buffer if it is full then the calculation task will be performed and its result sent to intermediate node. In this case, first iteration, sending task is triggered by a certain condition (if buffer is full) while for next iteration, it is initiated by timer periodically (every 5 sec). In addition, all of the nodes in the model have same timer rate to send a packet but since, boot time and their range are different from each other, intermediate node will receive their data in different time. When a group of nodes communicate using an RF, MAC protocol determines the communication schedules and rules because at any time only one pair of nodes can use the frequency to send out data to each other. In worst condition collision may occur and the protocol has mechanism to re-transmit the data.

Definition of WSN-DSM metamodel: A metamodel as shown in Fig. 6 describes the concepts and rules of DSM language and gives us and developers more general and flexible model for reuse requirements. At logical level model, sensing tasks can be connected to processing and processing tasks can be linked to data collection task. Conversely, researchers can not make a relationship from processing tasks to sensing tasks or data collection task to sensing tasks. For simplicity, researchers can classify some tasks into a group and allocate the group to a group of nodes. Execution of tasks involves timer either it executed by periodically (sampling rate and sending period) or based on certain event (implicitly is described

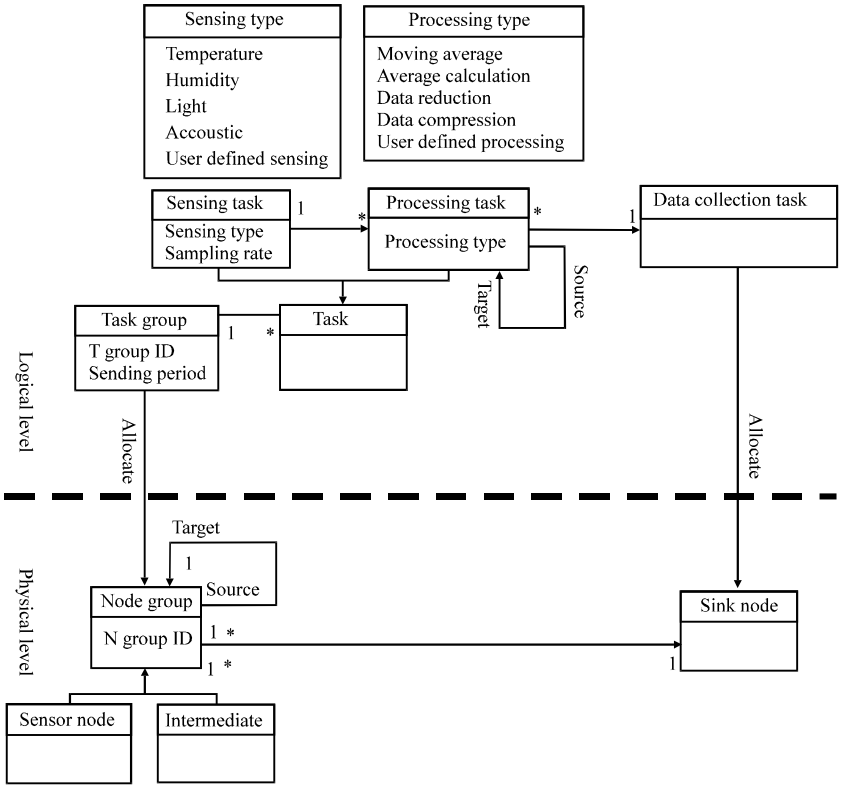

Fig. 6: WSN-DSM for environmental sensing

in processing algorithm). At physical level researchers can define a group of nodes. All the nodes performing similar tasks are grouped and physically deployed together are considered to belong to same region.

IMPLEMENTATION

In this study, researchers describe a schema of implementation which involves a DSM environment, nesC components and code generator. Figure 7 shows the implementation schema and its flow to guide application development process using the proposed model. Each activity (circle symbol) requires input document and will produce one or more output documents.

