ABSTRACT

Software Defined Radio (SDR) is one of possibilities to realize the structure of device with a high mobility, flexibility and reconfigurability**.** Extending the flexibility further, a system capable to sense the spectrum space available for communication and adapt to it is Cognitive Radio. Cognitive radio is envisioned as the ultimate system that can sense, adapt and learn from the environment in which it operates. Sensing the available bandwidth an SDR (Software defined radio) in a Cognitive System, tunes the circuits in the System for transferring data at optimum data rates, permissible by the space available. So it is a must for the SDR to accordingly add processing circuits to maintain the System performance at variable working frequencies. Cognitive radio and also presents some useful results obtained to configure the SDR for higher bandwidth available in Cognitive Radio.

In our project we have used algorithms like Particle Swarm Optimization (PSO) as proposed and Grass Hopper Optimization (GHO), Ant Colony Optimization (ACO), Whale Optimization (WO) and Butterfly Optimization (BO) as existing system. All are measured in terms of enegy and from the results the proposed Particle Swarm Optimization (PSO) performs well compared to other algorithms.

1. INTRODUCTION

1.1 Introduction of Project

1.1 Sensor network:

A network comprised of interconnected sensor nodes exchanging sensed data by wired or wireless communication.

1.2 Sensor node:

A device consisting of sensor(s) and optional actuator(s) with capabilities of sensed data processing and networking. Sensor node consists of a great number of nodes of the same type (sensor nodes), which are spatially distributed and cooperate with each other. Each such node has a sensing element (sensor), a microprocessor (microcontroller), which process sensor signals, a transceiver and an energy source. Distributed over the object, sensor nodes with the necessary sensors make it possible to gather information about the object and control processes which take place on this object.

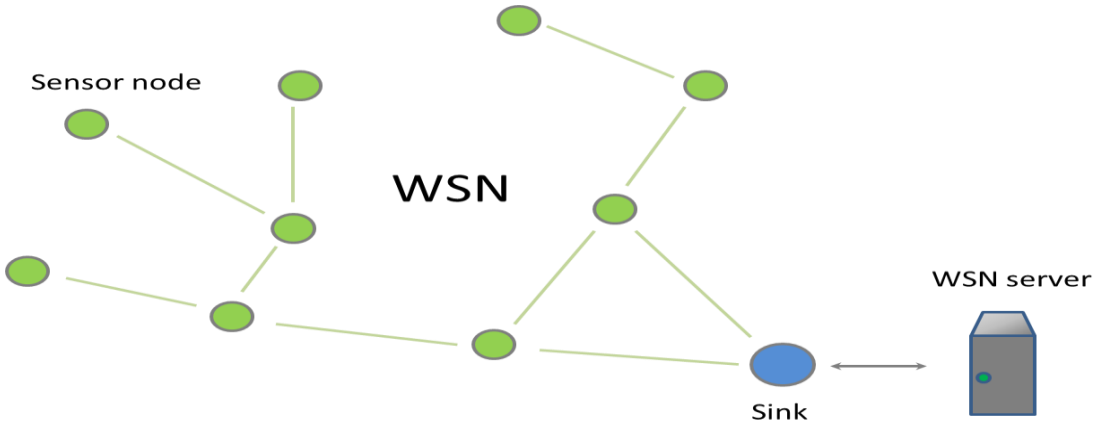


Fig 1 Overview of the network architecture

The above figure represents an example of a WSN. Here we can see a WSN which consists of twelve sensor nodes and a *network sink*, which also functions as a *gate*. Each sensor node is a device which has a transceiver, a microcontroller, and a sensitive element. Usually sensor node is an autonomous device. Each sensor node in WSN measures some physical conditions, such as temperature, humidity, pressure, vibration, and converts them into digital data. Sensor node can also

process and store measured data before transmission. Network sink is a kind of a sensor node which aggregates useful data from other sensor nodes. As a rule, network sink has a stationary power source and is connected to a *server* which is processing data received from WSN. Such connection is implemented directly, if server and WSN are placed on the same object. If it is necessary to provide a remote access to WSN, network sink also functions as a gate, and it is possible to interact with WSN through global network such as the Internet.

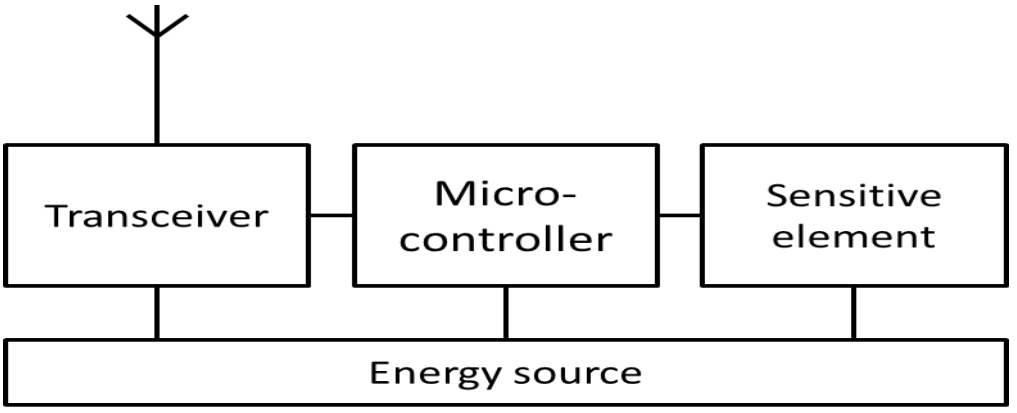


Fig 2 Sensor node inner structure

Some Sample Applications:

• Habitat and Ecosystem Monitoring

• Seismic Monitoring

• Civil Structural Health Monitoring

• Monitoring Groundwater Contamination

• Rapid Emergency Response

• Industrial Process Monitoring

• Perimeter Security and Surveillance

• Automated Building Climate Control

1.3 WSN Structure

WSN sink

Sensor nodes are the basis of a WSN. They collect and exchange data necessary for WSN functionality. Data collected by sensor nodes are the raw information and require processing. Depending on application, such data can be averaged statistical information or detailed measurements of parameters which define the condition of some object. A separate group of WSN applications is detecting and tracking of targets, for examples, vehicles, animals etc. Each of these cases requires processing of data provided by WSN. Usually it is impossible to perform this processing on sensor nodes themselves, by reason of energy saving and low computing power of sensor nodes. That is why in WSNs the final part of data processing is usually made beyond sensor nodes, on WSN server. WSN server is connected to only one sensor node which is called *sink* or *base station*. Sink is a collecting point of all data in the WSN and interact both with the sensor nodes and the WSN server.

1.4 WSNS With The Cluster Structure

Since energy content of sensor nodes is limited and non-renewable, it is important to use it in the most economical way. The figure illustrates the data streaming from sensor nodes to sink. Sink collects data from all the sensor nodes periodically. On the figure, every arrow between sensor nodes shows a transfer of a portion of measurements for a single period of data collection.

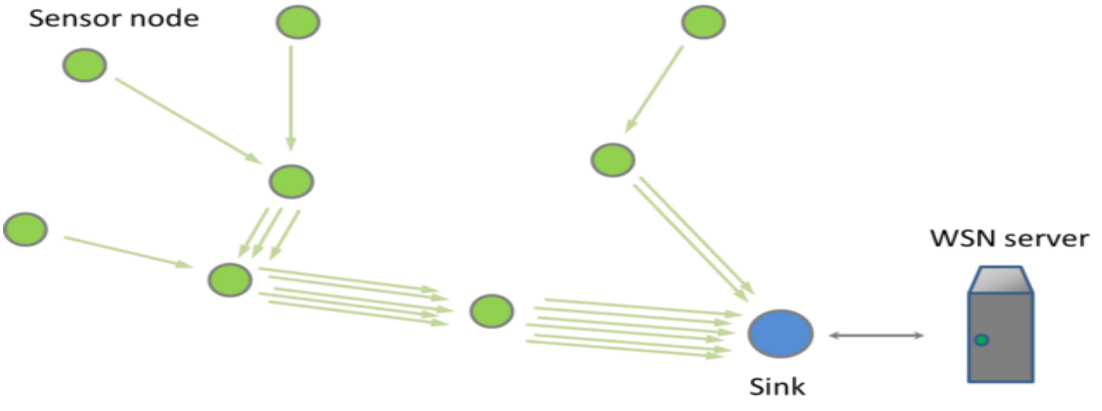


Fig 3 Data streaming from sensor nodes

Data is collected from all the sensor nodes; in result, the sensor nodes located closer to sink have to receive and transmit not only their own measurements, but also measurements from other sensor

nodes which are further from sink. So, transceivers of the nearest sensor nodes retransmit much more information, and hence they consume more energy than remote sensor nodes. And since sensor nodes are usually all of the same type and have equal energy content, it leads to the fact that the nearest sensor nodes fail much earlier than remote ones, and so the former disrupt the work of the rest of WSN.

So, if WSN application provides periodical data collecting (and it happens in the most cases), it turns out that time of autonomous operating of sensor nodes which are the nearest to sink is much reduced because of more frequent retransmitting. In the long run, traffic from all the sensor nodes is going through one sensor node that is nearest to sink. And the more sensor nodes are in a WSN, the higher is this traffic. From the point of view of energy saving, big WSNs with only one sink cannot consume resources effectively.

To solve this problem it is necessary to divide the WSN into clusters. Each cluster has its own sink and, in fact, is a separate, but smaller, WSN. And each sink communicates with the server directly. The figure represents a network with two sinks. On the figure the arrows also used to represent the amount of transmitted data. As we can see, the number of retransmissions is significantly decreased, and it reduces the load on the nearest to sinks sensor nodes.

