On Design Patterns for Sensor Networks

Syed Obaid Amin and Choong Seon Hong
Dept. of Computer Engineering, Kyung Hee University

요 약

A design pattern is a general solution to a commonly occurring problem. Design patterns have proven highly effective in representing, transferring, and applying the design knowledge in many engineering disciplines. However, these patterns have not addressed sensor network specifically. With a growth of sensors and sensor networks, and considering their profound applicability, there is a crucial need to articulate ones experience of application development or deployment of sensor nodes in the form of design patterns to avoid the future mistakes. This paper discusses the same issue and show applicability of design patterns in sensor networks.

1. Introduction

Shared vocabulary is a basic ingredient of any science or engineering discipline. It helps people of the same domain to exchange ideas more conveniently and easily. This vocabulary makes possible to explain the whole scenario with a single word. Design patterns, first introduced by Christopher Alexander [3], are a way to enrich this lexicon. According to Alexander, solutions of the daily life problems have a specific pattern which may be applied repeatedly to solve a problem. In design patterns, we articulate ones experience in a precise set of suggestions in the form of catalog. This catalog suggests that what measures should be taken and what steps should be avoided to complete any specific task. So that others do not make the same mistakes again and again and don't invest their time to rethink on the same problem. As it is said, good judgment comes from good experience, these experienced enriched catalogues help novice user to get the deeper insight of their relative domains within short period. Alexander proposed design patterns for roads, bridges and architectural world but his statement lies true for almost every aspect of life. Therefore, since its advent, design patterns have proven useful in many engineering disciplines such as Software Engineering, Business Information Processing, and Architectural Design etc.

With the inception of sensor networks, a new class of research topics is unleashed. Many of the differences lie almost in every layer of sensor networks as compare to traditional TCP/IP networks. Since sensor networks are in their evolutionary state, most of the research focus on the bottom four layers i.e. Physical, Link, Network and Transport layers to provide a high level of compatibility to the future applications of sensor networks and leave the application layer unattended. As sensor networks are application specific, we cannot find a generalized way of application development in them. However, with a growth of these miniaturized devices and considering their profound applicability, there is a crucial need of articulation of ones experience, of application development or deployment of sensor nodes, in form of catalog to avoid the future mistakes.

This paper elucidates the use of design patterns in sensor networks. The following section outlines the possible classification of design patterns for sensor networks. An issue which is still in veil and haven't got much attention so far.

2. Classifications

2.1. Identification and documentation of existing design patterns:

Creating a design pattern catalog can be divided in to two steps, identification of existing patterns in sensor network and document them in design pattern way. However, if this step is failed, the next step is to propose a new pattern and document them eloquently. Both of these issues require a great deal of research and would help novice user to grasp the key design issues more easily and swiftly.

The identification of these patterns could be at any level, within a sensor node, within the object(s) controlling the sensor nodes or at a system as a whole. For example, sensor networks software applications have some different trends than general programming paradigm. Strict energy constraints and limited resources compel developers to write application-specific services. Although, specialized software solutions enable developers to build efficient systems, but on the other hand, they have to compromise the reusability of software components. Features like, late binding or polymorphism may also not available in today's Sensor OS platform. The perfect example of this is TinyOS, in which interaction between components is defined at compile time rather at run time [2], as happens in OOP (Object Oriented Programming) realm. Therefore, it is difficult, although possible, for sensor network programmers to apply OOAD (Object Oriented Analysis and Design) or any other software engineering patterns easily and with an assurance that they will get the same results. These factors increase the need of a new set of design patterns specifically tailored for WSN (Wireless Sensor Networks).

2.2 Application of patterns of other domain:

Design patterns are not a new concept and have proven useful in many other engineering fields. Although sensor network have some specialized attributes, even then we can apply design patterns of other domains in sensor networks. For example, sensor networks posses some generalized properties of real time embedded systems. Issues which are common in both domains and have been resolved in real time embedded systems along with documentation can be very helpful for sensor networks as well.

