REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF - M'SILA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

DEPARTEMENT OF COMPUTER SCIENCE

N° :……..

DOMAINE: INFORMATICS

FILIERE: SOFTWARE ENGENEERING

OPTION: SOFTWARE ENGENEERING

A dissertation submitted in fulfilment of the requirements

For the Degree of master

By: Bouguerra Bilal

Subject

On the Verification of Internet of Things (IoT) Technologies using PRISM Model Checker

Presented to the jury :

Nom et prénom Enseignant	Université de M'sila	Président
Hichem DEBBI	Université de M'sila	Rapporteur
	Université de M'sila	Examinateur

Academic year : 2016 /2017

Acknowledgements

Praise be to God who enlightened us the path of science and knowledge and helped me to perform this duty and agreed to accomplish this work.

First, my thanks go to my supervisor because he helped me a lot to accomplish this work and wish him success in his future studies because he has great abilities.

Finally, I’m grateful for my family for their unlimited motivation and support.

Contents

CHAPTER 2 : MODEL CHECKING

CHAPTER 3 : GAME THEORY AND WIRELESS SENSOR NETWORKS

CHAPTER 4: MODELLING A WSN GAME THEORY IN PRISM-GAMES

List of figures

General introduction

Model checking is a set of techniques of automatic verification of temporal properties on reactive systems. In this dissertation, we want address the quantitative verification and analysis of Internet of things (Ubiquitous Computing), by taking a case study of the Wireless Sensor Networks (WSNs) that in principle collaborate to achieve some goals, such as reducing consumed energy. Nodes in WSN consume a lot of energy, because sensor nodes majorly depend on batteries for energy, which get depleted at a faster rate because of the computation and communication operations they have to perform.

There is a lot of research on the Wireless Sensor Networks (WSNs), among problems is how to conserve the energy and how estimate energy consumption at work time of WSNs, so the problem is how sensors nodes collaborate to achieve a better of conservation of energy in working time of WSNs. Among popular approaches proposed in WSN, is the use of game theory, which becomes more and more appropriate to deal with this kind of networks due to the aspect of collaboration in WSN.

In this dissertation, we will use probabilistic model checking to study and analyse this problem. We will use an extension of the popular probabilistic model checker PRISM, which is PRISM-Games to introduce a formal model of such communication and collaboration problem, and some specification properties to help us understand the behaviour of the network. This dissertation will be organized as follows:

We initially in chapter 01 give a history of Internet of Things, then we strategic research and the applications of (IoT), and infrastructure management, then we explained networks and communications in (IoT).In this part, we focus on wireless sensor networks because it is the case study of this dissertation. Finally we defineUbiquitous computing.

In chapter 02, we briefly describe system verification, after that we study model checking from different aspects. Then we talk about Markov Chain and the PRISM language. Finally we talk about the stochastic-games.

In chapter 03, we talk about the wireless sensor networks and game theory, showing how Game Theory Meets Wireless Sensor Networks.

In chapter 04, we try to create a model for the conservation of energy in wireless sensor networks, using the tool PRISM-games depending on the paper [14].

CHAPTER 1

INTERNET OF THINGS

1.1. Introduction

The next wave in the era of computing will be outside the realm of the traditional desktop. In the internet of things (IoT) paradigm, many of the objects that surround us will be on the network in one form or another. Radio Frequency Identification (RFID) and sensor network technologies will rise to meet this new challenge, in which information and communication systems are invisibly embedded in the environment around us. This result in the generation of enormous amounts of data which have to be stored, processed and presented in a seamless, effcient and easily interpretable form. This model will consist of services that are comodities and delivred in a manner similar to traditional commodities [1].

1.2. Definition

The internet of things (IoT) is the inter-networking of physical device, vehicles, buildings, and other items embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. The (IoT) allows objects to be sensed or controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit in addition to reduced human intervention [2].

Internet of Things is a concept and a paradigm that considers perva-sive presence in the environment of a variety of things that through wireless and wired connection and unique addressing schemes are able to interact with each other and cooperate with other things to create new applications and reach common goals [3].

The goal of the internet of tings is to enable things to be connected everytime, everywhere, with anything and anyone ideally using any path/network and any service.

1.3. History

The concept of the internet of things was invented by Peter T.Lewis in September 1985 in a speech he delivered at a U.S.Federal Communications Commission supported session at the Congressional Black Caucus 15th Legislative Weekend Conference. Mark Weiser seminal

1991 paper on ubiquitous computing ”The Computer of the 21st Century”, as well as academic venues such as UbiComp and PerCom produced the contemporary vision of IoT. In 1994 Reza Raji described the concept in IEEE Spectrum.

In 2016, the vision of the internet of things has evolved due to a convergence of multiple technologies, including ubiquitous wireless communication [2].

1.4. Internet of things strategic research

1.4.1. Internet of Things Vision

Internet of things is a new revolution of the internet . Objects make themselves recognizable and they obtain intelligence by enabling context related decisions , they can access information that has been aggregated by other things.

Networks of things connect things globally and maintain their identity online. Mobile allows connection to this global infrastructure anytime, anywhere. The result is a globally accessible network of things, users, and consumers.

Enabling technologies for the (IoT) for example : Radio Frequency Identification(RFID), mobile internet, IPv6…etc, we can be classified into three categories : (i) technologies that enable “things“ to acquire contextual information, and (ii) technologies that enable “things“ to process contextual information, and (iii) technologies to improve security and privacy. The first and second categories can jointly understood as functional building blocks required building ’’ intelligence ’’ into ’’ Things ’’. The third categories are not functional but rather a *de facto* requirement, without which the penetration of the IoT would be severely reduced. In 2011 there were over 15 billion things on the web, with 50 billion+ intermittent connections, in the future by 2020 over 30 billion connected things with over 200 billion with intermittent connections are forecast. This biscômes the Internet of everything [3].

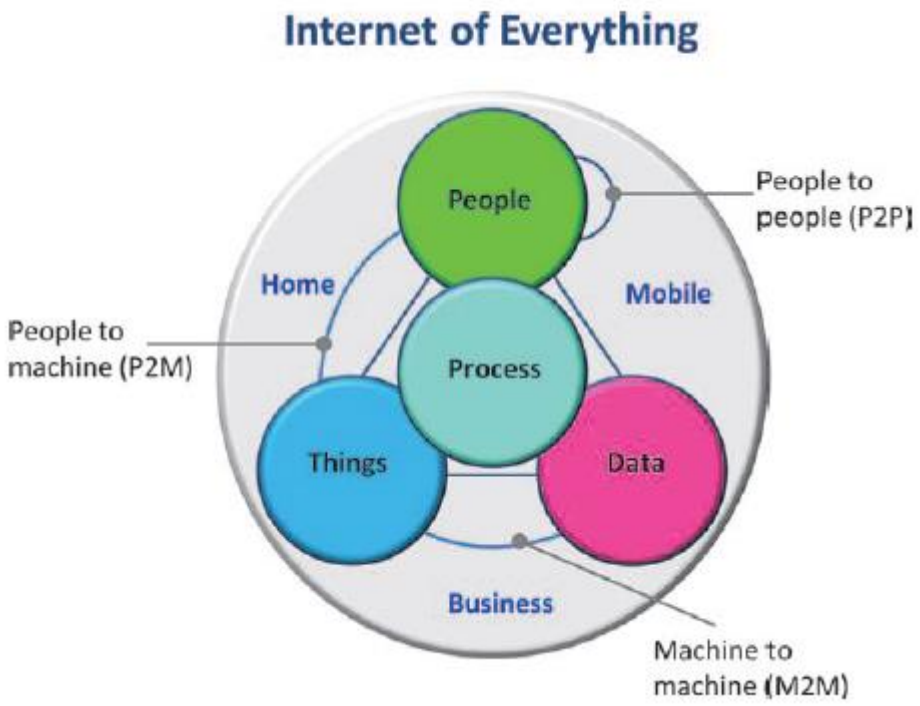


Figure 1-1 Internet of everything [1].

1.4.2. Internet of things Applications

The Internet of things applications are addressing the societal needs and the advancements to enabling technologies such as nanoelectronics and cyber-physical systems continue to be challenged by a variety of technical .

1.4.2.1. Smart Cities

By 2020 we will see the development of Mega city corridors and networked integrated and branded cities. With more than 60 percent of the world population expected to live in urban cities by 2025. This will lead to the evolution of smart cities with eight smart features, including smart economy, smart buildings, smart mobility, smart energy, smart information communication and technology, smart planning and smart citizen. There will be about 40 smart cities globally by 2025.

The role of the smart cities will be crucial for Internet of Things (IoT) deployment, running of the day-to-day city operations and creation of city development strategies will drive the use of the (IoT). Therefore, cities and their service represent an almost ideal platform for (IoT) research, taking into account city requirements and transferring them to solutions enabled by (IoT) technology.

The figure depicts several commons actions that may take place in the smart day, highlighting in each occasion which domain applies [3].

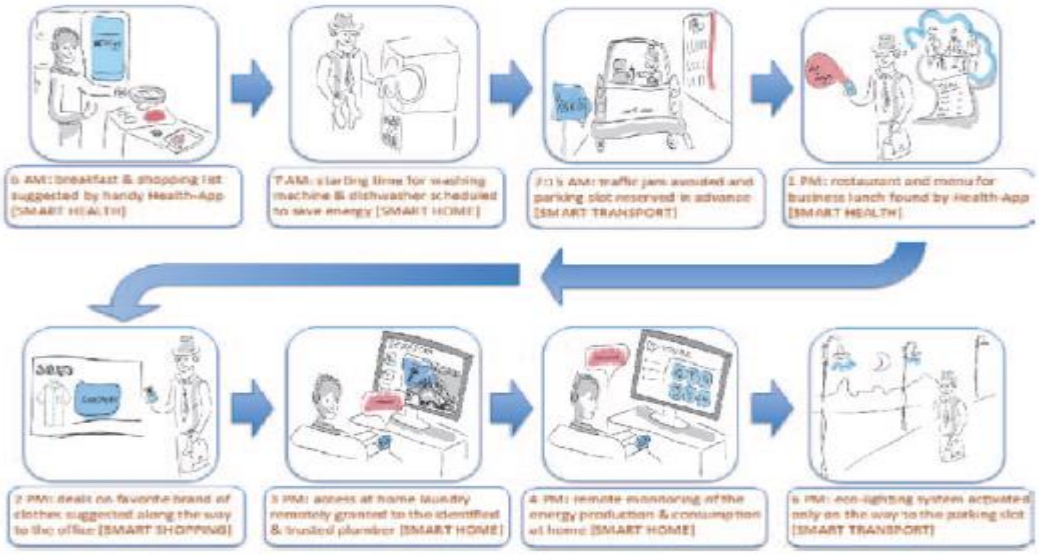


Figure 1-2 A day in the life of a typical European citizen of a smart city **[1].**

1.4.2.2. Smart Energy and the Smart Grid

There is increasing public awareness about the changing paradigm of our pol-icy in energy supply, consumption and infrastructure. For several reasons our future energy supply should no longer be based on fossil resources.

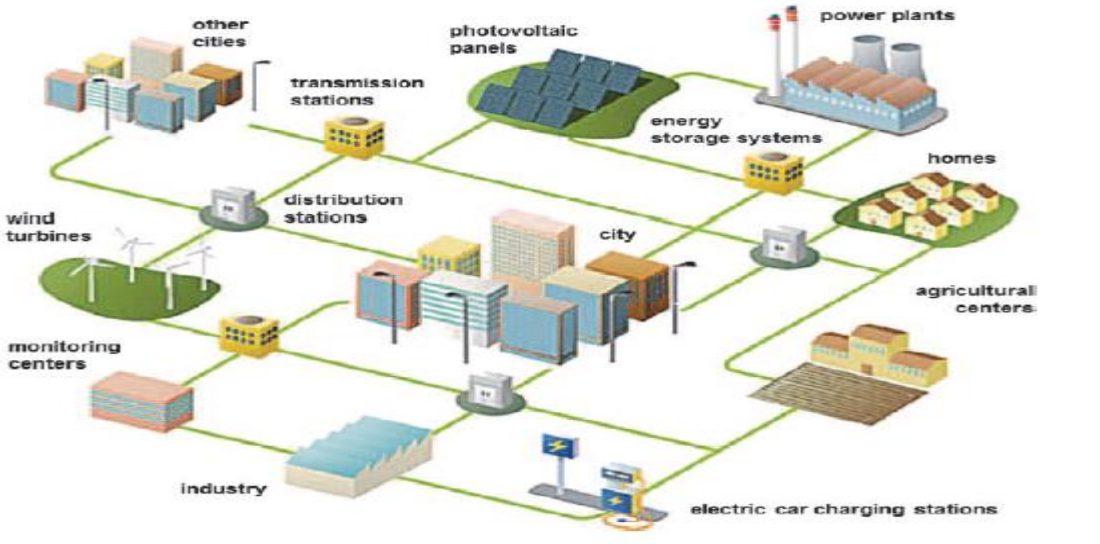


Figure 1-1 Smart grid representation[1].

1.4.2.3. Smart Home

The rise of Wi-Fi is role in home automation has primarily come about due to the networked nature of deployed electronics where electronic devices (mobile device , AV receivers …etc).

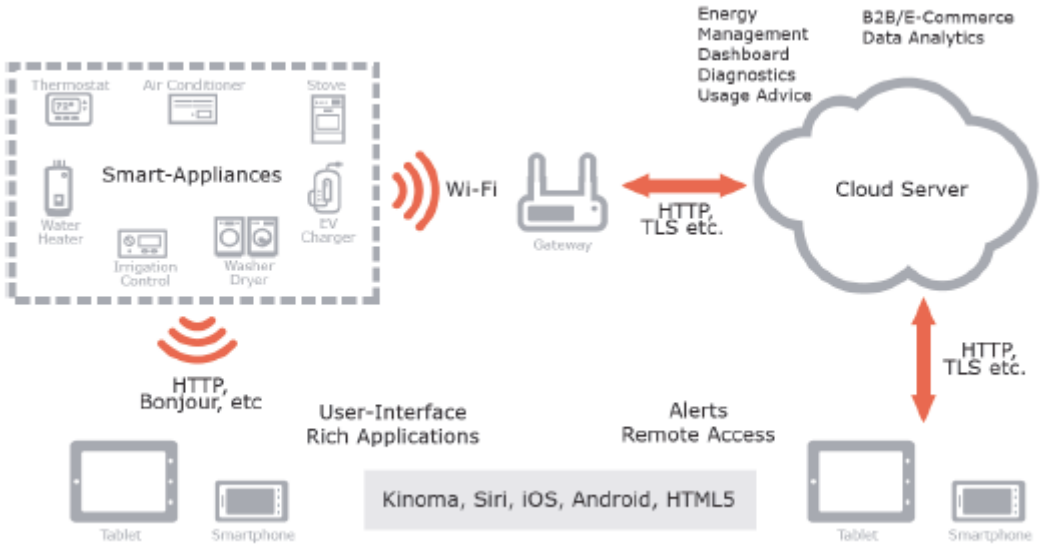


Figure 1-4 Smart home platform [1].