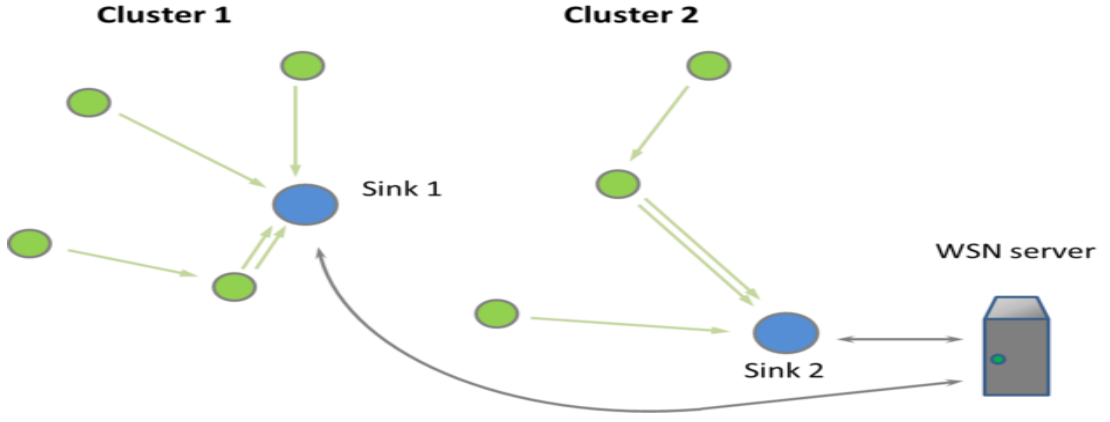


Fig 4 Multiple-sink WSN

Multiple-sink WSN is not a random division of one WSN into parts. In the most cases such division is made automatically when WSN is deployed and used. Sensor nodes automatically choose the sink to which they send data. This choice is made according to the algorithm of WSN protocol. Depending on requirements of the application, different criteria may be used, for example, the minimum time for data delivery, the minimum number of retransmissions, achieving the optimal traffic distribution in WSN and others.

1.5 WSN Gate

WSN organization schemes considered above suppose placement of all WSN elements in the same location. In practice, there is often necessary to have a remote access to WSN data. For example, WSN can be deployed in woodland in suburbs, and collecting and processing of WSN data have to be done in the office in a city. To organize data transmission from WSN to a remote server one uses specialized gates which receive sensor network data from sink and retransmit them using other (i. e. non-WSN) communication standard, wired or wireless. The figure represents such a network, which transmit collected data to server through the Internet using a gate.

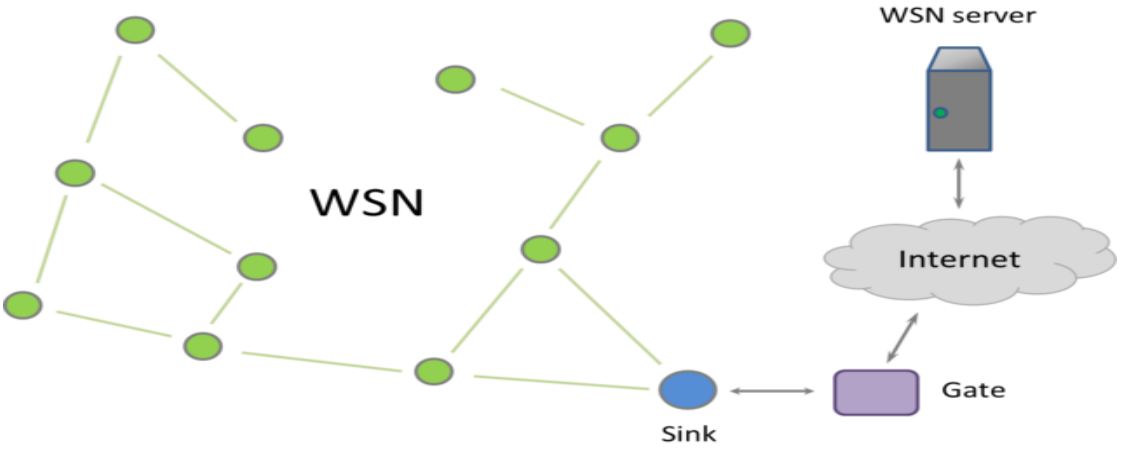


Fig 5 WSN server is connected via the Internet

Gates also provide the possibility to organize service provision. Nowadays, when access to the Internet via cellular, cable and satellite networks is available almost in any place in the world, connection of WSNs to the Internet in most cases is easy to implement. The figure represents the scheme of possible interaction between a user and a WSN.

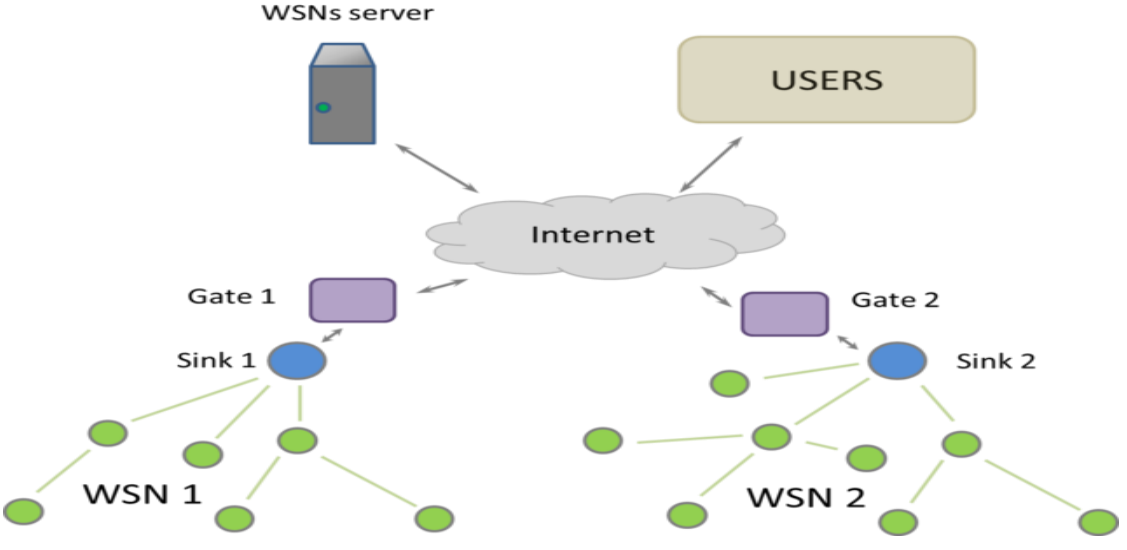


Fig 6 Scheme of provision of WSN services

1.6 Network Topology

Previously we have described traditional applications of WSN for data collecting and processing. Such applications have a special feature: they have one data collecting point, namely sink. But there are also applications where sensor nodes have not only to send information to sink, but to exchange data between themselves. That is why there are different schemes of organization of interaction between sensor nodes within WSN. These schemes are called network topologies. The main types of network topologies for WSNs are: *star*, *tree* and *mesh*. Different WSN standards support different types of network topologies.

Star

The star topology is widely used in computer networks, so when WSN appeared, it started being used also for organization of interaction between sensor nodes. The main characteristic of the star topology is connecting of all the sensor nodes to sink directly. The figure schematically represents this topology. In such cases sensor nodes are not connected between themselves, and all interactions between sensor nodes are taking place only via sink. Disadvantage of this topology is limited number of sensor nodes in such WSN. This limitation appears because all the sensor nodes have to be placed in the vicinity of sink, in order to connect to it directly. Another limiting factor is sink's performance, i. e. the maximum number of supported connections.

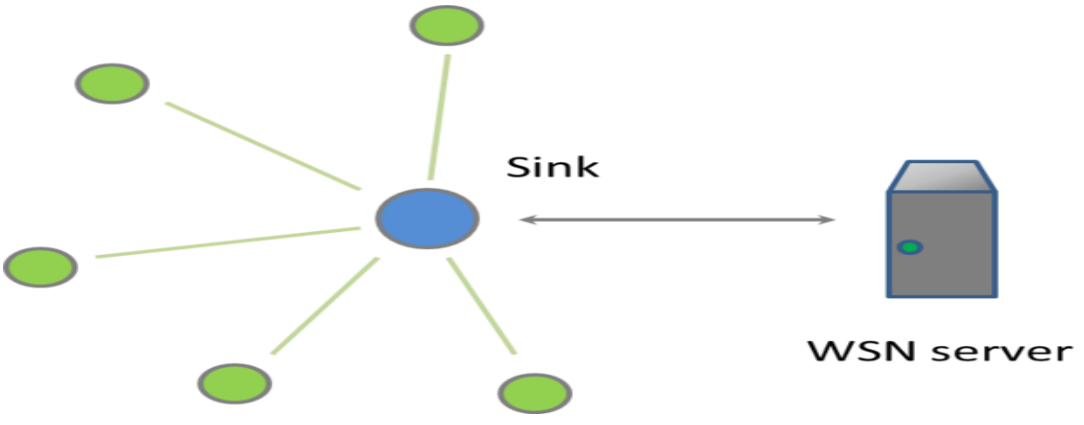


Fig 7 The star topology

Tree

The tree topology, in contradiction to the star topology, is much better suitable for WSN with the large number of sensor nodes. It has a hierarchical structure, as it is illustrated on The figure .Sensor nodes which are the nearest to sink interact with the sink directly. And more remote sensor nodes interact with the nearest ones according to the rules of the star topology. The tree topology also does not provide direct data exchange between all the sensor nodes. Data transmissions only from any sensor node to the sink and in the opposite direction are allowed. Also, in this topology data flow from the levels with greater numbers (i. e. “leaves”) can be delivered only through the levels with smaller numbers (i. e. “root” and “branches”). So, if on the first level there are only two sensor nodes, and the whole WSN consists of eleven sensor nodes, traffic will be delivered through these two sensor nodes much longer, because of data retransmission from nine sensor nodes on lower levels. Such network can fail quickly, because of energy consuming by the nearest to sink sensor nodes.

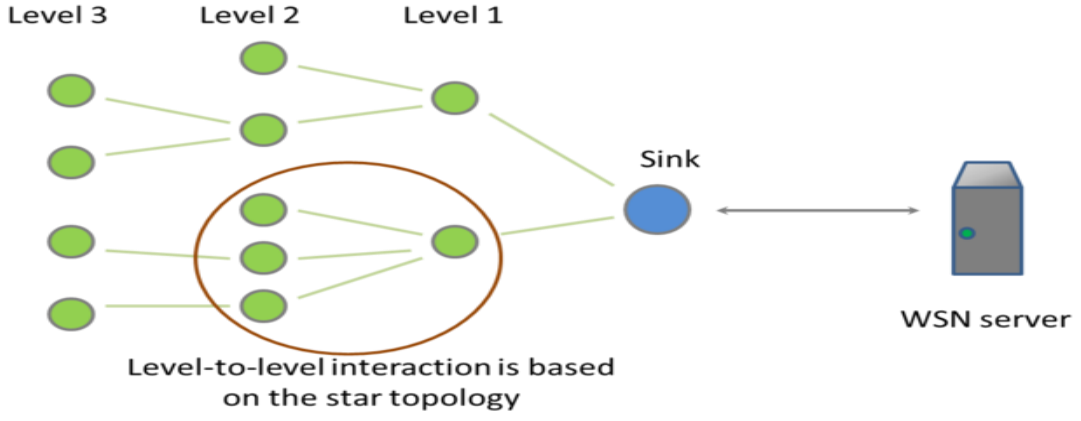


Fig 8 The tree topology

Mesh

The mesh topology is the most difficult one for implementation, but it provides much more opportunities for data exchange between sensor nodes. In WSN with the mesh topology interaction between sensor nodes is taking place according to the principle “with every nearest one”, as shown on The figure. It means that every sensor node cooperates with other sensor nodes, which are in its transceiver’s proximity. In such WSN data exchange between sensor nodes goes through the shortest ways and with the smallest number of retransmissions, what has a positive effect on the energy consumption of the sensor nodes.

