Author: Sidney Cadot	Date of issue: 2005-Oct-06 Kind of issue: Public	Scope: Project Documentation Doc.nr.: LOFAR-SHM-ADD-001
	Status: Initial Release Revision nr.: 1.0.2	File:

LOFAR System Health Management ADD

Sidney Cadot - Science [&] Technology BV

Verified:			
Name	Signature	Date	Rev.nr.
André Bos (S&T)	....	2005-Oct-06	1.0.2
Tilman Müller (Ordina)	....	2005-Oct-06	1.0.2

Accepted:		
Work Package Manager	System Engineering Manager	Program Manager
.... *date:*	.... *date:*	.... *date:*

Distribution list:

Group:	For Information:

Document revision:

Revision	Date	Section	Page(s)	Modification

Contents

List of Abbreviations

AC	Alternating Current
ACM	Antenna Correlation Matrix
ADC	Analog-to-Digital Converter
ADD	Architectural Design Document
ASP	Active Server Pages
AP	Antenna Processor
CDR	Critical Design Review
CEP	Central Processing
CCU	Central Control Unit
DB	Database
DC	Direct Current
FD	Fault Detection
FI	Fault Isolation (or Fault Identification)
FTS	First Test Site
GCF	Generic Control Framework
HBA	High Band Antenna (a.k.a. HFA)
HFA	High Frequency Antenna (a.k.a. HBA)
ITS	Initial Test Site
LAN	Local Area Network
LBA	Low Band Antenna (a.k.a. LFA)
LCU	Local Control Unit
LFA	Low Frequency Antenna (a.k.a. LBA)
LNA	Low-Noise Amplifier
LOFAR	Low-Frequency Array
MAC	Monitoring And Control
MIS	MAC Information Server
MS	Model Synthesis
ODBC	Open Database Connectivity
OTDB	Observation Tree Database
PHP	PHP Hypertext Processor
RCOF	Root Cause of Failure
RCU	Receiver Unit
RSP	Remote Station Processor
SAS	Specification, Administration, and Scheduling
SHM	System Health Management
SI	Sensor Ingestion
TBD	To Be Determined
WAN	Wide Area Network
WS	Web Services

1 Introduction

This document is the architectural design document (ADD) of the LOFAR System Health Management subsystem.

The SHM design discussed in this document reflects the status at the time of writing (September/October 2005). While it is expected that the high-level design and functional decomposition as discussed in this document (depicted in Figure 1) is very close to the operational SHM module for LOFAR, many details, including lower-level design decisions, still need to be worked out.

In particular, many design decisions are driven by continuing interaction with the MAC (Monitoring and Control) and SAS (Specification, Administration, and Scheduling) groups. Also, many design details of related LOFAR subsystems are not yet fixed. For example, the precise definition of station hardware is not fully known at the time of writing.

These facts necessitate a flexible approach to the SHM development process. Fortunately, because of the model-based SHM approach, incorporation and adaption to new circumstances outside the realm of SHM is relatively easy (though by no means completely trivial).

1.1 Introducing the LOFAR System Health Management Subsystem

The percentage of time during which the LOFAR system is effectively operational i.e., the *system uptime* is an important issue that warrants a lot of attention. The reason for this is that due to the complexity of the LOFAR system and the harsh operating environment, it is almost certain that at any moment in time many of LOFAR's components (antennas, amplifiers, network links, computing nodes, ...) will be non-functional. Within reasonable bounds, this should not impact the usability of LOFAR for performing useful scientific measurements; rather, the system performance should *gracefully degrade* with each failing component.

Any faulty component may affect the quality of the measurements in a negative way, and may also jeopardize the operational capabilities of the LOFAR network. The objective of the System Health Management (SHM) module of the LOFAR system is to support the efforts to maximize the system uptime. The main functions of the module will be the early detection of system failure, the accurate identification of failing components, and the support for remedial actions.

As daily or weekly on-site inspections cannot be performed in an economical viable way (at least for the remote stations), the System Health Management subsystem will primarily be guided by the data that is generated by the LOFAR system in an automated fashion. This data consist of both the scientific data (generated by the antennas) and the 'housekeeping' data of the equipment that controls the sensor network. Deviations in system health will be reflected in sensor data that deviate from the normative measurements. These deviations are called *symptoms* and are used by the SHM module to detect system failure and identify the responsible system component.