1.5. Infrastructure management:

The Internet of Things will become part of the fabric of everyday life. It will become part of our overall infrastructure just like water, electricity, telephone, TV and most recently the internet. Whereas the current Internet typically connects full-scale computers, the Internet of Things will connect everyday objects with a strong integration into the physical world.

1.5.1. Plug and Play Integration

This is only possible if connecting a thing to the internet of things becomes as a simple as plugging it in and switching it on. Such plug and play functionality requires an infrastructure that supports it, starting from the networking level and going beyond it to the application level. On the networking level, the Plug and Play functionality has to enable the communication, features like the ones provided by IPv6 are in the directions to help in this process [3].

1.5.2. Infrastructure Functionality

The infrastructure needs to support applications in finding the things required. An application may run anywhere, including on the things themselves. Finding things is not limited to the star-up time of an application. Automatic adaptation is needed whenever relevant new things become available, things become unavailable or the status of the things changes. The infrastructure has to support the monitoring of such changes and the adaptation that is required as a result of the changes [3].

1.5.3. Physical Location

As the Internet of Things is strongly rooted in the physical world, the notion of physical location and position are very important, especially for finding things, but also for deriving knowledge. Therefore, the infrastructure has to support finding things according to location (e.g. geo-location based discovery). Taking mobility into account, localization technologies will play an important role for the Internet of Things and may become embedded into the infrastructure of the Internet of Things [3].

1.5.4. Security and Privacy

In addition, an infrastructure needs to provide support for security and privacy functions including identification, confidentiality, integrity, non-repudiation authentication and authorization. Here the heterogeneity and the need for interoperability among different ICT systems deployed in the infrastructure and the resource limitations of (IoT) devices (e.g., Nano sensors) have to be taken into account [3].

1.6. Networks and Communication

Present communication technologies span the globe in wireless and wired networks and support global communication by globally-accepted communication standards. The evolution and pervasiveness of present communication technologies has the potential to grow to unprecedented levels in the near future by including the world of things into the developing Internet of Things.

1.6.1. Wireless Networks

Wireless networks especially will grow largely by adding vast amounts of small Internet of Things devices with minimum hardware, software and intelligence, limiting their resilience to any imperfections in all their functions. Based on the research of the growing network complexity, caused by the Internet of Things, predictions of traffic and load models will have to guide further research on unfolding the predicted complexity to real networks, their standards and on-going implementations. The idea of internet of things (IoT) was developed in parallel to WSNs.

1.6.2. Wireless Sensors Networks (WSN)

A wireless sensor network (WSN) is a network formed by a large number of sensor nodes where each node is equipped with a sensor to detect physical phenomena such as light, heat, pressure, etc. WSNs are regarded as a revolutionary information gathering method to build the information and communication system which will greatly improve the reliability and

efficiency of infrastructure systems. Compared with the wired solution, WSNs feature easier deployment and better flexibility of devices. With the rapid technological development of sensors, WSNs will become the key technology for IoT. (Book: Internet of things: wireless sensor networks) [4].

1.6.2.1. Characteristic of WSNs:

- A WSN can generally be described as a network of nodes that cooperatively sense and control the environment, enabling interaction between persons or computers and the surrounding environment.
- WSNs nowadays usually include sensor nodes, actuator nodes, gateways and clients.
- A large number of sensor nodes deployed randomly inside of or near the monitoring area (sensor field), form networks through self-organization.
- During the process of transmission, monitored data may be handled by multiple nodes to get to gateway node after multi-hop routing.
- Reach the management node through the internet or satellite [4].

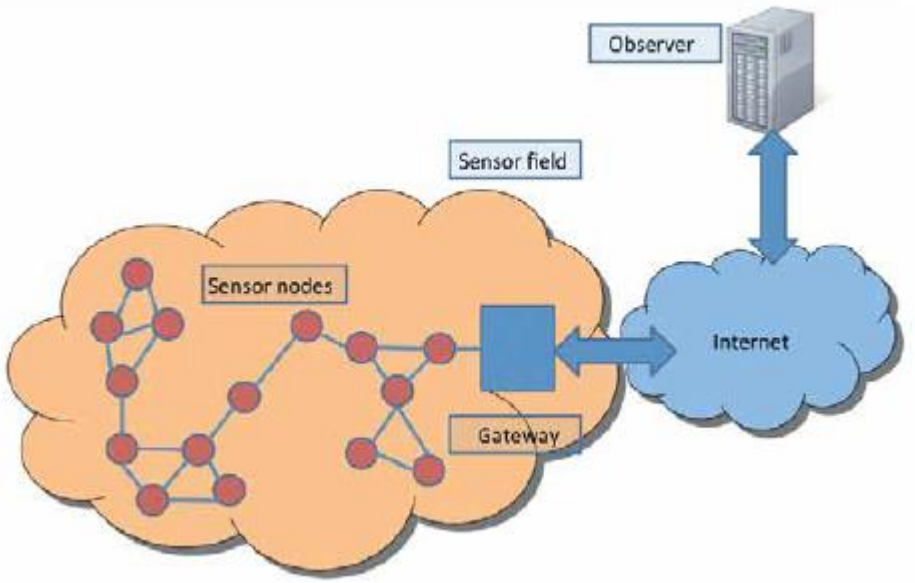


Figure 1-5 Wireless sensor networks [4].

1.6.2. 2. Sensor nodes of WSNs

The sensor node is one of the main parts of a WSN. The hardware of a sensor node generally includes four parts: the power and power management module, a sensor, a microcontroller, and a wireless transceiver, see Figure

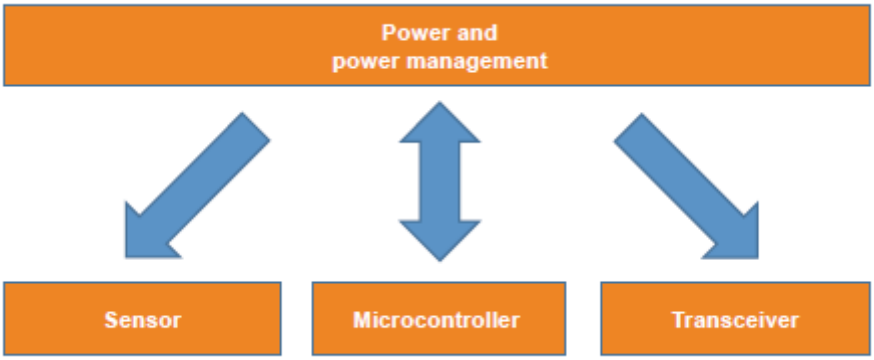


Figure 1-6 Hardware structure of a WSN sensor node [3].

The power module offers the reliable power needed for the system. The sensor is the bond of a WSN node which can obtain the environmental and equipment status. A sensor is in charge of collecting and transforming the signals, such as light, vibration and chemical signals, into electrical signals and then transferring them to the microcontroller [4].

1.6.2.3. Topology of WSNs

A WSN consists of a number of sensor network nodes and a gateway for the connection to the internet. The general deployment process of a WSN is as follows (see Figure).

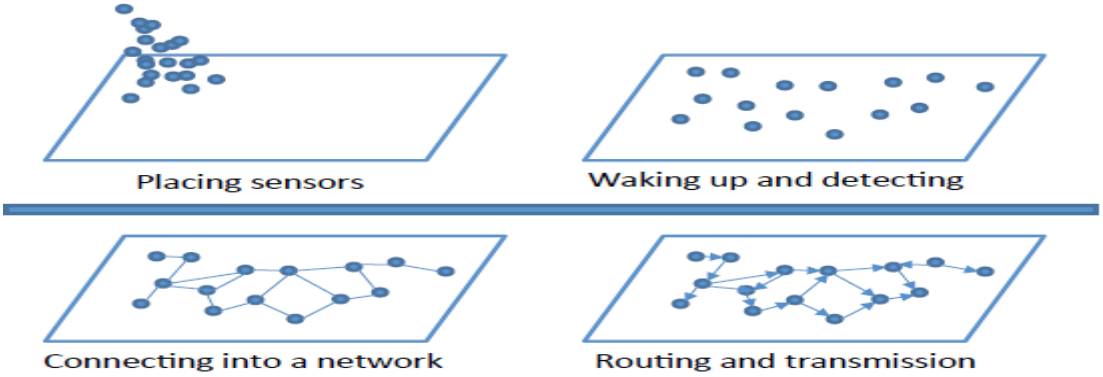


Figure 1-7 Organizing and transmitting process of WSNs [4].

Firstly, the sensor network nodes broadcast their status to the surroundings and receive status from other nodes to detect each other. Secondly, the sensor network nodes are organized into a connected network according to a certain topology (linear, star, tree, mesh, etc.). Finally, suitable paths are computed on the constructed network for transmitting the sensing data [4].

1.6.3. Self-organizing and reliable networking technology

The positions of WSN nodes are random, and the nodes can be moved, sheltered and interfered. The self –organizing management approach of network nodes can greatly improve the robustness of the network, resulting in a smart mesh networking technology, as shown in Figure.

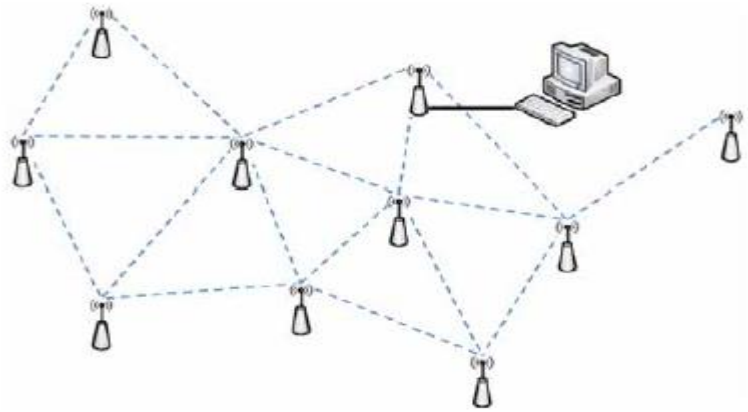


Figure 1-8 Mesh self-organizing network [4].

1.6.4. Self-adaptive flow control technology

In WSNs, the communication between nodes is susceptible to interference and occlusion, resulting in signal transmission failure. The principle of flow control is that the data sender adjusts the sending traffic according to the loss situation of data transmission. When data loss occurs, the sender decreases the transmission rate. And when the data is not lost, the sender increases the transmission rate. Such flow control mechanisms are no longer suitable for WSNs. Adaptive flow control checks the reason for packet loss and adjusts the transmission flow [4].

1.6.5. Low cost IP interconnection technology

The address length was relatively short and suitable for implementing in low-power embedded sensor network nodes. However, the internal address management method is not compatible with the IP method of the internet, which increased the difficulty of interacting between the sensor network nodes and the traditional IP network nodes. Traditional IPv4 addresses have been gradually depleted, and the new IPv6 technology has an enormous address resource which is suitable for a wide range of sensor network deployment. As a result, 6LoWPAN low-power wireless technology based on IPv6 has emerged [4].

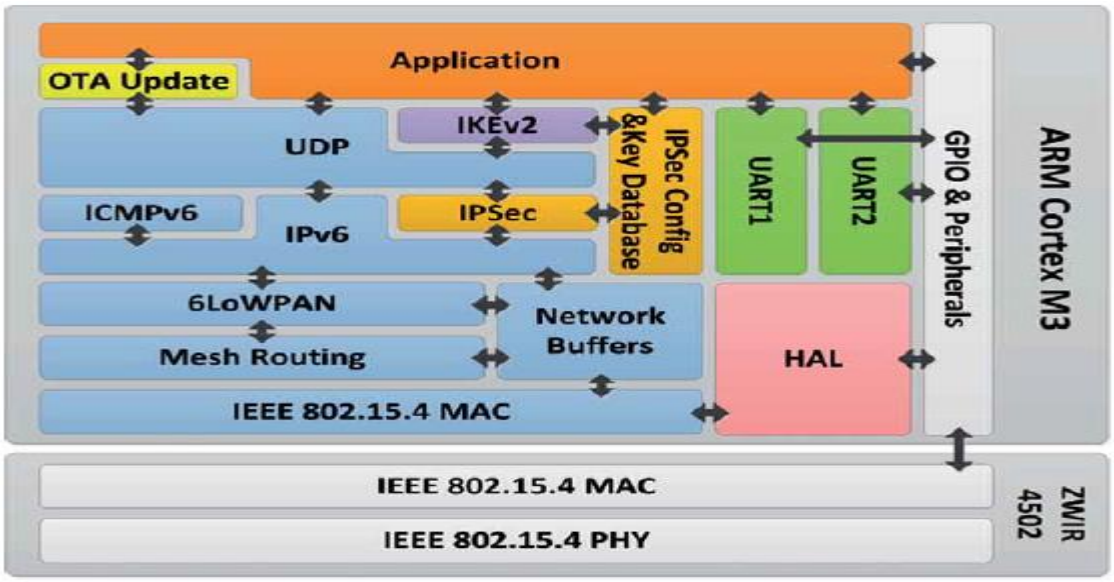


Figure 1-9 6LoWPAN protocol stack [4].

1.7. Platforms of WSNs

1.7.1. Hardware

One major challenge in a WSN is to produce *low cost* and *tiny* sensor nodes. There are an increasing number of small companies producing WSN hardware and the commercial situation can be compared to home computing in the 1970s. Many of the nodes are still in the research and development stage, particularly their software. Also inherent to sensor network adoption is the use of very low power methods for radio communication and data acquisition.

In many applications, a WSN communicates with a Local Area Network or Wide Area Network through a gateway. The Gateway acts as a bridge between the WSN and the other network. This enables data to be stored and processed by devices with more resources, for example, in a remotely located server [4].

1.7.2. Software

Energy is the scarcest resource of WSN nodes, and it determines the lifetime of WSNs. WSNs may be deployed in large numbers in various environments, including remote and hostile regions, where ad hoc communications are a key component. For this reason, algorithms and protocols need to address the following issues:

- Increased lifespan
- Robustness and fault tolerance
- Self-configuration

Lifetime maximization: Energy/Power Consumption of the sensing device should be minimized and sensor nodes should be energy efficient since their limited energy resource determines their lifetime. To conserve power, wireless sensor nodes normally power off both the radio transmitter and the radio receiver when not in use [4].

1.7.3. Online collaborative sensor data management platforms

Online collaborative sensor data management platforms are on-line database services that allow sensor owners to register and connect their devices to feed data into an online database for storage and also allow developers to connect to the database and build their own applications based on that data. Examples include Xively and the Wikisensing platform. Such platforms simplify online collaboration between users over diverse data sets ranging from energy and environment data to that collected from transport services. Other services include allowing developers to embed real-time graphs & widgets in websites; analyse and process historical data pulled from the data feeds; send real-time alerts from any datastream to control scripts, devices and environments.

The architecture of the Wikisensing platform describes the key components of such systems to include APIs and interfaces for online collaborators, a middleware containing the business logic needed for the sensor data management and processing and a storage model suitable for the efficient storage and retrieval of large volumes of data [4].