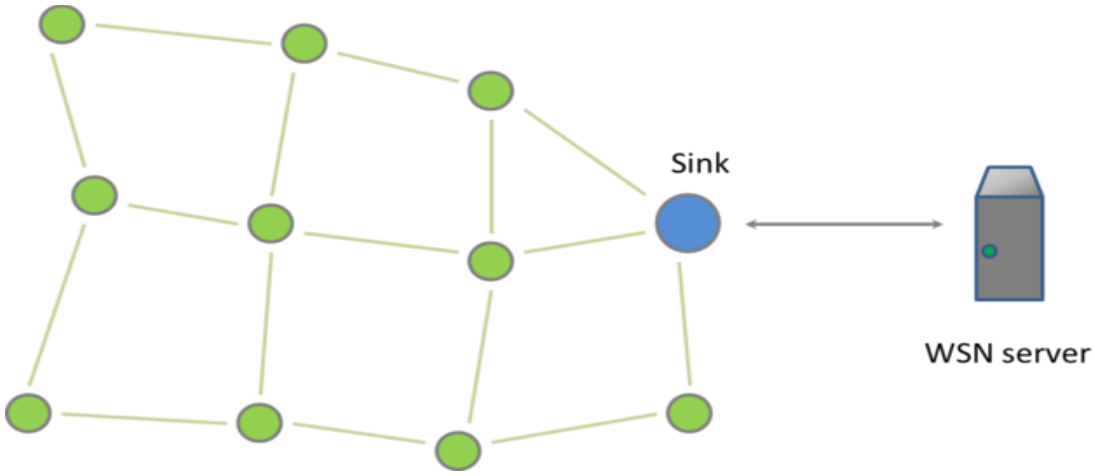


Fig 9 The mesh topology

1.7 General Design Issues

One of the most important tasks which have to be solved when working out a WSN for a specific application is the choice of the hardware platform which will serve as a basis for creating a sensor node. There are a lot of sensor nodes implementations from different vendors, but all the platforms have common elements. Choosing one or other existing platform or development of a new one from scratch have to be made in order to meet WSN functional requirements. Any hardware platform provides its own set of sensor node parameters. Variety of available platforms is caused by a wide range of WSN applications, and each existing platform has its own features according to the set of the tasks it is meant for. Also we have to understand that the current level of technology makes it necessary for the researchers to constantly find a balance between such parameters as size, productivity, battery lifetime, communication range, coverage, reliability, functionality, cost etc. The figure illustrates the correlation between the primary sensor node parameters. The arrows link directly related parameters, so improving a parameter in one end of the arrow will lead to worsening of the parameter in the other end of the arrow. For example, refinement of functionality (such as increasing the number of controlled parameters , improving the microcontroller performance) will inevitably cause an increase in cost, a decrease in battery lifetime and/or an increase in the size of a sensor node. So, the task of developing new hardware/software platforms which would support new technologies, expand the application scope, facilitate the deployment of WSNs is still relevant.

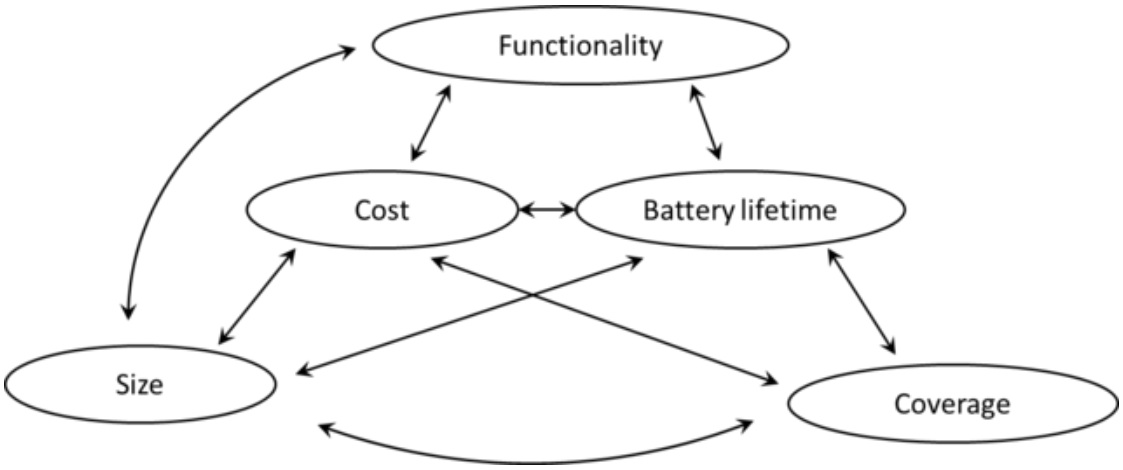


Fig 10 Relationships Of The primary sensor node parameters

In the next sections we are going to consider the internal structure of a sensor node as well as the main problems of sensor node development more precisely.

1.8 The Key Features Of Sensor Nodes

Before starting the consideration of the sensor node's structure in more detail, we should pay more attention to the main features of sensor nodes. They directly affect the capabilities of the whole WSN. That is why the requirements made to a WSN by a concrete application can always be converted to requirements for sensor nodes.

Energy Efficiency (Autonomy)

Unlike other common battery-powered mobile devices, sensor nodes deal with much more stringent requirements on energy efficiency, and it imposes restrictions on the all sensor node components. For example, for a mobile phone it is acceptable to keep working autonomously for a few days because the user usually have the possibility to charge the battery if necessary. But with sensor nodes we have another situation. The WSN parts may be spatially distributed on the area of many kilometers, especially if a WSN user is managing it via the Internet. At the same time, sensor nodes can be located in the inaccessible places, or the concrete location of each sensor node can be unknown. Also, a WSN may consist of dozens, hundreds or even thousands of sensor nodes. Under these conditions charging of sensor nodes by the user is out of question. That is why a sensor node must have high energy efficiency in order to keep working on small

and inexpensive battery for a few months and even years. This ultra-low-power operation can only be achieved by using low-power hardware components. Also, one of the key techniques extending the sensor node battery life is reduction of duty-cycle. This parameter is defined as the ratio of the sensor node active functioning time and the time when it is in the low power mode (the sleep mode). In WSNs with long lifetime sensor nodes most of the time are in the sleep mode, where sensor node power consumption is reducing in 3-4 times due to switching off all the main components excepting the part which is responsible for returning from the sleep mode when needed. After returning from the sleep mode the sensor node exchanges data with surrounding sensor nodes, takes readings from sensing elements, and then the sleep mode in turned on again.

Platform Flexibility

The majority of real applications require flexibility and adaptability of the WSN platform. In one application a user may need a WSN able to keep working for a few years, and herewith data update speed and data transmission delay won't play a significant role. For example, to monitor the soil temperature and humidity there is no need of frequent readings update and fast data transmission (because the soil temperature cannot change quickly), but it is very important that WSN which performs these functions keeps working as long as possible. In other applications such as monitoring of the spread of forest fires, fast detecting and fast data transmission will be more important, and the WSN lifetime will be less important parameter. So, each sensor node platform must have ability to be adjusted to meet the requirements of a specific application.

Reliability

Certainly, every WSN developer and manufacturer is interested in cost reduction of sensor nodes taking into consideration that every WSN has a great number of sensor nodes. Nevertheless, each concrete sensor node has to be reliable to such extent that it could work without breaking from the moment of turning on until the complete using of battery supply. In addition to increasing reliability of each sensor node, to provide the whole WSN reliability one may use adaptive protocols of data transmission management (*adaptive routing*). They are meant

for providing WSN general robustness when certain sensor nodes are failing. For example, if traffic from one or a few sensor nodes is going through the other sensor node and it suddenly fails, as it is illustrated on The figure, the WSN will change its structure and reconnect the “lost” node through the others nearest to it. It is worth mentioning that the main modern WSN platforms support this function.

There is also another common threat to WSN reliability which doesn’t deal with reliability of any concrete sensor node. It is interference with the signals of other wireless networks and household or industrial devices’ radiation. WSNs are often fully or partially located in places with significant electromagnetic fields of other wireless connection systems and appliances. In such cases these electromagnetic fields interfere with low-power transmitters in sensor node. This interference can be significant if it falls on radio spectrum in operation frequency range of sensor nodes’ transmitters. In this case connection between nodes in the interference area can get much worse or even break down, and here even operable sensor nodes cannot transmit collected data. In such situations to increase the system’s robustness to a node failure, a wireless sensor network must also be robust to external interference. The robustness of wireless links can be greatly increased through the use of *multi-channel and spread spectrum radios*. The figure represents principal of operation of the sensor nodes which support multi-channel radios. So, the possibility to change frequency channel for data transmitting is a necessary function for WSNs that are supposed to be deployed in a harsh electromagnetic environment.

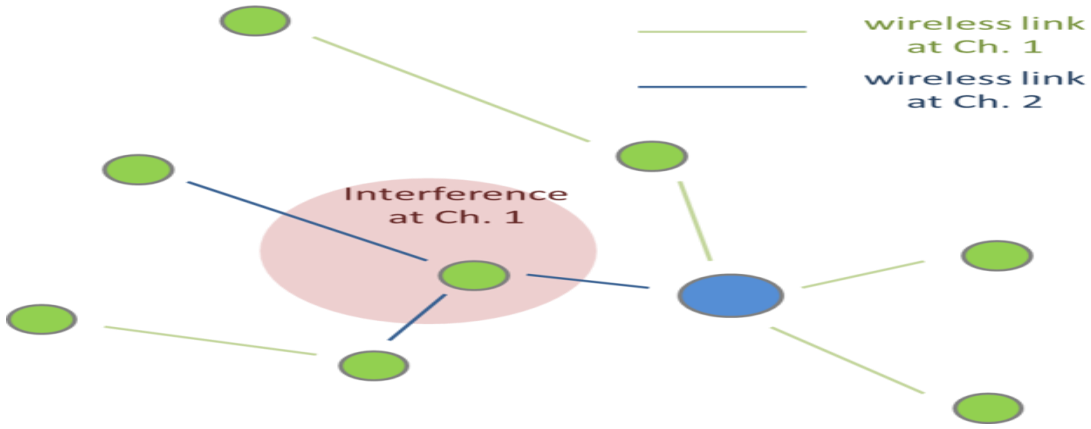


Fig 11 WSN working under conditions of strong interference on communication channel1

Information Security

Certain WSN applications make stringent requirements to information security. And this requirement becomes increasingly important, by reason of growth of cybernetic threats when WSNs are connected to the Internet. In order to meet the security requirements, sensor nodes must be capable of performing complex encrypting and authentication algorithms. In fact, radio communication channels can be easily tapped and become available for intruders. The only way to avoid it is encrypting of all data transmitted in the WSN. Many modern sensor nodes make it possible to flexibly set traffic encryption in the network. In some platforms it is made by means of software, but some sensor nodes include special hardware encryption blocks. But in any case, encryption requires additional expenditure of energy, and it has negative impact on WSN lifetime.