The evident use of design patterns is in OOAD (Object Oriented Analysis and Design) where they are used to manage the objects, their relationships and operations. The major work of this domain is by Erich Gamma et al. [1], commonly known as GoF Design patterns. In [1], they capture the experience in designing object-oriented software as design patterns. They named, explained, and evaluated important recurring designs in object-oriented systems and presented a catalog for OOAD.

Being a member of a real world objects, each sensor can also be represented as an Object of OOP (Object Oriented Programming) realm. Each sensor does the same basic things – sensing the environment and responding it, which is analogous to the 'methods' of an object. Based on these actions sensor may alters its state as well which can be regarded as 'attributes' of an OOP object. Design patterns have proven their potential in the OOP domain; and during modeling if we regard each sensor as an OOP object then the application of similar patterns would enhance the management of sensor networks and we could come up with good sensor network design. Of course, not all of the design patterns of OOAD world are applicable in this scenario because some patterns deals with abstraction as well. However, a larger set of OOAD patterns could help us not only in application programming for sensor networks but also in the management and deployment of sensor networks.

3. Design patterns for Sensor Networks:

This paper mainly elaborates the applicability of the design patterns of other domain on sensor networks. Due to page limitation, we only present one design pattern in detail and briefly discuss few more design pattern in Section 4. In general, each pattern starts with the pattern name, follows with its intent and then outlines its motivation. The following example outlines the motivation with respect to the sensor networks. We believe that this is a succinct but self-explanatory way of documenting one's experience.

3.1. Mediator:

Intent:

- *"Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently."[1]*
- *Design an intercessor to decouple many peers.*

Motivation:

Consider a multi agent system, an agent can be defined as a system which can perceive its environment through sensors and act upon it through effectors [5]. A multi agent system is a collection of these agents. Usually an agent is divided in to different components or modules according to their behavior to enhance reusability. However, such distribution of behavior can result in a structure with many connections between the various components and having too many interconnections tend to reduce the reusability again. Numerous inter-connections make it more difficult for a component to work without the support of others. In addition to that, making significant changes to the overall behavior of the system becomes unnecessarily difficult, since behavior is distributed among many modules. These problems can be avoided by encapsulating collective behavior in a separate mediator module. The mediator controls and co-ordinates the interactions of the various modules within the agent. Each module is only required to know its mediator and all communication with other modules is done indirectly through this channel. The foremost advantages are localization of the overall behavior in one module and the system can easily change its behavior by modifying or replacing just the mediator module.

Another example of mediator can be seen in smart ubiquitous environment where mediator contains the control logic for the entire system. When any new smart appliance is added to a system or any appliance requires a new rule the only place that will require modification would be mediator. Mediator pattern *decouples* all the appliances within the system from each other.

Participants:

Mediator: An intermediary node/component or module which provides an interface for communicating with other nodes/components or modules
Colleagues: Each colleague communicates with its mediator whenever it would have otherwise communicated with another colleague.

Applicable When:

In sensor network scenario is applicable in following situations

- When a set of modules communicate in well-defined but complex ways.
- When it is difficult to reuse a module because it refers to and communicates with many other modules.

Consequences:

Mediator pattern allows system managers to vary and reuse Colleague and Mediator independently. It also simplifies the maintenance of the system and any new functionality can be added at mediator with out affecting colleagues. Moreover, mediator pattern can simplify the communication protocol by replacing many-to-many relationship to one-to-many relationship as it is easy to inspect one-to-many relationship. A drawback of the mediator pattern is that without proper design the mediator itself can become overly complex.

Structure:

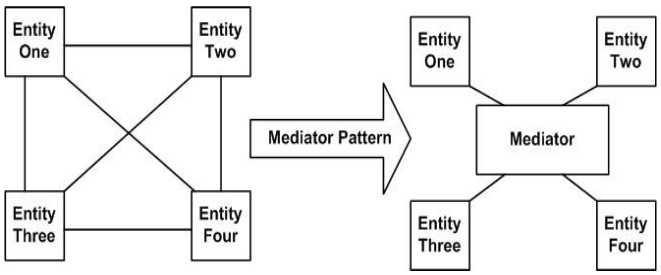

Fig. 1. Mediator pattern

4. Few more patterns at a glance

The intent of this paper is to introduce design patterns for sensor networks, a proven design strategy in many engineering fields and seems promising for sensor networks as well. However above example, due to space limitation, just provides a gist of an idea and might not be able to prove the given claim. Therefore, this section very briefly describes few more patterns and how they can be useful for sensor networks.

