MILKOG – a cognitive radio electronics warfare agent architecture

Tore Ulversøy and Thomas Thoresen

Norwegian Defence Research Establishment (FFI)

08 May 2012

FFI-rapport 2012/00419
1107

P: ISBN 978-82-464-2107-0
E: ISBN 978-82-464-2108-7

Keywords

Elektronisk krigføring

Kommunikasjon

Kognitiv radio

Dynamisk spektrums-tilordning

Multiagentsystem

Approved by

Tor-Odd Høydal	Project Manager
Anders Eggen	Director

English summary

The report documents investigations, design and development of agent controlled combined communication (COM), electronic surveillance (ES) and electronic attack (EA) nodes.

The report outlines an overall architecture for and the conceptual design of the MILKOG communications and electronic warfare agent. The objective of MILKOG is to manage radio transceivers and the radio spectrum in such a way that both communication and electronic warfare requirements are taken into account.

The report also describes a reduced-functionality experimental version of the MILKOG agent and multifunction (COM+ES+EA) platform, and documents some initial tests of a small system of MILKOG nodes. The experimental version of MILKOG demonstrates simple dynamic spectrum access where each agent is allowed to make its independent decision on which part of the spectrum to utilize. The demonstration functionality includes video transmission. It also demonstrates the combination of monitoring and jamming to form a reactive communications jammer functionality.

It is expected that MILKOG will form an expandable basis for research on combined COM and EW radio nodes and smart communications electronic attack features. While the prototype system and the initial tests of this system provide some level of support for the hypothesis that agent coordinated communication and electronic warfare is beneficial, further work is clearly needed. Suggestions for such further work are provided.

Sammendrag

Rapporten skisserer en overordnet arkitektur og konseptuelt design for en software-agent for kommunikasjon og elektronisk krigføring, MILKOG. Hensikten med MILKOG er å styre og koordinere transceivere slik at krav fra både kommunikasjon og elektronisk krigføring blir hensyntatt.

Rapporten beskriver også utviklingen av en eksperimentell versjon av MILKOG og en eksperimentell multifunksjons (kommunikasjon, monitorering, elektronisk angrep) plattform. Den eksperimentelle versjonen har redusert funksjonalitet i forhold til totalkonseptet, men demonstrerer en enkel form av dynamisk aksess til radiospektrumet. Demonstrasjonsfunksjonaliteten inkluderer videooverføring, og den demonstrerer også kombinasjonen av elektronisk monitorering og jamming som sammen utgjør en reaktiv jammer.

Det er forventet at MILKOG vil utgjøre en utvidbar basis for forskning på kombinerte kommunikasjons- og elektronisk krigføringsnoder, samt for forskning på smarte måter å designe funksjonalitet for elektroniske angrep på kommunikasjonsnettverk.

Prototype-systemet, og foreløpige tester av dette, gir støtte for en hypotese om at agent-koordinert kommunikasjon og elektronisk krigføring er nyttig, men det er klart at mer arbeid gjenstår for å belyse denne hypotesen. Rapporten inneholder forslag til videre MILKOG-aktiviteter.

Contents

1 Introduction

Electronic Warfare (EW) traditionally is using both separate personnel, information handling processes as well as equipment, from that of Communications (COM). A common challenge is that of coordination of the actions between these disciplines, such that the EW activities do not interfere with and destroy the performance of the COM networks. Moreover, EW may benefit from observations made in the COM systems. With the evolution of several key technologies, such as Multi-Agent Systems (MAS), distributed processing and interaction and Software Defined Radio (SDR), a technological basis for a closer integration of EW and COM is appearing.

Cognitive radio was proposed by Mitola and Maguire [1] in 1999 as a merge between SDR and artificial intelligence in the form of model-based reasoning. The radios were envisioned to include a software agent that would be capable of reasoning about a number of radio-domain aspects, e.g. when the radio should do simplified signal processing in order to save battery power. As well known, the issues related to dynamic and opportunistic access of the radio spectrum has since dominated the research agenda. The counterparty to the cognitive radio, the cognitive jammer [2] has however also been put forward and is likely a hot research topic in military research organizations. Cognitive radio is hence pushing the balance between COM and EW systems to a higher complexity level, where both COM and EW may utilize cognition to optimize their responses to counter-parties.

This report documents investigations, design and development of multifunction radio units, where by multifunction is meant the combination of COM and EW. The research goals are:

- Outline a system and an agent architecture for a MILKOG cognitive radio electronic warfare agent.
- Document the implementation of a simplified, reduced functionality, prototype MILKOG agent, based on the system and agent architecture.
- Design and implement the COM, ES and EA functionalities of the prototype.
- Document some initial small-scale prototype measurements with the agent, using real radio channels for the data payload, in order to provide some levels of support for the hypothesis that the suggested system and agent architecture will work in a real environment. The large-scale verification of the concepts and detailed solutions for the MILKOG system is however outside the scope of this report and outside the scope of the MILKOG activities in the OPEK II project.

MILKOG is intended to be a combined decision agent for Dynamic Spectrum Access (DSA) of the COM networks as well as handling Electronic Surveillance (ES) and Electronic Attack (EA). It aims to balance its own local COM and EW requirements with requests received from other networks or entities, as well as with infrastructure databases when available. The term ‘agent’ implies an entity that collects information and uses this information to make decisions, on behalf of a user.

MILKOG is aimed to serve as a system for further detailed studies within cognitive radio and cognitive EW, and it is expected that it will be expanded and changed many times during the course of the MILKOG activities in OPEK II and successor projects. The overall goal with the MILKOG architecture is to visualize benefits of co-locating and co-optimizing COM and EW functionality, and to have a prototype system for experimenting and verifying radio agent behavior.

The current report focuses on the architectural aspects of MILKOG, on documenting the prototyping of elements of this architecture, and on the prototyping of the multifunction (COM+ES+EA) platform functionalities. Future work will deal with the issue of the 'cognitive engine' of the agent and advanced optimization of agent spectrum decisions.

This report is organized as follows: Chapter 2 outlines the overall system architecture that the MILKOG agent aims to work in, and describes the elements of that architecture. Chapter 3 outlines the internal MILKOG agent architecture including its interfaces to the environment. Chapter 4 documents the development of a simplified MILKOG agent prototype, as well as the COM, ES and EA functionalities. Some initial tests of the prototype are described in Chapter 5. Chapter 6 provides a 'lessons learnt' summary from the MILKOG prototype development work. Chapter 7 provides suggestions for future MILKOG-related work to be carried out during OPEK III. Chapter 8 summarizes conclusions.

2 System Architecture

The purpose of this chapter is to outline the conceptual principles for the MILKOG architecture.

2.1 Infrastructure Perspective

Figure 2.1 and Figure 2.2 provide an overview of the different elements in the MILKOG architecture. The architecture includes the radio networks themselves ('green' units), where each transceiver is assumed to include a MILKOG agent. There is also an assumption that there are adversary networks present ('red' units), controlled by similar software agents, but with unknown functionality and strategies. Additionally, conventional non-agent controlled transceivers, electromagnetic monitoring units and jammers are assumed to be present.

The architecture also contains database elements, as will be explained in detail further below. The databases may be either in the form of replicated servers, or they may be distributed across the radio networks e.g. in the form of peer-to-peer databases.

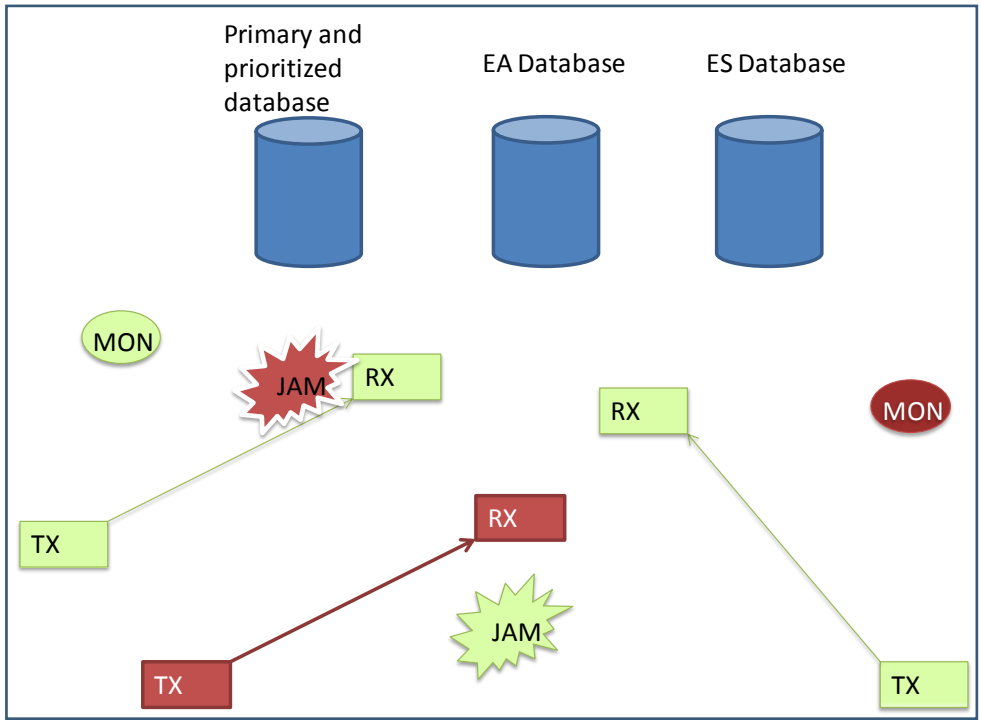


Figure 2.1 Overview of elements in the MILKOG architecture. 'Interfering simplex links' network model.

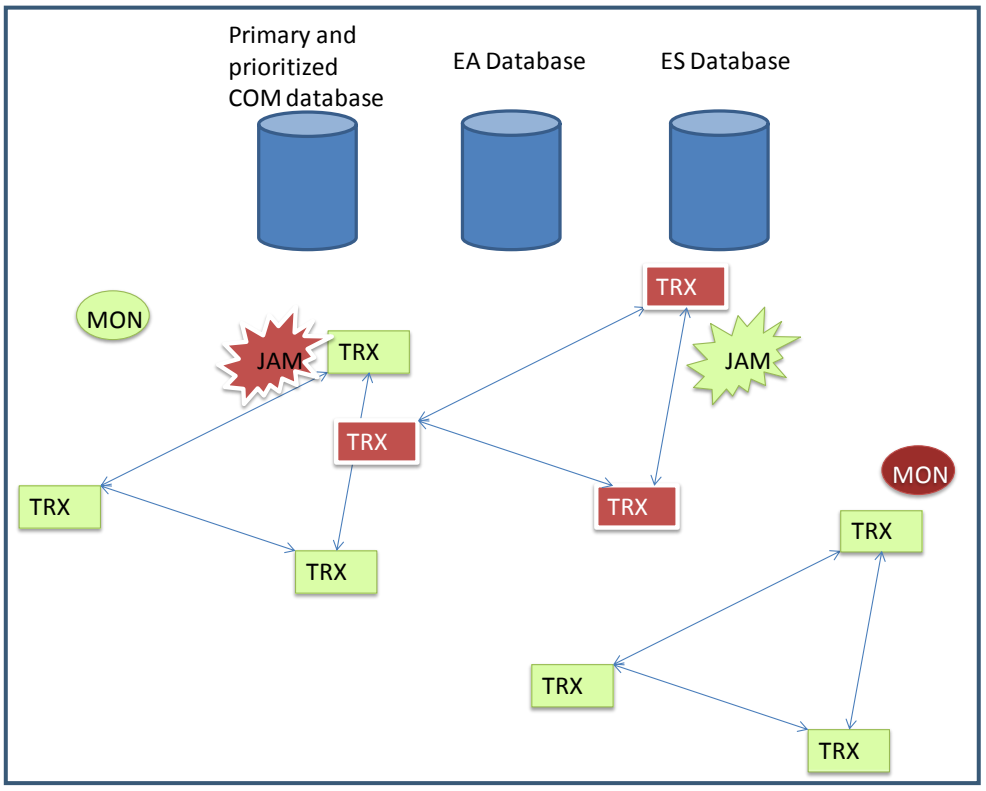


Figure 2.2 Overview of elements in the MILKOG architecture. Combat Net Radio Network (CNR) (all-hear-all) network model.

Both within radio networks, between radio networks as well as between networks and databases, there is supposed to be self-configuring coordination channels, as described in more detailed below.

2.2 Network Models

Two types of network models are taken into account. The simplest model is the *Interfering simplex links* one, which is the one sketched in Figure 2.1. Here, each one-way link between two transceivers is considered a physical network entity with individual waveform parameters and spectrum use. The links may be viewed as individual links in an ad hoc network, and where the links may dynamically access the medium in both the time and the frequency dimension, such as visualized in DARPA's *Wireless Network After Next* (WNAN) [3]. The second type is what is typically referred to as a *Combat Net Radio (CNR)* type of network, where several transceivers use the same single frequency interval (or the same hop-set), and where each transceiver broadcasts to the other transceivers in the network.

For the experimental implementation of MILKOG, see Chapter 4, the CNR type of networks are assumed, as those are closest to how military tactical networks operate today.

2.3 Spectrum Model

The sharing of spectrum resources may be realized along multiple dimensions, including time, frequency, code division, space, directive antenna angle, antenna polarity etc. In this report, and in the developed MILKOG prototype, we limit ourselves to discussing pure frequency division. We assume a spectrum model which is slotted in frequency, implying that the available spectrum band is divided into segments, see Figure 2.3. Here, B is the width of the spectrum band. The band is divided into M segments, each with a width of $\Delta f = B/M$. Defining f_0 as the center frequency of B, the center frequency of each segment k in B is

$$f_k = f_0 + ((k-1) - \frac{M-1}{2})\Delta f \tag{2.1}$$

where $k \in [1..M]$.

Each agent, or each CNR network, may use one or several segments.

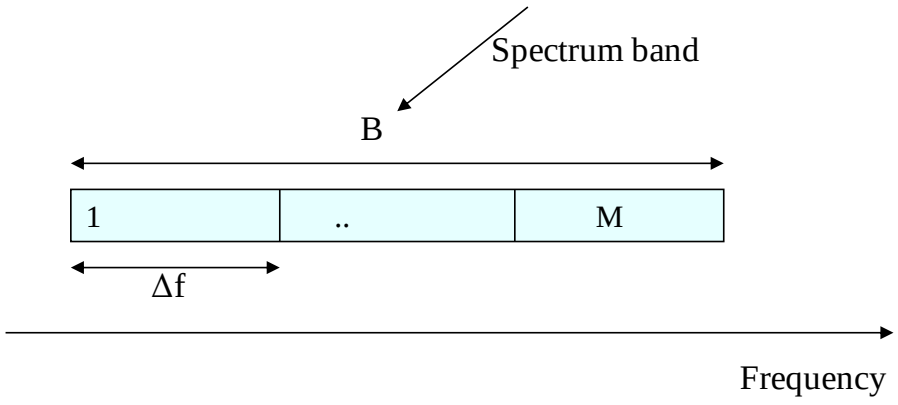


Figure 2.3 The segmented spectrum model that is assumed in this report

Methodology and mechanisms for optimized spectrum use and coordination between ES, EA and COM will be treated in OPEK III and documented in a separate report.

2.4 The Agent

The MILKOG agent is to be implemented as a software module that is co-located with and controls each MILKOG transceiver. The MILKOG agent manages the transceiver and controls the communication as well as the electronic surveillance and electronic attack properties.

The COM, ES and EA decisions made by the agent are to be based on a number of different inputs, both local and from the remote environment and infrastructure. The local stimuli include

- user commands and preferences
- communication traffic demands
- the sensed electromagnetic environment
- the transceiver limitations

The stimuli from within one's own network unit additionally include

- feedback of signal-to-noise and signal quality measures
- the sensed electromagnetic environment
- agreement on spectrum decisions

The stimuli from other networks include

- feedback on performance versus its target performance
- other optional feedback such as priority and interference measurements

The infrastructure information includes

- information about receivers that should be protected, and their priorities
- electronic attack targets
- electronic monitoring targets

2.5 Databases

The databases are infrastructure elements and provide a way of having accumulated information in the agent's environment, and that is accessible for all the agents. The interfacing to the databases is proposed to be implemented as a Web Service. For continuous coverage of large geographical areas, the databases should be implemented as replicated sets, where replication mechanisms copy location-relevant information between neighbor databases.

Not relying entirely on direct agent-to-agent communication but also including having accumulated information as part of the agent's environment, is inspired from [4]. Having infrastructure database elements is also in line with how cognitive reuse of the TV bands in the US have developed, where the FCC issued a request for companies willing to establish and run such spectrum databases.

Investigations on the databases and their communication interfaces are proposed as future MILKOG activities, see Chapter 7.

2.5.1 Primary and Prioritized COM Database (PPCD).

The Primary and Prioritized COM Database (PPCD) is to contain information about radio communication nodes, broadcasters and other forms of electromagnetic emitters, that do not have DSA abilities and whose emissions need to be protected from interference.

From the view of each MILKOG agent, the internal representation of information in PPCD is not important, but the agent needs to be able to obtain information about the query interface of the database. Two alternatives for the queried information are proposed:

- Alternative 1: The geographical area is divided into a cell structure with non-overlapping, equal-size cells. Each such cell has an address in a geographical addressing system. Upon requesting info from the PPCD, the MILKOG agent will receive information about maximum radiated power-density, for each spectrum segment within a specified interval. Obviously, there is a tradeoff between information accuracy (small cells) and complexity (large cells give a less complex data structure).
- Alternative 2: Same as alternative 1, the geographical area is divided into a cell structure with non-overlapping, equal-sized addressed cells. That is, in this case the MILKOG agent needs to inquire the tolerated interference level both in the cell in which it is located, and also in neighboring cells.

Alternative 1 is judged as the simplest approach.

2.5.2 EA Database (EAD).

The EA database (EAD) contains information about jamming targets. In its simplest form the return information from a query would be a specification of jamming power densities in specific spectrum segments. At a more advanced level, the return information could be characteristics of targets to be jammed, such as identities, types of networks or specific modulation features.

2.5.3 ES Database

The ES database (ESD) contains information about monitoring targets. In its simplest form the ESD specifies which frequencies are to be monitored at a given location. At a more advanced level, it could specify characteristics or identities of units to be monitored.

2.6 Coordination Communication

In order for agents to be able to coordinate decisions within and between radio networks, obviously channels for coordination communication need to exist. Also, between the agents in the radio networks and the infrastructure elements such as the various databases, coordination communication is needed.

The coordination communication is proposed to take place as high-layer communication, e.g. using an overlay addressing layer on top of Internet Protocol (IP) addressing. The data packets carrying the coordination information may then be physically routed depending on what physical connections are available in the specific time instant. This allows that the coordination communication is routed through the payload connections when these are active. Dedicated physical coordination channels are intended to be present as fallback when the payload connections are intermittently not available.

In the initial MILKOG prototype implementation, the coordination communication, for reasons of implementation simplicity, is using IP addressing and IP socket communication, and run through a wired Ethernet. This approach is realistic in the sense that it models the agents as truly separate entities, with all communication between them being explicit messages. It is partly idealistic though, since the probability that a coordination message gets lost is very small. For later tests outside of laboratory environment, we will include solutions for wireless coordination communication.

2.6.1 Coordination Within one Logical Network

For the one-way link type of network, this coordination is simply the feedback from the receiver back to the transmitter, such that the transmitter may make decisions based on its own link performance and sensed spectrum.

For the second logical network type considered, where multiple receivers and transmitters share a set of frequency segments, all transceiver nodes feedback their performance and sensed spectrum data to the other nodes. Further, coordination information that individual nodes have gathered from other networks is shared within the network. Lastly, the independent spectrum and jamming decisions made by the transceivers are fused into network-wide decisions.