Another aspect of information security in WSNs is protection of sensor nodes' internal memory. Sensor node internal memory includes not only information meant to be transmitted in the WSN, but also private keys for traffic encryption. So it must be reliably protected from external intervention.

These information security aspects have to be taken into account simultaneously. On the one hand, weak protection of internal memory will make WSN private keys available making it possible to "crack" the WSN no matter how complex encryption algorithm is. On the other hand, weak traffic encryption will make data transmitted in the network available for sniffing and alteration by the intruder, even if internal memory of each sensor node is well protected.

The security issues in WSNs will be considered in detail in Section 6.3.

Transceiver Performance

One of the key sensor node characteristics is transceiver performance. The main parameters of transceiver performance which affect the sensor node characteristics are maximum data transfer rate, frequency range, modulation method, receiver sensitivity and transmitter power.

All these sensor node technical parameters affect such main WSN characteristic as reliability, the minimum spatial density of sensor nodes, the maximum readings update rate and lifetime. So, sensor node transceiver parameters are one of the main characteristics of WSN

platform.

Above we have considered how the interference affects WSN reliability on a qualitative level. It is possible to estimate quantitatively how interference affects wireless link between sensor nodes with the help of the mentioned transceiver characteristics. For estimating the impact of noise on the quality of signal reception, in information theory the *signal-to-noise ratio* (SNR) is used. This ratio shows in how many times the wanted signal (signal from other sensor node) received by the sensor node receiver exceeds the power level of interference. The higher the SNR is, the more powerful is useful signal as compared with noise, and the higher is probability to receive the signal correctly. For every method of signal modulation there is a special SNR value at which or above which communication between receiver and transceiver is possible. Also, in information theory there is a fundamental principle, which can be expressed (in a simplified form) by the following statement: the higher SNR is, the higher is maximum data transfer rate.

Now it is obvious that the SNR affects reliability and quality of wireless link between two sensor nodes. And the higher SNR is, the better is the quality of communication. So, the more powerful is emission of the first sensor node's transmitter, the higher is SNR of receiving sensor node, and the higher is wireless link quality, hence the whole WSN reliability. In addition, the closer the sensor nodes to each other are located, the better is the SNR for both of them. It means that the maximum distance between sensor nodes in WSN is inextricably linked with power of sensor node transmitter, and the maximum distance between sensor nodes specifies the minimum number of sensor nodes necessary for covering the given space by WSN.

Sensor node receiver *sensitivity* represents the ability to receive weak signals, for example, at a great distance from other sensor nodes, and it, as well as transmitter power, affects the maximum distance between the sensor nodes. It is worth mentioning that increasing the power and sensitivity of sensor node transmitter and receiver leads to higher energy consumption and cost of the sensor nodes. But this dependence is not linear, and the benefit in increasing the range of sensor node is not so great. That is why the most common characteristics of transceivers measure up with ones mW of power, which is acceptable in terms of energy consumption and provides reliable wireless connection between sensor nodes at the distance of about 10 meters.

Frequency range of transceiver affects the maximum possible rate of data exchange and the maximum possible distance between the sensor nodes. At the heart of this dependence are the physical laws of the radio signal. According to these laws, the higher is frequency used as carrier, the stronger is the signal attenuation with the distance. That is why the sensor nodes platforms which operate in lower frequency ranges allow having higher value of the maximum distance between sensor nodes in WSN. But the basic physical laws don't allow using very low frequencies for connecting sensor nodes in the majority of WSNs, because the size of the transceiver's antenna has to be the bigger, the lower is the frequency, and it affects the size of the sensor nodes.

The maximum speed of data transmission and reception by sensor node transceiver restrains the maximum speed of data gathering in the WSN. In addition, the higher is the maximum speed of data transmission, the higher is the energy consumption of transceiver during transmission and reception. On the other hand, the higher is speed of transmission, the less time is necessary for transmitting the same data; hence, transceiver will be switched on for less time. But high speed of transmitting also requires more computing power and energy for this computing, which is not always acceptable.

1.9 characteristics of the wsn which utilizes such sensor nodes computing power

Sensor node's microcontroller (and hence, consumes battery energy) uses its computing powers for two kinds of tasks. First kind of these tasks deals with supporting WSN functioning, the second task is reading and processing measurements of sensing element. Both kinds of tasks require certain computing power and take the time of the microcontroller. When the micro controller is busy, its energy consumption becomes significant.

The task of supporting WSN functioning, in the first place, is implementation data reception and further transmission algorithms that are part of the WSN communication protocol. Every sensor node is permanently receiving data from other surrounding sensor nodes. Microcontroller identifies these data and depending on the content transmits to the nearest sensor nodes, ignores them or saves to internal memory for further processing. All it happens in

accordance with the WSN communication protocol. Computing power of sensor node microcontroller has to be the higher, the higher is the maximum rate of data exchange, so that to have time for data decoding.

We can see the same situation with computing powers necessary for reading and processing of sensor measurements. Sensitive elements can produce a plenty of data which have to be timely processed. And the types of necessary processing can vary a lot, from simple averaging, digital filtration, tracking of some threshold exceeding to calculation of autocorrelation and spectral analysis. The last two operations are the example of the especially resources-consuming ones.

Size And Cost

Miniaturization, price reduction, and improvement of other parameters are the most important priorities from the very first researches in WSNs. The good example is the SmartDust project which took place in the end of 1990s and the beginning of 2000s. Miniaturization and price reduction of sensor were constantly expanding the possibilities of WSN applications, and in future they can lead to the widespread use of WSNs and to uprising of ubiquitous WSNs.

Above we have already considered the dependence between different WSN characteristics, and now, after considering the additional characteristics, it is possible to imagine how difficult is to find balance between them when developing the sensor nodes.

1.10 Introduction of Domain

Machine Learning:

In the statistical context, Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. While Machine Learning involves concepts of automation, it requires human guidance. Machine Learning involves a high level of generalization in order to get a system that performs well on yet unseen data instances.

Machine learning is a relatively new discipline within Computer Science that provides a collection of data analysis techniques. Some of these techniques are based on well established

statistical methods (e.g. logistic regression and principal component analysis) while many others are not.

Most statistical techniques follow the paradigm of determining a particular probabilistic model that best describes observed data among a class of related models. Similarly, most machine learning techniques are designed to find models that best fit data (i.e. they solve certain optimization problems), except that these machine learning models are no longer restricted to probabilistic ones.

Therefore, an advantage of machine learning techniques over statistical ones is that the latter require underlying probabilistic models while the former do not. Even though some machine learning techniques use probabilistic models, the classical statistical techniques are most often too stringent for the oncoming Big Data era, because data sources are increasingly complex and multi-faceted. Prescribing probabilistic models relating variables from disparate data sources that are plausible and amenable to statistical analysis might be extremely difficult if not impossible.

Machine learning might be able to provide a broader class of more flexible alternative analysis methods better suited to modern sources of data. It is imperative for statistical agencies to explore the possible use of machine learning techniques to determine whether their future needs might be better met with such techniques than with traditional ones.

1.11 Objective of the Problem

The main objectives of the problem are as follows:

1. Less throughput.
2. No energy checkup after each communication.
3. Time delay is high.
4. High packet loss.
5. Delivery ratio is less.

6. Attacking is very high.

1.12 Scope of the project

The main scope of this project is to provide communication continuously by checking the energy at every communication and also to find out the intrusion that is happening inside the network. The main objectives of the projects are as follows:

1. To find out early loss.
2. To minimize packet loss.
3. More throughputs.
4. To reduce time consumption.
5. Continuous energy check up of all nodes to avoid node failure.

2. LITERATURE REVIEW

2.1 HMM Based Channel Status Predictor for Cognitive Radio, Chang-Hyun Park; Sang-Won Kim; Sun-Min Lim; Myung-Sun Song

Publisher: IEEE 2022

Nowadays, many researchers are interested in cognitive radio (CR) technology. We can say that the CR is the extended technology of software defined radio. Both technologies seem to be similar but there is an important difference. That is a sensing and channel management function. CR always senses incumbent users (IU) (or primary users) appearing on the channel the CR has been used and the CR must evacuate from that channel for preventing IU from interferences. For this purpose, CR should include a functionality of being able to find new relevant channel to move. So, CR must evaluate the quality of empty channels. From the point of view, we propose the HMM based channel status predictor, which helps the CR evaluate the quality. We will implement a HMM channel predictor and it will predict next channel status based on past channel states.

2.2 Study of inter-base station beam switching considering asymmetric broadband transmission in cognitive radio, Satoshi Imata; Mitsuo Nohara; Kanshiro Kashiki

Publisher: IEEE 2022

The authors considered applying cognitive radio to facilitate the efficient use of frequency resources over multiple radio links. They proposed a beam-switching scheme for radio links among inter-base stations and analyzed transmission performance characteristics by response time for Web browsing. The current modeling assumes mobile WiMAX transmissions to be a realistic and efficient form of broadband radio communications. In this paper, the transmission characteristics of inter-base station beam switching are evaluated by software simulations, assuming broadband asymmetric transmission of up- and downlinks. Furthermore, the evaluations extend to cascaded topologies applicable to disaster relief deployment. The results show that beam switching of broadband transmission has better transmission performance than

fixed capacity assignment of a band-division transmission depending on conditions in basic and cascaded cases.

2.3 Throughput comparison for Cognitive Radio network under various conditions of primary user and channel noise signals, Neelam Chandwani; Anjana Jain; Prakash D. Vyavahare

Publisher: IEEE 2022

Spectrum scarcity is increasing with the growing demand of number of wireless technologies and services. Cognitive Radio (CR) techniques have become reliable solution for solving spectral congestion problem. For utilization of unused electromagnetic spectrum in a cognitive radio network, it is necessary for secondary users to monitor the available spectrum (allocated to primary users) and identify the free spectrum (spectrum holes). In this paper primary user signal and channel noise conditions are discussed. Earlier literature has worked on one of these conditions. What are these conditions? Primary user signal and channel noise can be real valued signal or complex valued signal. There modulation schemes may also be different. In this paper classification of primary user signal and channel noise is discussed. The probability of false alarm, probability of detection and throughput of secondary network (user) is calculated and a comparison of throughput of secondary user for different conditions of primary user signal and channel noise is being presented.