1.8. Data Management of (IoT)

Data management is a crucial aspect in the Internet of Things. When considering world of objects interconnected and constantly exchanging all types of information, the volume of the generated data and the processes involved in the handling of those data become critical. There are many technologies and factors involved in the “data management” within the (IoT) context [3].

Some of the most relevant concepts which enable us to understand the challenges and opportunities of data management are:

1.8.1. Data Collection and Analysis (DCA)

Data Collection and Analysis modules or capabilities are the essential components of any (IoT) platform or system, and they are constantly evolving in order to support more features and provide more capacity to external components (either higher layer applications leveraging on the data stored by the DCA module or other external systems exchanging information for analysis or processing).

The DCA module is part of the core layer of any IoT platform. Some of the main functions of a DCA module are:

- **User/customer data storing:**

Provides storage of the customer's information collected by sensors.

- **User data and operation modeling**: Allows the customer to create new sensor data models to accommodate collected information and the modeling of the supported operations [3].
- **Customer rules/filtering:** Allows the customer to establish its own filters and rules to correlate events.
- **Customer task automation:** Provides the customer with the ability to manage his automatic processes. Example: scheduled platform originated data collection.

1.8.2. Big Data

Big data is about the processing and analysis of large data repositories, so disproportionately large that it is impossible to treat them with the conventional tools of analytical databases. Some statements suggest that we are entering the "Industrial Revolution of Data" where the majority of data will be stamped out by machines. These machines generate data a lot faster than people can, and their production rates will grow exponentially with Moore's Law. Storing this data is cheap, and it can be mined for valuable information. Examples of this tendency include: Web logs, RFID, Sensor networks….etc.

Big data requires exceptional technologies to efficiently process large quantities of data within a tolerable amount of time. Technologies being applied to big data include massively parallel processing (MPP) databases, data-mining grids, distributed file systems, distributed databases, cloud computing platforms, the Internet, and scalable storage systems. These technologies are linked with many aspects derived from the analysis of natural phenomena such as climate and seismic data to environments such as health, safety or the business environment [3].

1.9. Ubiquitous Computing

1.9.1. Introduction

The effort by researchers to create a human-to-human interface through technology in the late 1980 resulted in the creation of the Ubiquitous Computing, whose objective is to embed technology into the back ground of everyday life. Currently, we are in the post-PC era where smart phones and other handheld devices are changing our environment by making it more interactive as well as informative. Mark Weiser, the forefather of Ubiquitous Computing,

defined a smart environment as the physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network [5].

The creation of the internet has marked a foremost milestone towards achieving Ubiquitous Computings vision which enables individual devices to communicate with any other device in the world. The inter-networking reveals the potential of a seemingly endless amount of distributed computing resources and storage owned by various owners.

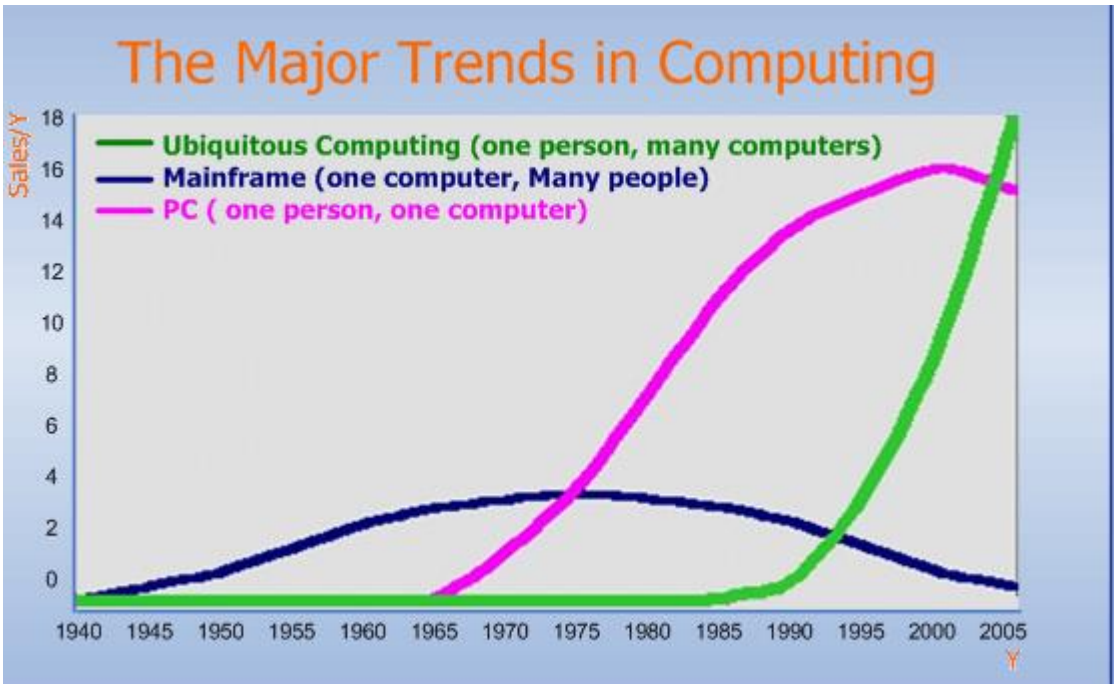


Figure 1-10 The Major Treands in Computing [5].

1.9.2 What is UbiComp (Ubiquitous Computing) ?

Ubiquitous computing (or "**ubicomp**") is a concept in software engineering and computer science where computing is made to appear anytime and everywhere. In contrast to desktop computing, ubiquitous computing can occur using any device, in any location, and in any format. A user interacts with the computer, which can exist in many different forms, including laptop computers, tablets and terminals in everyday objects such as a fridge or a pair of glasses. The underlying technologies to support ubiquitous computing include Internet, advanced middleware, operating system, mobile code, sensors, microprocessors, new I/O and user interfaces, networks, mobile protocols, location and positioning and new materials.

Ubiquitous computing touches on a wide range of research topics, including distributed computing, mobile computing, location computing, mobile networking, context-aware computing, sensor networks, human-computer interaction, and artificial intelligence.

The goal of Ubiquitous computing is create an environment where the connectivity of devices is embedded in such a way that the connectivity is unobtrusive and always available [6].

1.9.3. History:

Mark Weiser is the father of ubiquitous computing, he coined the phrase « Ubiquitous Computing » around 1988, during his tenure as Chief Technologist of the Xerox Palo Alto Research Center (PARC). Both alone and with PARC Director and Chief Scientist John Seely Brown, Weiser wrote some of the earliest papers on the subject, largely defining it and sketching out its major concers [6].

1.9.4. Core concepts of UbiComp

Ubiquitous computing presents challenges across computer science: in systems design and engineering, in systems modelling, and in user interface design. Contemporary human-computer interaction models, whether command-line, menu-driven, or GUI-based, are inappropriate and inadequate to the ubiquitous case.

Mark Weiser proposed three basic forms for ubiquitous system devices: tabs, pads and boards.

- Tabs: wearable centimeter sized devices
- Pads: hand-held decimeter-sized devices
- Boards: metre sized interactive display devices.

These three forms proposed by Weiser are characterized by being macro-sized, having a planar form and on incorporating visual output displays. If we relax each of these three characteristics we can expand this range into a much more diverse and potentially more useful range of ubiquitous computing devices. Hence, three additional forms for ubiquitous systems have been proposed.

- Dust: miniaturized devices can be without visual output displays, e.g. Micro Electro-Mechanical Systems (MEMS), ranging from nanometers through micrometers to millimetres.
- Skin: fabrics based upon light emitting and conductive polymers, organic computer devices, can be formed into more flexible non-planar display surfaces and products such as clothes and curtains, see OLED display. MEMS device can also be painted onto various surfaces so that a variety of physical world structures can act as networked surfaces of MEMS.
- Clay: ensembles of MEMS can be formed into arbitrary three dimensional shapes as artefacts resembling many different kinds of physical object [5].

Ubiquitous computing may be seen to consist of many layers, each with their own roles, which together form a single system:

Layer 1: task management layer

- Monitors user task, context and index
- Map user's task to need for the services in the environment
- To manage complex dependencies

Layer 2: environment management layer

- To monitor a resource and its capabilities
- To map service need, user level states of specific capabilities

Layer 3: environment layer

- To monitor a relevant resource
- To manage reliability of the resources

1.9.5. Examples

One of the earliest ubiquitous systems was artist Natalie Jeremijenko's "Live Wire", also known as "Dangling String", installed at Xerox PARC during Mark Weiser's time there. This was a piece of string attached to a stepper motor and controlled by a LAN connection; network activity caused the string to twitch, yielding a *peripherally noticeable* indication of traffic. Many of mobile phones supporting high speed data transmission, video services, and mobile devices with powerful computational ability. Although these mobile devices are not necessarily manifestations of ubiquitous computing, there are examples, such as:

- Japan's Yaoyorozu ("Eight Million Gods") Project in which mobile devices, coupled with radio frequency identification tags demonstrate that ubiquitous computing is already present in some form.
- Ambient Devices has produced an "orb", a "dashboard", and a "weather beacon": these decorative devices receive data from a wireless network and report current events, such as stock prices and the weather, like the Nabaztag produced by Violet Snowden.
- The Australian futurist Mark Pesce has produced a highly configurable 52-LED LAMP enabled lamp which uses Wi-Fi named MooresCloud after Moore’s Law.

1.10. Conclusion

In the future, computation will be human centered. It will be freely available everywhere, like batteries and power sockets, or oxygen in the air we breathe...We will not need to carry our own devices around with us. Instead, configurable generic devices, either handheld or embedded in the environment, will bring computation to us, whenever we need it and wherever we might be. As we interact with these "anonymous" devices, they will adopt our information personalities. They will respect our desires for privacy and security. We won't have to type, click, or learn new computer jargon. Instead, we'll communicate naturally, using speech and gestures that describe our intent [6].

CHAPTER 2

MODEL CHECKING

2.1. System verification

System Verification is a set of actions used to check the *correctness* of any element, such as a system element, a system, a document, a service, a task, a requirement, etc. These types of actions are planned and carried out throughout the life cycle of the system. Verification is a generic term that needs to be instantiated within the context it occurs. As a process, verification is a transverse activity to every life cycle stage of the system [9].

System verification techniques are being applied to the design of ICT systems in a more reliable way. Briefly, system verification is used to establish that the design or product under consideration possesses certain properties. The properties to be validated can be quite elementary, e.g., a system should never be able to reach a situation in which no progress can be made (a deadlock scenario), and are mostly obtained from the system's specification. This specification prescribes what the system has to do and what not, and thus constitutes the basis for any verification activity. A defect is found once the system does not fulfill one of the specification's properties. The system is considered to be "correct" whenever it satisfies all properties obtained from its specification. So correctness is always relative to a specification, and is not an absolute property of a system. A schematic view of verification is depicted in Figure [7].

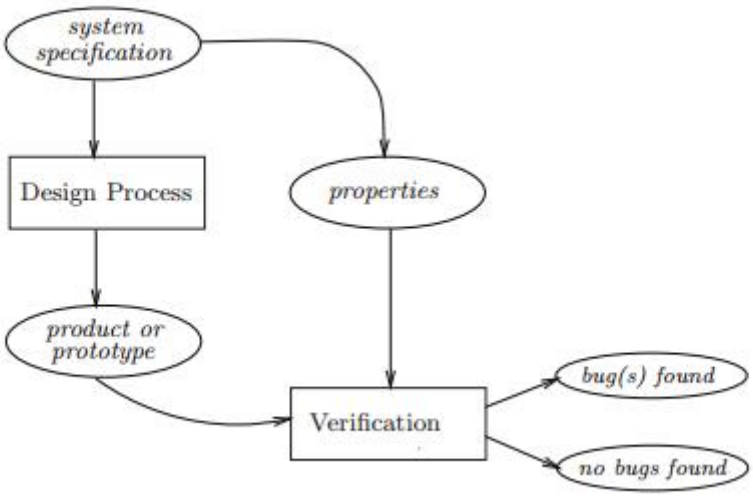


Figure 2.1 Schematic view of a posteriori system verification [8].

2.1.1. Why Verify?

In the context of human realization, any human thought is susceptible to error. This is also the case with any engineering activity. Studies in human reliability have shown that people trained to perform a specific operation make around 1-3 errors per hour in best case scenarios. In any activity, or resulting outcome of an activity, the search for potential errors should not be neglected, regardless of whether or not one thinks they will happen or that they should not happen; the consequences of errors can cause extremely significant failures or threats [7].

The definition of a verification action applied to an engineering element includes the following:

- Identification of the element on which the verification action will be performed
- Identification of the reference to define the expected result of the verification action

The performance of a verification action includes the following:

- Obtaining a result by performing the verification action onto the submitted element
- Comparing the obtained result with the expected result
- Deducing the degree of correctness of the element.

2.1.2. Software Verification

Peer reviewing and testing are the major software verification techniques used in practice. Apeer review amounts to a software inspection carried out by a team of software engineers that preferably has not been involved in the development of the software under review. The uncompiled code is not executed, but analyzed completely statically. Empirical studies indicate that peer review provides an effective technique that catches between 31 % and 93 % of the defects with a median around 60%. While mostly applied in a rather ad hoc manner, more dedicated types of peer review procedures, e.g., those that are focused at specific error-detection goals, are even more effective. Despite its almost complete manual nature, peer review is thus a rather useful technique. It is therefore not surprising that some form of peer review is used in almost 80% of all software engineering projects. Due to its static nature, experience has shown that subtle errors such as concurrency and algorithm defects are hard to catch using peer review [9].

2.1.3. Hardware Verification

Preventing errors in hardware design is vital. Hardware is subject to high fabrication costs; fixing defects after delivery to customers is difficult, and quality expectations are high. Whereas software defects can be repaired by providing users with patches or updates –

nowadays users even tend to anticipate and accept this hardware bug fixes after delivery to customers are very difficult and mostly require fabricating and redistribution [9].

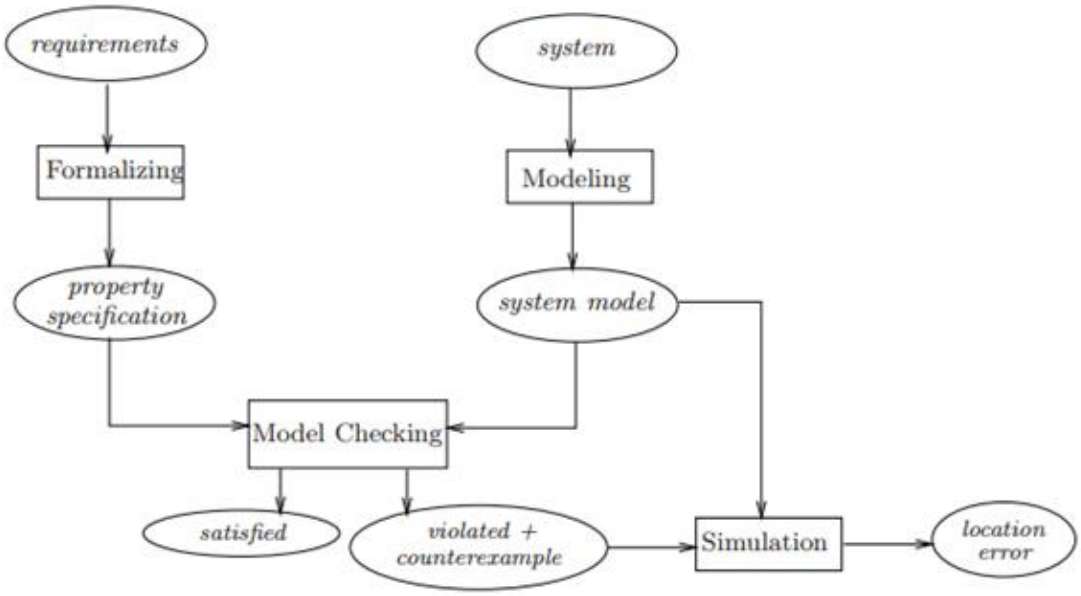
2.2. Model checking

With the evolution of modern technologies dependability of organizations and individuals over software applications is remarkably increased. In many situations these software applications are critical for the safety of individuals and the environment in which they are operating. So it has become a strict requirement that these applications must be fail safe and error free. But due to the complex nature of software, this requirement cannot be achieved 100%. There exist different testing strategies but it is very expensive, time consuming and difficult to make a complete and comprehensive list of testing scripts. To cope with these limitations formal verification technique called model checking can be used. By using formal verification techniques verification activity can be initiated very early in the design process which makes verification activity more effective and less time consuming [7].