The specific implementation of intra-network coordination for the MILKOG prototype is described in Section 4.3.5.

2.6.2 Coordination Between Different Logical Networks

This is coordination information between different networks within a coalition of such networks. The coordination comprises information about spectrum use, monitoring and jamming.

The specific implementation of inter-network coordination for the MILKOG prototype is described in Section 4.3.5.

3 MILKOG Agent, Internal Architecture

In this chapter, a conceptual description of each the modules in the MILKOG agent architecture is provided. Further details on the specific prototype implementation are provided in Chapter 4.

3.1 Overview

An overview of the internal architecture of the MILKOG agent is shown in Figure 3.1.

The functionality inside the dashed line is considered the internal agent functionality. The agent interfaces with the radio user through the *Graphical User Interface* (GUI) functionality. Further it interfaces with other agents and the infrastructure through the *coordination channel* interface. It acquires information on the observed spectrum through the *senser* interface, and manages the transceiver and receives *Quality-of-Service* (QoS) information through the *transceiver* interface.

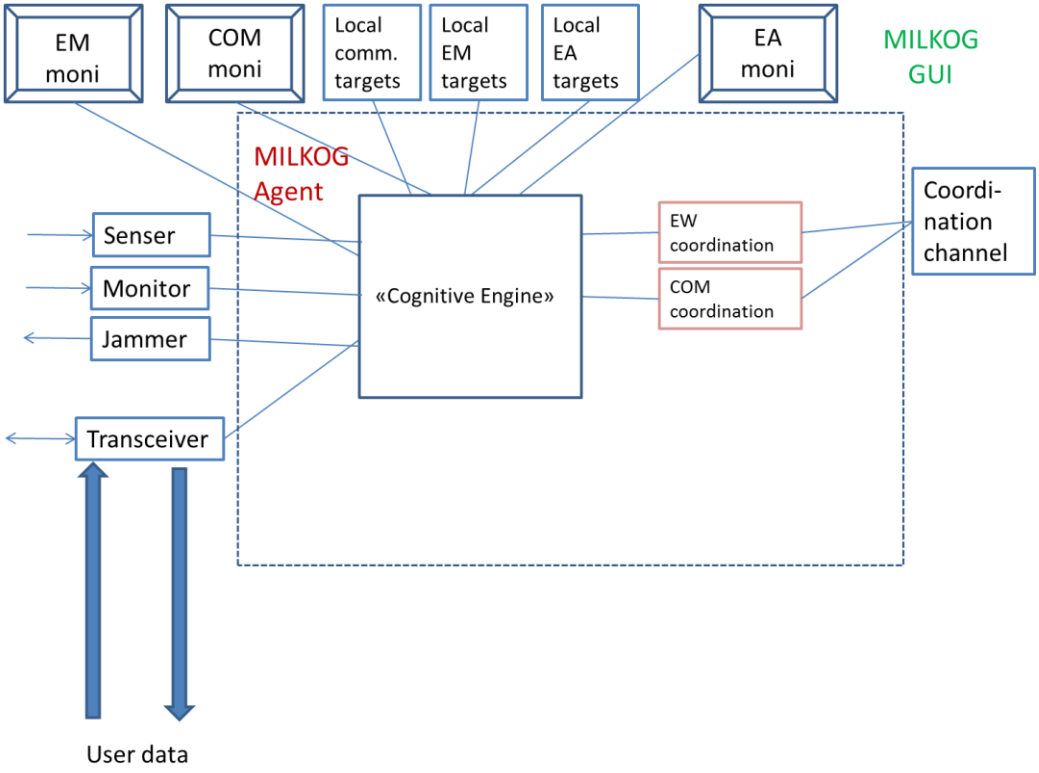


Figure 3.1 Overview of the MILKOG agent internal architecture.

Each block is described in further detail, below.

3.2 MILKOG Agent Blocks

3.2.1 Cognitive Engine

The MILKOG "cognitive engine" is the core of the agent, collecting information, doing the reasoning and optimization, and making spectrum decisions. The goal of the cognitive engine is to arrive at close-to-optimal solutions for transceiver power and operating modes in each of the used parts of the spectrum, subject to transceiver limitations, user-input restrictions and performance goals, and global restrictions and performance goals.

To calculate the transceiver power solutions, the cognitive engine may include various types of optimizers, e.g. ones based on classical optimization, and ones based on genetic algorithms.

The cognitive engine is assumed to include persistence elements such that it may aggregate observations and decisions, and experience related to past decisions.

Cognitive engine architecture is a vast subject in itself, and will be deferred to OPEK III. For the prototype described in Chapter 4, the cognitive engine block has been realized as a continuously operating reactive loop that evaluates bit error rates, senses the spectrum, suggests new spectrum decisions, and fuses these decisions with other agents in the same network.

3.3 GUI

This section outlines the principles for the agent GUI. The reader might want to have a side look at the prototype GUI in Chapter 4.

3.3.1 Local Communication Goals

This is a local GUI where the user is allowed to input her communication preferences. The preferences may be in the form of target frequency intervals, or in the form of a specific minimum bit rate that the MILKOG agent should target.

3.3.2 Local ES Targets

This is the local interface for inputting ES target preferences from the operator. This may be in the form of specific frequency or location targets for surveillance, or in the form of rules that describe which type of surveillance activity that should take place.

3.3.3 Local EA Targets

This is the local interface for inputting information on which radio nodes, networks, frequencies, operating modes and locations that should be subject to electronic attacks, and which forms of electronic attacks that are preferred.

3.3.4 COM Monitor

The COM monitor presents an overview of which MILKOG nodes this MILKOG node is connected to, along with performance and QoS information, and optionally sense data.

3.3.5 ES Monitor

The ES monitor displays surveillance information to the user. Based on the user's preferences, this may be only local EM information, fused information within own network, or globally fused information for a chosen area.

3.3.6 EA Monitor

The EA monitor displays information on currently active EA tasks. To the extent possible, it should display the effectiveness of EA tasks.

3.4 Interfaces

3.4.1 Coordination Channel Interface

The coordination channel interface handles the receiving and sending of coordination messages. The messages may be to other agents within own network, to agents in nearby networks that the agent needs to coordinate with, or to the database infrastructure elements.

If the normal payload physical communication is up and running, the coordination interface may choose to forward the coordination communication through these payload channels. If the payload physical channel is not available, the coordination communication will be forwarded on control channels.

The two modules in the SW agent that handles coordination messages are:

- The EW policies/coordination module takes care of messages related to locally identified EA targets that are to be coordinated with other nodes, and receives information about targets from other nodes or from the EA database. It also prepares ES data to be sent for fusion at other nodes or in a fusion database, and receives ES data from other nodes or fused data from a database.
- The COM coordination module deals with the formation of local networks and COM spectrum coordination with other agents. It also downloads aggregated spectrum information from a COM spectrum database (information on "which frequencies or receivers need to be protected in a certain area", see 2.5.1). Optionally it may upload spectrum decisions to the database.

3.4.2 Transceiver Interface

This is the interface to the COM transceiver unit controlled by each MILKOG agent.

3.4.3 The Sensing Interface

This is the interface to the sensing part of the radio node. The MILKOG agent will through this interface send sensing parameters, and receive corresponding sensing values. The sensing functionality may be implemented on the same hardware platform as COM, or on complementary hardware. The prototype implementation is described in Section 4.6.

3.4.4 The Monitoring Interface

This is the interface to the monitoring part of the radio node. The MILKOG agent will through this interface send monitoring parameters, and receive corresponding monitoring data values. The monitoring functionality may be implemented on the same hardware platform as COM, or on complementary hardware. The prototype implementation uses the COM platform, as described in Section 4.6.

3.4.5 Jammer Interface

This is the interface to the jammer part of the radio node. The MILKOG agent will through this interface send commands to the jammer functionality. The jammer may be implemented on the same transceiver hardware as COM, or on complementary transceiver hardware. The prototype implementation is described in Section 4.8.

4 Prototype Implementation

4.1 Introduction

A reduced-functionality prototype MILKOG system has been implemented to satisfy the project requirement to 'design and develop multifunction radio units' and for the purposes of experimentation and verification of system principles and optimization algorithms. It is expected that it will be a living prototype that will be gradually expanded during the course of the OPEK II and OPEK III projects.

The implementation of a full-functionality MILKOG prototype is a time-consuming task in terms of man-hours of programming. It is complex in terms of implementation details that need to be attended to in a system that should operate with real RF waveforms and data transmissions, and with fully distributed spectrum decision processes. In order to reach a working prototype within the timeframe of OPEK II, not all of the MILKOG functionality, as described in Chapter 3, has been implemented in the prototype. The prototype has basic functionality (as detailed in 4.2 through 4.9) that includes

- Transmission of real user data using an OFDM waveform
- Monitoring of data bit error rates for the purpose as input to the spectrum decision algorithms
- User-initiated formation of logical networks through control channel messages
- Basic spectrum sensing and electronic monitoring functionality (limited to "energy-sensing")
- A basic agent decision entity (using preprogrammed reactive strategies)
- The combination of the electronic monitoring functionality and jammer functionality to form a reactive jammer

Referring to Chapter 3, in particular the following features of the MILKOG architecture have not been implemented in the OPEK II MILKOG prototype:

- The central COM and EW databases and the interfaces to these databases
- Wireless control channels
- Advanced spectrum optimization algorithms
- Cognition features like advanced awareness, learning, planning and reasoning

Several of the topics above will be addressed in the OPEK III MILKOG activities, as described in Chapter 7.

4.2 Prototype Overview

The prototype consists of the following modules (see Figure 4.1):

- The MILKOG software agent (residing on a PC)
- The MILKOG traffic forwarder (co-located on the same PC as the software agent)
- The COM waveform running on a Lyrtech SFF SDR platform (See Section 4.5)
- Sense and ES functionality ("energy sensing"), also implemented on a Lyrtech SFF
- EA functionality implemented on the Lyrtech SFF (the MILKOG software agent is also prepared for using a USRP as EA unit, as a possible extension)
- The control channel (using wired Ethernet)
- The traffic source and sink, that uses the open-source freely available VLC software [5]. On the prototype, the traffic source and sink runs on the same computer as does the MILKOG software agent.

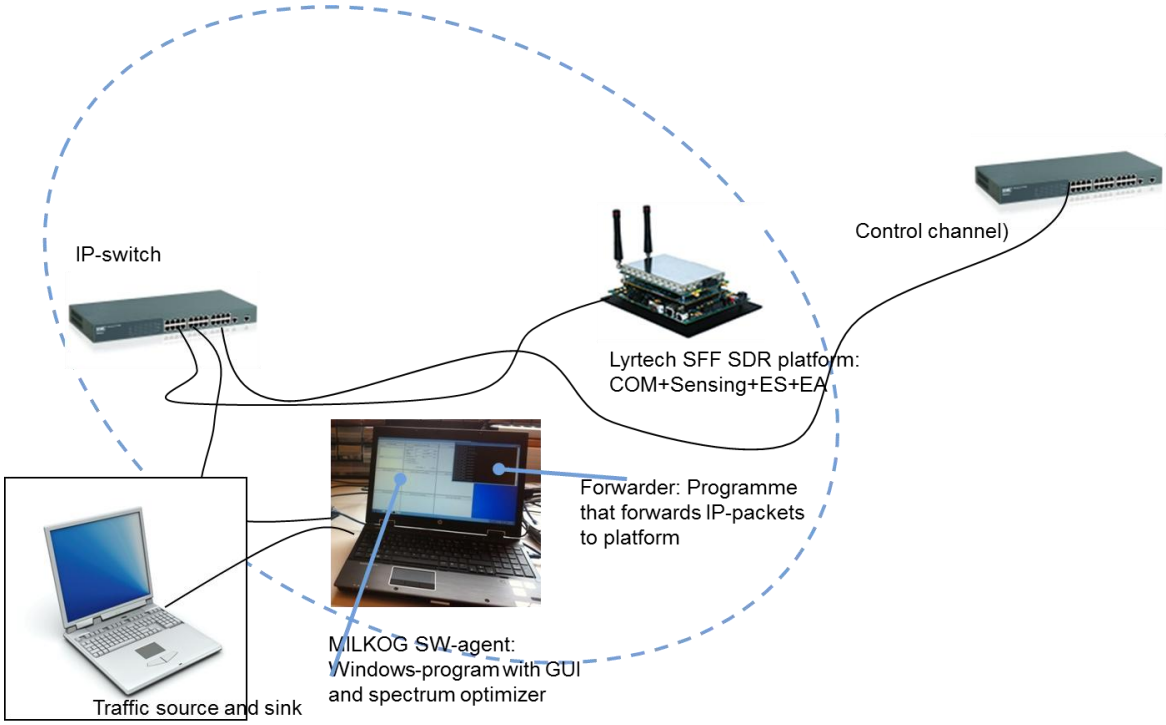


Figure 4.1 The prototype MILKOG radio node

4.3 The MILKOG Software Agent

The prototype MILKOG software agent has been developed in Microsoft Visual C++ for Windows.

4.3.1 The Internal Architecture of the MILKOG Agent

In order to handle the user interaction and various inter-agent and agent-to-platform communications concurrently with the COM-ES-EA decision making in a close-to-real time manner, the SW agent has been implemented as a multi-threaded[1] structure. Figure 4.2 illustrates the thread structure in the agent, where rectangles illustrate threads and the rectangles with rounded corners illustrate internal queue structures.

The MILKOG functionality is split between the following concurrent threads:

- The Main & Windows GUI thread, that creates all the other threads and runs the *Graphical User Interface* (GUI). If activity within the GUI results in coordination messages that needs to be sent to other agents, these messages are put on the SendQueue.
- SendThread, that reads messages from SendQueue and sends them out on the coordination interface.
- ReceiveThread, that receives messages from the coordination interface and puts them on ReceiveQueue.
- ReceiveFromForwarderThread, that receives bit error rate and bit rate measurements from the Forwarder, and puts these on the receive queue for the DSA thread to handle.
- The DSA thread, that interprets received messages, sense data and error rate data, and makes spectrum decisions.

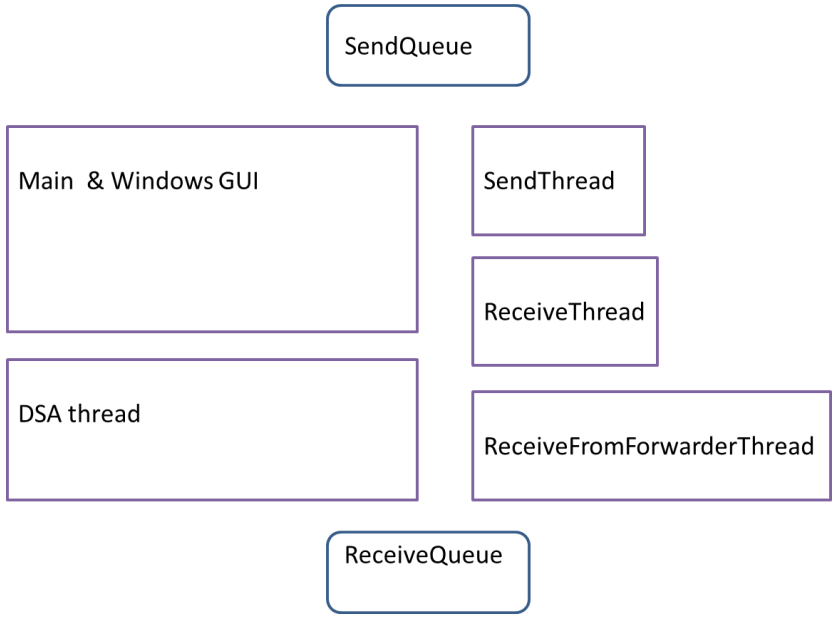


Figure 4.2 The thread structure of the MILKOG agent.

[1] A thread is the smallest unit of processing that can be scheduled by an operating system.

4.3.2 MILKOG GUI

The MILKOG GUI has been developed using the standard *win32* API. A screenshot of the main menu is shown in Figure 4.3. By selecting 'View', the various submenu choices appear. The ones that are activated in the current prototype, are:

- MILKOG GUI
- Configuration Settings
- Radio Network Configuration

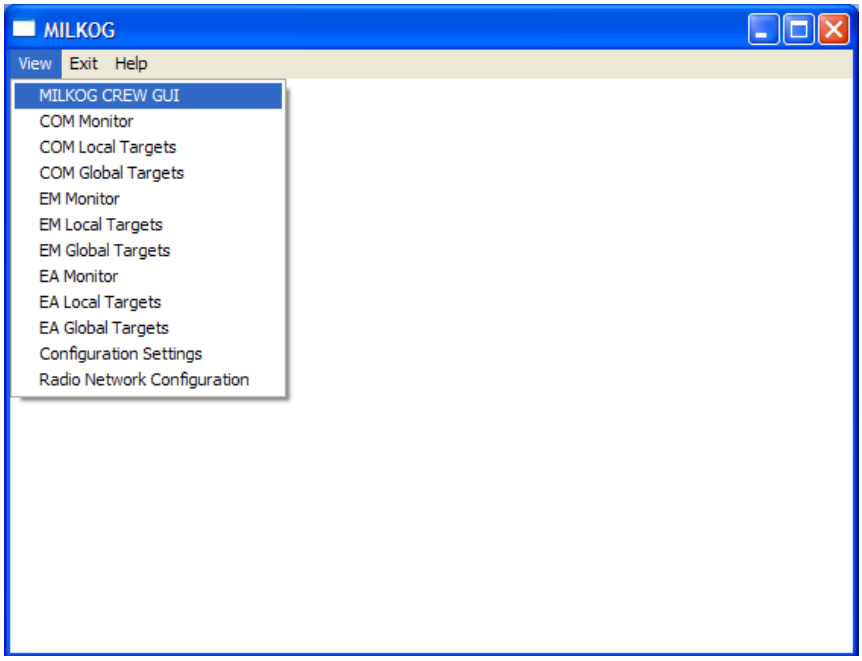


Figure 4.3 The main menu of the MILKOG agent.

4.3.2.1 Configuration Settings

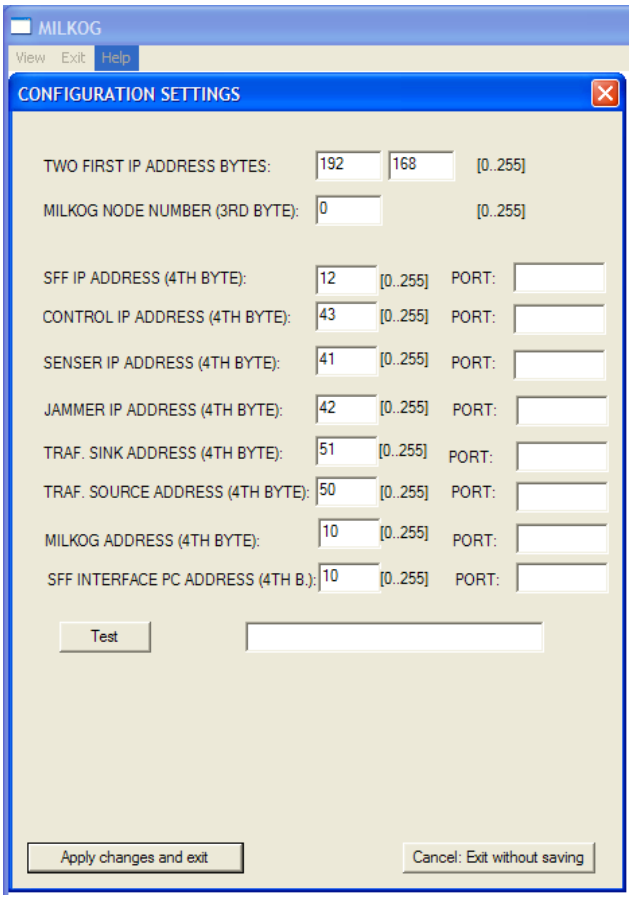


Figure 4.4 The Configuration Settings menu.