2.4 Research project concerning cooperative heterogenious radio networks for reliability improvements, Kanshiro Kashiki; Kazunori Takeuchi; Akira Yamaguchi

Publisher: IEEE 2022

In Japan, several new Research and Development (R & D) projects have been started, whose main purpose is to attain the efficient use of the restricted resources of frequency bands. This paper introduces the objective and outline of the project in which we have been engaged. The test-bed facility has been developed and their features, specifications and some experimental results are also described. Our project focuses on cooperative heterogeneous radio networks to improve the reliability on the radio communication link. The technology developed in the project

is considered to be advanced network technology that includes the cognitive radio system being studied in many research organizations.

2.5 Demonstration of plug-and-play cognitive radio network emulation testbed, Sohraab Soltani; Yalin E. Sagduyu; Jason H. Li; Jared Feldman; John Matyjas

Publisher: IEEE 2022

In this demonstration, we present the capabilities of a comprehensive cognitive radio system, CREATE-NEST. We provide hardware demonstration of a Cognitive Radio nEtworking ArchiTecturE (CREATE) over the Network Emulator Simulator Testbed (NEST). CREATE is a distributed and scalable network architecture operating without a common control channel and is equipped with cross-layer design functionalities. NEST provides plug-and-play cognitive network implementation with software-defined radios (SDRs). In this demonstration, we use eight USRP N210 radios, each running full network protocol stack and communicating with each other over multiple hops and different frequency bands. Instead of over-the-air transmissions, NEST uses RFnest (Radio Frequency Network Emulator Simulator Tool) to generate the air environment by changing channel properties digitally. We demonstrate the capability of cognitive network nodes to discover local neighborhood, sense the spectrum, estimate channels, dynamically access the spectrum, select channels, route traffic, and avoid congestion, primary users and RF interferers. We present different topology and mobility scenarios and show the cognitive radio performance (throughput, delay, energy, and overhead) with interactive GUI.

2.6 Versatile reconfiguration of radiation patterns, frequency and polarization: A discussion on the potential of controllable reflectarrays for software-defined and cognitive radio systems, Julien Perruisseau-Carrier

Publisher: IEEE 2022

The application of the reconfigurable reflectarray structure to software-defined and cognitive radio is discussed for the first time. The main idea is that the reflectarray has the potential-in addition to dynamical beamforming - of simultaneous frequency and polarization reconfiguration or diversity/multiplexing gains, at low complexity and cost. We discuss the expected benefits of this potentially universally reconfigurable antenna system in the context of (MIMO) software-

defined and cognitive radio. We also deduce the requirements on the controllable reflectarray unit cell phase characteristics and feed antennas, to achieve the desired novel reconfiguration capability.

2.7 Multiuser diversity analysis in spectrum sharing cognitive radio networks, F. A. Khan; T. Ratnarajah; M. Sellathurai

Publisher: IEEE

Cognitive radio has recently been proposed as an efficient paradigm to solve the spectrum scarcity in modern communication systems. In this paper, we consider spectrum sharing cognitive radio networks that utilize spectrum bands licensed to primary network. Spectrum sharing cognitive radio networks operate by ensuring that interference power at primary receiver remains below a certain threshold. We analyze the performance of the spectrum sharing cognitive radio networks with multiuser selection. Analytical and simulation results show that multiuser diversity gain in each network is substantially larger than that in the conventional wireless networks.

2.8 Prototype of a Cognitive Radio System with Cooperative Sensing and Interference Alerting, Munehiro Matsui; Kazunori Akabane; Hiroyuki Shiba; Kazuhiro Uehara

Publisher: IEEE 2022

Cognitive radio systems provide dynamic spectrum access, which is expected to improve frequency use efficiency. These systems need a sensing function to avoid interference from other radio stations. However, it is impossible to completely avoid interference because no perfect sensing function has been developed. To solve this problem, we propose using a new cognitive radio system with three main functions: cooperative sensing, interference alerting, and frequency management. We developed and evaluated a prototype of the system and found that the system is effective.

2.9 A novel approach for cognitive radio sensing using wideband chirp signal, Ahmed Barnawi

Publisher: IEEE 2022

Sensing the radio environment is probably the most important and challenging aspect in cognitive radio systems. Traditional sensing techniques have proven shortcomings ranging from inaccuracy to implementation complexity. Chirp signal is a wideband signal with variety of applications in communication and signal processing. In this paper, we introduce a novel approach for sensing the radio spectrum using chirp signal. This method is showing potential to improve sensing in cognitive radio environment at tolerable level of complexity.

2.10 Constructing accurate Radio Environment Maps with Kriging Interpolation in Cognitive Radio Networks, Danlei Mao; Wei Shao; Zuping Qian; Hong Xue; Xin Lu; Hongsheng Wu

Publisher: IEEE 2022

The increased development of information industry and relevant radio communication services is making the spectrum management problem more challenging to solve. The Radio Environment Map (REM) can be a powerful tool for solving spectrum scarcity and spectrum access problems, making context-aware resource allocation more efficient. We can optimize the accuracy of the REM using a geostatistical tool named Kriging interpolation in cognitive radio networks. In this paper, the proposed REM construction method combines residual maximum likelihood-based radio propagation parameter estimation with Kriging-based transmission power prediction. Additionally, we compare the performance of the constructed REMs with the metric of root mean square error (RMSE) in three methods: the path loss-based method, the Kriging-based method without the fit of a path loss model beforehand, and the proposed method. With the Monte Carlo simulation, the result indicates that the path loss-based method provides the most unsatisfied performance and the interpolation accuracy can be improved by 2 dB with the proposed method, which proves that our approach can effectively determine sharing conditions of radio spectrum use both in time and space.

2.11 A study of consolidating OFDMA radio networks for downlink in cognitive radio system, Takashi Fujimoto; Kazunori Takeuchi; Akira Yamaguchi

Publisher: IEEE 2022

Cognitive radio is a great candidate for creating a new fundamental network architecture improving the efficiency of frequency resource usage. In a heterogeneous network, a cognitive radio system composed with a multi-transmission link is required to select the optimized wireless access network to the terminal. To make the proper selection of a wireless access network to the terminal, the relative amount of radio resources in use for each network is an important parameter. We call this parameter the ¿occupation level¿ in this paper, which is one of the most useful parameters for the selection as well as the Received Signal Strength Indicator (RSSI). A wireless access network adopting an Orthogonal Frequency Division Multiple Access (OFDMA) scheme has a certain occupation level in its Up Link/Down Link (UL/DL) resource allocation. The UL/DL resource allocation is not easily accessible from equipment outside the base station; how to extract these data and utilize them is unresolved. In this paper, we propose a new scheme for extracting and utilizing the UL/DL resource allocation for the occupation level of the cognitive radio in a Long Term Evolution (3GPP LTE) network. From the system load point of view, we confirmed the method is practical by examining these data being extracted from an LTE radio system while transmitting full buffered payload data.

2.12 Towards cognitive radio networks: Spectrum utilization measurements in suburb environment Vaclav Valenta; Zbynek Fedra; Roman Marsalek; Genevieve Baudoin; Martine Villegas

Publisher: IEEE 2022

This paper deals with spectrum utilization measurements in the frequency band from 100 MHz up to 3 GHz. The measurement is based on the energy detection principle using wideband logarithmically periodic antenna. The results point out the fact, that the frequency spectrum is not utilized in an optimal manner and that there do exist less or more utilized licensed frequency bands that could be possibly used by cognitive radios in an opportunistic way. Cognitive radio concept for better spectrum utilization is introduced here along with an overall approach regarding spectrum utilization in the next generation wireless networks.

3. PROBLEM STATEMENT AND METHODOLOGY

3.1 PROBLEM DEFINITION

- ✓ Commercial Wireless network standards are continuously evolving from 2G to 2.5G/3G and then to 4G. The difference in networks of each generation being significantly in link-layer protocol standards cause problems to subscribers, wireless network operators and equipment vendors.

- ✓ Subscribers are forced to buy new handsets whenever a new generation of network standards is deployed. Wireless network operators face problems during migration of the network from one generation to next due to presence of large number of subscribers using legacy handsets that may be incompatible with newer generation network. The network operators also need to incur high equipment costs when migrating from one generation to next. Equipment vendors face problems in rolling out newer generation equipment due to short time-to-market requirements.

- ✓ The air interface and link-layer protocols differ across various geographies (for e.g. European wireless networks are predominantly GSM/TDMA based while in USA the wireless networks are predominantly IS95/CDMA based). This problem has inhibited the deployment of global roaming facilities causing great inconvenience to subscribers who travel frequently from one continent to another. Handset vendors face problems in building viable multi-mode handsets due to high cost and bulky nature of such handsets.

- ✓ Wireless network operators face deployment issues while rolling-out new services/features to realize new revenue-streams since this may require large-scale customizations on subscribers' handsets.

3.2 METHODOLOGY

3.2.1 Existing System

Grass Hopper Optimization (GHO):

Grasshopper are insects. They are considered a pest due to their damage to crop production and agriculture. Although grasshoppers are usually seen individually in nature, they join in one of the largest swarm of all creatures. The size of the swarm may be of continental scale and a nightmare for farmers. The unique aspect of the grasshopper swarm is that the swarming behaviour is found in both nymph and adulthood. Millions of nymph grasshoppers jump and move like rolling cylinders. In their path, they eat almost all vegetation. After this behaviour, when they become adult, they form a swarm in the air. This is how grasshoppers migrate over large distances. The main characteristic of the swarm in the larval phase is slow movement and small steps of the grasshoppers. In contrast, longrange and abrupt movement is the essential feature of the swarm in adulthood. Food source seeking is another important characteristic of the swarming of grasshoppers. As discussed in the introduction, nature-inspired algorithms logically divide the search process into two tendencies: exploration and exploitation. In exploration, the search agents are encouraged to move abruptly, while they tend to move locally during exploitation. These two functions, as well as target seeking, are performed by grasshoppers naturally. Therefore, if we find a way to mathematically model this behaviour, we can design a new nature-inspired algorithm

Ant Colony Optimization (ACO):

In computer science and operations research, the **ant colony optimization** algorithm (**ACO**) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Artificial ants stand for multi-agent methods inspired by the behavior of real ants. The pheromone-based communication of biological ants is often the predominant paradigm used. Combinations of artificial ants and local search algorithms have become a method of choice for numerous optimization tasks involving some sort of graph, e.g., vehicle routing and internet routing.