The design of the SHM-module is driven by a desire to maximize diagnostic accuracy, and to minimize the effort required to construct and maintain the diagnostic knowledge to map symptoms to diagnoses. In order to keep the construction and maintenance effort at a reasonable level, the SHM module will make heavy use of model-based principles. A model-based design of a SHM-module differs from a rule-based design (the preferred technique of the previous generation of SHM systems) in the way the knowledge to map symptoms to an identification of failing system components is defined and captured. In a rule-based design, the symptom/diagnosis relation is given in an explicit way and is often defined manually.

The designer of a rule-based SHM must track how the effects of a system failure are propagated through the system components. It is clear that any change in the design or implementation of the system under diagnosis means that the a re-examination of the diagnostic knowledge as captured in the rules has to be performed.

Model-based SHM systems avoid this error-prone definition of diagnostic knowledge as they don't require that the symptom-to-diagnosis relation is defined explicitly. Instead, this type of diagnostic knowledge is given in an implicit fashion by a model of normative system behavior. A model-based SHM system is equipped with a *reasoner* that performs the failure effect propagation automatically. This greatly simplifies the definition of diagnostic knowledge. Furthermore, the structure of the model can closely follow the physical component structure of the system under diagnosis. This means that in case the design of the system under diagnosis changes, it is much easier to update the diagnostic knowledge contained in the model. In this way, the maintenance costs are greatly reduced.

1.2 A birds-eye view of LOFAR SHM

Figure 1 shows a bird's-eye overview of the LOFAR System Health Management subsystem and its software environment. From the LOFAR point-of-view, SHM is one of the so-called 'central systems'. Together with the 'Specification, Administration, and Scheduling' (SAS) and 'Monitoring and Control' (MAC) components, SHM implements the software control infrastructure of LOFAR.

It is also quite valid to view SHM as a 'sub-functionality' of MAC that enables MAC to detect and deal with failing components. Indeed, SHM's main channel of interaction is with MAC, via the MAC Information Server (MIS), from which SHM obtains structural and status information, and to which it requests reconfigurations and reports back diagnoses. The one-way interaction with the Observation Tree Database (OTDB) is mainly in place to lighten the load on MAC (specifically, the PVSS system on which MAC is built); all data that SHM receives via the OTDB could, in principle, also be retrieved directly from MAC.

From the perspective of SHM, there are three main types of interaction with the outside world (MAC and the OTDB):