DSM environments: To develop a proof of the proposed models, researchers have implemented the model and evaluated it. The development tools that researchers used in this research are as follows:

- MetaEdit+4.5 to develop DSM language and its generator
- nesC language 1.2.4 on TinyOS-2.x as the target codes
- Tossim-2.x to simulatethe generated codes
- Micaz mote is hardware platform target
- Perl script to analyze the log files (simulation results) and produce network static (Fig. 7)

nesC components model and code generator: In this case study, researchers use TinyOS and nesC language as a final target. A nesC program is a collection of components. These components are defined by using module and configuration concepts. Module implementation sections consist of nesC code, declares variables and function, call functions, etc. Configuration implementation sections consist of nesC wiring code which connects components together. Every component is in its own source file and there is one to one mapping between component and source file names. For example, the file TimerMilliC.nc contains the nesC code for the component TimerMilliC. Figure 8 shows nesC main components used in this research. MainC, LedsC, ActiveMessageC and TimerMilliC components are nesC predefined modules while SensorC and RoutingC components are self predefined modules, these modules dependent on user requirements. For Routing components researchers use collection tree and P2P protocol as multi-hop protocol. The last component is DataCollection which contains codes of application and it will be generated by generator.

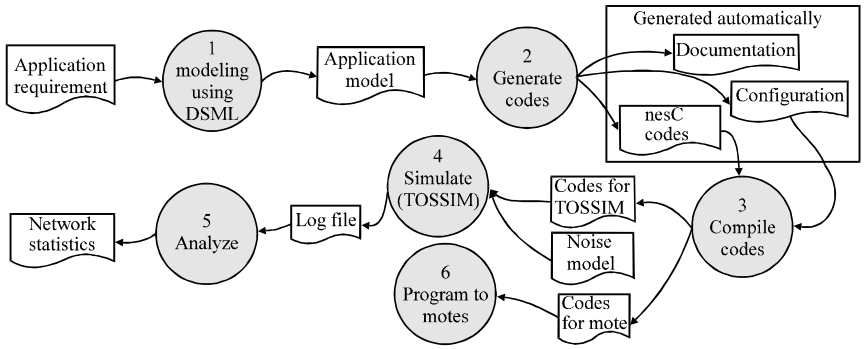

Fig. 7: Implementation schema

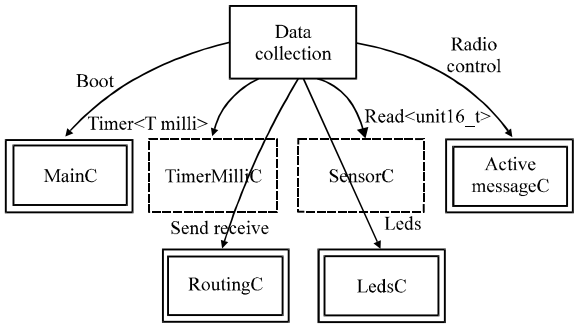

Fig. 8: nesC component model

The final step is obtaining final application codes via code generation automatically. A nesC generator in this research has been implemented using the Meta Edit Report Language (MERL) script. The generator reads application model and produces nesC codes for each node group, intermediate nodes and sink node. The core codes are generated into two files (nesC configuration and module file), the nesC configuration file contains wiring components and nesC module file contains the behaviour of application including algorithms for processing. For each provided interface in the nesC model, all its Commands must be implemented. Conversely, for each required Interface, all its Events must be handled. The generated solution, obtained from the corresponding model in the case study and the nesC component model in the Fig. 8 have been successfully compiled and tested on TinyOS simulator, demonstrating satisfactory results.

CASE STUDY: AN ENVIRONMENTAL SENSING APPLICATION

In this study, researchers use the proposed models to develop environmental sensing applications and show their evaluation results.

Modeling the application: Application 1 and 2 have two sensor nodes (with temperature sensing task), one head and one sink node, respectively. The nodes are connected via a tree multi-hop architecture. Head node for application 1 is a forwarder (without a sensor device and calculation task). While head node for application 2 is a processor (with a humidity sensor device and average calculation task). The head node in application 2 will intercept packets from sensor node 1 and 2, calculate the packets and send its average to sink node. The model of application 2 is shown in Fig. 9.

Table 1: Coding cost in time

Application	No. of functions	No. of lines	Time (min)
Application 1	8	129	33
Application 2	9	144	38

Table 2: Modeling cost in time

Application	No. of objects	No. of relationship	Time (min)
Application 1	9	9	5
Application 2	10	10	6

Development cost: Researchers measure the cost of development time for application 1 and 2. Final target is two nesC files (module and configuration). For coding activity/handwriting code researchers use nesC template and programmer editor with assumptions, developers have experience in nesC programming. Using handwriting code to produce nesC codes for application 1 researchers need 33 and 38 min for application 2 (Table 1).