As an example, ant colony optimization is a class of optimization algorithms modeled on the actions of an ant colony. Artificial 'ants' (e.g. simulation agents) locate optimal solutions by moving through a parameter space representing all possible solutions. Real ants lay down pheromones directing each other to resources while exploring their environment. The simulated 'ants' similarly record their positions and the quality of their solutions, so that in later simulation iterations more ants locate better solutions.[4] One variation on this approach is the bees algorithm, which is more analogous to the foraging patterns of the honey bee, another social insect.

This algorithm is a member of the ant colony algorithms family, in swarm intelligence methods, and it constitutes some metaheuristic optimizations. Initially proposed by Marco Dorigo in 1992 in his PhD thesis, the first algorithm was aiming to search for an optimal path in a graph, based on the behavior of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants. From a broader perspective, ACO performs a model-based search and shares some similarities with estimation of distribution algorithms.

Whale Optimization (WO):

Whale Optimization Algorithm (WOA) is a recently proposed (2016) optimization algorithm mimicking the hunting mechanism of humpback whales in nature.

Whales are fancy creatures. They are considered as the biggest mammals in the world. An adult whale can grow up to 30 meters long and 180 tons weight. There are seven different main species of this giant mammal such as Killer, Minke, Sei, Humpback, Right, Finback, and Blue whales. Whales are mostly considered as predators. They never sleep because they have to breathe from the surface of oceans. In fact, half of the brain only sleeps. The interesting thing about the whales is that they are considered as highly intelligent animals with emotion.

According to Hof and Van Der Gucht , whales have common cells in certain areas of their brains similar to those of human called spindle cells. These cells are responsible for judgment, emotions, and social behaviors in humans. In other words the spindle cells make us distinct from

other creatures. Whales have twice number of these cells than an adult human which is the main cause of their smartness. It has been proven that whale can think, learn, judge, communicate, and become even emotional as a human does, but obviously with a much lower level of smartness. It has been observed that whales (mostly killer whales) are able to develop their own dialect as well.

Another interesting point is the social behavior of whales. They live alone or in groups. However, they are mostly observed in groups. Some of their species (killer whales for instance) can live in a family over their entire life period. One of the biggest baleen whales is humpback whales (Megaptera novaeangliae). An adult humpback whale is almost as size of a school bus. Their favorite prey are krill and small fish herds.

The most interesting thing about the humpback whales is their special hunting method. This foraging behavior is called bubble-net feeding method. Humpback whales prefer to hunt school of krill or small fishes close to the surface. It has been observed that this foraging is done by creating distinctive bubbles along a circle or '9'-shaped path. Before 2011, this behavior was only investigated based on the observation from surface. However, Goldbogen et al. investigated this behavior utilizing tag sensors. They captured 300 tag-derived bubble-net feeding events of 9 individual humpback whales. They found two maneuvers associated with bubble and named them 'upward-spirals' and 'double-loops'. In the former maneuver, humpback whales dive around 12 meters down and then start to create bubble in a spiral shape around the prey and swim up toward the surface. The later maneuver includes three different stages: coral loop, lobtail, and capture loop. Detailed information about these behaviors can be found.

It is worth mentioning here that bubble-net feeding is a unique behavior that can only be observed in humpback whales. In this work the spiral bubble-net feeding maneuver is mathematically modeled in order to perform optimization.

Butterfly Optimization (BO)

BOA mimics the food foraging behavior of butterflies. To understand this algorithm some biological facts and how to model them in BOA are discussed in following subsections.

There are three phases in BOA:

(1) Initialization phase,

(2) Iteration phase

(3) Final phase. In each run of BOA,

First initialization phase, the algorithm defines the objective function and its solution space. The values for the parameters used in BOA are also assigned. After setting the values, the algorithm proceeds to create an initial population of butterflies for optimization. As the total number of butterflies remains unchanged during the simulation of BOA, a fixed size memory is allocated to store their information. The positions of butterflies are randomly generated in the search space, with their fragrance and fitness values calculated and stored. This finishes the initialization phase and the algorithm starts the iteration phase, which performs the search with the artificial butterflies created.the initialization phase is executed, then searching is performed in an iterative manner and in the last phase, the algorithm is terminated finally when the best solution is found.

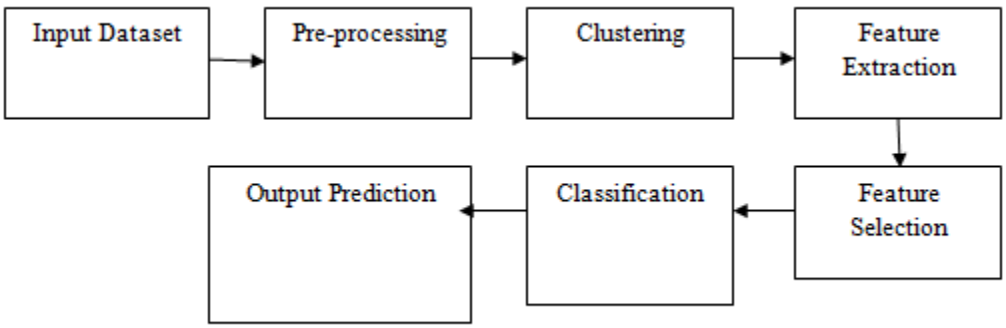
3.2.2 Proposed System

Particle Swarm Optimization (PSO):

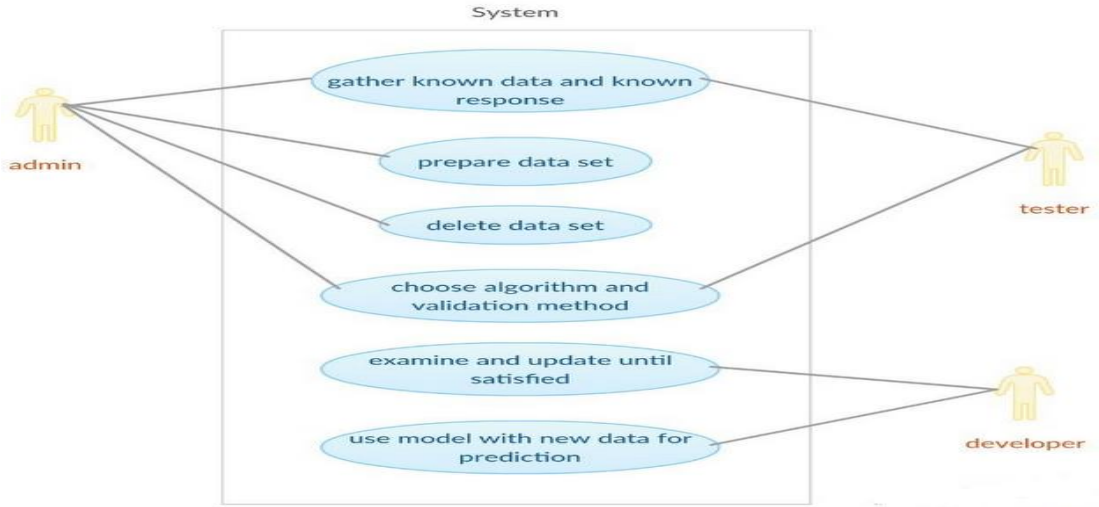
In computational science, **particle swarm optimization** (**PSO**) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

PSO is originally attributed to Kennedy, Eberhart and Shi and was first intended for simulating social behaviour, as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. The book by Kennedy and Eberhart describes many philosophical aspects of PSO and swarm intelligence. An extensive survey of PSO applications is made by Poli.