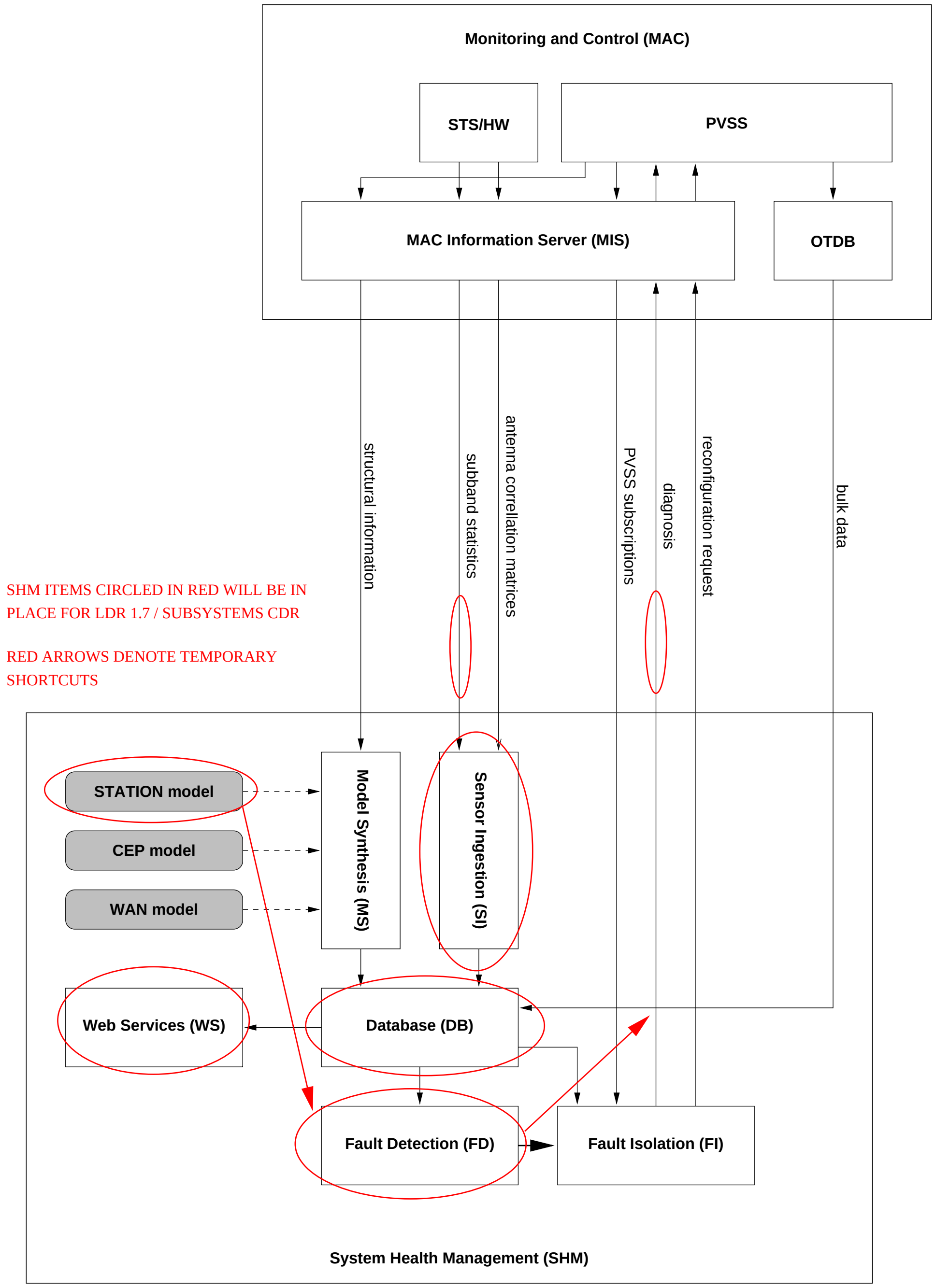

Figure 1: SHM and its software environment

- incoming status information on LOFAR components: bulk data (via OTDB), PVSS subscriptions (via MIS; used sparingly in order to lighten the load on MAC), antenna subband statistics and correlation matrices (via MIS).
- incoming structural information on LOFAR components; i.e., which versions of which hardware are installed, and in what topology? This data is available in PVSS, and SHM can (or rather, will be able to) access it through the MIS.
- outgoing reconfiguration requests and diagnoses: SHM uses all the structural and status information it sees to detect issues relating to system health. In case a problem is suspected, SHM can ask for a reconfiguration of a certain component (via a 'reconfiguration request') and, when enough confidence is reached to warrant a diagnosis, SHM can issue a 'diagnosis' message about a certain component, which will be delivered to the MIS.

The interaction of SHM with the MIS and OTDB components is described in Section 2.

SHM consists of a handful of closely interacting sub-components. Section 3 describes their function, and the way in which they work together to provide the SHM functionality.

LOFAR SHM will be *model-based*. Models of LOFAR sub-systems are therefore a crucial part of the diagnostic infrastructure. Section 4 describes the different models that together will describe the entire LOFAR system.

Finally, Appendix A shows a listing of the FTS-1 model that was developed for the CDR planned for October 2005. This should convey some flavor of what an SHM model looks like.

1.3 The diagnostic loop

The components that were introduced in the preceding section work together to provide the health management infrastructure for the LOFAR system. It is instructive to describe the nominal mode of operation of the diagnostic system when LOFAR is operational.

The LOFAR System Health Management subsystem is built around a database that holds status-data and sensory information as collected from LOFAR components. The database is continuously filled with status and sensor updates, such as OTDB data (bulk status information), PVSS data (incidental status information), antenna subband statistics and correlation data (which provide low-level data indicative of station hardware health), and more. All these data sources together provide information from which the health state of LOFAR must be deduced.

Apart from the current system status as maintained in the database, SHM also maintains a behavioral model of subsystems, which is synthesized from behavioral models of sub-components and up-to-date information on the actual system layout as maintained by MAC. Model Synthesis is responsible for the maintenance of the up-to-date models.