As explained previously, the MILKOG nodes are comprised of one or more software radio platforms and one or more computing platforms, connected using an ordinary Gigabit Ethernet switch. The 'Configuration Settings' menu allow to set the individual IP addresses of the platforms, see Figure 4.4. The two first IPv4 address bytes are fixed at '192.168'. The third address byte is the MILKOG node number (which imposes a restriction on the number of nodes in the experimental system). The fourth byte is used to address the various computers and SDR platforms, as seen in Figure 4.4. The port number input is currently not used (the port numbers are preset in the system).

4.3.2.2 Radio Network Configuration Settings GUI

The Radio Network Configuration Settings menu allows the user to inform MILKOG which nodes it wants to communicate with, and that should form a network. The user first selects the 'Detect available MILKOG nodes'. This creates PING messages to the range of other MILKOG units to discover which ones are present. The PING messages are added to SendQueue, for the Send thread to handle.

The nodes that receive PING messages send responses ("PONG"), which are used to build a list of available MILKOG nodes. The user is allowed to select nodes to connect to, then presses 'Try connecting to selected nodes', which sends out JOIN messages to the selected nodes. This action

only needs to be carried out from one of the nodes in the network. A full explanation of the messages associated with radio network configuration is documented in Appendix I.

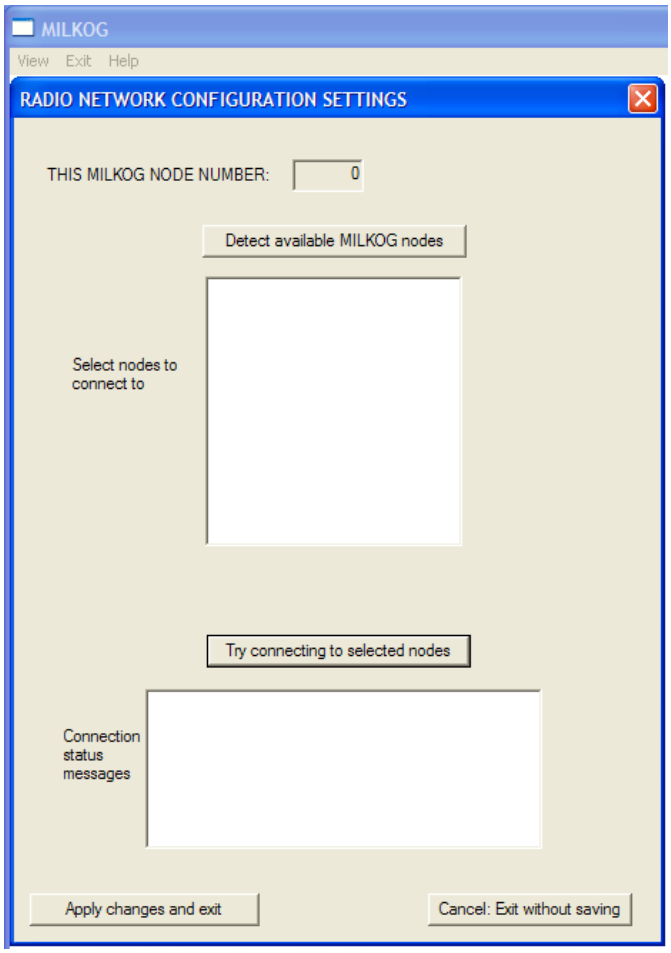


Figure 4.5 The Radio Network Configuration menu.

4.3.2.3 MILKOG GUI

The MILKOG GUI is the main dialog for the combined optimization of COM, ES and EA.

The dialogue is organized in a 3 by 3 blocks fashion. The left column displays status information to the user. The middle column is used to input local restrictions from the user to the DSA optimization functionality (or to the 'cognitive engine'). The right-hand column is supposed to show global restrictions, downloaded from infrastructure databases, but in this version the right-hand column is unused.

Row-by-row, the upper row contains the COM information and restrictions. The middle row contains ES information and restrictions, and the lower row the EA information and restrictions.

It must be stressed that the MILKOG GUI should be considered as a draft one. Its main purpose is that of being an engineering tool for experimentation with the MILKOG system. However, we hope that the draft GUI also can serve to trigger discussions with military operations personnel as to requirements for practical such GUIs.

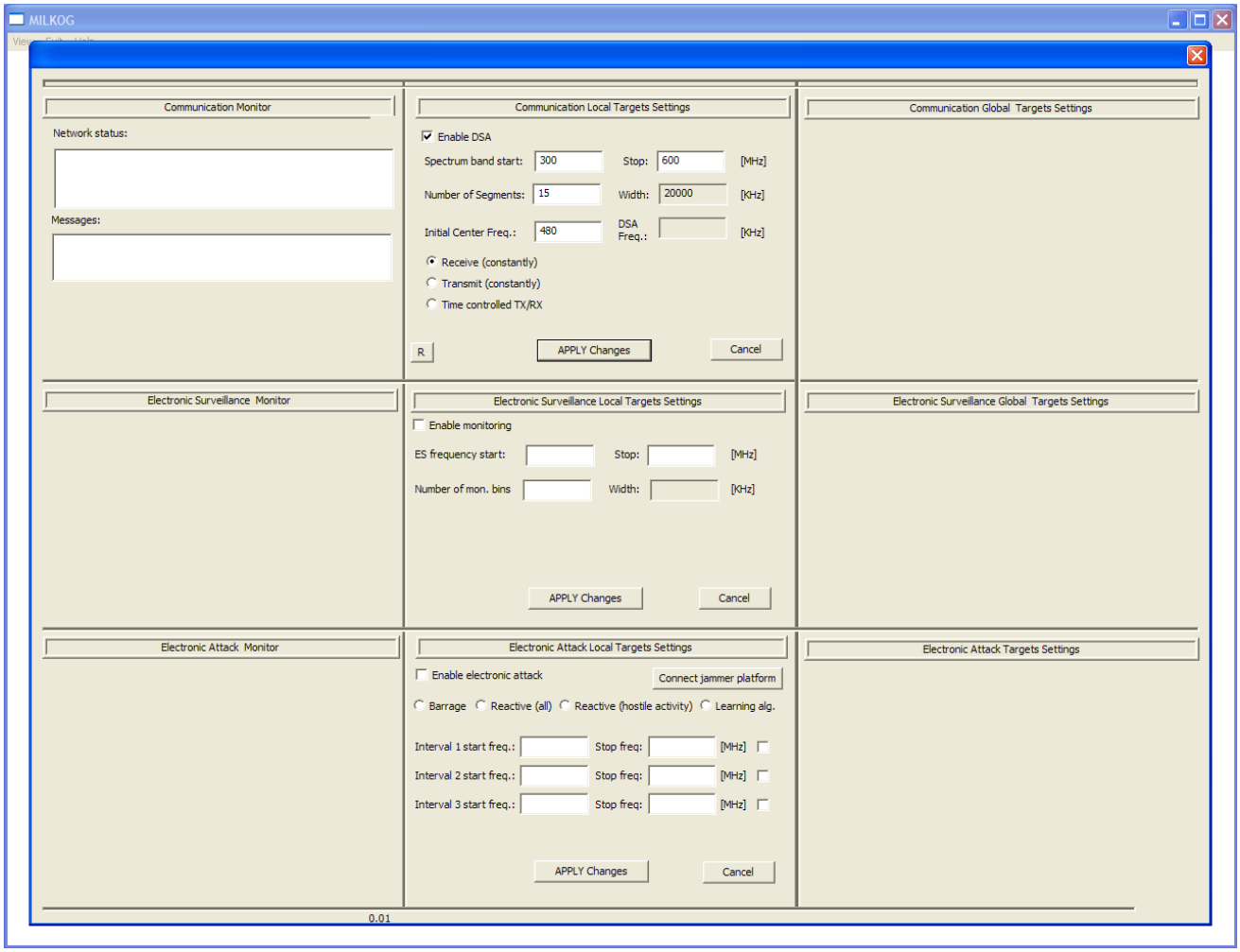


Figure 4.6 The combined COM-EW menu of the MILKOG agent.

4.3.3 The Send, Receive and ReceiveFromForwarder Threads

The Send thread monitors SendQueue. For each new message that appears in SendQueue, the Send thread sends the message to the recipient, as UDP (i.e. best-effort) packets.

The Receive thread receives UDP messages on the SW agent's monitoring port. It puts the messages on the ReceiveQueue, for the DSA thread to handle.

The ReceiveFromForwarder thread is responsible for receiving bit error rate and bit rate data as UDP messages from the Forwarder program, and adding these to the ReceiveQueue.

An overview of all the defined messages is provided in Table 4.1 in Section 4.3.5.3.

4.3.4 The DSA Thread

The DSA thread runs the core functionality of the current version of the MILKOG SW. The DSA thread interprets all incoming information, maintains the internal data representations, sends new spectrum proposals to other agents and handles fusion of arrived spectrum proposals.

In the current version, only COM spectrum proposals and decisions are made (not fused COM-EW decisions), and only a very simple spectrum decision algorithm has been implemented. It is

planned to expand this functionality into demonstrating a much more advanced COM-EA-ES decision functionality, as explained in Chapter 7.

The DSA thread continuously interprets new information (e.g. sense data), puts the information into internal structures, based on the current existing information makes new decision proposals and contributes in finding the fused decision among the agents. In this way the DSA thread implements a purely reactive architecture, where new information trigger new decisions.

The functionality in the DSA thread also handles the medium access (MAC) mechanism, as explained below.

4.3.4.1 Interpretation of New Information

All new information to the DSA core, which may range from information about discovered nodes to bit rates, sense data and monitoring data, all arrive through the ReceiveQueue. As explained previously, the reasoning behind this is that the DSA thread should be operating continuously, and without the risk of being stuck polling the various interfaces to platforms and other agents.

The DSA thread runs a continuous loop where it constantly reads the messages from ReceiveQueue, and puts them into internal information structures according to message type. In the prototype, new information merely replaces older information, there are no statistical aggregation of information or learning mechanisms implemented.

4.3.4.2 Internal Data Representation in the Agent

Network:

Each agent builds up her internal representation of the local network that she belongs to. The network representation is in the form of a doubly linked list structure termed LOCALNET.

There are three cases that lead to updating of the LOCALNET structure:

- The DSA thread receives a JOIN message from another agent
- The DSA thread receives a JACK message from another agent, as an acknowledgment of a JOIN
- The DSA thread receives an UPDN message from another agent
- An UPDN is sent to all nodes in the network, each time a change is made to LOCALNET. This is to make sure all the nodes in LOCALNET have the same information about the structure of the network.

Refer again to Table 4.1 and Appendix I.

Sensing and Monitoring Data:

As described previously, the spectrum model assumes that the available spectrum is divided into a number of segments. The sensing data is referenced to each such segment.

The monitoring in the same way uses a monitoring spectrum bin size. The monitoring bin size may be of a different size than the COM segment.

The sense data for the individual node is stored in the structure *segmentLocalTxReferencedNoiseAndInterference.* The lowest signal-to-sense level for the network of nodes, for each segment, is to be stored in the structure *segmentWorstCaseTxReferencedNoiseAndInterference* (not fully implemented).

In the same manner, local node monitoring data is stored in the data structure *ESnodeNoiseAndInterference*. The highest monitoring level for the network of nodes, for each segment, is to be stored in the data structure *ESnetworkNoiseAndInterference* (not fully implemented).

Quality-Of-Service:

MILKOG monitors bit error rate and data rate for the communications channel. The bit error rate is measured by sending known data. The known bit error rate test data is always sent when the unit is in TX mode, and there is not any payload (e.g. video) data available to be sent. Also, 20 packets of known data is always sent prior to a sequence of video data, and after a sequence of video data.

4.3.4.3 DSA proposals

The DSA thread implements a state machine for the spectrum decisions, refer to Figure 4.7. The following is a description of the different states in the state machine:

- IDLE: This is a waiting state, where the DSA thread waits for other agents to send a spectrum proposal, or it waits for a need to initiate a proposal, based on evaluation of QoS data and sense data. If a spectrum proposal is received, the next state is PROP RECEIVED. If instead a need to initiate a proposal is identified, the next state is INITIATE PROP.
- INITIATE PROP: When this state is entered, a need for a spectrum change has been identified. The DSA thread in this state uses available information to calculate a new spectrum proposal, and adds this spectrum proposal to SendQueue, then sets a timer. The next state is RECEIVE FURTHER PROPS.
- PROP RECEIVED: When this state is entered, through a received proposal from another node it has been indicated that there is a need for a spectrum change. The DSA thread in this state uses available information to calculate a new spectrum proposal, and adds this spectrum proposal to SendQueue, then sets a timer. The next state is RECEIVE FURTHER PROPS.

- RECEIVE FURTHER PROPS: In this state the DSA thread accumulates proposals from other agents until timer expires (in the prototype: 200 milliseconds).
- ACT: This is the state where individual decisions are evaluated, to arrive at a common decision for the network. After implementing the decision, the next state is IDLE.

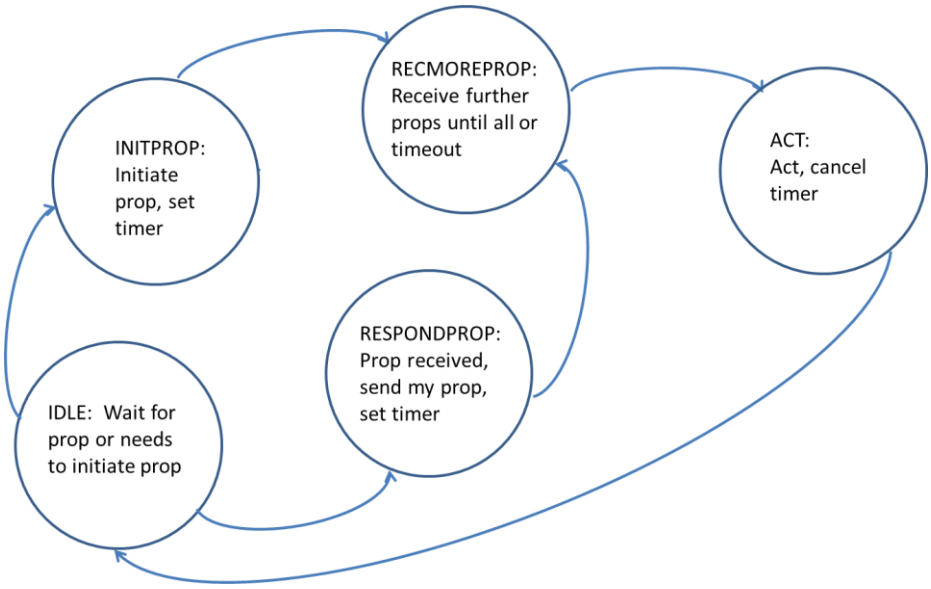


Figure 4.7 State diagram for spectrum decisions

The state diagram naturally triggers more questions, like:

- When is there a need to initiate a new PROP?
- How to calculate / determine new PROPs?
- How are individual PROPs evaluated, in order to arrive at a common fused decision?

These questions are at the core of the MILKOG functionality, with multiple choices being possible. As outlined in Chapter 7, these are questions that will be subject to future research. For the current prototype, a simple set of assumptions and implementations have been made:

When to send a new PROP: There are two conditions that both need to be met in order to switch to the INITIATE PROP state. The first condition is a minimum time since the last spectrum decision. This minimum time is a stochastic parameter

$$T_{DSA} = T_{DSAmin} + rand(0,1) * T_{DSAmin} \quad (4.1)$$

T_{DSAmin} is a settable parameter (value used in the tests in Chapter 5: 6 seconds). The second condition is based on *Bit Error Rate* (BER): The agent receives a BER measurement each second. Let BER(0) denote the last measured BER value that the agent has received, and BER(1) the previous value, and so on. Then

$$lowBER = \min_{i=0..4} BER(i) \quad (4.2)$$

The second condition is then that lowBER > minBER (value used in the tests in Chapter 5: 3.5%)

Calculation of a new PROP: This is based on the agent's local sense data. The agent proposes to switch to the spectrum segment that has the lowest sense noise level, of all spectrum segments between the spectrum sense start frequency and the sense stop frequency.

Common decision: The decision chosen is based on each individual agent's evaluation according to the diagram in Figure 4.8. There are no confirmations returned. In the unlikely event of a wrong decision having been made, this merely results in a new INITIATE PROP.

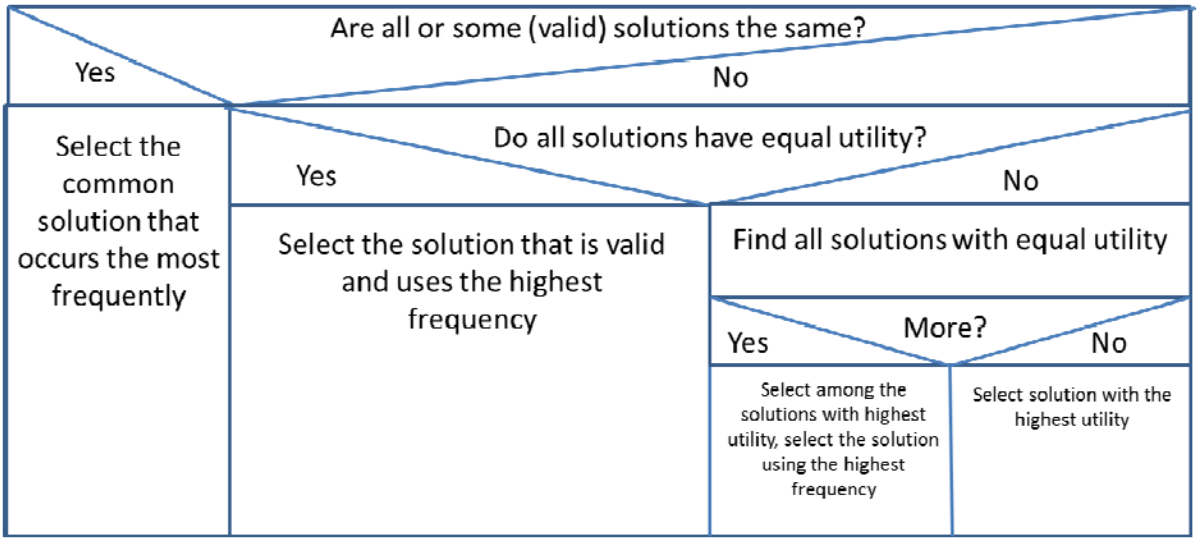


Figure 4.8 The criteria for selecting which of the proposed spectrum decisions to use

4.3.5 Coordination Between Agents

4.3.5.1 Out-of-band Coordination Signaling

By out-of-band coordination signaling we mean signaling that does not utilize the payload waveform.

Out-of-band coordination signaling has been implemented as IP datagrams (UDP). In the very first prototype, the out-of-band physical data carrier is wired Ethernet. This is to be replaced with wireless physical layer connections at a later stage.

As explained previously, coordination messages are added to SendQueue, which is then read by SendThread and sent via UDP to the recipient agent. Here, the message is received by ReceiveThread of that agent and added to ReceiveQueue for handling by the information processing and decision making core of the agent.

4.3.5.2 In-band Coordination Signaling

In-band coordination signaling refers to signaling that utilizes the payload waveform.

Preparations have been made for using in-band signaling for the feedback of bit error rates from radio nodes within a common network, but this has not been activated in the prototype.