While for modeling activity, researchers measure the time since, they run the DSM graphical model, select the requirements tasks and input their parameters (timer rate), connect the task then design network architecture, allocate the tasks to the nodes and finally, generate the codes using generator automatically. To produce nesC codes for application 1 researchers need 5 min while for application 2 researchers need 6 min (Table 2).

The results show that the use of the proposed model to develop the two applications will reduce the time of development significantly, the productivity improvement of about 6 times than manual approach.

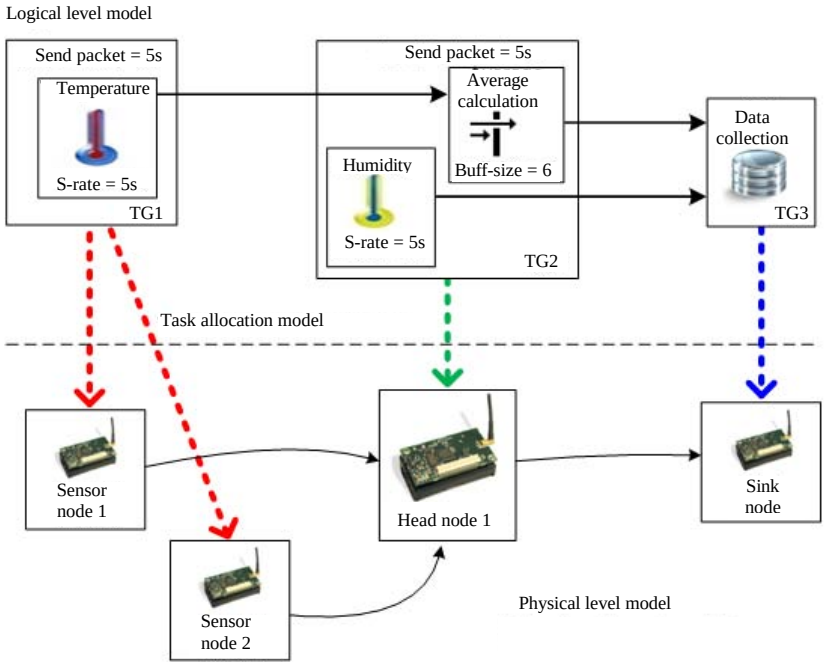

Fig. 9: An environmental sensing application (Application-2) depicted using the proposed model

Table 3: Network performance of application 1

From	To	No. of packets sent	No. of packets loss	Delay time average (sec)
2	1	1018	20	0.00751943
3	1	1018	18	0.00776123
1	0	339	5	0.00753571

Packet loss ratio = 1.7%

Table 4: Network performance of application 2

From	To	No. of packets sent	No. of packets loss	Delay time average (sec)
2	1	1018	15	0.00750677
3	1	1018	13	0.00743656
1	0	679 (Original data)	12	0.00748064
1	0	339 (Calculated data)		

Packet loss ratio = 0.3%

Performance evaluation: Researchers use TinyOS Simulator (TOSSIM) to observe and evaluate the quality of generated codes in terms of network performance such as packet loss, delay time and ability to reduce the cost of communication. TOSSIM captures the behaviour and interactions of networks of thousands of TinyOS motes at network bit granularity and it runs the same code that runs on sensor network hardware (Levis *et al*., 2003). Researchers present here, one of testing results that researchers have done. Table 3 shows a network performance of application 1 after researchers added average calculation task to head node.

Head node (node 1) made the number of packet sent to sink node (node 0) was decreased significantly. From 1998 packets sent successfully to the head node, it generates 339 packets and sends 334 packets to sink node successfully. The packets are average results of data from node 2 and 3. This is one of calculations types of space domain requirement. For average calculation task, the number of packets sent and data accuracy are influenced by buffer size (in this case the buffer size is 6 elements). Similarly, for application 2, Table 4 shows head node (node 1) made the number of packet sent to sink node was decreased significantly (339 packets). Besides, it also sent its own humidity data (679 packets) to sink node. This result shows even if the head node sent its own data, its performance was not different significantly with application 1.

CONCLUSION

Researchers present a high level of abstraction model in logical and physical level model as well as task allocation model which allows developers to build their environmental sensing applications more flexible, reusable and configurable including a clear way how to specify the WSN architecture via physical level model and how to control tasks allocation using task allocation model. While the use of generator to generate a final target automatically makes the applications development more quickly with high quality of codes.