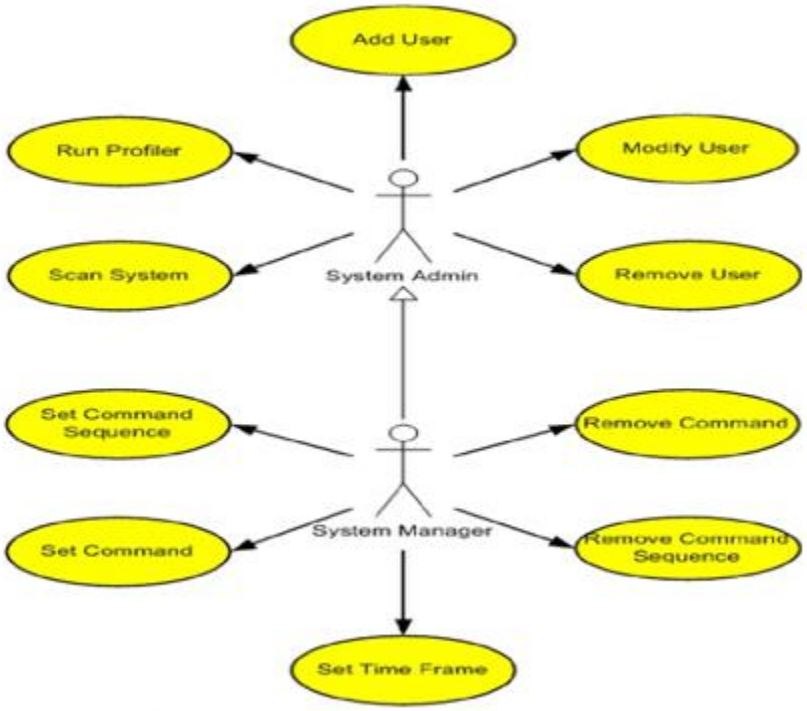
3.3 SYSTEM ARCHITECTURE



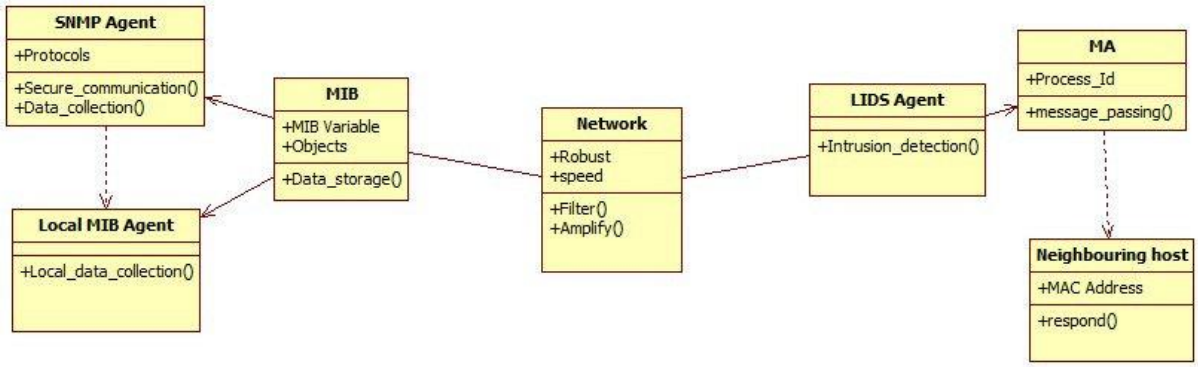
3.4. UML Diagram



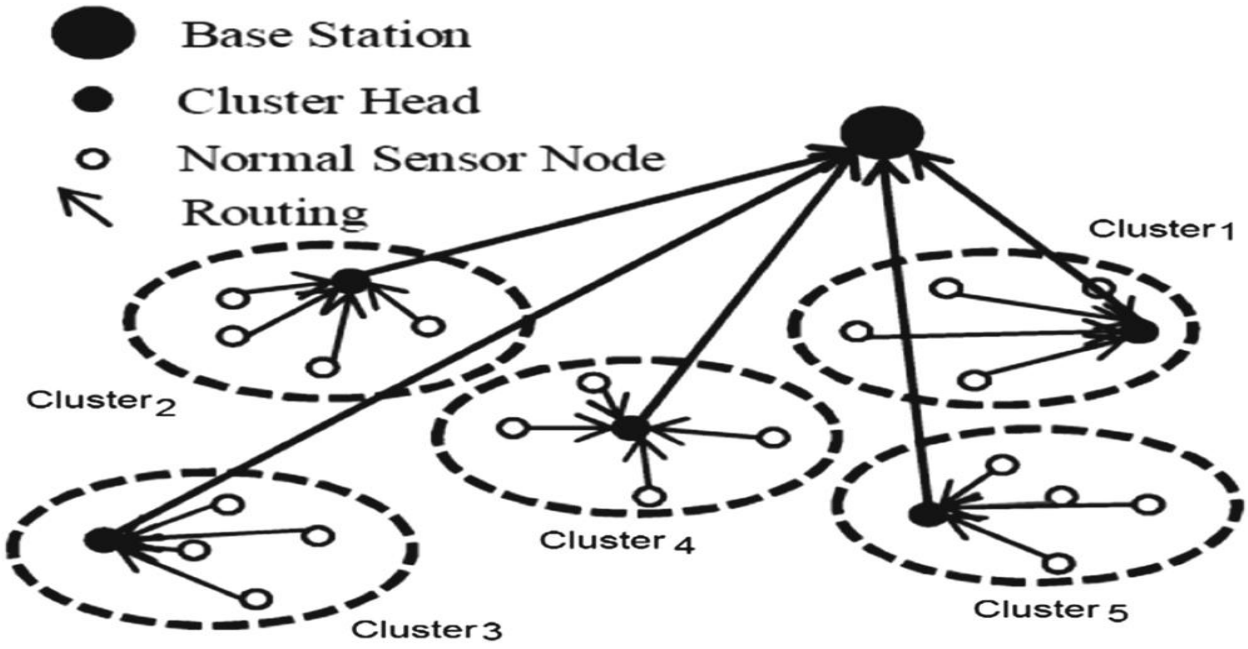
3.4.1 Use Case Diagram



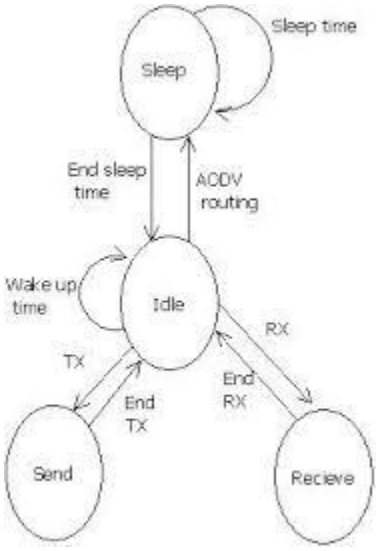
3.4.2 Class Diagram



3.4.3 Component Diagram



3.4.4 Deployment Diagram



4. SYSTEM IMPLEMENTATION

4.1 MODULE DESCRIPTION

There are 8 components in the system. They are

- Incoming packet
- Packet Capture
- Packet scanner
- Packet analyzer
- Labeling
- Training model
- Prediction:
- Output

Incoming packet: In our project directly connected to network (we capture online packet) and transfer to packet scanner.

Packet Capture: In Real time networking many packets are transferred from source to destination. In this module live packets from the network are captured and passed to the pre-processor module. According to the concept of machine learning the data is grouped or clustered based on its features or characteristics for e.g. protocols used by the packets.

Packet scanner: After catching the packet we use packet scanner for the purpose of scanning the packet. Packet scanning is the important part of our system.

Packet analyzer: Packet analyzer is also known as Packet sniffer. As data streams flow across the network sniffer capture each packet and if needed decodes the packet, raw data showing the values of various fields in the packet and analyzes its content accord- ing to specification.

Labeling: Labeling is used for defining the corresponding packet.

Training model: Training model contain set of trained data set which used to detect attack in the packet.

Prediction: Based on the training model we make the prediction that the packet is normal packet or abnormal packet.

Output: Based on the prediction (i.e. normal or abnormal) we generate output in the form for abnormal packet

4.2 ALGORITHM OF PROPOSED WORK

i. Dataset is divided into small overlapping or either non-overlapping blocks.

ii. Extract the features using traditional techniques.

iii. Extracted feature values corresponding to each key point are stored in matrix.

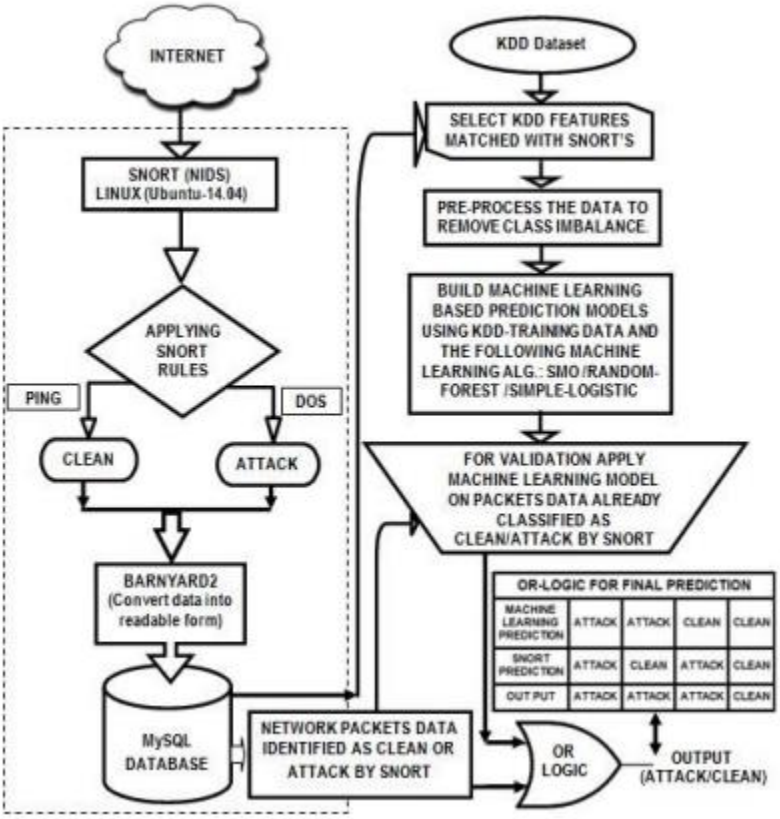
iv. Apply sorting techniques to get similar features that lie in nearness.

v. Introduce shift vector concept to find key point with similar shifting.

vi. Use the counter vector to count the occurrence same shifting key point and set the counter to 1.

vii. Similar regions are identified with the help of threshold value above steps are used.

4.3 FLOWCHART



4.4 COMPARATIVE STUDY OF EXISTING AND PROPOSED SYSTEM

In our project we have used algorithms like Particle Swarm Optimization (PSO) as proposed and Grass Hopper Optimization (GHO), Ant Colony Optimization (ACO), Whale Optimization (WO) and Butterfly Optimization (BO) as existing system. All are measured in terms of enegy and from the results the proposed Particle Swarm Optimization (PSO) performs well compared to other algorithms.

5. SYSTEM STUDY

5.1 FEASIBILTY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution. The analyst conducting the study gathers information using a variety of methods, the most popular of which are:

- Developing and administering questionnaires to interested stakeholders, such as potential users of the information system.
- Observing or monitoring users of the current system to determine their needs as well as their satisfaction and dissatisfaction with the current system.
- Collecting, examining, and analyzing documents, reports, layouts, procedures, manuals, and any other documentation relating to the operations of the current system.
- Modeling, observing, and simulating the work activities of the current system.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components. These components are:

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY

- SOCIAL FEASIBILITY
- OPERATIONAL FEASIBILITY

5.2 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.4 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.5 OPERATIONAL FEASIBILITY

The ability, desire, and willingness of the stakeholders to use, support, and operate the proposed computer information system. The stakeholders include management, employees, customers, and suppliers. The stakeholders are interested in systems that are easy to operate, make few, if any, errors, produce the desired information, and fall within the objectives of the organization.

6. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

6.1 Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is

correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

6.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.7 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed

- All links should take the user to the correct page.

6.8 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.9 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7. PYTHON SOFTWARE

A. Python

Python is a remarkably powerful dynamic, object-oriented programming language that is used in a wide variety of application domains. It offers strong support for integration with other languages and tools, and comes with extensive standard libraries. To be precise, the following are some distinguishing features of Python:

- Very clear, readable syntax.
- Strong introspection capabilities.
- Full modularity.
- Exception-based error handling.
- High level dynamic data types.
- Supports object oriented, imperative and functional programming styles.
- Embeddable.
- Scalable
- Mature

With so much of freedom, Python helps the user to think problem centric rather than language centric as in other cases. These features makes Python a best option for scientific computing.

B. Open CV

Open CV is a library of programming functions for real time computer vision originally developed by Intel and now supported by Willogarage. It is free for use under the open source BSD license. The library has more than five hundred optimized algorithms. It is used around the

world, with forty thousand people in the user group. Uses range from interactive art, to mine inspection, and advanced robotics. The library is mainly written in C, which makes it portable to some specific platforms such as Digital Signal Processor. Wrappers for languages such as C, Python, Ruby and Java (using Java CV) have been developed to encourage adoption by a wider audience. The recent releases have interfaces for C++. It focuses mainly on real-time image processing. Open CV is a cross-platform library, which can run on Linux, Mac OS and Windows. To date, Open CV is the best open source computer vision library that developers and researchers can think of.