4.3.5.3 Local Parameters Synchronization

Whenever a change of local parameters settings is done at one of the agents in a logical network, the new parameter settings are communicated to the other agents in the network. The messages are COMP (for local communication parameters) , EAPA (local electronic attack parameters) and ESPA (local electronic surveillance parameters), see Table 4.1.

For the full details on the message formats, please refer to Appendix I.

Table 4.1 An overview of the implemented administrative command messages in the MILKOG system. The complete format of each message may be found in Appendix I.

Local parameters exchange messages:	
COMP	Local COM parameters (sent to other nodes in network when changes made)
EAPA	Local EA parameters (sent to other nodes in network when changes made)
ESPA	Local ES parameters (sent to other nodes in network when changes made)
Network association messages:	
PING	Discovery type message. Sent to all node numbers.
PONG	The response to a received PING message.
JOIN	Request from sender node to receiver node to join his logical network.
JACK	Response to JOIN, an accepted JOIN request
UPDN	Network update notification sent to all nodes in the logical network, following any change of the logical network (this is to make sure that all nodes in the logical network have knowledge about all the other nodes).
DISC	Disconnect message sender node from network (reserved, not implemented in the prototype)
Time synchronization messages:	
SYNC	Sends the node's current time (to other nodes in the current network)
Spectrum proposal messages:	
PROP	Sends a spectrum proposal (to other nodes in the current network)

4.3.5.4 Time Synchronization

Due to not having absolute time references (e.g. GPS clocks) available in the prototype agent, clock synchronization between agents in each LOCALNET has been implemented. Each time an update has been done to the LOCALNET, a SYNC command is sent to all nodes in the network. The SYNC command includes a time-stamp, which is compared to the clock in the receiving agent. The transmission delay of the SYNC messages is neglected. The clock of the highest-numbered node in the LOCALNET is adopted as a common time for all the nodes in the network, and correction factors are calculated in each agent.

The purpose of synchronizing the time between the agents in this manner is to enable the clock-controlled MAC.

4.3.5.5 Performance Feedback Within a Network

Each agent maintains a BER monitoring for the received signal, which is a moving average BER where the BER of each new received packet contributes by 10% relative to the accumulated BER. In the same manner, it maintains a moving average Bit-Rate monitoring. BER data and Bit-Rate data is sent from the Forwarder to the MILKOG agent SW (in the prototype each second), by using the defined BERM and BRAT messages, see Table 4.1.

In order for the transmitter end of each one-way link to know the quality of the link, the last recorded BER data needs to be fed back. The BER data is to be fed back by in-band signaling, however this is not yet implemented in the prototype.

4.3.5.6 Spectrum Sensing Feedback Within a Network

Each agent may make better decisions if, in addition to its own observations, it is informed about the observations made by the other agents in the same network. In terms of the exchanging of and cooperation on gathering sensing data, this is referred to as cooperative sensing in the literature. Such cooperative sensing is deferred to future work.

4.3.5.7 Spectrum Proposals

Spectrum proposals are made as described in 4.3.4.3. The spectrum proposals are sent to the other nodes within a network using the PROP message, see Table 4.1.

4.4 The MILKOG Forwarder

The MILKOG Forwarder is a separate program that is targeted to do fast forwarding of data to/from the SDR platform to/from other node entities. It is implemented as a Visual Studio c++ multi-threaded Windows console program.

The Forwarder contains the following functionality:

- The UDP reception of data from a data source (e.g. a video server), and the reformatting and TCP/IP socket transmission of this data to the SFF SDR platform. The reformatting includes putting redundancy in the data stream, such that each block of video data is actually sent 3 times. This is in order to compensate somewhat for the lack of error correction in the communication waveform.
- The TCP/IP forwarding of commands from the MILKOG agent to the SFF platform.
- The reception of data from the SFF SDR platform, including block synchronization of the data stream. For BER test data, BER is calculated and sent to the MILKOG agent each second. For payload data (e.g. video), the data is reformatted and sent to the data sink (e.g. to the video client). Since the payload data contains each video block occurring three times, the Forwarder in the reformatting process selects which data bytes to keep.

4.5 The COM Functionality

The COM functionality is implemented on a Small Form Factor (SFF) SDR platform, purchased from Lyrtech Inc. The SDR platform consists of three major modules: Digital processing module, Data conversion module and a RF module. A simplified block diagram of the SFF SDR platform is presented in Figure 4.9.

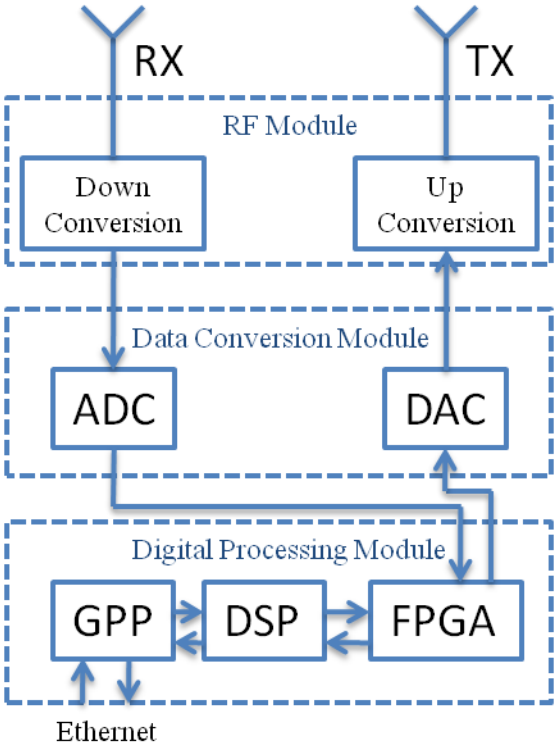


Figure 4.9 Simplified block diagram of SFF SDR

The Digital processing module consists of one Texas Instruments System on Chip (SoC) and a Field-programmable Gate Array (FPGA) from Xilinx. The SoC includes one General Purpose Processor (GPP) and one Digital Signal Processor (DSP). On the purchased SFF SDR platform the GPP runs with a fixed Ethernet communication software that moves data from / to the Ethernet port to/from the DSP. Communication between the SDR platform and a host PC is achieved via FTP, through the Ethernet connection. The Data conversion module converts the digital processed signal in the FPGA to and from an analogue representation. These analogue signals are converted to and from RF by the RF module, with a RF range of 250MHz to 1GHz. For more information about the SFF SDR see appendix IV.

The following software was used to program the SFF SDRs DSP and FPGA.

DSP development software:

- Code Composer studio v.3.3
- Real Time Workshop

FPGA development software:

- MATLAB R2008a
- Xilinx ISE Design Suite 10.1.03 IP update 3
- Xilinx System Generator for DSP 10.1.3.1386.

Code Composer Studio is an integrated development environment for Texas Instruments embedded processor family, which includes C/C++ compilers. The SFF SDR is programmed by C-code and the compiled *.out* file is stored in an onboard flash memory. The DSP may also be programmed by Simulink, via Real Time Workshop. Real Time Workshop is a Simulink coder that generates and executes C and C++ code from Simulink diagrams, state-flow charts and MATLAB functions.

Xilinx System Generator is a Digital Signal Processing tool for Xilinx ISE Design Suite, enabling the use of MATLAB for designing Xilinx FPGAs, via specific Xilinx blocksets integrated in Simulink. The high-level programming in Simulink makes the implementation of complex hardware designs an easy task for designers, and Xilinx System Generator compiles the completed design to a *.bit* FPGA configuration file. This configuration file is stored in the same onboard flash memory as the *.out* DSP file. The GPP came with a default *streamGPP.iae* configuration file, stored in the same flash memory, that streams data from/to the Ethernet connection to/from the DSP. When the SFF SDR platform reboots, the GPP, DSP and FPGA are loaded with their respectively configuration files, from the onboard flash memory.

4.5.1 PHY

For the COM waveform in MILKOG, our goal has been to adopt a waveform that has the potential to conform to the spectrum model described in Section 2.3. This implies that the waveform should allow flexibility in using a set of frequency segments $\Delta \in \{1..M\}$. To adapt to different SNR levels, it additionally should be possible to use different modulation and/or coding in each of the frequency segments.

One way of approaching the above goals is to use several, individually configured, narrowband waveforms running in parallel. Another way, and one which is frequently seen in cognitive radio test-bed implementations, is *Non-Contiguous Orthogonal Frequency Division Multiplexing* (NC-OFDM), which allows a set of sub-channels within an OFDM waveform to be deactivated. With the popularity of OFDM in current wireless systems as well as in cognitive radio research, and since Lyrtech had a base example waveform available at a moderate cost of $1000, it was decided to proceed with OFDM.

The OFDM waveform obtained from Lyrtech is mainly defined as functionality to be deployed in the SFF SDRs FPGA. The design has configurable parameters, enabling the possibility to generate new FPGA bit files with different parameter sets (such as different modulation, different number of sub-channels, different sets of nulled sub-channels etc.).

Some minor modifications have been done to the waveform, but the main structure is kept the way it was delivered from Lyrtech. The design has the potential to be turned into a real-time

configured NC-OFDM one, but due to the amount of work involved it is currently used with a static number of sub-channels activated. This subsection is based on the documentation supplied with the Lyrtech example OFDM waveform [6].

4.5.1.1 Description of the Base OFDM Waveform

OFDM is a multicarrier modulation or multiplexing technique that takes a high rate data stream and divides it into N parallel low rate data streams, and sends them simultaneously over N sub-channels. A block diagram of the COM transceiver architecture is shown in Figure 4.10.

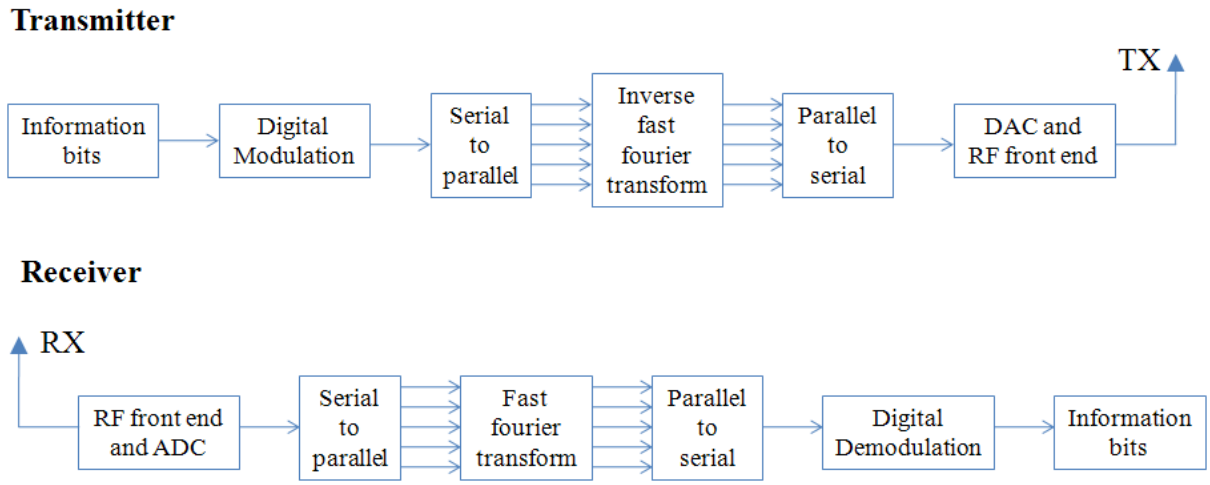


Figure 4.10 Transceiver block diagram

The information bits are modulated by a *Quadrature Amplitude Modulation* (QAM), according to the constellation diagram shown in Figure 4.11.

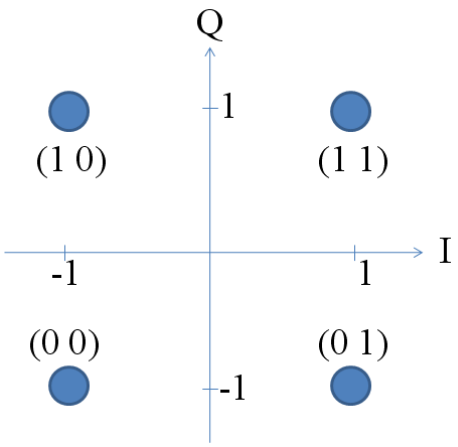


Figure 4.11 4-QAM Constellation Diagram with Gray Coding.

The high rate QAM-symbols, of rate R_s symbols per second, are subdivided into N parallel data streams, each with a rate of R_s /N. An N-point *Inverse Fast Fourier Transform* (IFFT) centers each low rate QAM-symbol on a sub-channel $k\Delta f$ relative to the carrier frequency. According to [7] the spectrum of the output OFDM waveform may be written as

$$X(f) = \frac{1}{\sqrt{T_S}} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} Q(k) \frac{\sin(\pi f - k\Delta f)}{\pi f} \tag{4.3}$$

where $Q(k)$are the QAM-symbols with a finite duration of $T_S = N/R_S$. If Δf is set to $1/T_s$ each sub-channel will be zero at the centre of every other sub-channel, and they are set to be orthogonal, as shown in Figure 4.12 where N=8.

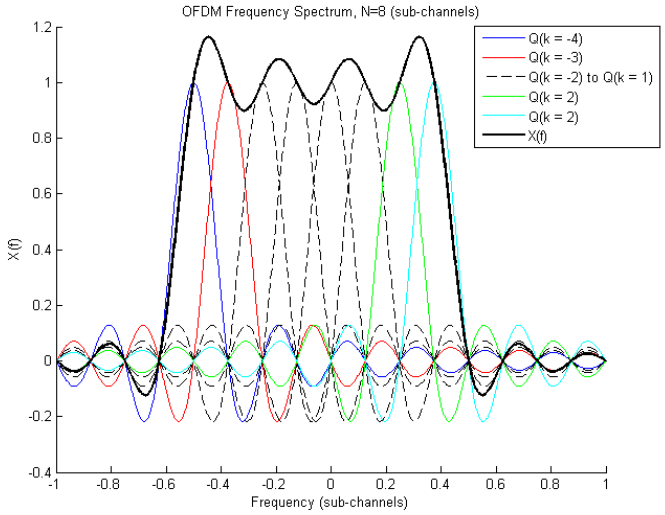


Figure 4.12 Frequency spectrum of an OFDM-signal with 8 sub-channels (N=8).

Figure 4.12 shows how the OFDM waveform contains all the QAM-symbols, and how they are spaced in frequency.

The OFDM waveform is implemented on the digital processing module's FPGA, giving it a reconfiguration possibility. The IFFT size can vary from 64 sub-channels to 2048, where a default configuration of 64 is used. The model also allows a use of other modulation forms (16- or 64-QAM), but the default modulation of 4-QAM is used. Other parameters that are reconfigurable are the symbol rate, number of unused sub-channels and general packet structure. One downside to the Lyrtech OFDM waveform is that it`s implemented in such form that the radio parameters cannot be changed after the FPGA configuration file is loaded to the platform. This is not ideal for a cognitive radio, where it`s desirable to configure the radio parameters instantaneously according to the surrounding environment. For instance would it be desirable to individually change the modulation type on each sub-channel and switch on and off unused sub-channels, without having to reloading a configuration file.

A total of nine OFDM blocks, of size 64, are spaced in time and put together to form a packet to be transmitted through the communication channel. These are preceded by a cyclic prefix, to reduce the *Inter-Symbol Interference* (ISI). The two first OFDM blocks are dedicated to AGC, packet detection and block boundary acquisition. The next two are channel estimation pilots, dedicated to a coarse *Carrier Frequency Offset* (CFO) estimation. These blocks also contain unused sub-channels at DC and band edges allowing for non-ideal filters in the analog front end.

Continuous frequency training pilots are also embedded in specific sub-channels for a fine CFO estimation and sampler offset tracking, as well as to estimate and correct the common phase error portion of phase noise. The total packet structure is summarized in Figure 4.13.

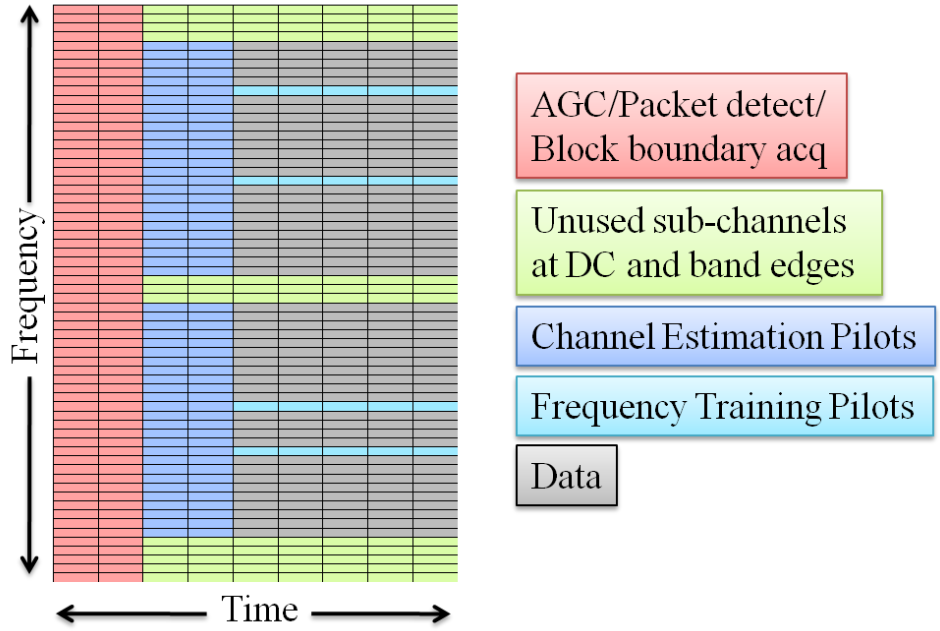


Figure 4.13 Default packet structure

Figure 4.13 shows how one packet consists of 5 data blocks, where each block holds 48 QAM-symbols. This means that one packet transmits 5*48=240 QAM-symbols, which according to Figure 4.11 equals 480 bit or 60 Byte. The total OFDM-packet is then pushed forward to the DAC and antenna, for transmission through the communication channel.

According to Equation (4.3) one sub-channel will occupy $\Delta f = 1/T_s = R_s/N$ of the frequency spectrum, where R_s is the QAM-symbol rate. This means that the total OFDM waveform occupies $N * \Delta f = R_s$ of the frequency spectrum. The OFDM waveform is implemented with a QAM-symbol rate of 20 MSymbols/s, generating a total spectrum usage of 20MHz.

The received OFDM-packet is detected via block boundary detection after down conversion to baseband, which does a correlation to a known PN-sequence in time domain. When the received samples coincide with the known sequence, a large peak occurs. This peak is compared to the average received power and a threshold, which determinates if a packet is present. A coarse estimate of CFO is also performed in the block boundary detection module. After packet detection and coarse CFO estimation the cyclic prefix is removed on each OFDM-waveform, before the packet is demodulated by performing a *Fast Fourier Transform* (FFT). This is the inverse operation of an IFFT, and the QAM-symbols modulated by the IFFT in the transmitter, is the output of the FFT in the receiver. Post the FFT operation a fine CFO estimation is performed, by using the continuous frequency training pilots in the packet. All the sub-channels will be rotated by same the amount in one OFDM block, and the amount of angular rotation from one block to the next is proportional to the CFO. After the fine CFO, every QAM-symbol is

demodulated according to the constellation diagram in Figure 4.11, and the information bits are obtained.