With the availability of the system model (which describes how the system *should* behave) and the status information (which describes how the system actually *does* behave), it is possible to detect problems as they manifest themselves, and (when enough sensory information is available) to pinpoint the Root-Cause-Of-Failure (RCOF). The latter step (RCOF identification) is often performed by combining sensory information gathered when the system is in different configurations. In fact, SHM will have the ability to suggest system reconfigurations through the 'reconfiguration request' mechanism, which is specifically intended to aid the process of RCOF identification.

Once a diagnostic conclusion has been reached, SHM will notify MAC (via the MIS). A diagnosis consists of a *component*, a *diagnosis*, and a *confidence level*. For simplicity of handling the diagnosis at the receiving end (the MIS), the diagnosis can only be 'faulty' or 'healthy'; more information is available to aid a (human) operator, however.

The MAC layer is responsible for handling incoming diagnoses in an appropriate way as defined by the operator. MAC can decide to ignore the diagnosis (in this case, the diagnosis is logged which may lead a human operator to investigate the issue), or to mark a component as 'defect' or 'suspect'[1].

The diagnostic loop is closed when SHM is notified (through a PVSS subscription or OTDB update) of the action taken by MAC in response to receiving a diagnosis, e.g., powering off a faulty component, or marking a component as 'defect'. SHM will take such status information into account when considering the behavior of the subsystem/component later on. Incorporation of the status information by the SHM system will, in general, lead to more accurate diagnoses.

[1] Please note the terminology: SHM diagnoses a component as *FAULTY* which may lead to a resource state of *DEFECT* as registered by MAC. There is no deep meaning to this.

2 The SHM Environment

LOFAR SHM is embedded in an extensive software environment that comprises the LOFAR 'control' infrastructure. To perform its work, SHM communicates with MAC (through the MIS service that runs on LCUs and CCUs) and with SAS's Observation Tree Database (OTDB). This section describes the interaction with these external software systems.

2.1 The MAC Information Server (MIS)

MAC Information Servers run on LCU and CCU nodes and are accessed via TCP port 27009. Communication with the MIS is performed using the existing Generic Control Framework (GCF) packet-based protocol infrastructure[2].

A MIS-specific GCF protocol has been defined that covers seven types of SHM/MAC interaction:

generic messages ('ping' and 'identification'); these messages are intended to test the basic communication infrastructure (Is the link up? Who is on the other side?) and can also be used for the purpose of sending 'keep-alive' messages, timing peer-to-peer latency, and checking clock synchronization.

diagnosis messages that flow from SHM to MAC provide information to MAC about the failure of specific components. Components can be classified as 'healthy' or 'faulty'; each diagnosis is annotated with a 'confidence level'. It is the responsibility of the receiver (MAC) to act (or not act) upon reception of a diagnosis.

reconfiguration messages are requests made by SHM to MAC to put a certain piece of equipment in a certain state. An example could be a request to put a specific RCU in 'high band 2' mode to distinguish between an LBA and an RCU failure [Note: reconfiguration is not implemented for the October 2005 FTS-1 test setup].

LOFAR structure messages allow SHM to interrogate MAC about the structure/topology of (parts of) LOFAR, such as a specific Remote Station. This is necessary for the process of Model Synthesis as performed by SHM. [Note: model synthesis is not implemented for the October 2005 FTS-1 test setup; instead, a static, hand-crafted model of the system is used for modeling].

PVSS subscriptions provide SHM with the opportunity to interrogate PVSS about detailed status and settings of LOFAR components. [Note: although this *is* currently implemented, the October 2005 FTS-1 test setup does not currently depend on this].

Subband Statistics messages enable SHM to inquire the subband-statistics that are generated once every second on each station. Subband Statistics provide a rich source of diagnostic information;

[2] This protocol infrastructure is used extensively within MAC to communicate between its components, and was chosen for MAC/SHM communication to minimize the implementation effort on the MAC side.

the current FTS-1 test setup relies heavily on the classification of subband data as input to its diagnostic model.

Antenna Correlation Matrix messages provide on-demand subband-dependent ACMs from remote stations to SHM. This type of information is not yet available from the RSP driver (October 2005) and can therefore not be requested routinely.

2.2 The Observation Tree Database (OTDB)

The Observation Tree Database (OTDB) is used for managing Observation Trees that represent the logical and physical resources needed to perform an observation. Of particular interest to System Health management are the so-called 'PIC Trees' that describe the settings and status of physical components.