Models describing the behavior of the system in a precise and clear manner are used by the model based verification techniques. Correct modeling of the system behavior can discover the inconsistencies in system behavior with respect to its specifications.

In model checking all possible systems scenarios and states are explored in a brute force manner just like the chess program that examines all possible moves. By using this way of verification it can be proved that the system truly satisfies certain properties. Certain errors which remain undiscovered during simulation, testing and emulation can be potentially discovered by using model checking. As the name implies, model checking checks the model of the system that is an abstract and high level description of the system, it does not checks the actual program of the system. It is a great challenge for the developers to make a model of the system that truly represents the actual system. The main obstacle in model checking is the abstraction of the software application. It states that during abstracting the real world problem, the important and critical aspects of the system model should not be left out. The abstraction should be comprehensive enough to contain all the critical aspects of the system. Otherwise, the model checker will be useless if it cannot explore the states which could possibly give rise to errors and compromise the safety in some critical software applications.

Model checking is a software verification technique that uses model of the software application and tries to verify certain properties on the software model during execution. Precisely it can be said that model checking process is basically composed of designing system model, properties definition, running the model checker and analyzing the results [7].



Fi

Figure 2-2 Schematic view of the model-checking approach [8].

2.2.1. The Model-Checking Process

Following are the different phases in model checking process:

2.2.1.1. Designing system model

Model description language provided by the model checking tool under hand is used to model the system under consideration. Designing a system model usually involves making finite state machines which specify desired behavior of the system that can be defined in terms of variables, initial values for the variables, environmental assumptions, and description of conditions under which values of variables will change.

2.2.1.2. Properties definition

Properties of the system to be verified against the system model are defined by using some temporal logic constraints for example linear temporal logic (LTL) or computation tree logic (CTL). Each model checking tool supports different kind of property specification language.

2.2.1.3. Running the model checker

Model checker is executed to check the validity of the defined properties against the system model. Results of the model checker are stored to be analyzed at the later stages.

2.2.1.4. Analyzing the results:

Results of the model checker are analyzed. If the current property for which model checker was executed is satisfied by the system model then check the next property if there is any. And if the property is violated by the system model then analyze the counter example or error

path generated by the model checker and based upon that refine the system model, design of system or may be the property. Repeat the entire above procedure until all the properties are satisfied by the system model. During model checking lot of states are generated by the model checker so there is a possibility of getting out of memory error. In such case either reduces the system model or use more sophisticated techniques for model checking [7].

1. analyze generated counterexample by simulation;

2. refine the model, design, or property;

3. repeat the entire procedure.

- out of memory?→try to reduce the model and try again

- property satisfied?→check next property (if any);

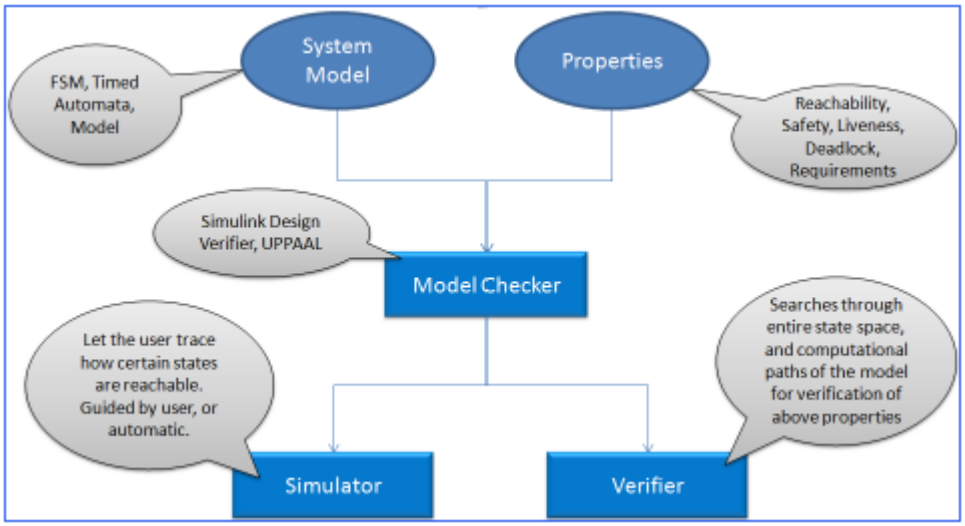


Figure 2.3 Model Checking Architecture [8]

2.3. Types of model checkers

Based on the model specification and state space, there exist different types of model checkers. Mainly the model checkers can be divided into two categories

1) Explicit model checkers.

2) Implicit model checkers.

Explicit model checkers construct a searchable representation of the design model and store a representation of each state visited. Implicit model checkers also called symbolic model checkers use logical representations of sets of states (such as binary decision diagrams) to describe the regions of model state space that satisfy the properties being evaluated. Such

compact representations generally allow symbolic model checkers to handle a much larger state space than explicit model checkers.

Some recently developed model-checkers use satisfiability modulo theories (SMT) solvers for reasoning about infinite state models containing real numbers and unbounded arrays. These model checkers use a form of induction over the state transition relation to automatically prove that a property holds over all executable paths in a model. While these tools can handle a larger class of models, the properties to be checked must be written to support inductive proof [8].

2.4. Model Checking Tools

Different types of temporal logics are used by the model checking tools to express the properties of system as logical formulas to be verified on the system model. Most of model checking tools use either linear time or branching time logics. Properties that can be expressed in either of these logics they can also be expressed in the other one as well. But some of the tools use different types of logic to formulate the properties [8].

- **Linear Time:**

Linear time also called linear temporal logic or linear-time temporal logic (LTL) is a modal temporal logic with modalities that refer to time. In LTL formulas future can be encoded. For example a condition will finally be true; a condition will be true until another fact becomes true, etc.

- **Branch Time:**

Branch time also called computation tree logic (CTL) is branching-time logic. Its model of time is a structure like tree where the future is not determined. There are different paths in the future that can be followed; any one of them can be selected as future path. In CTL it can be specified that when an initial condition is true then all possible executions of a program avoid some unwanted state or condition.

- **Other:**

There are model checking tools which use different type of logics to formulate the properties of the system to be verified as mentioned in the table below in the column named as other, next to the column named linear time.

- **Real Time:**

Real time systems are the systems that operate under time sensitive environment and provide guaranteed response within strict time constraints. Real time column shows that the tool supports the modeling, verification and analyses of real time systems.

- **Probabilistic:**

Probability is an important component in the design and analysis of complex systems. Probabilistic model checking is a formal verification technique for modeling and analyzing the systems.

Probabilistic column in the table below shows the tools that support the modeling and analysis of systems that exhibit probabilistic behavior.

- **Hybrid:**

Hybrid column shows that the tools marked as hybrid supports both real time and probabilistic model checking approaches.

- **GUI:**

Graphical user interface (GUI) column shows that which of the tools provide graphical user interface for modeling the system and for formulating the properties of the system to be verified.

- **Availability:**

Availability column shows that under what conditions the tools is available for use. Availability of the model checking tools can be divided into three types

1. Free - Free means that the tools are freely available to be downloaded and to be used for all kinds of users.

2. Free - Under condition - Free under condition means that tool is available to be downloaded and used for the academic purposes but if someone wants to use it for the commercial purposes then licenses are required.

3. Commercial - Commercial means that license must be acquired for downloading and using the model checking tool [8].

2.5. Advantages of Model Checking

Model Checking has a number of advantages compared to other verification techniques such as automated theorem proving or proof checking. A partial list of some of these advantages is given below:

- No proofs! The user of a Model Checker does not need to construct a correctness proof. In principle, all that is necessary is for the user to enter a description of the circuit or program to be verified and the specification to be checked and press the \return" key. The checking process is automatic.
- Fast In practice, Model checking is fast compared to other rigorous methods such as the use of a proof checker, which may require months of the user's time working in interactive mode.
- Diagnostic counterexamples, If the specification is not satisfied, the Model Checker will produce a counterexample execution trace that shows why the specification does not hold). It is impossible to overestimate the importance of the counterexample feature. The counter examples are invaluable in debugging complex systems. Some people use Model Checking just for this feature.
- Temporal Logics can easily express many of the properties that are needed for reasoning about concurrent systems. This is important because the reason some concurrency property holds is often quite subtle, and it is difficult to verify all possible cases manually [7].

2.6. Markov Chain

A Markov chain is a stochastic process with the Markov property. The term "Markov chain" refers to the sequence of random variables such a process moves through, with the Markov property defining serial dependence only between adjacent periods (as in a "chain"). It can thus be used for describing systems that follow a chain of linked events, where what happens next depends only on the current state of the system. The system's state space and time parameter index needs to be specified. The following table gives an overview of the different instances of Markov processes for different levels of state space generality and for discrete time vs. continuous time [14].

	Countable state space	**Continuous or general state space**
Discrete-time	(discrete-time) Markov chain on a countable or finite state space	Harris chain (Markov chain on a general state space)
Continuous-time	Continuous-time Markov process or Markov jump process	Any continuous stochastic process with the Markov property, e.g. the Wiener process

Table 2.1 Discrete Time VS. Continuous Time [9].

2.6.1. Probabilistic Models

Definition. (Discrete Time Markov Chain (DTMC)) A Discrete-Time Markov Chain (DTMC) is a tuple D = (S, sinit, P , L) , such that S is a finite set of states, sinit ∈ S the initial state, P : S × S → [0, 1] represents the transition probability matrix, L : S → 2AP is a labeling function that assigns to each state s ∈ S the set L(s) of atomic propositions[11].

Definition (Continuous Time Markov Chain (CTMC)) A Continuous Time MarkovChain (CTMC) is a tuple C = (S, sinit, ℜ, L) , such that S is a finite set of states, sinit ∈ S the initial state, ℜ : S × S → R>0 represents the transition rate matrix, L : S → 2AP is a labeling function that assigns to each state s ∈ S the set L(s) of atomic propositions Comparing to DTMC, the main difference is that with DTMC we have the transition probability matrix that corresponds to discrete-time steps, whereas with CTMC, the transition can occur in real-time, and thus are presented by the transition rate matrix, where every time rate of transition from s to s′ is given by ℜ(s, s′). This parameter represents a negative exponential distribution that contributes to computing the transition probability within time units [11].

Definition (Markov Decision Process (MDP)) A Markov Decision Process (MDP)

is a tuple M = (S, S init, A, P , L), where S is a finite set of states, s init ∈ S is the initial state, A is a set of actions , P : S × A × S → [0, 1] is a probability transition function

such that for every state s ∈ S and an action α ∈ A : ∑s′∈S P(s, α , s′) ∈ {0, 1}, and L : S → 2 AP is a labeling function that assigns to each state s ∈ S a set of atomic propositions [11].

2.6.2. Probabilistic Logics

The Probabilistic Computation Tree Logic (PCTL) [108] has appeared as an extension of CTL for the specification of systems that exhibit stochastic behaviour. We use the PCTL for

defining quantitative properties of DTMCs. PCTL state formulas are formed according to the following grammar:

$$\varphi = \text{true} \mid a \mid \neg\varphi \mid \varphi 1 \wedge \varphi 2 \mid P{\sim}p(\phi)$$

Where $a \in AP$ is an atomic proposition, ϕ is a path formula, P is a probability threshold operator, $\sim \in \{<, \leq, >, \geq\}$ is a comparison operator, and p is a probability threshold. The path formulas ϕ are formed according to the following grammar:

$$\phi = \varphi 1 U \varphi 2 \mid \varphi 1 W \varphi 2 \mid \varphi 1 U{\leq}n\varphi 2 \mid \varphi 1 W{\leq}n\varphi$$

Where $\varphi 1$ and $\varphi 2$ are state formulas and $n \in N$. As in CTL, the temporal operators (U for strong until, W for weak (unless) until and their bounded variants) are required to be immediately preceded by the operator P. The PCTL formula is a state formula, where path formulas only occur inside the operator P. The operator P can be seen as a quantification operator for both the operators $\forall$ (universal quantification) and $\exists$ (existential quantification), since the properties are representing quantitative requirements. The semantics of a PCTL formula over a state s (or a path σ) in a DTMC model $D = (S, s_{init}, P, L)$ can be defined by a satisfaction relation denoted by $\models$. The satisfaction of $P{\sim}p(\phi)$ on DTMC depends on the probability mass of set of paths satisfying ϕ. This set is considered as a countable union of cylinder sets, so that, its measurability is ensured [10].

The semantics of PCTL state formulas for DTMC is defined as follows:

$s \models \text{true} \Leftrightarrow \text{true}$

$s \models a \Leftrightarrow a \in L(s)$

$s \models \neg\varphi \Leftrightarrow s \not\models \varphi$

$s \models \varphi 1 \wedge \varphi 2 \Leftrightarrow s \models \varphi 1 \wedge s \models \varphi 2$

$s \models P{\sim}p(\phi) \Leftrightarrow P(s \models \phi) \sim p$

Given a path $\sigma = s0\ s1...$ in D and an integer $j \geq 0$, where $\sigma[j] = s_j$, The semantics of PCTL path formulas for DTMC is defined as for CTL as follows:

$\sigma \models \varphi 1 U \varphi 2 \Leftrightarrow \exists\, j \geq 0.\sigma[j] \models \varphi 2 \wedge (\forall 0 \leq k < j.\sigma[k] \models \varphi 1)$

$\sigma \models \varphi 1 W \varphi 2 \Leftrightarrow \sigma \models \varphi 1 U \varphi 2 \vee (\forall k \geq 0.\sigma[k] \models \varphi 1)$

$\sigma \models \varphi 1 U{\leq}n\varphi 2 \Leftrightarrow \exists 0 \leq j \leq n.\sigma[j] \models \varphi 2 \wedge (\forall 0 \leq k < j.\sigma[k] \models \varphi 1)$

$\sigma \models \varphi 1 W{\leq}n\varphi 2 \Leftrightarrow \sigma \models \varphi 1 U{\leq}n\varphi 2 \vee (\forall 0 \leq k \leq n.\sigma[k] \models \varphi 1)$

The satisfaction of P∼p (ϕ) on DTMC depends on the probability mass of set of paths satisfying ϕ . This set is considered as a countable union of cylinder sets, so that, its measurability is ensured. A formula P∼p (ϕ) is satisfied on an MDP M if only if for every d ∈ D:

P(ϕ)∼ p, where D represents the set of all schedulers and P(ϕ) represents the probability of the set of all finite paths satisfying ϕ under the scheduler d . The semantics of PCTL state and of path formulas for MDPs are defined as the same as for DTMCs, except that for model checking of MDPs we have to consider either maximizing or minimizing schedulers. let P max(ϕ) be the maximal probability of ϕ where P max(ϕ) =max{Pd(ϕ)|d ∈ D}, and dually the minimal probability P min(ϕ) be the minimal probability of ϕ where P min(ϕ) = min{Pd(ϕ)|d ∈ D}. For instance for properties of upper threshold, it is evident that (M⊭ = P ≤p(ϕ)) ⇔ Pmax(ϕ)>p [11].