4.5.1.2 Modifications Done to the Waveform

To improve the communication performance some minor modifications are done to the waveform. The packet structure has been modified, where the number of unused sub-channels in the upper and lower sideband has been increased from respectively 4 to 8 and 5 to 9. To transmit the same amount of data, it`s been added one more data block of size 64 to the total OFDM symbol, increasing the total symbol length. The new packet structure is summarized in Figure 4.14.

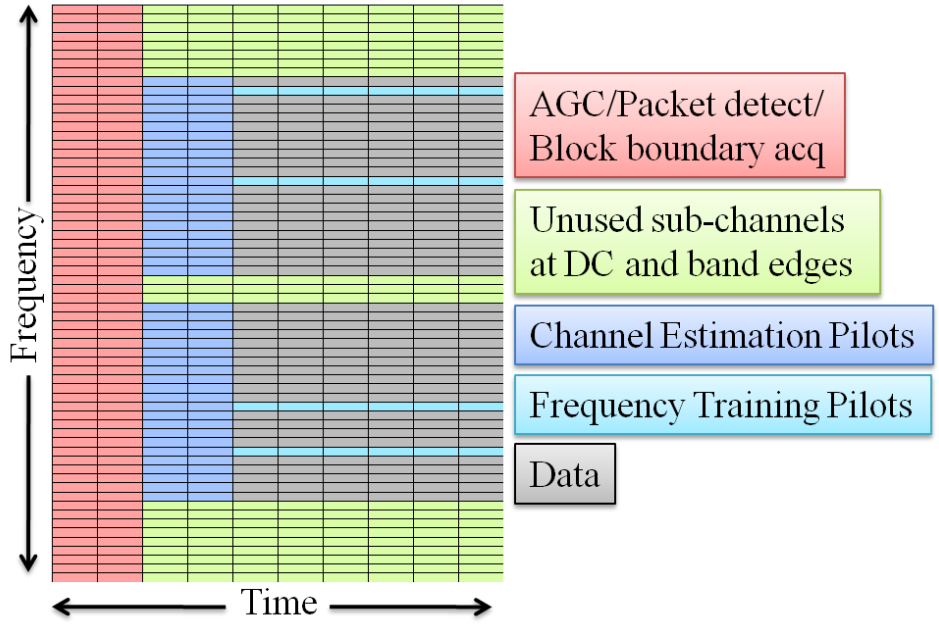


Figure 4.14 Modified packet structure

Figure 4.14 shows how the OFDM packet transmits the same amount of data, with 6 data blocks of 40 QAM-symbols, resulting in a total data amount of 60Byte (60*40=240 QAM symbols = 480 bit = 60 Byte). The threshold value for the block boundary detection in the receiver structure has also been tuned, to give the best communication performance regarding BER and receiving sensibility.

4.5.2 MAC

For reasons of simplicity, and inspired by elements in the 802.22 standard, the *Medium Access Control* (MAC) in the prototype system is a time controlled one only. Each node has relative accurate knowledge about a common absolute time, reached through sending time-stamped SYNC messages to all nodes in the local network, as explained previously in 4.3.5.4. (Alternatively, in a future version, the establishment of a common absolute time may be accomplished through the use of GPS receivers).

For a local network with N nodes, a chosen fundamental time interval T is split into N+1 equal slots

$$t = \frac{T}{N+1} \tag{4.4}$$

where t is divided into a transmit time t_t followed by a guard time t_g, such that t=$t_t + t_g$.

The slots are numbered 0..N. Each node sends in one of the slots, in correspondence with its sequence number in the local network (i.e. the node having the lowest node number sends in slot 0, the node having the highest node number in the local network sends in slot N-1.) Slot N is a sensing slot, where none of the nodes are transmitting and all of the nodes are instead sensing the spectrum noise levels. The sequence is illustrated in Figure 4.15.

The transmit/receive/sense state is controlled by the agent SW itself, inside the continuous DSA loop. This is also for implementation convenience only, one could argue that it is a detail at a too low level for the agent SW to really need to bother with.

The simple MAC implementation in the prototype is reflecting that we have not put a large emphasis on this in the prototype work, partly because we have an electronic warfare viewpoint. It is expected that practical, deployed systems will have much more advanced MACs.

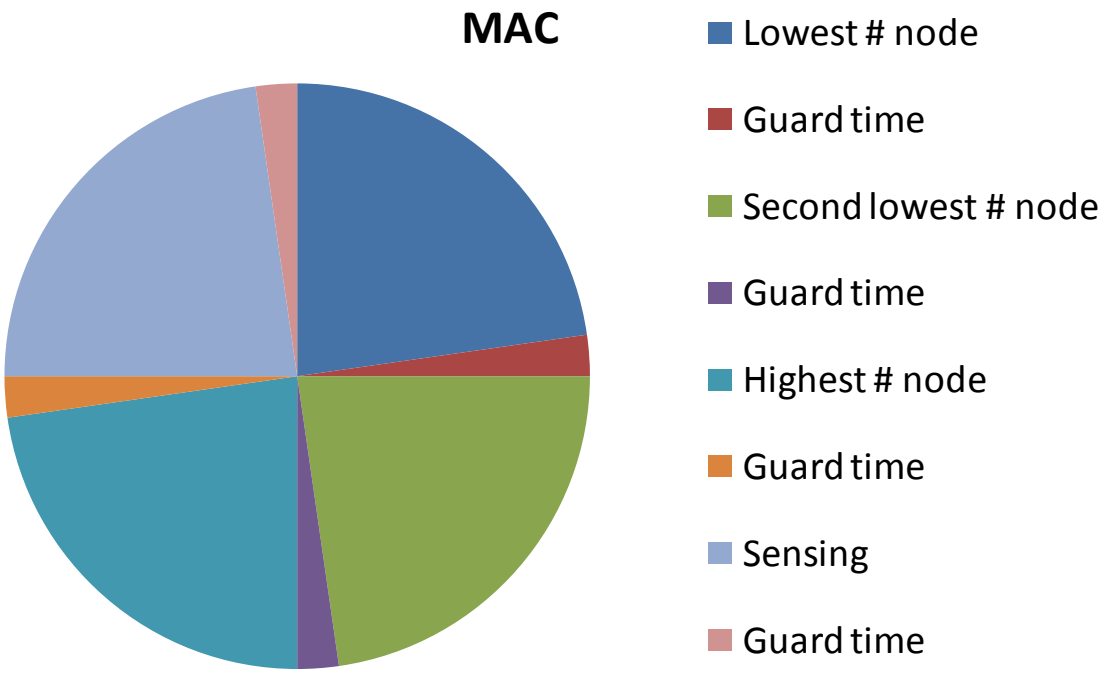


Figure 4.15 Illustration of the time controlled MAC of each local network, in the prototype implementation, for a local CRN network with 3 nodes. The complete circle corresponds to a time interval T.

4.6 The Sensing Functionality

The sensing functionality is implemented by using energy detection. This detection method computes the received energy and compares it to a preset threshold λ. If the received energy exceeds this threshold, a signal is present. The principle is shown in Figure 4.16.

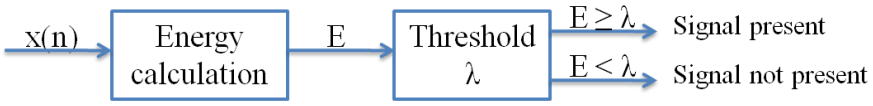


Figure 4.16 Principle energy detection

If $x(n)$ describes the received signal over the finite interval $0 \leq n \leq N-1$, the total energy may be calculated as [8]

$$E = \sum_{n=0}^{N-1} |x(n)|^2 \tag{4.5}$$

This equation summarizes the received power $|x(n)|^2$ over a given time interval ΔT, specified by N and the sampling frequency Fs (ΔT=N/Fs). Over the same finite interval, equation (4.5) may be written as

$$E = y(n) = \sum_{k=0}^{N-1} |x(n-k)|^2 \tag{4.6}$$

The only difference between equation (4.5) and (4.6), is that equation (4.5) calculates the energy over the next N samples, whereas equation (4.6) calculates the energy of the last N samples. For a real time system equation (4.6) is the only realizable, and may be expressed as

$$y(n) = \sum_{k=0}^{N-1} |x(n-k)|^2 = y(n-1) + [\,|x(n)|^2 - |x(n-N)|^2\,] \tag{4.7}$$

and be implemented by using the structure in Figure 4.17

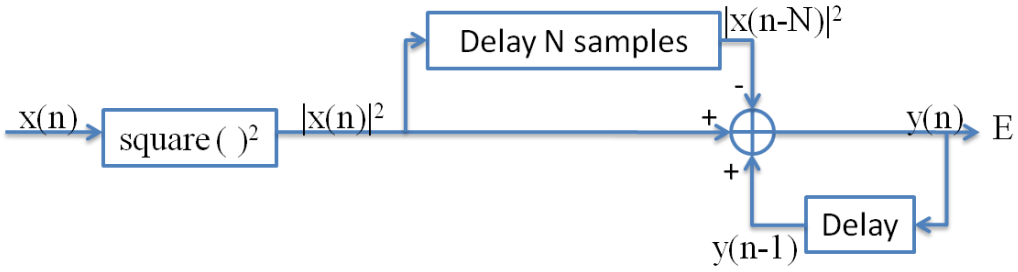


Figure 4.17 Moving Energy Calculator.

This structure is implemented on the SFF SDR platform`s FPGA design, alongside the COM functionality, and the calculated Energy is sent to a custom register for threshold comparison by the DSP. The structure is implemented with N=4096 and a FPGA speed of 80MHz, resulting in a time interval ΔT of 51.2 µs (4096/80 MHz = 51.2 µs).

The Moving Energy Calculator is the simplest form for energy detection, and calculates the energy in full signal bandwidth 20MHz. To extract information about where the energy is located in frequency, a Frequency Energy Calculator is implemented alongside in the FPGA design. By using Parseval's theorem [8] the Energy may be calculated as

$$E = \frac{1}{N}\sum_{k=0}^{N-1}|X(k)|^2 \tag{4.8}$$

where $X(k)$ is the Discrete Fourier Transform of $x(n)$ and $0 \leq k \leq N-1$. To compute the energy in equation (4.8), the structure in Figure 4.18 is implemented. This figure shows how an input signal is transformed to a frequency representation by a FFT-algorithm, squared to form a power spectrum and scaled to represent the energy in each frequency bin, over the calculated time interval ΔT.

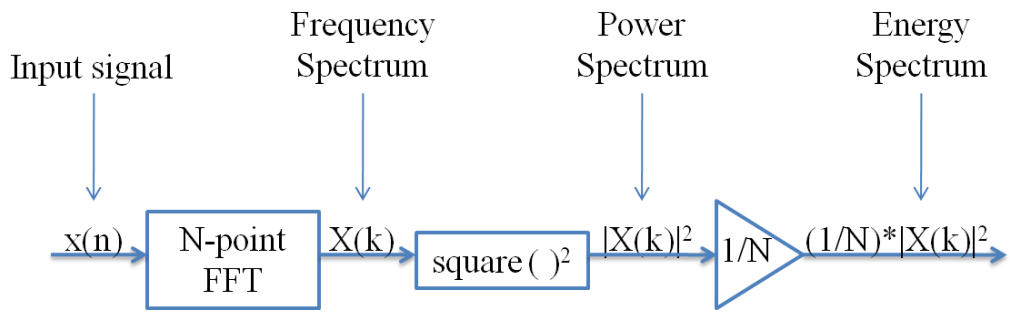


Figure 4.18 Frequency Energy Calculator

The FFT-algorithm is implemented with a length of 4096, resulting in a time interval ΔT of 51.2 µs and a frequency resolution of 19.53 kHz, when the FPGA runs at 80 MHz. Since the signal bandwidth is limited to 20MHz by the analogue RF front end, only 1024 of the total 4096 FFT-calculations represent the signal bandwidth. Therefore only 1024 FFT-calculations are pushed forward to the DSP for threshold comparison. By studying these 1024 FFT-calculations, the DSP is able to determine where the energy is located within the signal bandwidth of 20 MHz, with a frequency resolution of 19.53 kHz (20 MHz/1024 = 19.53 kHz).

4.7 The ES Functionality

The ES functionality is implemented by performing a spectrum monitoring based on energy sensing. When monitoring is enabled, the MILKOG GUI allows the user to specify a frequency span and a number of spectrum segments to be monitored. If the segment size is a multiple of 20 MHz, the Moving Energy Calculator is enabled. For monitoring smaller segments the Frequency Energy Calculator is used, down to a minimum segment size of 19.53 kHz.

Before the ES data is displayed in the MILKOG GUI, the sensed energy in each segment is averaged, to minimize variability in the energy calculations. When the Moving Energy Calculator is enabled, the DSP reads 1000 energy calculations for each segment, and sends the average energy of each segment to the ES monitor. When the Frequency Energy Calculator is enabled, the FPGA calculates the energy in each segment 10 times, and sends the averaged energy of each segment to the ES monitor via the DSP.

4.8 The EA Functionality

The EA functionality has been implemented as a reactive jammer. The reactive jammer has a look-through phase, in which it scans the spectrum to discover emitters, and a jamming phase, in which it performs the jamming. The jamming is full signal-band jamming.

The MILKOG GUI allows the user to specify the spectrum interval in which to scan for emitters. The scanning starts at the lower limit of the spectrum interval, and then increments with 10MHz until the frequency is outside the specified interval. For each frequency setting, the RF chain is allowed to settle for 0.5 milliseconds. Then 50 values are measured, with a wait interval of 1 microsecond between each value, using the moving energy functionality in the FPGA. The EA jammer functionality then selects the highest sensed emitter as the target for the jamming. If no emitters are found above a certain threshold, the jammer is not activated.

When the reactive jammer is activated, the MILKOG agent SW initiates a new jamming look-through phase approximately each second. The duration of the look-through phase depends on the size of the scanning spectrum interval. As an example, if the allowed EA spectrum band is between 400 and 600MHz, the theoretical look-through time is $21 \cdot 0.55$ milliseconds.

The jammer waveform is implemented by using a white noise generator in the SFF SDR platform`s FPGA design. To set the noise bandwidth approximately equal the signal bandwidth (20 MHz), a lowpass filter is used for spectrum shaping. This filter has a passband from DC to 10 MHz and a stopband from 14 MHz, which gives a 3-dB bandwidth of 22.8 MHz after up conversion to RF.

Obviously, for a reactive jamming pattern to be effective, the look-through phase needs to be far shorter than the one implemented here. Also, a practical reactive jammer needs to be able to target multiple emitters instead of just one, and should be able to also use far narrower bandwidths. Reactive jamming will be explored further in future work.

4.9 The Traffic Source and Sink

The traffic source and sink both utilize the open source video server and client VLC from www.videolan.org. This enables a nice visual demonstration of communication link quality, with minimum implementation effort.

The dataflow is illustrated in Figure 4.19. VLC captures the video+audio from the PC camera and microphone, then sends it to the Forwarder. The Forwarder reformats the video stream to the

default packet sizes handled by the SFF SDR platform and sends it over on the TCP connection. The data is then reformatted again to be sent in 60-byte packets over the OFDM waveform, to other nodes in the network. The Forwarder on the receiving nodes formats the data back into the packet sizes expected by the video client. To compensate somewhat for the lack of error correction in the communication waveform implementation, the Forwarder sends each video block three times.

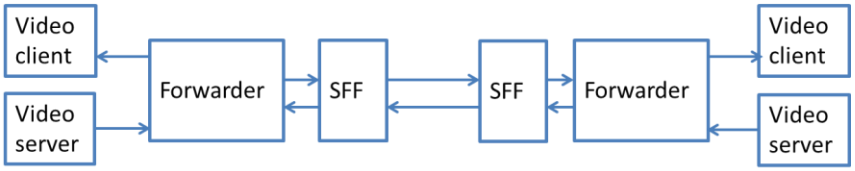


Figure 4.19 Dataflow from data source (VLC videostreamer) to data sink (VLC video client).

For the case of node 1, the video server is started using the command line sequence in Figure 4.20. This makes VLC send a UDP video+audio stream to IPv4 address 192.168.1.10 port 50020, which is the port that the 'receive from data source' thread in the Forwarder is listening to. The command sequence to create the video client is provided in Figure 4.21.

```
c:\Dev\Videolan\VLC\VLC dshow://
:sout=#transcode{vcodec=DIV3,vb=200,scale=0.1,acodec=mp3,ab=32,channels=2,samplerate=44100}
:udp{dst=192.168.2.10:50020} :no-sout-rtp-sap :no-sout-standard-sap :ttl=1 :sout-keep
```

Figure 4.20 Command sequence to VLC to create the video streamer.

```
c:\Dev\Videolan\VLC\VLC udp://@192.168.1.10:50000
```

Figure 4.21 Command sequence to VLC to create the video client.

5 Prototype Verification Tests

To document the prototype verification tests, 3 nodes were put together to form a radio network. Each radio node was set up with a MILKOG SW-agent and connected together via an IP control network for spectrum decisions, as specified in Chapter 4 and Figure 4.1. QoS data and payload data was sent between the radio nodes according to the RF setup displayed in Figure 5.1.

Figure 5.1 shows how each node is connected though a communication channel, via attenuators and splitters/combiners. The number displayed inside these boxes illustrates the attenuation on each component. In addition a spectrum analyzer and a signal generator were added, to act as respectively a spectrum displayer and a jammer/signal generator. In the RX section the attenuators were tuned to give the best communication performance, with respect to BER and missed packages percentage.

The following components were used for the prototype verification tests:

- 3 Lyrtech SFF SDR radio nodes.
- 1 Rohde & Swartz FSW Signal and Spectrum Analyzer (2Hz – 8GHz).
- 1 Rohde & Swartz SMIQ-06B Signal Generator (300kHz – 6.4GHz).
- 2 Mini-Circuits ZFSC-4-1-S+ splitters/combiners.
- 2 Narda 4747-60 step attenuators (DC – 18GHz).
- 1 Hewlett Packard 355C UHF step attenuator (DC – 1000MHz).
- Various Mini-Circuits 15542 attenuators of different values.
- Various coaxial cables.

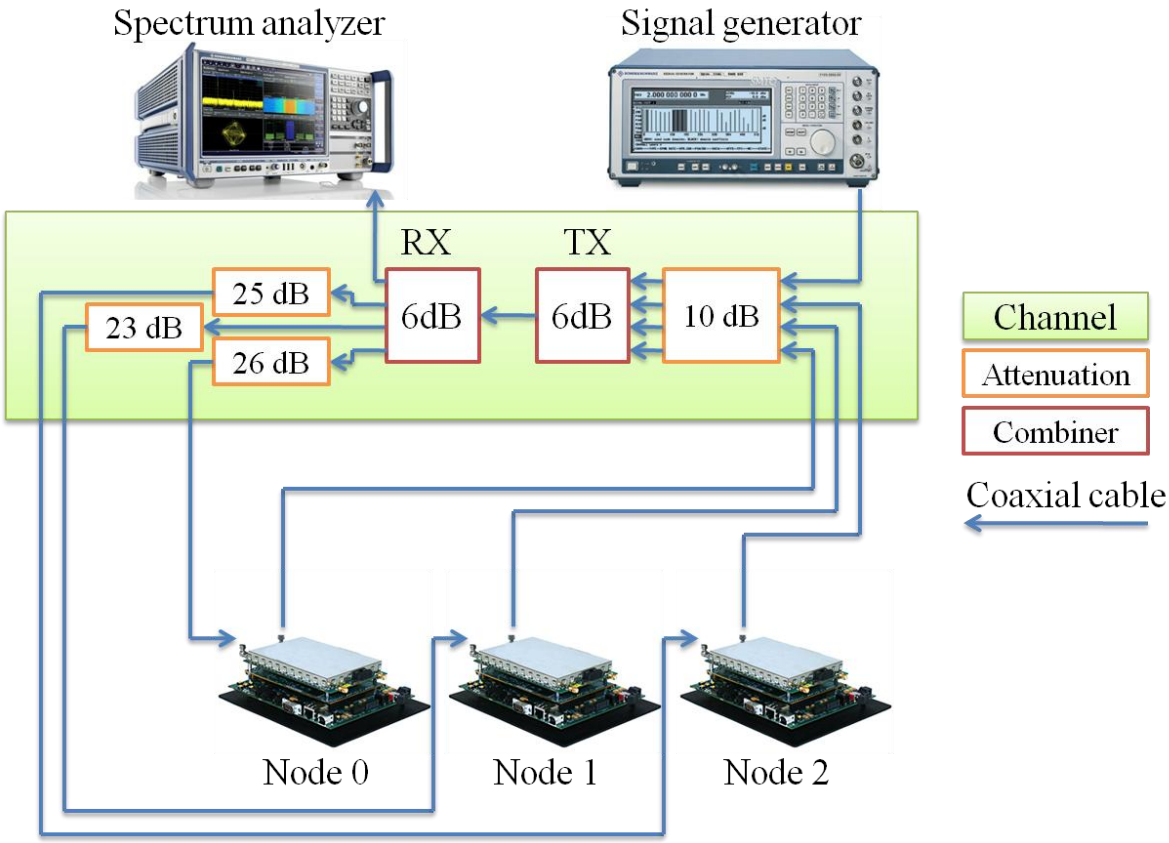


Figure 5.1 RF setup

5.1 COM Functionality Tests

5.1.1 Test Pattern Bit Rate and Bit Error Rate Measurements

The purpose of this test was to examine the MILKOG`s QoS data, when a test pattern was sent between the radio nodes.