C. Tesseract

Tesseract is a free software OCR engine that was developed at HP between 1984 and 1994. HP released it to the community in 2005. Tesseract was introduced at the 1995 UNLV Annual Test OCR Accuracy and is currently developed by Google released under the Apache License. It can now recognize 6 languages, and is fully UTF8 capable. Developers can train Tesseract with their own fonts and character mapping to obtain perfect efficiency.

7.1 ABOUT THE SOFTWARE "PYTHON"

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python - Environment Setup

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2

- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS
- BeOS
- Amiga
- VMS/OpenVMS
- QNX
- VxWorks
- Psion
- Python has also been ported to the Java and .NET virtual machines

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

You can download Python documentation https://www.python.org/doc/The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms −

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the *Modules/Setup* file if you want to customize some options.
- run ./configure script
- make
- make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.
- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Macintosh Installation

Recent Macs come with Python installed, but it may be several years out of date. See http://www.python.org/download/mac/ for instructions on getting the current version along

with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available.

Jack Jansen maintains it and you can have full access to the entire documentation at his website − http://www.cwi.nl/~jack/macpython.html. You can find complete installation details for Mac OS installation.

Setting up PATH

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The **path** variable is named as PATH in Unix or Path in Windows (Unix is case sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

Setting path at Unix/Linux

To add the Python directory to the path for a particular session in Unix −

- **In the csh shell** − type setenv PATH "$PATH:/usr/local/bin/python" and press Enter.
- **In the bash shell (Linux)** − type export PATH="$PATH:/usr/local/bin/python" and press Enter.
- **In the sh or ksh shell** − type PATH="$PATH:/usr/local/bin/python" and press Enter.
- **Note** − /usr/local/bin/python is the path of the Python directory

Setting path at Windows

To add the Python directory to the path for a particular session in Windows −

At the command prompt − type path %path%;C:\Python and press Enter.

Note − C:\Python is the path of the Python directory

Python Environment Variables

Here are important environment variables, which can be recognized by Python −

Sr.No.	Variable & Description
1	**PYTHONPATH** It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by the Python installer.
2	**PYTHONSTARTUP** It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH.
3	**PYTHONCASEOK** It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.
4	**PYTHONHOME** It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy.

Running Python

There are three different ways to start Python −

Interactive Interpreter

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter **python** the command line.

Start coding right away in the interactive interpreter.

```
$python # Unix/Linux
or
python% # Unix/Linux
or
C:> python # Windows/DOS
```

Here is the list of all the available command line options −

Sr.No.	Option & Description
1	**-d** It provides debug output.
2	**-O** It generates optimized bytecode (resulting in .pyo files).
3	**-S** Do not run import site to look for Python paths on startup.

4	**-v** verbose output (detailed trace on import statements).
5	**-X** disable class-based built-in exceptions (just use strings); obsolete starting with version 1.6.
6	**-c cmd** run Python script sent in as cmd string
7	**File** run Python script from given file

Script from the Command-line

A Python script can be executed at command line by invoking the interpreter on your application, as in the following −

```
$python script.py # Unix/Linux

or

python% script.py # Unix/Linux

or

C: >python script.py # Windows/DOS
```

Note − Be sure the file permission mode allows execution.

Integrated Development Environment

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

- **Unix** − IDLE is the very first Unix IDE for Python.
- **Windows** − PythonWin is the first Windows interface for Python and is an IDE with a GUI.
- **Macintosh** − The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine.

Note − All the examples given in subsequent chapters are executed with Python 2.4.3 version available on CentOS flavor of Linux.

We already have set up Python Programming environment online, so that you can execute all the available examples online at the same time when you are learning theory. Feel free to modify any example and execute it online.

PANDA

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc. In this tutorial, we will learn the various features of Python Pandas and how to use them in practice.

Audience

This tutorial has been prepared for those who seek to learn the basics and various functions of Pandas. It will be specifically useful for people working with data cleansing and analysis. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages is a plus. Pandas library uses most of the functionalities of NumPy. It is suggested that you go through our tutorial on NumPy before proceeding with this tutorial. You can access it from Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.

- High performance merging and joining of data.
- Time Series functionality.

Standard Python distribution doesn't come bundled with Pandas module.

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. This tutorial explains the basics of NumPy such as its architecture and environment. It also discusses the various array functions, types of indexing, etc. An introduction to Matplotlib is also provided. All this is explained with the help of examples for better understanding.

Audience

This tutorial has been prepared for those who want to learn about the basics and various functions of NumPy. It is specifically useful for algorithm developers. After completing this tutorial, you will find yourself at a moderate level of expertise from where you can take yourself to higher levels of expertise.

Prerequisites

You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.umPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy – A Replacement for MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

It is open source, which is an added advantage of NumPy.tandard Python distribution doesn't come bundled with NumPy module. A lightweight alternative is to install NumPy using popular Python package installer, **pip**.

```
pip install numpy
```

The best way to enable NumPy is to use an installable binary package specific to your operating system. These binaries contain full SciPy stack (inclusive of NumPy, SciPy, matplotlib, IPython, SymPy and nose packages along with core Python).

8. CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

In this project the CRs technology was presented. Energy Signal Detection is introduced as a figure of merit on which to base quantitative assessment of a radiometer's design including its calibration architecture and algorithm. The problem of the spectrum detection schemes was formulated which include Energy detection in time and frequency domain. Energy detection has been adopted as an alternative spectrum sensing method for CRs due to its simple circuit in the practical implementation and no information requires about the signal needed to detect.

8.2 Future Scope

Future research should consider other machine learning algorithms to ascertain more efficient ways to perform the classification technique on the datasets. It is recommended that further research should be carry out on other parameters that can improve the accuracy of detection

9. REFERENCES

[1] J. Mitola III and G. Q. Maguire, Jr., “Cognitive radio: Making software radios more personal," IEEE Personal Commuicatios Magazine, vol. 6, no. 4, pp. 13-18, Aug. 1999. DOI: 10.1109/98.788210.

[2] P. Pawełczak, “Cognitive Radio: Ten Years of Experimentation and Development,” IEEE Communications Magazine, vol. 49, no. 3, pp. 90-100, Mar. 2011. IEEE DOI: 10.1109/MCOM.2011.5723805.

[3] NTIA, “U.S. frequency allocations,” [online] http://www.ntia.doc. gov/osmhome/allochrt.pdf.

[4] K.G. Smitha and A.P. Vinod, “Cluster based power efficient cooperative spectrum sensing under reduced bandwidth using location information,” AEUE - International Journal of Electronics and Communications, vol. 66, no. 8, pp. 619-624, 2012. Elsevier DOI: 10.1016/j.aeue.2012.03.016.

[5] P. Wang and I. F. Akyildiz, “On the Origins of Heavy-Tailed Delay in Dynamic Spectrum Access Networks,” IEEE Transactions on Mobile Computing, vol. 11, no.2, pp. 204-217, 2012. IEEE DOI: 10.1109/TMC.2011.187.

[6] J. Ma, G. Ye Li, and B. H. Juang, “INVITED PAPER: Signal Processing in Cognitive Radio,” JournalProceedings of the IEEE, vol. 97, no. 5, pp. 805-823, 2009. IEEE DOI: 10.1109/JPROC.2009.2015707.

[7] M.B.H. Weiss, “Spatio-temporal spectrum holes and the secondary user,” Journal: 2011 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN), pp. 216-222, 2011. IEEE DOI: 10.1109/DYSPAN.2011.5936209.

[8] J. Ch. Clement, K. V. Krishnan, and A. Bagubali “Cognitive Radio: Spectrum Sensing Problems in Signal Processing,” International Journal of Computer Applications, vol. 40, no. 16, pp. 37-40, 2012. Foundation of Computer Science (FCS) DOI: 10.5120/5067-7475.

[9] B. Wang and K. J. Ray Liu, “Advances in Cognitive Radio Networks: A Survey,” IEEE Journal of Selected Topics in Signal Processing, vol. 5, no. 1, pp. 5 - 23, 2011. IEEE DOI: 10.1109/JSTSP.2010.2093210.

[10] K. Chowdhury, R. Doost-Mohammady, W. Meleis, M. Di Felice, and L. Bononi “Cooperation and Communication in Cognitive Radio Networks based on TV Spectrum Experiments,” 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 1-9, 2011. IEEE DOI: 10.1109/WoWMoM.2011.5986378.

[11] S. Haykin, “INVITED PAPER: Cognitive Radio Brain-Empowered Wireless Communications,” IEEE Journal on Selected Areas in Communications, vol. 23, no. 2, pp. 201-220, 2005. IEEE DOI: 10.1109/JSAC.2004.839380.

[12] FCC, ET Docket No. 03-108, “Facilitating Opportunities for Flexible, Efficient, and Reliable Spectrum Use Employing Cognitive Radio Technologies,” FCC Report and Order, 2005.

[13] IEEE Standard Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management, IEEE Std. 1900.1-2008, 2008. [online] http://standards.ieee.org/findstds/standard/1900.1-2008.html.

[14] Y. Tachwali, F. Basma, and H. H. Refai, “Adaptability and Configurability in Cognitive Radio Design on Small Form Factor Software Radio Platform,” Wireless Personal Communications, vol. 62, no. 1, pp. 1-9, 2012, Springer DOI: 10.1007/s11277-010-0035-3.

[15] J. Mitola, Cognitive Radio – An Integrated Agent Architecture for Software Defined Radio, Ph.D. Dissertation, Royal Institute of Technology, Kista, Sweden, 2000.

[16] R.W. Thomas, L.A. DaSilva, and A.B. MacKenzie, “Cognitive networks,” First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, pp. 352-360, 2005. IEEE DOI: 10.1109/DYSPAN.2005.1542652.

[17] I. F. Akyildiz, Won-Yeol Lee, M. C. Vuran, and Sh. Mohanty, “NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey,” Computer Networks, vol. 50, no. 13, pp. 2127-2159, 2006. Elsevier DOI: 10.1016/j.comnet.2006.05.001.

[18] J Mitola, "The Software Radio", IEEE National Telesystems Conference, pp. 13/15-13/23, 1992. IEEE DOI: 10.1109/NTC.1992.267870.

[19]. J. Mitola, “The software radio architecture,” IEEE Communications Magazine, vol. 33, no. 5, pp. 26–38, 1995.

[20]. J. Mitola III, “Software radio architecture: a mathematical perspective,” IEEE Journal on Selected Areas in Communications, vol. 17, no. 4, pp. 514-538, 1999.