2.7. The PRISM Language

2.7.1. Introduction

In order to construct and analyse a model with PRISM, it must be specified in the PRISM language, a simple, state-based language, based on the Reactive Modules formalism of Alur and Henzinger. This is used for all of the types of model that PRISM suppojrts: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), Markov decision processes (MDPs) and probabilistic timed automata (PTAs).

The fundamental components of the PRISM language are ***modules*** and ***variables***. A model is composed of a number of *modules* which can interact with each other. A module contains a number of local *variables*. The values of these variables at any given time constitute the state of the module. The ***global state*** of the whole model is determined by the *local state* of all modules. The behaviour of each module is described by a set of *commands*. A command takes the form:

[] guard -> prob_1 : update_1 + ... + prob_n : updating

The ***guard*** is a predicate over all the variables in the model (including those belonging to other modules). Each *update* describes a transition which the module can make if the guard is true. A transition is specified by giving the new values of the variables in the module, possibly as a function of other variables. Each update is also assigned a probability (or in some cases a rate) which will be assigned to the corresponding transition [13].

2.7.2. Example

We will use the following simple example to illustrate the basic concepts of the PRISM language. Consider a system comprising two identical processes which must operate under mutual exclusion. Each process can be in one of 3 states: {0,1,2}. From state 0, a process will move to state 1 with probability 0.2 and remain in the same state with probability 0.8. From state 1, it tries to move to the critical section: state 2. This can only occur if the other process is not in its critical section. Finally, from state 2, a process will either remain there or move back to state 0 with equal probability. The PRISM code to describe an MDP model of this system can be seen below. In the next sections, we explain each aspect of the code in turn [13].

```
mdp

module M1

    x : [0..2] init 0;

    [] x=0 -> 0.8:(x'=0) + 0.2:(x'=1);
    [] x=1 & y!=2 -> (x'=2);
    [] x=2 -> 0.5:(x'=2) + 0.5:(x'=0);

endmodule

module M2

    y : [0..2] init 0;
    [] y=0 -> 0.8:(y'=0) + 0.2:(y'=1);
    [] y=1 & x!=2 -> (y'=2);
    [] y=2 -> 0.5:(y'=2) + 0.5:(y'=0);
endmodule
```

2.8. Stochastic Multi-Player Games

Stochastic multi-player games (SMGs) are a generalisation of Markov decision processes, where we distinguish between several types of nondeterministic choices, each corresponding to a different player. SMGs thus allow us to reason about strategic decisions of multiple players competing or collaborating to achieve the same objective. Several stochastic game models exist, which include concurrent games and partial-observation games. Here we focus on ***turn-based*** games as studied in, in which a single player controls the choices made in a given state [14].

Definition 1

A stochastic multi-player game (SMG) is a tuple G = (π, S, (Sπ, Sp), S_{init}, Δ, A, L), where π is a finite set of players; S is a finite nonempty set of states partitioned into player states Sπ = S Si∈Π and probabilistic states Sp ; S init ∈ SΠ is an initial state; Δ: S × S → [0, 1] is a probabilistic transition function such that for all player states s, t ∈ SΠ and probabilistic states s0 ∈ Sp we have Δ(s, t) = 0, Δ(s, s0) ∈ {0, 1} and Pt∈SΠ Δ(s0, t) = 1; A is a finite nonempty set of actions; and L : Sp → A is an action labelling function [14].

Definition 2

(Reward structure)**.** Given a game *G* = (Π, *S*, (*S*Π, *Sp*),*S* init, Δ, A, *L*), a *reward structure* for *G* is a function *r* : *S* → R.

To resolve the non determinism in an SMG, similarly to MDPs we use strategies, except we now have a strategy for each player *i* ∈ Π. We work with an explicit memory representation of strategies due to [14].

Definition 3

(Strategy)**.** For an SMG *G* = (Π, *S*, (*S*Π, *Sp*), *S* init, Δ, A, *L*), a *strategy σi* for player *i* of *G* is a tuple *σi* = (*M*, *σiu*, *σin*, *σi*init), where *M* is a countable set of memory elements, *i* : *M* × *S* → *D*(*M*) is a memory update function, *σin* : *M* × *Si* → *D*(*S*) is a next move function such that *i* (*m*, *s*)(*s0*) > 0 only if Δ(*s*, *s0*) > 0, and *σi*init : *S* → *D*(*M*) is an initial memory element function. If the memory update function maps to point distributions, i.e. is of type *σiu* : *M* × *S* → *M*, the strategy *σi* is *deterministic memory update*, and otherwise it is *stochastic memory update*. The set of all strategies for player *i* ∈ Π is denoted by Σ*i G*. A *strategy profile σ* = *σ*1 …*σ*|Π| comprises a strategy for every player in *G* [14].

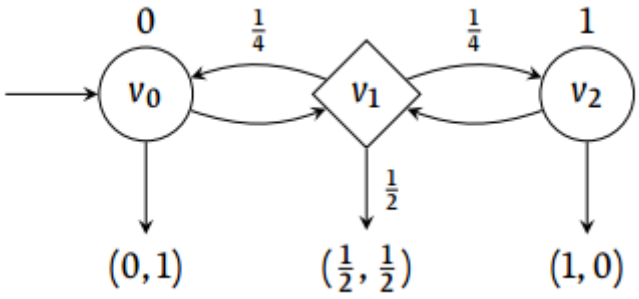


Figure 2-4 An example of a two-player SSMG [8].

2.9. Conclusion

As autonomous systems are becoming an integral part of our society, their failure carries potentially unacceptable and life-endangering risks. Rigorous model-based verification technologies incorporated within the design process can improve their safety and reliability and reduce development costs. This chapter briefly summarised quantitative verification and strategy synthesis techniques developed for autonomous systems modelled as turn-based stochastic multi-player games as implemented in the tool PRISM-games and outlined future research challenges in this challenging yet exciting field [14].

CHAPTER 3

GAME THEORY AND WIRELESS SENSOR NETWORKS

3.1. Introduction

Game theory (GT) is a mathematical method that describes the phenomenon of conflict and cooperation between intelligent rational decision-makers. In particular, the theory has been proven very useful in the design of wireless sensor networks (WSNs). In this chapter following developments and findings of GT, its applications in WSNs, and provides the community a general view of this vibrant research area. We first introduce the typical formulation of GT in the WSN application domain. The roles of GT are described that include routing protocol design, topology control, power control and energy saving [17].

The primary goal of my thesis is the conservation of energy in wireless sensor networks WSNs. Then, three variations of game theory are described, namely, the cooperative, non-cooperative, and repeated schemes. Finally, existing problems and future trends are identified for researchers and engineers in the field.

3.2. Wireless Sensor Networks

A wireless sensor network (WSN) is a network of thousands of resource-constrained sensors whose communication with a central station are conveyed by means of wireless signals. A sensor node is generally comprised of four basic elements, including a sensing unit, a processing unit, a transceiver unit, and a power unit. The WSN is frequently deployed for sensing the area of interest where data captured encompass light, pressure, sound, and others. Sensor nodes in WSN mainly use a broadcast communication paradigm where the sensor signals are used in further analyses of the sensed environment. WSN is preferred as the sensor system architecture with regard to its inherent redundancy but is susceptible to disadvantages caused by limited operation life-time. Differ from other wired networks, the use of WSNs are usually restricted by energy stored, computation capability, memory, plethoric information flow, and short communication distance. Since the sensor nodes are often densely deployed in a sensing field, it is difficult and costly to replace faulty sensor nodes manually. Furthermore, sensor nodes may have no global information of the whole network and the topology of a WSN varies frequently [17].

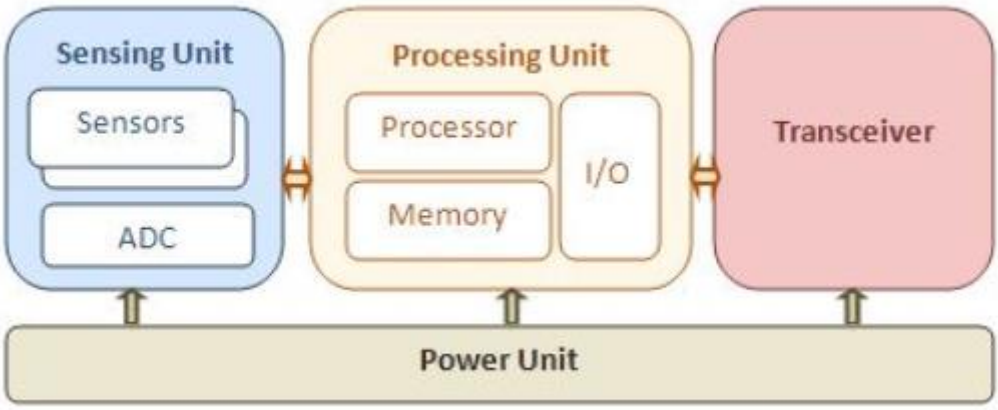


Figure 3-1 Architecture of a Sensor Node [22].

3.2.1. Application Examples

With the high degree of deployment flexibility, applications of WSN are vast and can be broadly classified into the monitoring and tracking categories. Monitoring applications include environmental monitoring such as forest fire detection, biocomplexity mapping of the environment, flood detection, precision agriculture; health monitoring contains telemonitoring of human physiological data, monitoring doctors and patients conditions and drug administration in hospitals; inventory location monitoring; factory, machine, chemical and structural monitoring. Military monitoring examples can be found in monitoring friendly forces, equipment and ammunition, battlefield and terrain surveillance, reconnaissance of opposing forces, targeting, battle damage assessment, nuclear, biological and chemical attack detection. Tracking applications include objects, animals, humans, vehicles, and military enemy tracking. These applications are made possible due to the fact that WSN has a short system setup time and sensors can be disposed with acceptable operation cost [20].

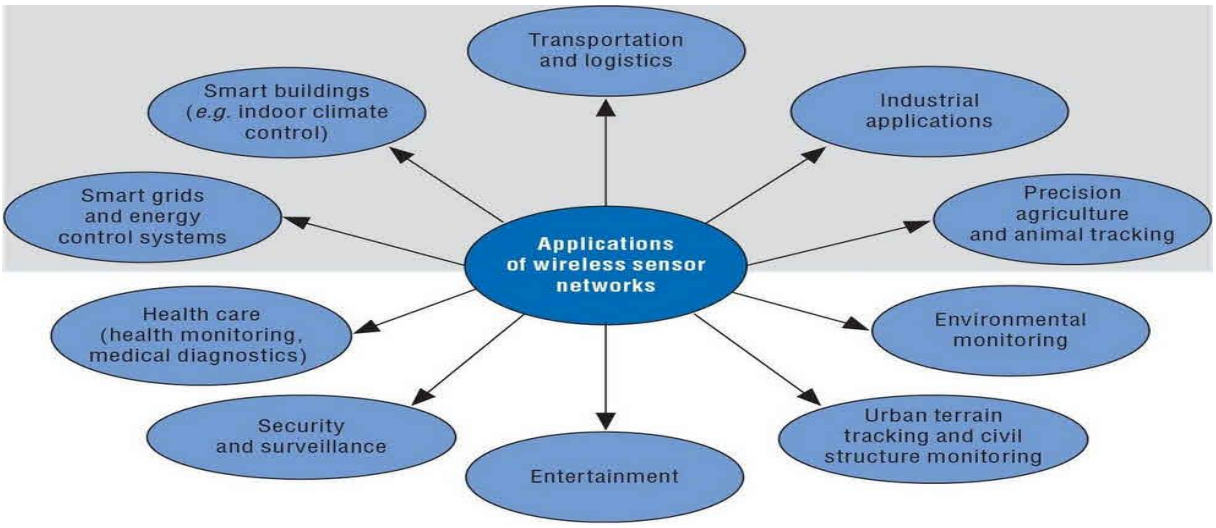


Figure 3-2 WSNs Applications [12].

3.2.2. Need for Game Theory

The flexibility, fault tolerance, high sensing fidelity, low-cost and rapid deployment characteristics of WSNs are desirable features in creating many new and exciting application areas for remote sensing, detecting, tracking, and monitoring. However, it is non-trivial and very involved to design an optimal WSN to satisfy performance objectives such as maximum sensing coverage and extended operation periods. In order to obtain a practical and feasible WSN and due to the operation nature of the network, game theory (GT) is regarded as an attractive and suitable basis to accomplish the design goal. Game theory is a branch of mathematics and can be used to analyze system operations in decentralized and self-organizing networks. GT describes the behavior of players in a game. Players may be either cooperate or non-cooperative while striving to maximize their outcomes from the game. In this regard, sensors manage their operations in terms of power resources devoted to sensing and communicating among themselves and with a global controller such that the assigned task could be completed effectively as desired [17].