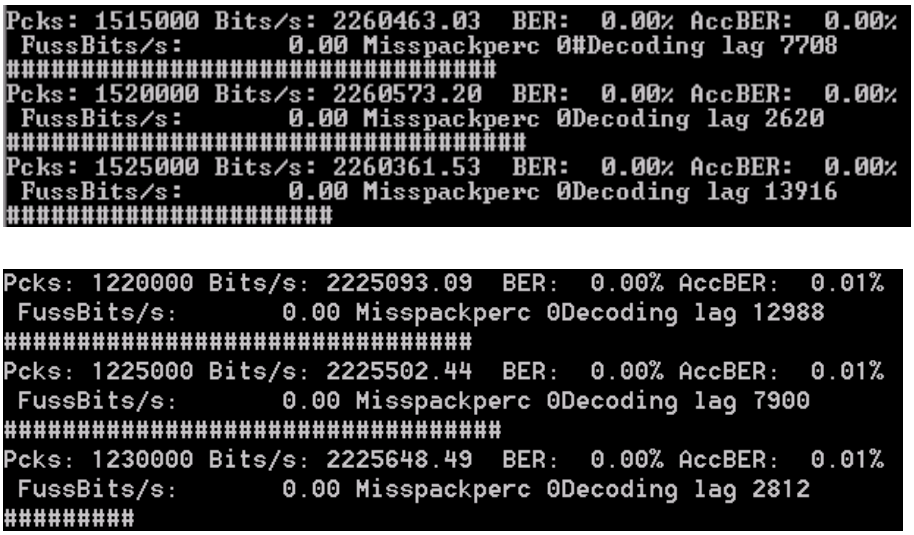
Prior to monitoring QoS data, the following initializations were done:

- The RF network was set up according to Figure 5.1.
- RX/TX control was set to ‘Receive (constantly)’.

Node 1 and 2 were then set up to send QoS data between each other, by setting the RX/TX control to ‘Transmit (constantly)’ in turn. The results of these tests are displayed in Figure 5.2.

Figure 5.2 shows how the bit rate between the nodes exceeds 2,25 Mbps, with a BER at 0.00% missed package percentage of 0. The bit rate difference between Figure 5.2.a and 5.2.b is because the displayed rate is the accumulated bit rate, not the instantaneous bit rate. The snap shot in Figure 5.2 is taken when the accumulated bit rate increase has leveled off.

By default the MILKOG SW agent uses a transmission frequency of 480 MHz. A snapshot of the spectrum analyzer, see Figure 5.3, displays the transmission frequency and spectrum usage, when QoS data is transmitted between the radio nodes.



a)

b)

Figure 5.2 QoS data. a) Node 2 to node 1. b) Node 1 to node 2

Figure 5.3 shows how the waveform uses more frequency resources than the 20MHz predicted in waveform description in subsection 4.5. The main reasons for this are believed to be image frequency components generated in the analogue frequency up conversion and sideband effects generated by the OFDM algorithm.

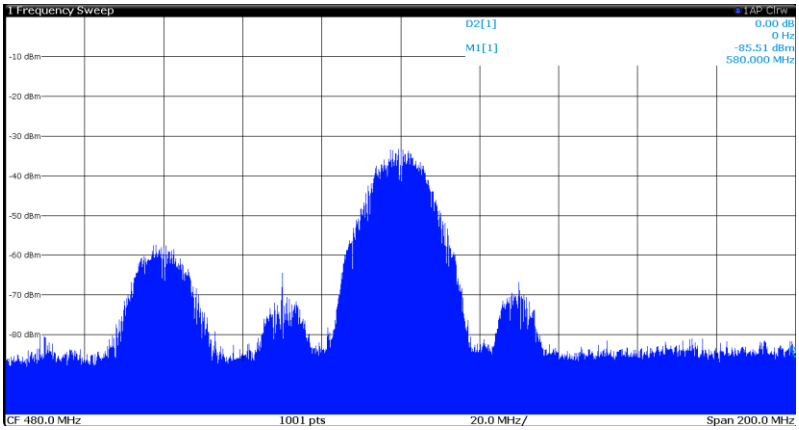


Figure 5.3 Frequency spectrum of QoS data, with center frequency at 480MHz.

5.1.2 Video Transfer Test

The purpose of this test was to visually demonstrate the MILKOG`s communication link quality, when payload data was sent between two radio nodes.

Prior to sending payload data, the following initializations were done:

- The RF network was set up according to Figure 5.1.
- RX/TX control was set to 'Receive (constantly)'.
- A video client and a video server were started at each node.

First, node 1 was set to 'Transmit (constantly)', enabling the video client and server to send live video and audio from node's webcam. A snapshot of the received video on node 2 is displayed in Figure 5.4.a. Thereafter, node 2 was set to 'Transmit (constantly)'. A snapshot of the live video on node 1 is displayed in figure 5.4.b. The default transmission frequency of 480MHz was used for these tests.

a) Video stream on node 1, from node 2 *b) Video stream on node 2, from node 1*

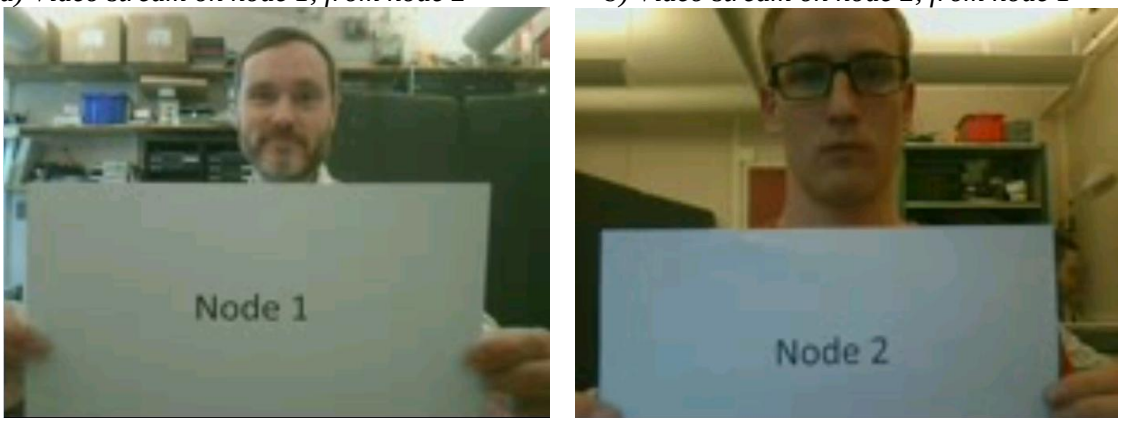


Figure 5.4 Visual demonstration of communication link quality.

Figure 5.4 shows that the video is transferred with acceptable quality. The video resolution displayed in Figure 5.4 is 160x120, which is quite poor when taking into account the bit rate of 2.25Mbps. The main reason for this low resolution is that the waveform has no bit error correction algorithm. Each video block is transferred 3 times, and the video block that is the most similar to another block is displayed. If convolutional error correcting codes and a Viterbi decoder had been included, the resolution would be far better.

5.2 COM DSA Test:

5.2.1 Avoiding a Slow Follower Jammer

The purpose of this test was to verify that MILKOG's DSA functionality, when the COM channel was being jammed, was able to change to a new unjammed frequency.

Prior to applying the jamming signals, the following initializations were done:

- The RF network was set up according to Figure 5.1.
- Node 1 and node 2 were associated in a network by using the *radio network configuration menu* in node 2
- DSA was enabled on both nodes
- Spectrum band start was set to 400MHz, band stop to 600MHz and number of segments to 10.
- RX/TX control was set to 'Time controlled TX/RX' to enable the time controlled MAC previously described.

After these initial steps and prior to applying jamming, the MILKOG agents of node 1 and node 2 had coordinated to use a center frequency of 530 MHz, refer to Figure 5.5 a).

Jamming was then applied at this center frequency, using a 40MHz wide flat spectrum signal, with a generator power output of -1.4dBm. It was observed that the COM center frequency now changed to 590 MHz, see Figure 5.5 b), and that the noise jamming was observable in the sense spectrum.

Acting as a slow follower jammer, we adjusted the jammer center frequency to the new COM frequency, i.e. 590 MHz. It was observed that the COM center frequency changed back to 530 MHz, see Figure 5.5 c).

Tuning the jammer once again to 530 MHz, the COM center frequency changed to 590 MHz once again, see Figure 5.5 d).

The exact time required for MILKOG to change its COM frequency was not measured, however the response was in the order of several seconds. The slow response is primarily due to our primitive MAC, where one TX-RX-SENSE MAC cycle in the prototype (with two nodes connected) is approximately 5+5+5 seconds.

In conclusion, it was verified that basic DSA functionality was able to initiate frequency changes when being jammed, and correctly establish communication between the two nodes in a new, unjammed spectrum segment. However, it is clear that the slow DSA response of the system does not make the system suitable for coping with a fast follower jammer, and that faster responses and/or more advanced strategies are beneficial.

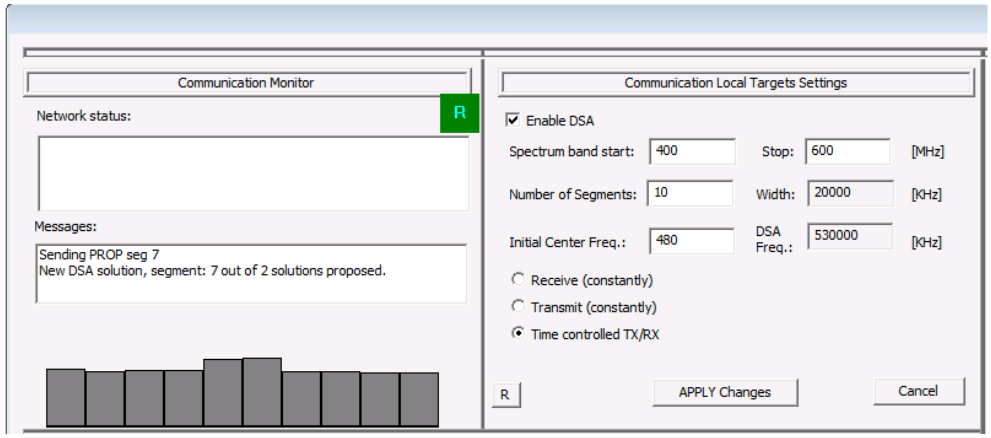
5.3 ES Functionality Test: Energy Spectrum Monitoring

The purpose of this test was to verify that MILKOG, when enabled as an energy spectrum monitor, was able to detect and display frequency resources in use.

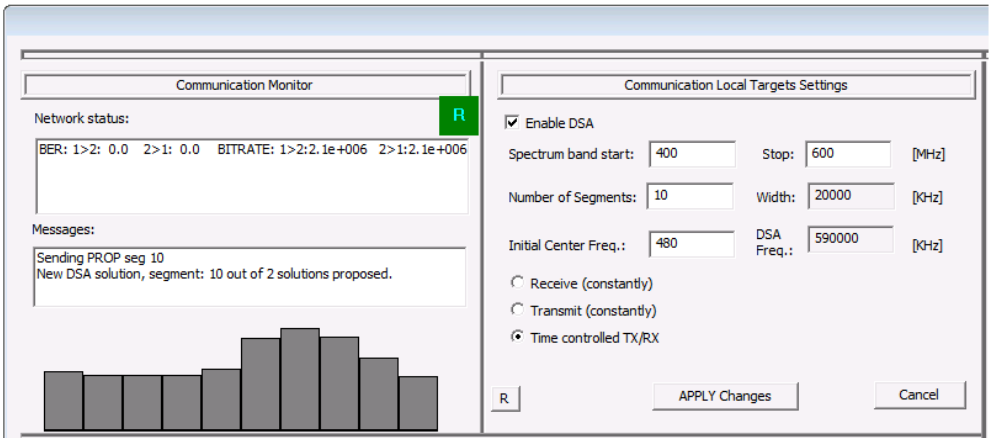
Prior to applying the power spectrum monitoring, the following initializations were done:

- The RF network was set up according to Figure 5.1.
- Monitoring was enabled on node 2.
- ES frequency start was set to 300MHz, frequency stop to 400MHz and number of segments to 100.
- The signal generator was set up with a 20MHz wide flat spectrum signal, with a generator power output of -1.4dBm, centered at 350MHz.

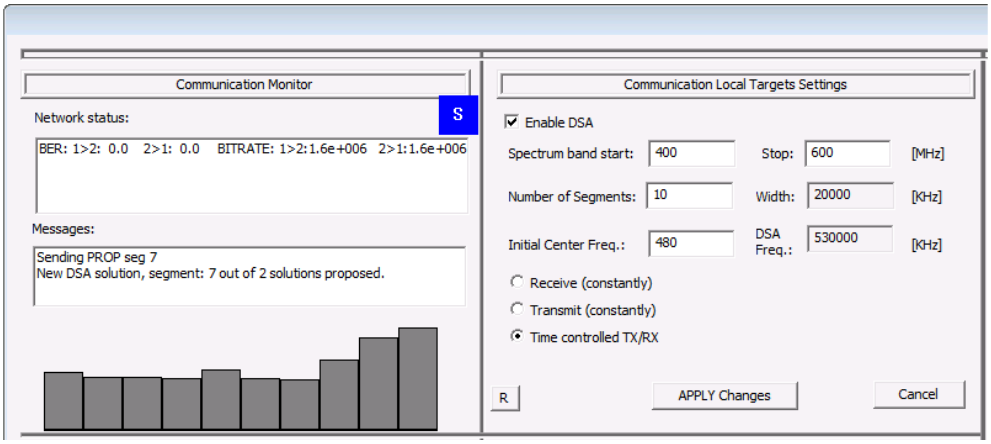
To verify the frequency spectrum generated by the signal generator, the spectrum analyzer displayed the same frequency area as set prior in the MILKOG agent, se Figure 5.6.



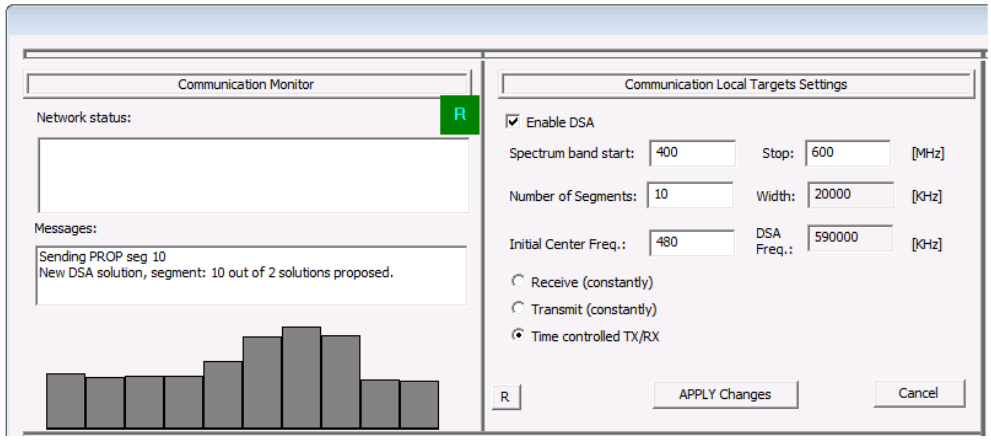
a)



b)



c)



d)

Figure 5.5 DSA test. a) Jammer off, COM freq. 530MHz. b) Jammer at 530MHz, COM moved to 590. c) Jammer at 590, COM moved to 530. d) J. at 530, COM moved to 590.

Figure 5.6 shows how the signal generator generates a flat spectrum with a bandwidth of approximately 20MHz. The parameter setting of 100 frequency bins over a span of 100MHz gives a spectrum resolution of 1MHz. This enables the MILKOG`s frequency energy calculator, since the frequency resolution is less than 20MHz. The monitored energy spectrum on the MILKOG agent is displayed in Figure 5.7.

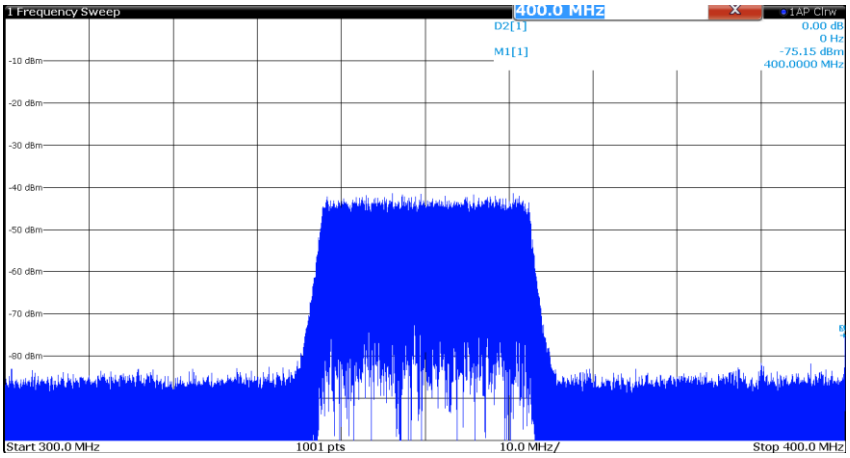


Figure 5.6 Spectrum usage of signal generator with flat spectrum

Figure 5.6 and 5.7 shows the same frequency spectrum displayed both on the spectrum analyzer and in the GUI of the MILKOG agent. There is a slight different in the spectrum shape, especially where the spectrum increases and falls off. The main reason for this is the length of the FFT used for the spectrum calculations. The frequency resolution in the spectrum analyzer is much finer, resulting in a larger FFT calculation noise for the spectrum displayed on the MILKOG agent.