Evaluation results on environmental sensing applications show that the proposed model has ability to

increase productivity of developers in development time about 6 times than manual or handwriting approach. While evaluation of quality of the generated codes shows the effectiveness of processing task in this case study average calculation at intermediate nodes can reduce the cost of communication significantly.

REFERENCES

Beckmann, K. and M. Thoss, 2010. A model-driven software development approach using omgdds for wireless sensor networks. Proceedings of the 8th IFIP WG 10.2 International Conference on Software Technologies for Embedded and Ubiquitous System, October 13-15, 2010, Austria, pp: 95-106.

Bonnet, P., J.E. Gehrke and P. Seshadri, 2000. Querying the physical world. IEEE Personal Commun., 7: 10-15.

Czarnecki, K. and U.W. Eisenecker, 2000. Generative Programming: Methods, Tools and Applications. Addison-Wesley, USA., ISBN-13: 9780201309775, Pages: 832.

Fajar, M., T. Nakanishi, S. Tagashira and A. Fukuda, 2010. Introducing software product line development for wireless sensor/actuator network based agriculture systems. Proceedings of the AFITA 2010 International Conference on Quality Information for Competitive Agricultural Based Production System and Commerce, October 3-7, 2010, IPB International Convention Center, Bogor, Indonesia, pp: 83-88.

Gay, D., P. Levis, R. Behren, M. Welsh, E. Brewer and D. Culler, 2003. The nesC Language: A holistic approach to networked embedded systems. ACM SIGPLAN Notices, 38: 1-11.

Greenfield, J. and K. Short, 2004. Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools. Wiley Publishing Inc., USA., ISBN-13: 9780471202844, Pages: 666.

Han, C.C., M. Goraczko, J. Helander, J.L.B. Priyantha and F. Zhao, 2006. CoMOS: An operating system for heterogeneous multi-processor sensor devices. Microsoft Research Technical Report No. MSR-TR-2006-177, pp: 14. http://research.microsoft.com/apps/pubs/default.aspx?id=55991.

Kelly, S. and J.P. Tolvanen, 2008. Domain-Specific Modeling: Enabling Full Code Generation. Wiley-IEEE Computer Society Press, USA., ISBN-13: 9780470036662, Pages: 427.

Levis, P., N. Lee, M. Welsh and D. Culler, 2003. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, November 5-7, 2003, Los Angeles, CA., USA., pp: 126-137.

Losilla, F., C. Vicente-Chicote, B. Alvarez, A. Iborra and P. Sanchez, 2007. Wireless sensor network application development: An architecture-centric MDE approach. Proceedings of the 1st European Conference on ECSA, September 24-26, 2007, Aranjuez, Spain, pp: 179-194.

Madden, S.R., M.J. Franklin, J.M. Hellerstein and W. Hong, 2005. TinyDB: An acquisitional query processing system for sensor networks. ACM Trans. Database Syst., 30: 122-173.

Mottola, L. and G.P. Picco, 2006. Logical neighborhoods: A programming abstraction for wireless sensor networks. Proceedings of the 2nd IEEE International Conference on Distributed Computing in Sensor Systems, June 18-20, 2006, San Francisco, CA., USA., pp: 150-168.

Newton, R. and M. Welsh, 2004. Region streams: Functional macroprogramming for sensor networks. Proceedings of the 1st Workshop on Data Management for Sensor Networks, August 30, 2004, Toronto, Canada, pp: 78-87.

Newton, R., Arvind and M. Welsh, 2005. Building up to macroprogramming: An intermediate language for sensor networks. Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, April 15, 2005, Los Angeles, CA., USA., pp: 37-44.

Sadilek, D.A., 2007. Prototyping domain-specific languages for wireless sensor networks. Proceedings of the 4th International Workshop on Software Language Engineering, October 2007, Johannes Gutenberg-Universitat, Mainz, Germany, pp: 76-90.

Sugihara, R. and R.K. Gupta, 2008. Programming models for sensor networks: A survey. ACM Trans. Sensor Network, 4: 1-29.

Welsh, M. and G. Mainland, 2004. Programming sensor networks using abstract regions. Proceedings of the 1st Symposium on Networked Systems Design and Implementation, March 29-31, 2004, San Francisco, CA., USA., pp: 29-42.