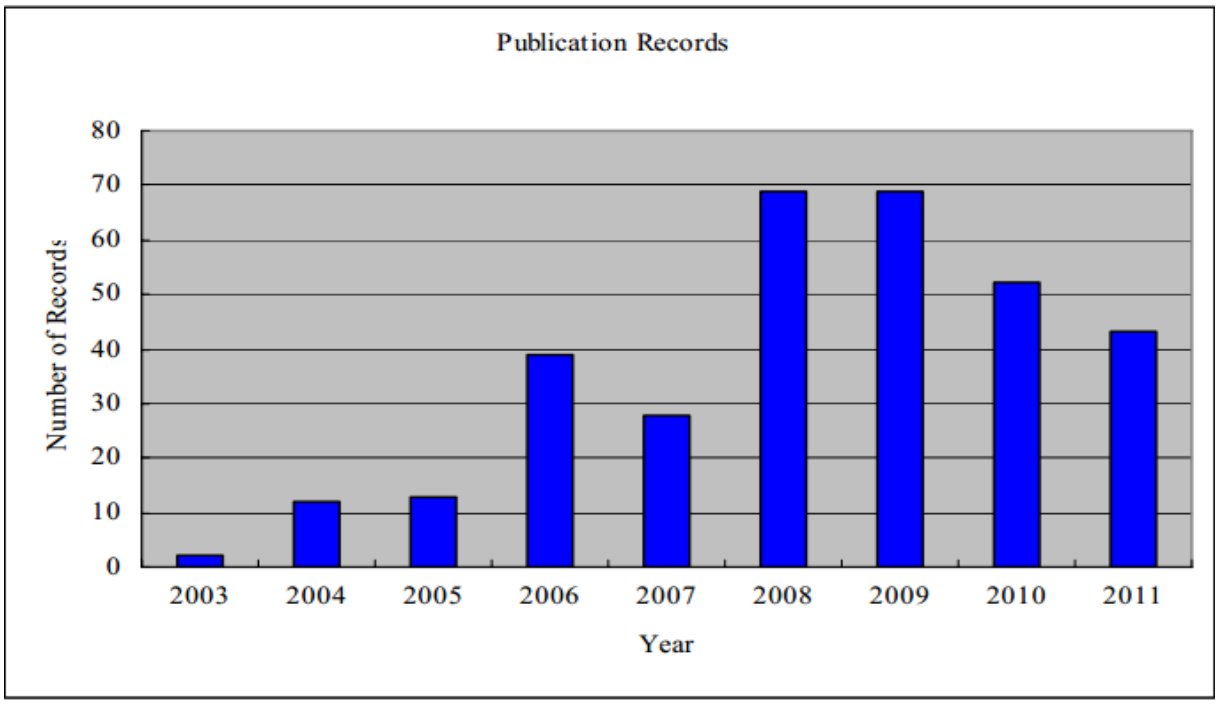


Figure 3-3 Yearly publications on GT of WSN [21].

3.3. Game Theory (GT):

Game theory is an advanced branch intelligent optimization. The model of game theory represents a game between player groups that choose to behave cooperatively or non-

cooperatively and try to promote their benefits (payoffs) through the used strategies executed through the cumulative players actions. The fundamental definitions of game parameters, which can be summarized as follows:

Definition 1:

A game is a description of the strategic interaction between opposing, or cooperating, interests where the constraints and payoff for actions are taken into consideration [18].

Definition 2:

A player is a basic entity in a game, which is involved in the game with a finite set of players denoted by N that is responsible for taking rational actions denoted by Ai, for each player i. A player can represent a person, machine, or group of people within a game [18].

Definition 3:

The Utility/Payoff is the positive or negative reward to a player for a given action within the game denoted by $u_i: A \rightarrow R$, which measures the outcome for player i determined by the actions of all players $A = \times i \in N Ai$, where the symbol × denotes Cartesian product [18].

Definition 4:

A strategy is a plan of action within the game that a given player can adopt during game play denoted by a strategic game **[** $N,(A),(u_i)$ **]**.

In the security field, game theory application is not only limited to counteracting the effect of external intruders; it can be used to detect the malicious nodes and reveal the nodes that behave selfishly and overburden the whole network. Generally, Nash equilibrium (NE) is the intelligent solution for the social problems that has become a promising concept for wireless networks and more specifically for WSN security [18].

3.3.1. Basics of Game Theory

Game theory is increasingly attracting more attention as a mechanism to solve various problems in WSNs. Generally, a game consists of a set of players, a set of strategies for each player and a set of corresponding utility functions. A norm form game of a WSN of n sensor nodes is given by a 3-tuple G = <N, S, U>. Here, G is a particular game, where N = {n_1, n_2, ...,n_n} is a finite set of the sensor nodes. S = {S1, S2, ..., Sn}, is the strategy space of the sensor node i can select from is represented by Si (i = 1, 2, ..., n). U = {u1, u2, ..., un} is the corresponding payoff function of node i represented by u_i (i = 1, 2, ..., n), u_i is a utility value of each node receives at the end of an action.

A strategy for a player is a complete plan of actions in all possible situations in the game. The players try to act selfishly to maximize their consequences according to their preferences. We have to formulate the payoff functions in a way that will help node i to select a strategy Si that represents the best response to the strategies selected by the other n-1 nodes. Here, si is the particular strategy chosen by node i and s-i is the particular strategies chosen by all of the other nodes in the game. For strategies $s = \{s_i , s_{-i}\}$, it is called a strategy profile or sometimes a strategy combination. Every different combination of individual choices of strategies can produce a different strategy profile. The strategy profile $s = \{s_1, s_2, ..., s_n \mid s_i \in S_i, i = 1, 2, ..., n\}$ needs to place the nodes responding to a Nash Equilibrium (NE). It is a solution concept that describes a steady state condition of a game involving two or more players, in which each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only its own strategy unilaterally. NE is identified wherein no nodes will rationally choose to deviate from his chosen strategy otherwise it will diminish its utility, i.e., $u_i (s_i, s_{-i}) \geq u_i (s_i^*, s_{-i})$ for all $s_i^* \in S_i$ [17].

3.4. Most known game types for WSNs

3.4.1. Cooperative Games

The common types of cooperative games that are used to resolve different WSN security issue presented as follows:

3.4.1.1. Bargaining Game

The Bargaining game represents a problem between two competitors (agents) who should cooperate; in other words, the bargaining or Nash bargaining game is modeled based on the bargaining interaction concept between two players, who request a fraction of the same benefits. It can be used to model resource allocation in wireless communication networks, in which the agents aim to exploit the same spectrum, which should be fairly allocated. In Nash bargaining game, if the total requests by the two players are greater than the available resources, both requests are discarded. Conversely, if their requests are less than the available resources, both requests are accomplished. Pareto-inefficient is a result of non-cooperating players, which is solved by a Nash bargaining solution [18].

3.4.1.2. Repeated Game

The repeated game is fundamentally considered as an interaction between two individual players who repeatedly play the game. This game is also known as iterated game which consists of some repetitive stages. Each stage has two players at which the current action is taken into consideration in the subsequent actions of the other players. The repeated games can be classified into two categories: finitely repeated games and infinitely repeated games. In the finitely repeated games, the time period is fixed and thoroughly known. This category has a drastic defect which permits the player to act selfishly and NE equals the minmax payoff. Consequently, the punishment is not sufficient in this case. The infinitely repeated game represents the most popular notion in which the game is probably played for infinite times. The punishment is defined in this case as reducing the payoff that the non-cooperating player earns based on his reputation. The reputation is computed based on the players' interactions [18].

3.4.1.3. Coalition Game

The coalition game is a result of cooperation among a set of players acting as one player against the others aiming at maximizing the mutual outcome. This coalition mutual benefit called coalition value. Coalition games are classified into two forms, namely, strategic form and partition form. In the former case, the coalition value depends on the number of participant players in the coalition regardless of their network establishment. Conversely, in partition form, the establishment represents the intrinsic role for the coalition value [18].

3.4.2. Non-Cooperative Games

The common types of non-cooperative games that are used to mitigate different WSN security problems are presented as follows.

3.4.2.1. Zero-Sum Game

The zero-sum game is one of the types of non-cooperative games between two players. One player is considered a maximizer that strives to maximize its gain while the other is considered to be the minimizer that aims to minimize its losses.

Apparently, constant-sum game could be transformed to an equivalent zero-sum game; and zero-sum game is a special case of constant-sum game given that the players add up their gains or losses to a constant value for any strategy profile [18].

3.4.2.2. Nonzero-Sum Game

Nonzero-sum game is played between two or more players where the sum of players' utilities is not constant during the course of the game. In nonzero-sum games, all players are considered maximizers or minimizers which have no constraints on the total utility as in the zero-sum game. Consequently, all the participants can gain or lose together [18].

3.4.2.3. Stackelberg Game

The Stackelberg game is used to model two competitive players; one is a game initiator (leader) who chooses an action from a set A1 then the second player traces the leader's action and chooses an action from a set A2. This scenario is widespread in securing different WSNs where the defender acts as a leader and the attacker plays the role of the follower accomplished. Pareto-inefficient is a result of non-cooperating players, which is solved by a Nash bargaining solution [18].

3.4.2.4. Stochastic Game

The stochastic game is one of dynamic games that are played in a sequence of stages.

The stage is formulated based on a probabilistic transitions by one or more players. The new state of the game is random which depends on the previous players' actions [18].

3.4.2.5. Bayesian Game

The Bayesian game falls under the non-cooperative game framework, in which the players have some information shortage while executing their actions. In other words, Bayesian game could be suitable for modeling the incomplete information interactions between players. Accordingly, a player can estimate the other players' payoffs. Moreover, a game theoretic approach based on Bayesian game has been developed to do intrusion detecting for wireless nodes [18].

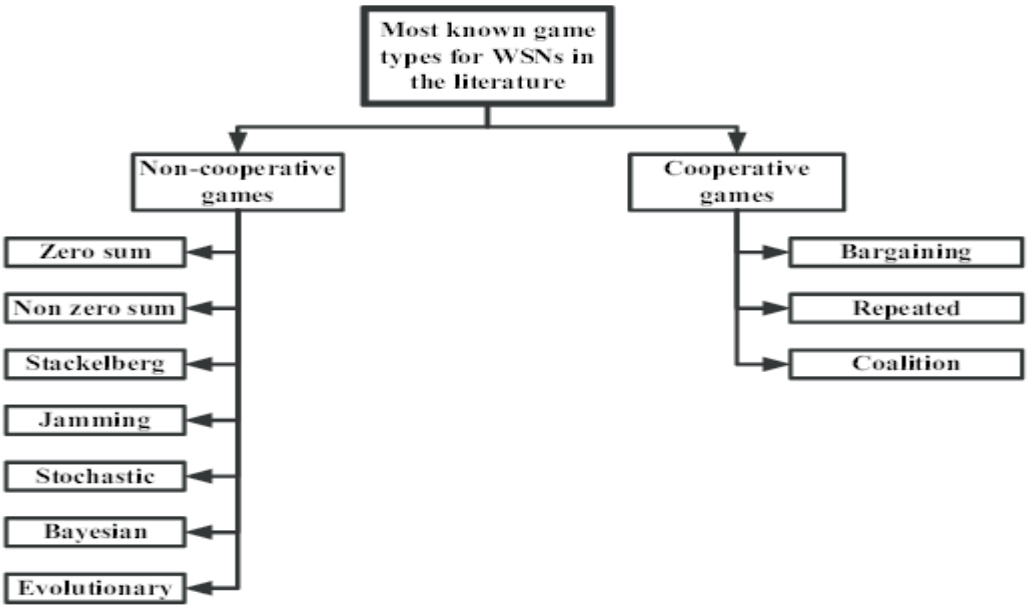


Figure 3-4 Cooperative and Non-cooperative Game Classification for Addressing WSN [21].

3.5. Applications of Game Theory in WSN

A game is a set of their fundamental components: a set of players, a set of strategies, and a set of payoffs. Players or nodes are the decision takers in the game. The strategies are the different choices available to nodes. Finally, a utility function (payoffs) decides the all possible outcomes for each player. Table 1 shows typical components of a wireless networking game [19].

Components of a game	Elements of a wireless network
Players	Nodes in the wireless network
A set of strategies	A modulation scheme, Coding rate ,transmit power level, etc.
A set of payoffs	Performance metrics (e.g. Throughput, Delay, SNR, etc.)

Table 3.1 Components of a wireless networking game [22].

3.5.1. Game Theory for Wireless Network Management in WSN

The design of a wireless network and optimization of its performance is a non-trivial and

Complicated process. The WSN has to fulfill straight requirements imposed from a set of operation goals. Game theory thus plays a supportive and critical role in designing and

operating a WSN. Figure illustrates the relationship between various WSN elements and the way that game theory is employed.

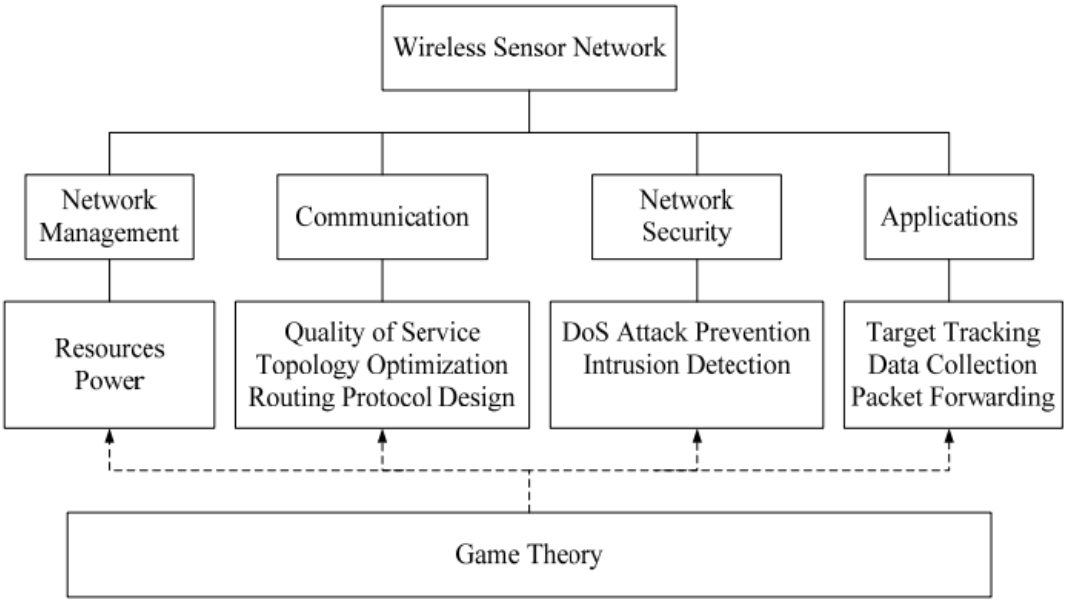


Figure 3-5 An illustration of the relation between WSN and game theory [21].

Representative requirements in a WSN include network management which is responsible for making the most economical use of power resources such that WSN can be put into operation for an extended period of time. A fundamental need for the WSN is to communicate with a centralized or base station that sensed data is fused and analyzed. In order to derive an effective WSN, one needs to consider the quality of service (QoS) specified, the topology used in the network architecture and how data is dispatched according to some preferred routing protocol. When the WSN is put into operation, it is always susceptible that the network may be attacked by hackers where data could be intercepted and retrieved illegally. To this end, the WSN has to be designed for deny of service (DoS) prevention and the incorporation of intrusion detection capability. With the desire to enlarge the application domain of WSN, attentions are also directed toward target tracking and data collection in the WSN design.

3.5.2. Power Management—Energy Saving and Power Control

In WSN, energy is a limited resource and must be used judiciously. Currently, the energy problem remains one of the major obstacles somehow preventing the complete exploitation of WSN technology.

Energy saving and power control strategies should be devised at sensor nodes as well as in the network to prolong the network lifetime. In practice, the Game-theoretical Total Link

(GTL) algorithm which sets each node's energy range is usually better than Critical Transmitting Range (CTR) in energy saving according to the topological changing, as shown by Zhang and Kyung. Considered a MAC scheme based on p-persistence slotted ALOHA. To determine the value of the attempt probability p, the authors constructed the p-persistence slotted ALOHA as a simple non-cooperative game which involves generalized payoffs reflecting energy saving and throughput.