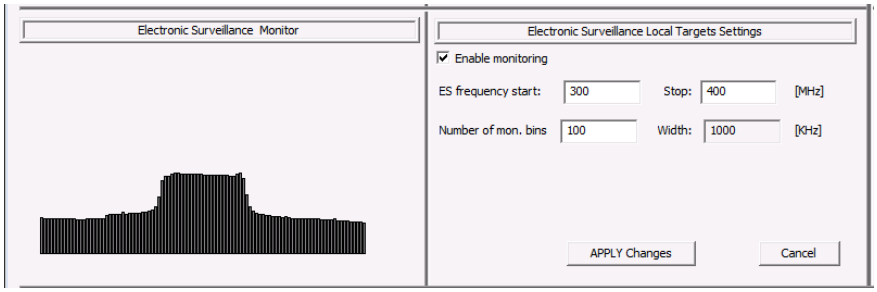


Figure 5.7 Electronic Surveillance Monitor, with frequency resolution of 1MHz.

The signal displayed in Figure 5.6 was also monitored by the moving energy calculator. This energy calculator is enabled when the frequency bin width $\geq$ 20MHz. Therefore, the frequency span was increased to 300 - 900 MHz, with a total of 30 frequency bins, giving a frequency resolution of 20MHz. This monitored spectrum is displayed in Figure 5.8, where the signal at 350MHz spans over three frequency bins.

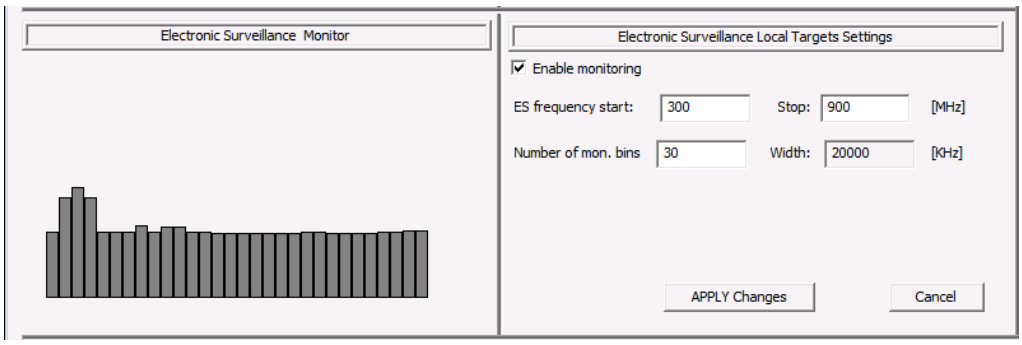


Figure 5.8 Electronic Surveillance Monitor, with frequency resolution of 20MHz.

5.4 EA Functionality Test: Reactive Jamming

The purpose of this test was to verify that MILKOG, when enabled as a reactive jammer, was able to place its jamming signal on top of a payload signal in a given frequency range.

Prior to applying the reactive jamming, the following initializations were done:

- The RF network was set up according to Figure 5.1.
- Node 1 and node 2 were associated in a network by using the *radio network configuration menu*
- DSA was enabled on both nodes
- Spectrum band start was set to 400MHz, band stop to 600MHz and number of segments to 10.
- RX/TX control was set to 'Time controlled TX/RX' to enable the time controlled MAC previously described.
- A video client and a video server were started at each node.

The two nodes were observed to transmit video signals to each other, with a typical observed peak bit rate on node 1 of 1.8E6 bits/second, and typical BER of 0.0%, see Figure 5.9.

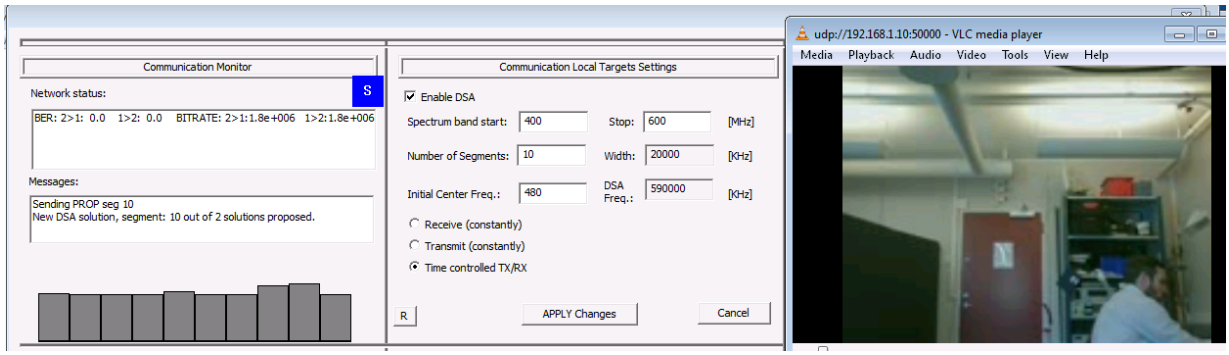


Figure 5.9 Node 1 GUI, prior to the reactive jamming.

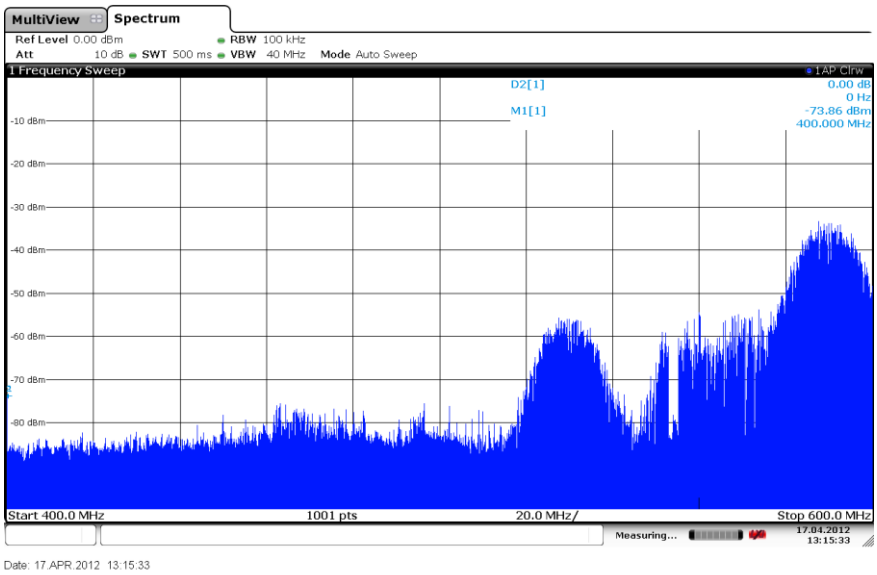


Figure 5.10 Non-jammed spectrum, network here operates at 590 MHz.

To enable jamming, the following configurations were then done to node 0, which was to act as the reactive jammer:

- Spectrum band start 400MHz, band stop 600MHz, jamming ON, reactive jamming (see Figure 5.11).

On the spectrum analyzer we observed that the jamming signal followed the spectrum changes of the payload signal. The video stream was observed to be partly broken down, with very few updates to the video pictures.

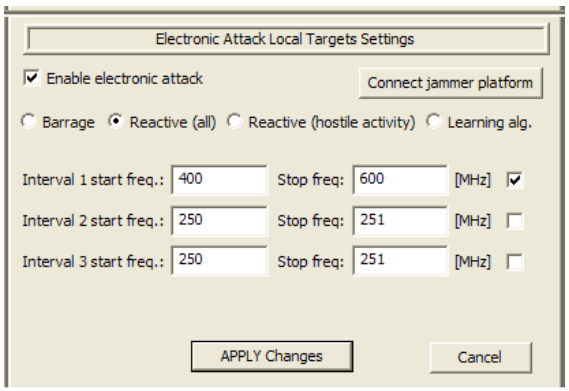


Figure 5.11 The settings of the reactive jammer.

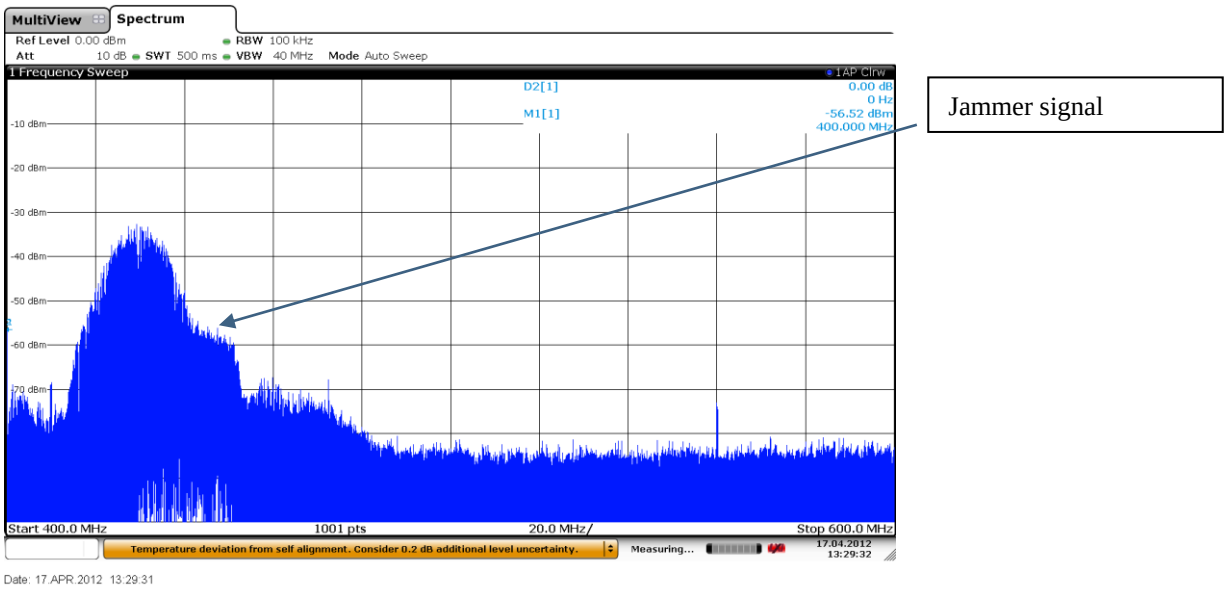


Figure 5.12 Reactive jamming test.

The MILKOG GUI on node 1, see Figure 5.13, reported a peak bit data rate of 5.8E6 bits/second, which implies that there is still some data coming through on the link, but the majority of the packets are lost. The reason for the jamming not being 100% effective is partly the look-through interval of the jamming functionality, in which the jammer is quiet such that the receiver can look for jamming targets. With the reactive jamming spectrum band set at 400 to 600MHz, the measured look-through interval was 41 milliseconds (out of a total of approximately 1 second). For a jammer to be used in real environments, both the look-through and the jamming duration need to be reduced.

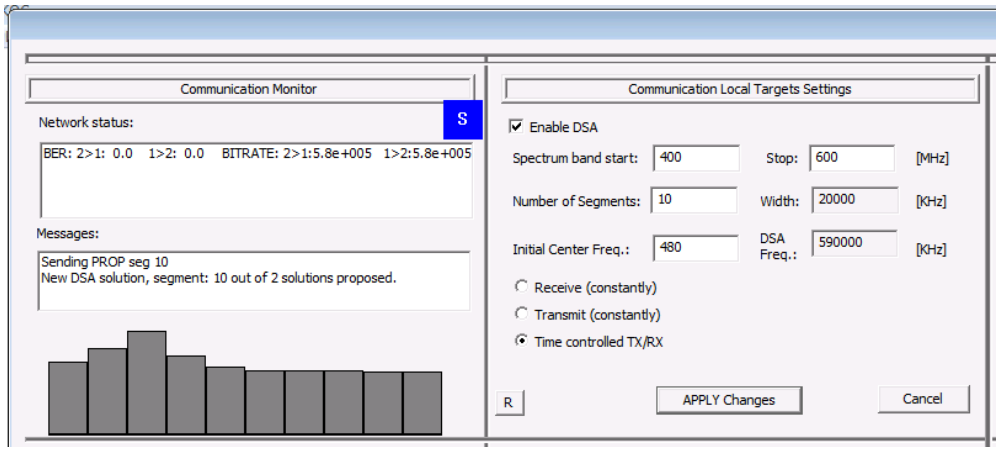


Figure 5.13 Reactive jamming test, BER and bit rate observation on node 1.

Another reason for the jamming not being 100% effective is that the jamming signal is (approximately 20dB) lower than the OFDM signal. This is probably a minor issue, the jammer output may be improved by tuning the digital-to-analog converter gain of the jammer.

6 Lessons Learnt from the MILKOG Prototype Development

6.1 Lyrtech SFF SDR HW Experiences

The Lyrtech SFF SDR platform was selected as the platform-of-choice for MILKOG based on a formal tender process initiated in November 2009. The evaluation criteria was a weighted balance of technical specifications, cost, time of delivery, training, support and vendor references.

We have made the following positive experiences with the SFFs:

+ The SFFs have sufficient digital processing power and Ethernet interface performance to run broadband waveforms (in our case above 2 Mbit/second works fine (on two of the platforms, see below)).
+ The SFF is well documented, and the platform is well integrated with the design tools (see Section 6.2).

We have observed the following issues:

– There are platform-to-platform variability issues that we do not currently understand, and that prevent us from using all of our five platforms in all roles (COM-ES-EA). Two of the five platforms are working in full speed (2MBits/second+) with the COM waveform. With the other three platforms the COM transmit speed is significantly slowed down. Our hypothesis is that the issues we observe are related to clock synchronization.
– We have had one platform that was repaired within the warranty period, and one unit that became defective outside of warranty and that was repaired at FFI. 2-3 spare RF modules are also defective (outside of warranty).
– The quality of the support provided has been variable but often slow and inadequate.
– The FPGA on the platform (Xinlinx Virtex IV) is too small to simultaneously include both the OFDM waveform functionality and all the ES and EA functionality that we would like to include

6.2 SFF Design Tools Experiences

As explained in Section 4.5, the software tools are well integrated and documented. By following demos and model-based design tutorials provided by Lyrtech, it was easy to learn how to program and implement own functionality in the SFF SDR.

We have made the following positive experiences with the SFF SDR design tools:

+ Simulink and Xilinx System Generator offer a high-level block based FPGA programming, without needing to know VHDL.
+ The integration with MATLAB allows the use of MATLAB code within the FPGA design. (Also possible to use C code and VHDL).
+ Since the FPGA design is implemented in Simulink, simulation of the design is an easy task.

+ Flexibility is provided in the programming of the DSP, either standard c-code using Code Composer or graphical programming using Simulink.

We also experienced some drawbacks with the design tools:

- The generation of a new FPGA configuration bit file takes a fair amount of time with Xilinx System Generator .
- Memory problems causing MATLAB to crash when simulating the FPGA design.

6.3 Pros and Cons of FPGA PHY

The OFDM waveform was implemented on the SFF SDRs FPGA, alongside EA and ES functionality. The EA and ES functionality was made from scratch, while the OFDM waveform was based on a Lyrtech example.

We made the following positive experiences while implementing the desired functionality:

+ The platform FPGA has allowed us to run a high-speed broadband OFDM waveform, at the same time as ES and EA functionality is implemented.
+ Relatively easy to modify the following parameters on the OFDM waveform; number of unused sub-channels, number of data blocks in each OFDM symbol and the OFDM symbol rate.
+ Allowed us to implement two types of sensing functionalities, one narrowband FFT-based and one broadband moving average based.

The following negative experiences were made during the implementation process:

- The OFDM waveform is implemented in such form that a fast reconfigurable complex waveform (with e.g. many optional modulations) is difficult to construct.
- The documentation of the example OFDM waveform was partly insufficient, resulting in some confusion about the waveform functionality.
- Cumbersome handling of the connection bus between the DSP and the FPGA, especially while implementing the FFT-based energy sensing.

6.4 Distributed Node SW Architecture

As was illustrated in Figure 4.1, each MILKOG node is implemented as a distributed system, where the different parts are connected through Ethernet. Part of the processing occurs on a multicore PC, and other parts of it on the DSP and FPGA of the SFF platform.

The primary reasons why this approach was selected, were:

- It was found to be practical to implement the agent SW on a PC, such that we were able to have a proper Windows GUI on it.

- The distributed approach makes it easy to add more signal processing platforms to each node
- The SFF required data in/out to go through its Ethernet interface

There are some drawbacks with the selected architecture:

- It is not possible to have very precise time control from the MILKOG agent and to the DSP and FPGA processing parts. This is due to the long communication path from the agent to the DSP on the SFF: The MILKOG agent sends its command to the Forwarder, which sends it to the GPP on the SFF, which forwards it to the DSP. The long path introduces significant latency and latency variation.
- Since the system has many interacting processes and communication paths, failure analysis gets more time-consuming than with a simpler system.

7 Planned Expansions of MILKOG

7.1 Functionality Improvement for Practical Tests

The MILKOG system described in this report is aimed at laboratory-environment type tests only. In order to be able to use the system to demonstrate operational benefits in more realistic environments, several improvements are needed:

- reduction of the minimum occupied bandwidth of the communication mode: The current version of the system occupies 20MHz minimum. However, to obtain temporary frequency assignments for several times 20 MHz for field tests is somewhat unrealistic. Bandwidths below e.g. 2 MHz would be easier to handle in field test environments.
- get rid of wirebound administrative traffic: The current version runs administrative traffic on Ethernet connections, for more practical tests this traffic needs to be wireless. To limit the work associated with this, a suggestion is to make this communication in-band using the same waveform as for the payload communication.
- improve the automatic gain control: With the current prototype, only a 'fine adjustment' AGC is present and receive signals within a fairly narrow range is needed. A course AGC has been partly implemented. For more practical tests, this course AGC needs to be optimized and put in action.

The following improvements are advantageous:

- add error-correction coding, for improved quality-of-service with practical use of the system
- investigate if the possibilities for non-contiguous OFDM, with individual suppression and/or modulation of the carriers in the OFDM waveform: This enables better adaptation to the spectrum conditions, but due to significant parts of the waveform residing in FPGA code, this is a comprehensive task.
- include other waveforms, such as e.g. the NATO narrow-band waveform

7.2 Cognitive Architecture and Optimization Methodology

The decision functionality in the current version of MILKOG is based on a state machine and simple programmed rules (if-then-else type). In the future work, we will focus on more advanced reasoning and decision making.

While cognitive architecture research in itself has matured for quite some time, with a number of different architectures having been suggested and downloadable software existing, cognitive architecture research in cognitive radio is not that mature. Most approaches rely on simple architectures, such as based on the OODA loop. Cognitive architecture for cognitive radio is therefore an area where there are possibilities for fundamentally new research.

With FFIs significant in-house competence on genetic algorithms, it is natural to include and test this in MILKOG. However, the concept of using genetic algorithms as part of the cognitive engine of a cognitive radio is not new from a scientific point of view, it has been suggested, tested and also patented by Virginia Tech [9].

7.3 Systems of Communication Nodes and Jamming Nodes

A significant portion of what makes multi-agent systems, of which cognitive radio systems are a special case, complex to analyze, is the interactive reaction patterns: A decision within one agent inflicts on decisions within other agents that again changes the environment for the first agent and may cause it to reevaluate its decision.

Game theory is a mathematical tool to study such systems of interacting agents. There exists a vast literature on this subject.

We expect to study node interaction by using both simulation models, which enables to study many-node-scenarios, and the MILKOG prototypes, which may provide real-scenario experience with few-node scenarios.

7.4 Reactive and Proactive Jamming

There are at least two ways of achieving superiority on the electromagnetic battlefield: One is to have superior speed in reactive responses. The other way is to have superior planning of smart strategic and tactical moves.

In the work so far we have demonstrated that the multifunction concept enables us to create sophisticated reactive jammer capabilities. We plan to continue this research, and try to answer central questions such as how fast such reactive responses can be made, and how targeted they can be made. E.g. how should one block adversary communication but not affect own communication.

Central in proactive jamming is the ability to create plans for electronic attack activities, and to modify plans as new information is observed.