Facade: Facade pattern provides a simple interface of a complex system and enforce layering in the architecture. Facade pattern can be useful in many scenarios. For example, in smart ubiquitous environment, as soon as house owner puts his keychain on Smart key stand many related work such as turning on the lights, switching on the answering machine and so on, can be started as a sub layer. Facade makes subsystem easier to use by hiding complex interaction between different components of a given system and provide simplified view to its users.

Chain of responsibility: This pattern is useful when we want to give a chance to more than one entity to handle a request. The intent of this pattern is to avoid coupling between the sender of a request and its receiver [1]. At network design level it can be useful when a user wants to know if some activity has been detected, so if one sensor has not detected any action, it can ask the next sensor, and then whichever sensor has detected the activity, it can respond. Within the sensor node, a vehicle may contain multiple sensors that can accomplish a given task. This pattern provides a way for the sensor to be selectable.

Watchdog [4]: Another specialized observer pattern called Watchdog is also useful for very light weight sanity check, one of the sanity check patterns discussed by Douglas [4]. In the watchdog pattern, an object called a watchdog is signaled periodically by various objects in the system. The lightweigh t nature of this pattern makes it a perfect candidate for sensors and sensor networks however, it has limited appl icability. Specifically, it determines the health of the syst em by receiving events within a specified time window. This is helpful in detecting hung system faults but not o ther kinds of faults. Watchdogs are mainly used for time critical sensor network deployment i.e. the computations have a deadline by which they must be applied. If th e computation occurs after that deadline, the result m ay either be erroneous or irrelevant. It could be data as well that if is it valid after specific period or not. Watchdog provides a mean to apply these sanity chec ks.The list of these patterns can goes up to many pages, in fact it requires a book for a complete discussion. This paper contributed to research community by introducing a proven design methodology for sensor networks along with its motivation. We also highlighted the possible research areas in this field and provided many examples to visualize the idea. As said above the pattern-based approach serves the purpose of providing a common language for expressing design knowledge across disciplines and design layers. It will also help the people of sensor network domain to describe the design decisions taken at various layers in terms of their associated pattern. Patterns can be technically precise; however, they do not adhere to a particular design language and offer a flexible medium for communication between system developers.

5. Conclusions:

In this paper, we introduced the concept of a design pattern for sensor networks. We envision that using design patterns leads to the systems which are scalable, modular, adaptable and understandable. Its use can have a profound impact on the way sensor networks are designed. This paper provides a pioneering work in this regard. This idea is still in evolutionary stages and needs a great deal of refinement. Due to space limitation, we are only able to provide a gist of an idea and discussed very few design patterns. Many useful and valuable patterns are left to be discussed and will be addressed in future papers. Patterns from real time systems like Recursive Containment Pattern, Hierarchical Control Pattern, and others can also be helpful for sensor networks [4]. This document is a result of our ongoing work and only talks about the applicability of the patterns, articulated for other domain, on sensor networks. However, our future work will not only discuss this issue but will also provide some new patterns specifically tailored for WSN.

References

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design Patters: Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1995.

[2] David Gay, Philip Levis, and David Culler, "Software Design Patterns for TinyOS", ACM SIGPLAN/SIGBED 2005 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05), Pages: 40 - 49, 2005.

[3] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel., “A Pattern Language”, Oxford University Press, New York, 1977.

[4] Bruce Powel Douglass, “Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems”, Addison Wesley Professional, 2002.

[5] Stuart Russell and Peter Norvig, “Artificial Intelligence: A Modern Approach, Prentice Hall Series in AI”, New Jersey, Prentice Hall, 1995.