They derived an NE in a closed form Investigated a cooperative communication scheme for WSN. They modeled the power allocation problem as a two-person bargaining game, and used the Nash Bargaining Solution (NBS) to achieve a win-win strategy.

A distributed optimization framework using Game-Theory (GT) was used to analyze the power-allocation and rate allocation in WSNs. Na proposed a distributed power control algorithm based on game theory for WSN, and its objectives are formulated toward reducing power consumption, decreasing overhead and increasing network lifetime. The simulation shows that this power control algorithm converges to a NE when decisions are updated according to a better response dynamic. The heterogeneous types of sensors are viewed as intelligent agents interacting with each other locally in a WSN. Sensors in this network interact with each other either in a cooperative way to form coalitions or in a non-cooperative way to deal with conflicts.

Kannan presented a novel formulation of the problem of energy misbehavior and develop an analytical framework for quantifying its impact on other nodes. They formulated two versions of the power control problem for WSN with latency constraints arising from duty cycle allocations. In the first version, strategic power optimization, nodes are modeled as rational agents in a power game, who strategically adjust their powers to minimize their own energy. In the other version, joint power optimization, sensor nodes adjust their transmission powers to minimize the aggregate energy expenditure [17].

3.5.3. Game Theory for Wireless Sensor Networks Security

The trustworthiness mechanisms are considered the foremost concern of the WSNs security specialists. Therefore, different network security frameworks are discussed in the literature.

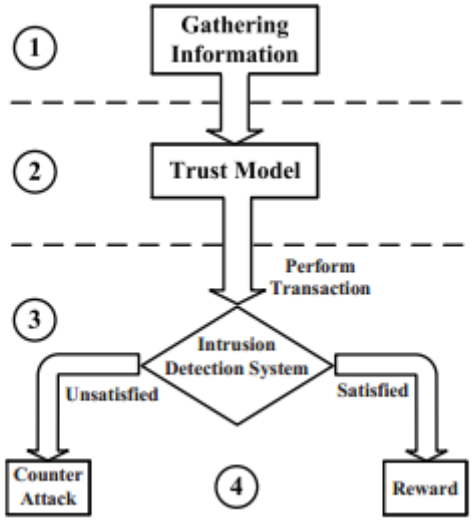


Figure 3-6 General Trust and Reputation Model Mechanisms [22].

3.6. Conclusion

A wireless network is identified by a distributed, dynamic, self-organizing architecture. Each node in the network is capable of independently adapting its operation based on the current environment according to predetermined algorithms and protocols. Analytical models to estimate the performance of wireless networks have been scarce due to the distributed and dynamic nature of such networks [19].

This paper gives a detailed insight in the game theory definition, classifications, game models and applications of games in wireless networks. We have presented recent works related to game theory in communication networks, cognitive radio networks, wireless sensor networks, resource allocation and power control. Game theory offers a suite of tools that may be used effectively in modelling the interaction between independent nodes in wireless network. Because of these numerous benefits, adopting analytic approach that emphasizes the use of game theory over wireless networks is preferable for analyzing the users' needs in such networks. However, game theory focuses on solving the Nash equilibrium and analyzing its properties and not to consider how players should interact to reach this equilibrium [19].

CHAPTER 4

MODELLING A WSN GAME THEORY IN PRISM-GAMES

4.1. Introduction

In wireless sensor networks (WSNs) we have a wide range of potential applications including, smart spaces, medical systems and robotic exploration. In this chapter, we will study WSNs in term of energy consumption and how to conserve energy in times of working, we use PRISM-games to model an incompletely cooperative game theory for WSN communication protocol.

4.2. Media Access Control (MAC)

Definition: is the lower sublayer of the data link layer (layer 2) of the seven-layer OSI model. The MAC sublayer provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multiple access network that incorporates a shared medium, e.g. an Ethernet network. The hardware that implements the MAC is referred to as a media access controller.

4.3. Classification of WSN MAC Protocols

Several MAC protocols have been successfully proposed to meet the stringent design requirements of WSNs. Actually; these protocols depend on how protocol allows nodes to access the channel. We have classified WSN based MAC protocol as depicted in Figure 4 into four categories as; contention based, scheduling based, channel polling based, and hybrid protocols [26].

An extended version of CSMA, called CSMA with collision avoidance (CSMA/CA),adds mechanisms to limit the number of messages lost when nearby devices transmit at the same time, CSMA/CA attempts to avoid collisions by using a control message exchange to reserve the wireless channel before each data message transmission. The benefit of CSMA/CA techniques in sensor networks depends on the traffic conditions, wireless channel characteristics, and network topology, so in some cases it may prove beneficial and in others an unnecessary overhead [27].

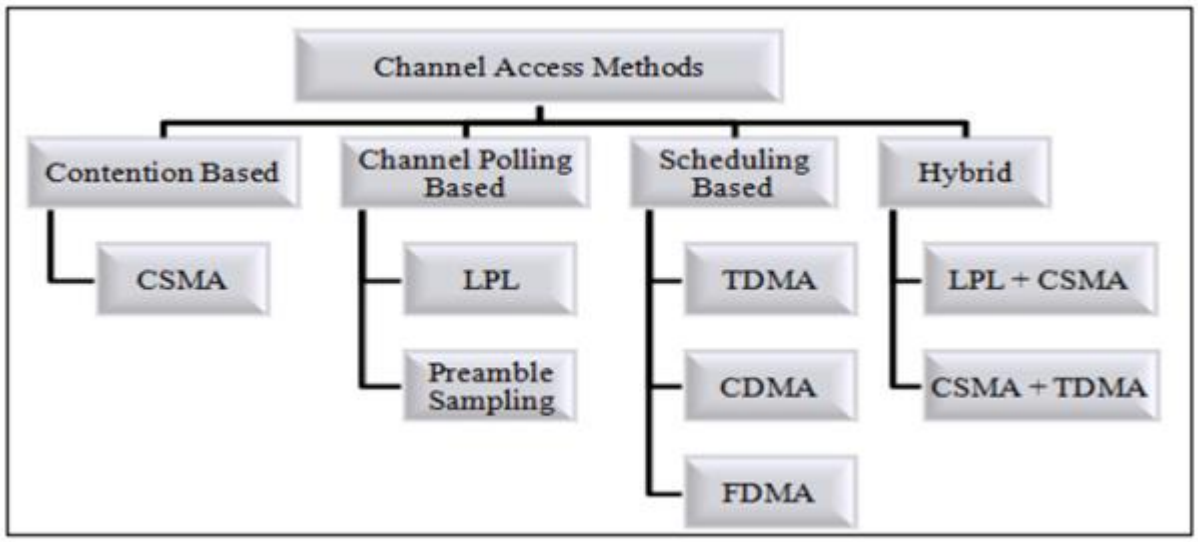


Figure 4-1 Channel Accessing Taxonomy in WSNs [22].

4.3.1. Carrier-sense multiple access with collision avoidance (CSMA/CA)

- Definition: is a network multiple access method in which carrier sensing is used, but nodes attempt to avoid collisions by transmitting only when the channel is sensed to be "idle". When they do transmit, nodes transmit their packet data in its entirety.
- Performance: CSMA/CA performance is based largely upon the modulation technique used to transmit the data between nodes. Studies show that under ideal propagation conditions (simulations), Direct Sequence Spread Spectrum (DSSS) provides the highest throughput for all nodes on a network when used in conjunction with CSMA/CA and the IEEE 802.11 RTS/CTS exchange under light network load conditions. Frequency Hopping Spread Spectrum (FHSS) follows distantly behind DSSS with regard to throughput with a greater throughput once network load becomes substantially heavy. However, the throughput is generally the same under real world conditions due to radio propagation factors.
- Usage of **CSMA/CA**: GNET,IEEE 802.15.4 (Wireless PAN) uses CSMA/CA, Bus networks.

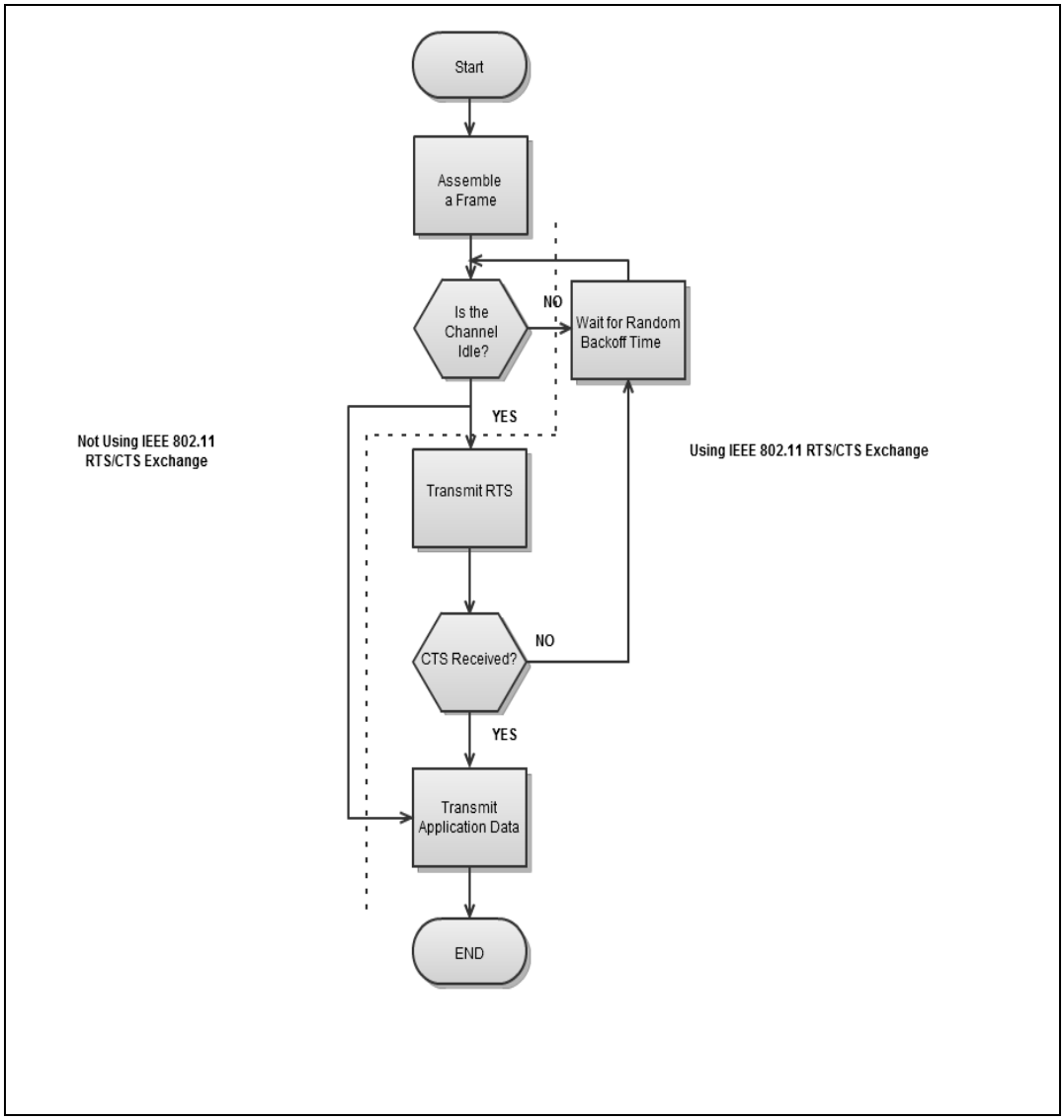


Figure 4-2 Simplified algorithm of CSMA/CA [28].

4.4. Incompletely Cooperative Game Theory

4.4.1. Framework of Incompletely Cooperative Game Theory

The incompletely cooperative game can be classified as a stochastic game, which starts when a new packet arrives at the node's transmission buffer and ends when the packet is transmitted successfully or discarded. In each timeslot, each player (i.e., node) estimates the current game state based on what happened in the past timeslots, then all the players take actions simultaneously, i.e., transmitting their packets, listening or sleeping.

In the game, each player takes a distributed approach of detecting and estimating and adjusting the current game state, and tuning its local contention parameters to the estimated game state. The framework of the game consists of three major components, a detector, an estimator and an adjustor. The first component, the detector, detects and records the

experienced PHY-specific information. The second component, the estimator, uses the above measurements to estimate the current game state, such as the number of competing nodes (n). The third component, the adjustor, makes a decision according to which strategy the player transmits its packets, and implements the optimal strategy by tuning contention parameters, as seen in Fig 4-2.

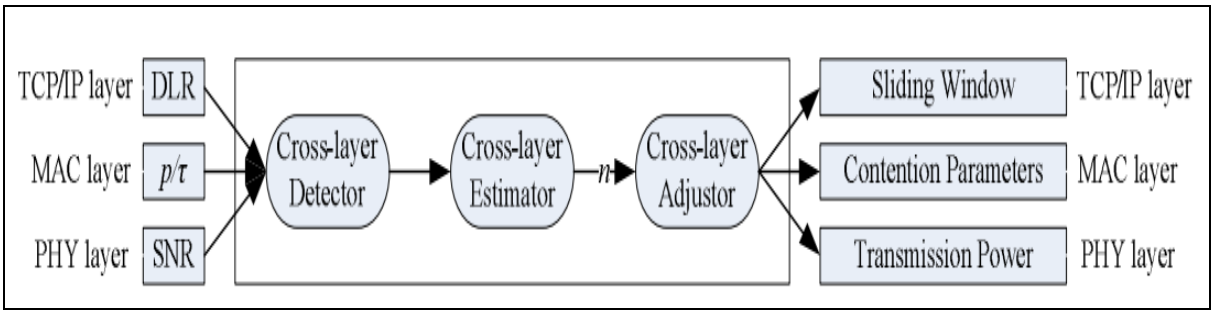


Figure 4-2. Framework of incompletely cooperative game theory [23].

Such cross-layer characteristics imply that to obtain the game state and to tune the equilibrium strategy in each timeslot, cross-layer detection, cross-layer estimation and cross-layer adjustment are needed. The term “cross-layer” in the incompletely cooperative game means that the knowledge of the PHY, MAC and TCP/IP layers is used by the detector, estimator and adjustor, to calculate the game state and implement the optimal strategy.

4.4.2. Equilibrium Strategy of Incompletely Cooperative Game Theory

In the incompletely cooperative game, the utility function of the player (i.e., node *i*),μi=f(si,si'). The strategy of theplayer1, si, includes 03 possible actions: transmitting, listening or sleeping.

The strategy of the player2, si', includes 04 possible actions: successful transmission, fail transmission, listening or sleeping, as seen in Fig 4-2.