Both reactive and proactive jamming are closely connected to the study of cognitive architectures: Reactive jamming benefits from cognitive architectures with fast reasoning and ones that may compile reasoning into fast ("sub-conscious") reaction patterns. Proactive jamming requires a cognitive architecture with good reasoning and planning capability.

7.5 MILKOG Databases

The MILKOG architecture assumes that there is a reachable infrastructure with three types of databases: One for prioritized communications (e.g. broadcasters) that are non-cognitive or that for good reasons should not be challenged by a cognitive radio node. The second is for aggregated monitoring data. The third is for electronic attack tasks. Neither of these databases have been focused nor prototyped in the work so far.

The databases represent infrastructure that, as viewed from each SW agent, creates an environment with persistence and aggregation of information. In multi-agent systems in general, at least some authors claim that communication with the environment is beneficial, something which is frequently illustrated with examples from biology and from human societies.

The spectrum database is of particular interest for the near term future. In order to allow cognitive radio nodes into military bands, spectrum managers are likely to want to maintain control and need an assurance that the cognitive nodes do not interfere with other prioritized military traffic. Having a spectrum database that the cognitive nodes may download information from on a regular basis, may contribute to this assurance. In a transition phase, or as required from spectrum managers, the cognitive radio nodes may also upload their decisions to such a database.

The activity should both investigate how COM and EA databases could be integrated in operations planning concepts in general, and suggest how they could be practically implemented.

8 Conclusions

This report documents efforts taken to investigate, design and develop multifunction radio units. The multifunction term in this case points to the combination of COM and EW on the same processing platform. The development has been carried out using commercially available software defined radio platforms from Lyrtech Inc, and a combination of high-level and lower-level design tools.

The main hypothesis motivating this work has been that having such multifunction radio units, provides a higher level of flexibility for the use of the units, and that the units may contribute e.g. in an electromagnetic monitoring role while simultaneously fulfilling their communication role. A further hypothesis has been that a SW agent in each node, managing both COM and EW functionalities simultaneously, may provide more coordinated and hence more effective COM and EW.

We have suggested an architectural environment for multifunction nodes, and an internal architecture for the SW agent and the multifunction nodes themselves. A prototype SW agent, managing the COM, ES and EA functionality and including a subset of the architecture, has been developed for purposes of verification and experimentation.

For dynamic spectrum access experimentation, we have adapted an OFDM waveform application, where the basis has been bought from Lyrtech, to run at above 2 Mbits/ second. The waveform application has been integrated with an open-source video server and video monitor. We have not included error correction in the waveform application.

The agent SW makes spectrum decisions for the COM functionality, but only based on simple rules (if-then-else) at this stage.

The first hypothesis above ("flexibility") has been practically demonstrated e.g. by the units being capable of monitoring the spectrum while communicating, and also by the combination of the monitoring and jamming capability to enable reactive jamming.

While the work in this report has provided some support for and suggested an architecture for the second hypothesis above ("better coordination"), this hypothesis clearly needs to be further analyzed and verified. This will be addressed in the future work on cognitive architectures and optimization.

The requirement to actually design and develop multifunction units, relative to just doing a paper exercise or using simulation models, meant that a lot of time had to be spent purchasing platforms, tedious design details and a wide range of issues also outside the core issues of cognitive radio and multifunction radio nodes. This time spent on practical issues that are not in themselves central research tasks is the main drawback of this type of 'design paradigm' research methodology.

An important benefit of the practical approach is that the prototypes allow us to verify principles in a close-to-real system environment. The prototypes also allow us to demonstrate working principles for military operations personnel, and to trigger discussions as to what their needs and concerns are with these types of systems.

Also, with exercises that are purely simulation oriented, simplifications and assumptions are very often made, some of which may actually be real issues. As an example, a majority of cognitive radio publications neglect the administrative communication between radio nodes, merely assuming that agent decisions are somehow instantly communicated to other nodes/agents. Our approach e.g. shows that the administrative communication between cognitive radio nodes, enabling coordination between receivers and transmitters and between neighboring nodes, is a real and laborious issue and that much attention needs to be devoted to it as it is a vulnerability that may be targeted by attackers. Also it shows that the speed of the administrative communication is important, e.g. when operating in reactive mode. Clever and efficient schemes

for such administrative communication need to be included into a practical system to be used in a military context.

The prototyping approach also has given us very valuable experience with software defined radio development in general, which we may draw upon in later EW projects.

We have outlined five candidate topics for future MILKOG work:

- Improve the general functionality further to enable practical tests of the system
  - > to test system features in more realistic environments and discuss and get feedback from military operations personnel
- Cognitive architecture and COM-EW optimization
  - > to study coordination in the system and enable smarter defensive and offensive action
- Study systems of communication and jamming nodes
  - > defensive and offensive strategies in a game theoretic context
- Reactive and proactive jamming
  - > superiority through speed of action or through the quality of the decisions
- MILKOG databases
  - > to investigate wide area coordination and concerted action and to protect non-cognitive communications

In the fields of spectrum management, communications planning and electronic warfare there is clearly a need for coordination and concerted action. Our hope is that the principles outlined for MILKOG, and the experimentation tool constituted by the MILKOG nodes, will be steps contributing towards this need.

References

[1] J. Mitola III and G. Q. Maguire, Jr., "Cognitive Radio: Making Software Radios More Personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13-18, Aug.1999.

[2] "Department of the Air Force" Cognitive Jammer, RFI-PKS-001-201020-1-2010

[3] Preston Marshall, "DARPA Progress Towards Affordable, Dense, and Content Focused Tactical Edge Networks," in *MILCOM 2008* 2008.

[4] David Keil and Dina Goldin, "Indirect Interaction in Environments for Multi-Agent Systems," in *Environments for Multi-Agent Systems II* Springer Berlin / Heidelberg, 2006, pp. 68-87.

[5] VideoLan - VLC http://www.videolan.org24-2-2012 Accessed: 5-3-2012

[6] Lyrtech Inc., "SISO FlexOFDM documentation," 2010.

[7] Simon R.Saunders and Aljenadro Argòn-Zavala, "Overcomming Wideband Fading," in *Antennas and Propogation for Wireless Comminication Systems*, 2 ed John Wiley & Sons, Ltd, 2007, pp. 414-435.

[8] John G.Proakis and Dimitri G.Manolakis, *Digital Signal Processing*, 4 ed Upper Saddle River, New Jersey 07458, 2007.

[9] A. B. MacKenzie, P. Athanas, C. W. Bostian, R. M. Buehrer, L. A. DaSilva, S. W. Ellingson, Y. T. Hou, M. Hsiao, Jung-Min Park, C. Patterson, S. Raman, and C. daSilva, "Cognitive Radio and Networking Research at Virginia Tech," *Proceedings of the IEEE*, vol. 97, no. 4, pp. 660-688, Apr.2009.

[10] Lyrtech Inc., "Small Form Factor SDR Evaluation Module/Development Platform User`s Guide," 2010.

Abbreviations

AGC	Automatic Gain Control
API	Application Programming Interface
BER	Bit Error Rate
CFO	Carrier Frequency Offset
CNR	Combat Net Radio
COM	Communication
DSA	Dynamic Spectrum Access
DSP	Digital Signal Processor
EA	Electronic Attack
EAD	Electronic Attack Database
ES	Electronic Surveillance
ESD	Electronic Surveillance Database
EW	Electronic Warfare
FCC	Federal Communications Commission
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
FTP	File Transfer Protocol
GPP	General Purpose Processor
GPS	Global Positioning System
GUI	Graphical User Interface
IFFT	Inverse Fast Fourier Transform
IP	Internet Protocol
IPv4	Internet Protocol version 4
ISI	Inter-Symbol Interference
MAC	Media Access Control
MAS	Multi-Agent Systems
MILKOG	Military Cognitive Radio
OFDM	Orthogonal Frequency Division Multiplexing
NC-OFDM	Non-Continuous Orthogonal Frequency Division Multiplexing
PC	Personal Computer
PN	Pseudo-random Noise
PPCD	Primary and Prioritized COM Database
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
SFF SDR	Small Form Factor Software Defined Radio
SDR	Software Defined Radio
SoC	System on Chip
SW	Software
RF	Radio Frequency
RX	Receiver
TX	Transmitter

UDP	User Datagram Protocol
VHDL	VHSIC Hardware Description Language
VLC	Media player/streamer from VideoLan
WNAN	Wireless Network After Next

Appendix A Overview of MILKOG Administrative Messages

Table A.1 An overview of the implemented administrative command messages in the MILKOG system.

Network association:									
Discovery type message. Sent to all node numbers.	IP-address of the sender of the message	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port number of the to-node	LSB of the port number of the to-node	PING\0	(Not used)	(Not used)
The response to a received PING message.	IP-address of the sender of the message	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port number of the to-node	LSB of the port number of the to-node	PONG\0	(Not used)	(Not used)
Request from sender node to receiver node to join his logical network.	IP-address of the sender of the message	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port number of the to-node	LSB of the port number of the to-node	JOIN\0	(Not used)	(Not used)
Response to JOIN, an accepted JOIN request	IP-address of the sender of the message	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port number of the to-node	LSB of the port number of the to-node	JACK\0	(Not used)	(Not used)
Network update notification sent to all nodes in the logical network, following any change of the logical network.	IP-address of the sender of the message	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port number of the to-node	LSB of the port number of the to-node	UPDN\0		81: Total number of nodes in the network (unsigned char) 82…140: The node numbers of the nodes in the network (up to Total number of nodes)
Disconnect message sender node from network.	IP-address of the sender of the message	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port number of the to-node	LSB of the port number of the to-node	DISC\0	(Not used)	(Not used)
Spectrum proposal:									
Message explanation	Bytes 0..15	Byte 16	Byte 17	Bytes 18..33	Byte 34	Byte 35	Bytes 36..40	Bytes 41..80	Bytes 81..140
Send a spectrum proposal .	IP-address of the sender of the message	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port number of the to-node	LSB of the port number of the to-node	PROP\0		81: Number of segment values in the PROP (nmbr) 82: Number of first segment 83: Power value of first segment 84: Number of second segment 85: .. 81+2*nmbr: Power value of last segment 2*nmbr+1: Utility of solution

Local parameters exchange messages:									
Message explanation	Bytes 0..15	Byte 16	Byte 17	Bytes 18..33	Byte 34	Byte 35	Bytes 36..40	Bytes 41..80	Bytes 81..140
Local communication	IP-address	MSB of port	LSB of port	IP to-address	MSB of the	LSB of the port	COMP\0	(Not used)	81..84: DSA start frequency 85..88: DSA stop frequency

parameters	of the sender of the messag e	number of the sender	number of the sender		port numbe r of the to-node	number of the to-node			89..92: Number of segments 93..96: Segment width
Local electronic attack parameters	IP-address of the sender of the messag e	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port numbe r of the to-node	LSB of the port number of the to-node	EAPA\0	(Not used)	81..84: EA start frequency interval 0 85..88: EA start frequency interval 1 89..92: EA start frequency interval 2 93..96: EA stop frequency interval 0 97..100: EA stop frequency interval 1 101..104: EA stop frequency interval 2 105: Jamming type 106..108: Jamming interval 0..2 active (Active=JAMON, non-active=JAMOFF)
Local electronic surveillance parameters	IP-address of the sender of the messag e	MSB of port number of the sender	LSB of port number of the sender	IP to-address	MSB of the port numbe r of the to-node	LSB of the port number of the to-node	ESPA\0	(Not used)	81..84: ES start frequency 85..88: ES stop frequency 89..92: ES number of bins 93..96: ES bin width
Time synchronization:									
Message explanation	Bytes 0..15	Byte 16	Byte 17	Bytes 18..33	Byte 34	Byte 35	Bytes 36..40	Bytes 41..80	Bytes 81..140
Send the node's current time (to other nodes in the current network)	IP-addres s of the sender of the messa ge	MSB of port number of the sender	LSB of port numbe r of the sender	IP to-address	MSB of the port numbe r of the to-node	LSB of the port number of the to-node	SYNC\0	(Not used)	81..88: Time sample (DWORD)

Appendix B The Formation of Local Networks

The network formation procedure in MILKOG is a radio-operator initiated procedure:

1) The radio operator which takes the initiative, starts the 'Radio Network Configuration' dialogue in the MILKOG agent software.
2) The radio operator selects the 'Detect available MILKOG nodes'. This initiates a sequence of 'PING' messages through the coordination channel. MILKOG agents that receive the 'PING', confirms by responding with a 'PONG'. The dialogue will show the numbers of the radio nodes that have sent confirmations.
3) Next, the radio operator highlights the radio nodes with which to form a network, then selects 'Try connecting MILKOG nodes'. This initiates a sequence of 'JOIN' messages through the coordination channel. Nodes that accept the invitation, respond with 'JACK', and update their LOCALNET structure. The initiating node also updates its LOCALNET structure as it receives the 'JACK' messages.
   Here, it is assumed that LOCALNET is a traditional Combat Net Radio all-hear-all type network. Hence, if node 1 asks node 2 and 3 to join the LOCALNET, we also want node 2 and 3 to understand that they belong to the same network. This, as well as other dynamic changes of the network, is facilitated through:
4) Each time an agent makes a change to its LOCALNET, it sends a network update message 'UPDN' to all its previous state and current state LOCALNET peers. The 'UPDN' contains a list of all the node numbers in the network (with the exception of the initiator). The 'UPDN' is handled in the following way:
   a. The receiving agent checks the 'UPDN' node numbers relative to his LOCALNET structure.
   b. If all the 'UPDN' numbers are in LOCALNET, do nothing.
   c. If a number is found that is not in LOCALNET, then JOIN is sent to this node. As above, nodes that accept the invitation, respond with 'JACK', and update their LOCALNET structure. The initiating node also updates its LOCALNET structure as it receives the 'JACK' messages.
5) Disconnection of nodes from the network is to be facilitated through the 'DISC' message (reserved, not implemented), with a list of node numbers to disconnect included in the message.
6) Each time a change is done to LOCALNET, the change is visualized in the 'Communications Monitor' dialogue window.

Appendix C Hardware Description of SFF SDR

All data and the figure in this subsection are obtained from [10]. Figure C.1 illustrates the module structure of the SFF SDR, consisting of the Digital processing module, the Data conversion module and the RF module.

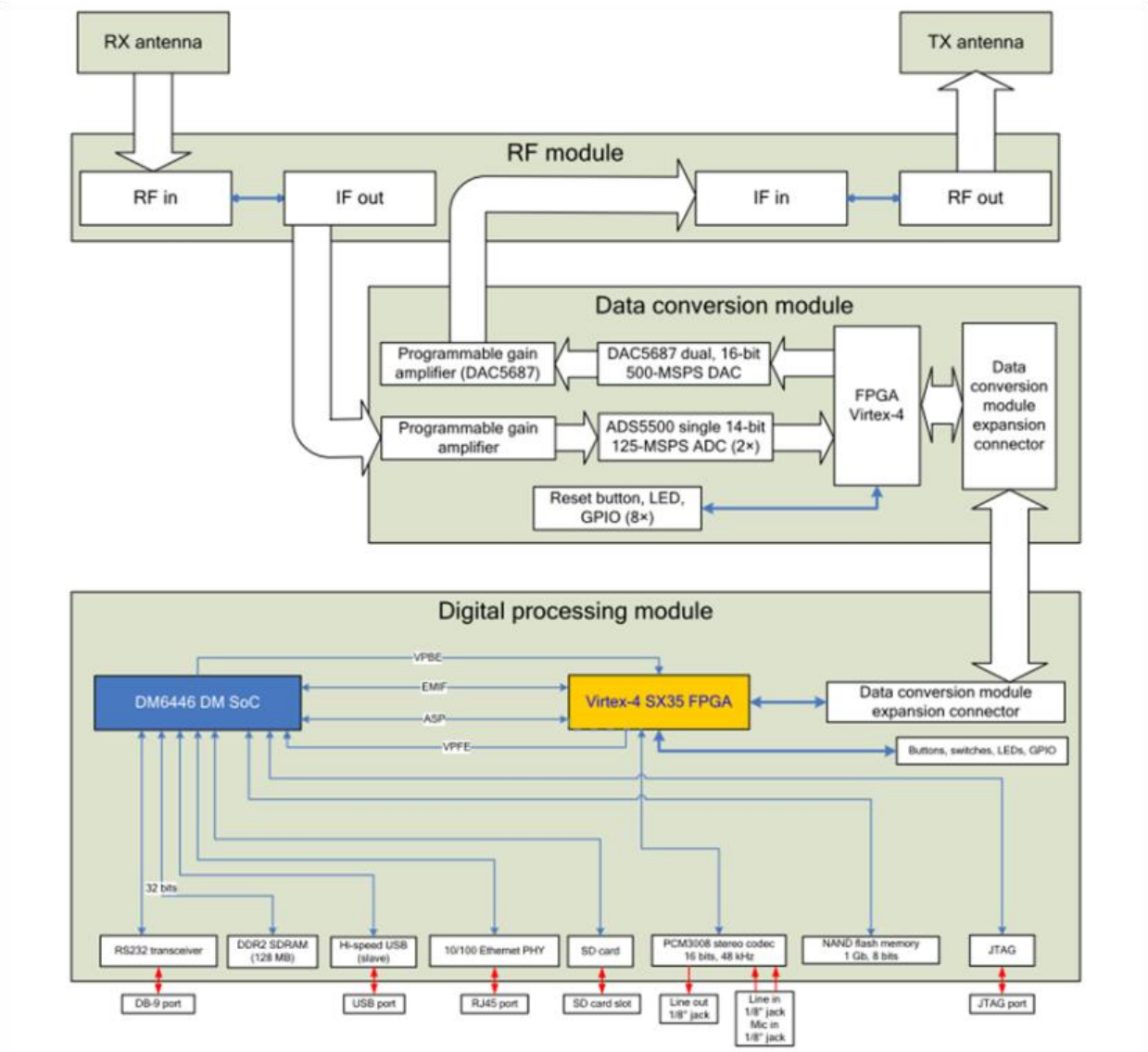


Figure C.1 Hardware block diagram SFF SDR platform

The Digital processing module consists of two main components, one DSP System on Chip (SoC) from Texas Instruments (TMS320DM6446) and a FPGA from Xilinx (Virtex-4 SX35). The SoC contain one 594-MHz DSP (C64x+) and one 297-MHz General Purpose Processor (GPP) (ARM926EJ-S). Communication between the DSP and FPGA is handled by a protocol called Video Processing Subsystem (VPSS), containing a Video Processing Front and Back End (VPFE/VPBE). VPFE serves as an input interface to the DSP, with a depth of 16 bit running at 75MHz. VPBE has an equal depth of 16 bit, but runs at half the rate 37,5MHz as an output interface from the DSP.

The Data conversion module converts the digital processed signal to an analogue representation and vice versa. A dual-channel 16 bit 500MSPS DAC from Texas Instruments (DAC5687) converts the digital signal to an analogue representation, and two 14 bit 125MSPS ADC from Texas Instruments (ADS5500) performs the reversed operation. The Data conversion module is also equipped with two external clock inputs (ADC and DAC), one 10 MHz onboard reference clock and a reference clock input for synchronization. In order to control the ADC and DAC a FPGA from Xilinx (Virtex XC4VLX25) is included. To provide the ADC, DAC and FPGA with their desired clock frequencies, a Phase Locked Loop (PLL) is included.

The RF module works as a half-duplex transceiver, with a RF range of 250 MHz to 1 GHz. It is a tunable RF module where the RX section is composed of a three stage superheterodyne receiver, with a final Intermediate Frequency (IF) at 30MHz and a selectable signal bandwidth of 5MHz or 20MHz. The TX section of the RF module is a 2 band (250-500MHz and 500-1000MHz) quadrature mixer. All the frequencies are kept stable with a PLL.