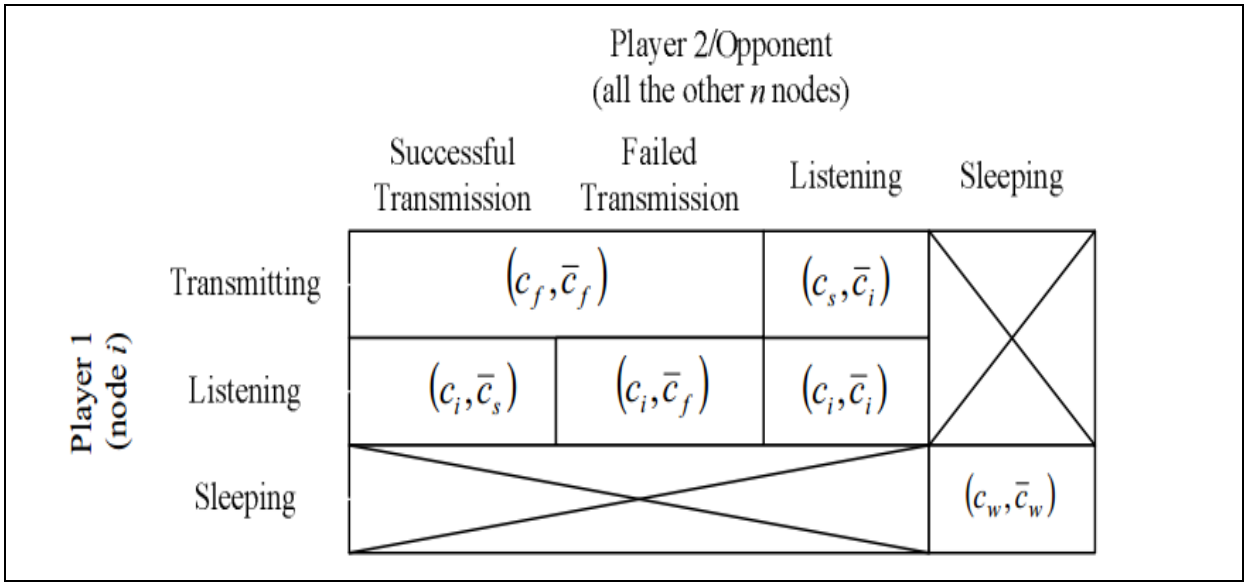


Figure 4-3. Incompletely cooperative game model of n+1 nodes [23].

The strategy table with 2 players (i.e., *n*+1 nodes), where *ci* and *ci'* are the payoff when Player 1 and Player 2 listen respectively, *cs* and *cs'* are the payoff when they transmit successfully respectively, *c f* and *c f '*are the payoff when they fail respectively, *cw* and *cw'* are the payoff when they sleep respectively.

For simplicity, assume that the probability of the following two cases is zero: firstly, Player 1 transmits or listens when Player 2 is sleeping; secondly, Player 2 transmits or listens when Player 1 is sleeping. Obviously, Player 1 changes its sleeping and transmitting probability not to obtain its own optimal utility (μi*), but to help Player 2 to achieve the optimal utility (μi*').Hence, it indicates that all the nodes play the cooperative gamebased on the estimated game states. On the other hand, the twoplayers get the optimal utility under one condition, the limit ofenergy consumption. It indicates that all the nodes play theconditional game.

4.4.3. Incompletely Cooperative Game Theory in WSNs

In the incompletely cooperative game is too complex to propose an explicit formulation for its utility function. Hence, a simple method is presented in the following, time is divided into super-frames has two parts:

- Sleeping part: during the sleeping part, each player turns off its radio to preserve energy.
- Active part: during the active part, the player contends for the channel according to the game, i.e., it estimates the current state of the game.

4.5. A Simplified Game -Theoretic MAC Protocol FOR WSNs

The problem in the incompletely cooperative game is how to estimate the game state accurately and timely. In the active part, after transmitting a packet, to maintain the current contention.

For simplified the game-theoretic we will work on the CSMA/CA working mechanism because it is like a mechanism of WSNs,In CSMA/CA, as soon as a node receives a packet that is to be sent, it checks to be sure the channel is clear (no other node is transmitting at the time). If the channel is clear, then the packet is sent. If the channel is not clear, the node waits for a randomly chosen period of time, and then checks again to see if the channel is clear. This period of time is called the backoff factor, and is counted down by a backoff counter. If the channel is clear when the backoff counter reaches zero, the node transmits the packet. If the channel is not clear when the backoff counter reaches zero, the backoff factor is set again, and the process is repeated [27].

4.6. Implementation

For the implementation, we use PRISM-games, a model checker for stochastic multi-player games (SMG), which supports modelling, automated verification and strategy synthesis for probabilistic systems with competitive or cooperative behaviour.

At first we have constant probabilistic values:

const double Cs1Cf2 = 0.75;

const double Cs1Ci2 = 0.25;

Their total = 1

//

const double Ci1Cs2 = 0.25;

const double Ci1Cf2 = 0.25;

const double Ci1Ci2 = 0.25;

const double Ci1Cw2 = 0.25;

Their total = 1

//

const double Cw1Cf2 = 0.25;

const double Cw1Ci2 = 0.25;

const double Cw1Cw2 = 0.5;

Their total = 1

//

The model consists of two players for control of modules, player p1 and player p2, each of them has their own works performed as follows:

player p1

[transmitting], [sleeping], [listening]

End player

player p2

[Successful_ transmission], [faild_transmission], [listening], [sleeping]

End player

Two players control two modules as follows:**modul**node01, **modul**node02.Firstly If first player in case of transmission or listening so second player in case sleeping, secondly, the player two in case of transmission or listening so first player in case sleeping.Finally if the first player and the second player are in case sleeping there is no energy consumption.

In first case, the first player consumes energy, but the second player does not consume energy. In second case, the first player does not consume energy, and second player consumes energy.

Each module contains the following:

module node01

// LOCAL STATE
s1 : [0..4];
[] s1=0 -> (s1'=1);
*[**Transmitting**] s1=0 -> Cs1Cf2: (s1'=1) + Cs1Ci2 : (s1'=2) ;*
*[**Listening**] s1=1 | s1=2 -> (s1'=3);*
*[**Sleeping**] s1=3 ->1: (s1'=4);*

End module

//

module node02

// LOCAL STATE

s2 : [0..4];

*[**Successful_transmission**] s2=0 -> Cs1Cf2: (s2'=2) + Cs1Ci2 : (s2'=3) ;*

*[**Faild_transmission**] s2=2 | s2=3-> Ci1Cs2: (s2'=1) + Ci1Cf2 : (s2'=2) + Ci1Ci2 : (s2'=3) + Ci1Cw2 : (s2'=4) ;*

*[**Sleeping**] s2=3 -> Cw1Cw2: (s2'=4) + Cw1Ci2: (s2'=3) +Cw1Cf2: (s2'=3) ;*

End module

Finally we will write the rewards of two players:

1. in state of **“sleeping”:**

rewards "sleeping"

// *Use state rewards:*

(s1=3) & (s2=3) :Ci1Cf2PayOff;

Endrewards

rewards

// *Use state rewards:*

(s1=4) & (s2=4) :Ci1Cf2PayOff;

Endrewards

2. in state of **"transmitting":**

rewards "transmitting"

// *Use state rewards:*

(s1=1) & (s2=2) :Ci1Cf2PayOff;

Endrewards

rewards

// *Use state rewards:*

(s1=2) & (s2=3) :Ci1Cf2PayOff;

Endrewards

3. in state of **"listening":**

rewards "listening"

// *Use state rewards:*

(s1=2) & (s2=3) :Ci1Cf2PayOff;

Endrewards

rewards

// *Use state rewards:*

(s1=3) & (s2=4) :Ci1Cf2PayOff;

Endrewards

4.6.1. Property Specification

- Players p1 and Player p2 have a strategy to ensure that the probability of reaching "transmitting" is exactly 0.35.
  <<p1,p2>> P=0.35 [F "transmitting"]
- which states that Player p1 has a strategy to ensure that the probability of reaching a state satisfying s1=1 within 5 time-steps is at least 0.99, regardless of the strategies of any other player.
  <<p1>> P>=0.99 [F<=5 (s1=1)]
- which states that Player p2 has a strategy to ensure that the probability of reaching a state satisfying s1=1 and s2=2 is less or equal 1.

  <<p2>> P<=1 [F (s1=1)&(s2=2)]

- Players p1 and p2 can ensure that the probability of reaching an "transmitting" state within 100 time-steps is < 0.95.

  <<p1,p2>> P<0.95 [F<=100 "transmitting"]
- Can both players collaborate to achieve an expected sleeping of precisely 0?

  <<p1,p2>> R{"sleeping"}=0 [Fc "sleeping"]
- Maximum probability with which Player p1 can listening.

  <<p1>>Pmax=? [F ("listening")]

General Conclusion

In this dissertation, we surveyed the Wireless Sensor Networks in model checking from many aspects, we have seen quantitative verification of Ubiquitous Computing(WSNs), and how this mechanism works and we found it effective, and we learned through this work how to work with PRISM-games tool, and we found it not much different from PRISM Language where the PRISM-games depends on players that incorporate competitive or collaborative behaviour, thus, we have seen Model Checker from a different aspect. We learned how to do research build on other researches. Through this work, we tried to achieve excellent results and reach all our goals, however we could not maintain completely the energy problem in WSN, we aim to reach this goal in detail manner in future works.

References

[1] J.Gubbi , R. Buyya, Internet of Things (IoT): A vision, architectural elements, and future directions, 2013

[2] Wikipedia, www.wikipedia.com/internet_of_things

[3] O.Vermesan,P.Friess,internet of things-Converging Technologies for Smart Environments and Integrated Ecosystems.

[4] Internet of Things:Wireless Sensor Networks, International Electrotechnical Commission, Switzerland,2014

[5] J. Krumm Microsoft Corporation Redmond, Washington, U.S.A. Ubiquitous Computing Fundamentals

[6] Wikipedia,www.wikipedia.com/Ubiquitous_computing

[7] Christel Baier, Joost-Pieter Katoen, Principles of Model Checking

[8] Edmund Clarke, Orna Grumberg, Model Checking

[9] Wikipedia, www.wikipedia.com/system_verification

[10] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert K. Brayton. Verifying continuous time Markov chains. In Rajeev Alur and Thomas A. Henzinger, editors, Proc. 8th International Conference on Computer Aided Verification (CAV'96), volume 1102 of LNCS, pages 269–276, Berlin, 1996. Springer

[11] Christel Baier, Joost-Pieter Katoen, and Holger Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In Jos C. M. Baeten and Sjouke Mauw, editors, Proc., Berlin, 1999. Springer..

[12] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst., 8(2):244–263, 1986.

[13] Prism, www.prismmodelchecker.org

[14] Wikipedia, www.wikipedia.com/Markov_Chain

[15] M.College ,Automatic Verification of Competitiv Stochastic Systems,A thesis submitted for the degree of Doctor of Philosophy, Oxford,2013

[16] Marta Kwiatkowska, David Parker, Taolue Chen, PRISM-games: A Model Checker for Stochastic Multi-Player Games, University of Oxford.

[17] Hai-Yan Shi, Wan-Liang Wang, Ngai-Ming Kwok, Sheng-Yong Chen,Game Theory for Wireless Sensor Networks:ASurvey,2012.

[18] Mohamed S. Abdalzaher, Karim Seddik, Maha Elsabrouty ,Game Theory Meets Wireless Sensor Networks Security Requirements and Threats Mitigation,2016.

[19] BADR BENMAMMAR1, FRANCINE KRIEF21 LTT Laboratory, University of Tlemcen, Algeria 2 LaBRI Laboratory Bordeaux 1 University, Talence, France ,Game theory applications in wireless networks: A survey,2014.

[20] Wikipedia, www.wikipedia.com/Wireless_Sensor_Networks.

[21] Vinoba.V1, Chithra.S.M2,The Study of Game Theory in Wireless Sensor Networks, Department of Mathematics, K.N. Government Arts college, Tamil Nadu, India, September-October 2014.

[22] Maria Svorenova, Marta Kwiatkowska, Quantitative Verification and Strategy Synthesis for Stochastic Games.

[23] Liqiang Zhao, Hailin Zhang, Jie Zhang, Using Incompletely Cooperative Game Theory in Wireless Sensor Networks, University of Bedfordshire,2014.

[24] www.prismmodelchecker.org

[25] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst., 8(2):244–263, 1986.

[26] Taolue Chen , Marta Kwiatkowska,David Parker ,Aistis Simaitis,Automatic verification of competitive stochastic systems

[27] Aistis Simaitis, A thesis submitted for the degree of Doctor of Philosophy,Automatic Verification of Competitive Stochastic Systems.

[28] Ahlam Saud Althobaiti1, Manal Abdullah2*Medium Access Control Protocols for Wireless Sensor Networks Classifications and Cross-Layering, Taif Univerfity , International Conference on Communication, Management and Information Technology (ICCMIT 2015).

[29] Kurtis Kredo, Prasant Mohapatra ,Medium Access Control in Wireless SensorNetworks, University of California.

الملخص:

صارت التكنولوجيا شيء ضروري في حياتنا اليومية و ذلك من خلال التواصل الدائم بين الأفراد و للقيام بذلك وضعوا شبكات.و في هذه المذكرة قمنا بدراسة لهذه شبكات الاستشعار اللاسلكيةوهي عبارة عن مجموعة من أجهزة الاستشعار التي تستخدم في نقل أو متابعة ظاهرة فيزيائية أو كيمائية و بعدها نقل المعلومات عن هذه الظاهرة لاسلكيا لمركز معالجة البيانات و قد توسعت استعمالاتها في القرن 21 في مجالات عدة مثل : الهياكل الأساسية، والاستكشاف العلمي، والمراقبة العسكرية، ومراقبة حركة المرور والسيطرة... الخ , ومع ذلك، لا تزال هناك العديد من المشاكل التي تعاني منها شبكة الاستشعار اللاسلكية. وتشمل قلة الموثوقيه لأنظمة الاتصالات اللاسلكية ، محدودية الطاقة المتاحة ، وإخفاق العقد، وما إلى ذلك. و من خلال هذا العمل قمنا بإنشاء نموذج يعمل على كيفية الحفاظ على الطاقة أثناء عمل هذه أجهزة الاستشعار و التقليل من استهلاك الطاقة.

Abstract

Technology has become a necessity in our daily lives through constant communication between individuals or objects, especially with the rise of Internet of Things (IoT). Wireless networks are the most prevalent today. In this thesis, we study wireless sensor networks, which are a set of sensors that are used to transport or follow a physical phenomenon, and then transfer the information about this phenomenon wirelessly to the data processing center. However, there are still many problems experienced by the wireless sensor network. They include low reliability for wireless communication systems, limited power availability, contract failure, etc. Through this work, we try to implement a formal analysis model in PRISM-games based on a collaborative game theory to reduce energy consumption.

Résumé

Dans cette mémoire, nous avons étudié ces réseaux de capteurs sans fil, qui est un ensemble de capteurs qui sont utilisés dans le transfert ou d'un phénomène physique de suivi ou chimique et après le transfert d'informations sur ce phénomène sans fil au centre de traitement des données et ont élargi leur utilisation au 21e siècle dans plusieurs domaines tels que: les infrastructures, l'exploration scientifique, ...etc. Cependant, il y a encore de nombreux problèmes qui affligent le capteur sans fil. Ils comprennent le manque de fiabilité des systèmes de communication sans fil, la puissance limitée disponible, et l'échec du contrat, et ainsi de suite. Grâce à ce travail, nous avons créé un modèle de travail sur la façon d'économiser l'énergie pendant les travaux de ces réseaux et de réduire la consommation d'énergie.