The OTDB is the primary mechanism for configuration and status information (on a per-component basis) from MAC back to SHM. The OTDB is used to replicate and archive large amounts of status and settings data. SHM can request the status of components residing below a certain point in the LOFAR system hierarchy at a certain point in time, and request updates, i.e., status changes between two given points in time. In this way, SHM will be able to construct a rather complete picture of observable component settings, component status readouts, and sensor values, which can subsequently be used as input for SHM in the 'Fault Detection' and 'Fault Identification' modules.

[NOTE: OTDB feedback is not used in the FTS-1 test setup. Since FTS-1 system health management is performed exclusively on the basis of subband statistics, which include the relevant RCU settings when read via the MIS, the OTDB/SHM link is not needed there.]

3 Inside SHM: Functional Components

This section describes the design of the functional components that make up the System Health Management subsystem of LOFAR. It is recommended that the reader keep Figure 1 in mind to understand in broad terms what kind of functionality is to be offered by each of the components discussed below.

3.1 SHM Sensor Ingestion (SI)

The SHM Sensor Ingestion component is comprised of processes that periodically ingest sensor signals. In the case of LOFAR, Sensor Ingestion components will be implemented for the 'subband statistics' and 'antenna correlation matrix' data that are produced at each station.

There will also be much simpler ingestions of sensor data, e.g. temperature sensors. These data points will generally be communicated via the OTDB (for 'bulk' data) or via the MIS (for incidental sensor readouts during the Fault Identification stage). These simple types of scalar sensor data will simply be classified according to range and stored in the SHM-DB; no complicated processing is necessary to handle that data properly.

For all incoming data, a data reduction step is performed where the data is classified in one of a small number of classes. This step is essential to bridge the gap between the highly domain-specific incoming signals (in the case of LOFAR: antenna data and sensor input) and the generic diagnostic search engine, which is a constraint solver able to handle finite domain data only. Below, we describe how this is done for the Subband Statistics and Antenna Correlation Matrices.

3.1.1 Subband Statistics Ingestion

The RSP Antenna Processors calculate so-called 'subband statistics' which provide a measure of the electro-magnetic power in the subbands averaged over the last second of measurement. This data is made available to SHM via the RSP driver and the MIS.

Periodically (in the FTS-1 test system: once every 5 seconds; in the operational system: approximately once every 10 minutes, TBD), SHM queries a station in order to obtain a subband spectrum for each of the RCUs residing in the station. Figure 4 shows a typical FTS-1 daytime test signal. Figures 5, 6, **??** show signals that show some sort of anomaly compared to the nominal daytime signal.

In order to make a robust characterization of nominal and non-nominal detector signals, several tests were conducted on FTS-1. A preliminary subband statistics characterization algorithm was devised that is based on two subband spectrum characteristics: the *median subband power* (which is a good measure for the signal noise floor) and the *peak subband power divided by the median subband power* which measures 'peak' above the background noise that usually come from a radio source in the vicinity of the station.

Figure 2 shows the median subband power as it varies over a quiet 48-hour weekend period for 8 nominally working antennas. Please note that the power as measured in the subband statistics (the vertical axis in Figures 2 to 7) are the values as read from the RSP board. This value is linear to the received power per subband, although no calibration factor has been determined that would enable conversion to SI units.

The median subband power can be seen to be relatively stable (few outliers; spread of roughly 10% of signal value).

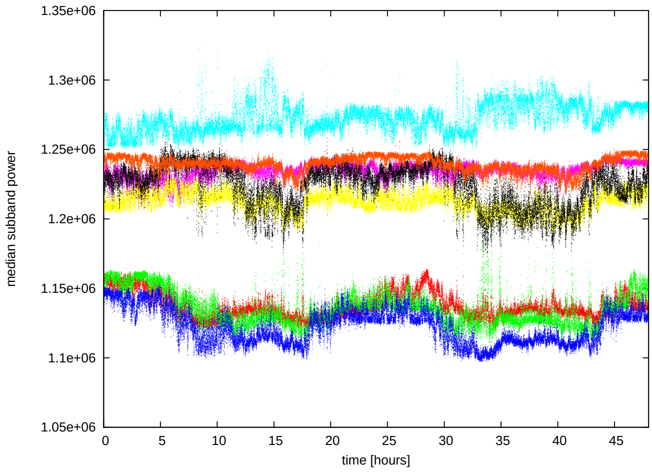

Figure 2: Mean subband power over 48 hour period

Figure 3 shows the 'peak/median' subband power as it varies over the same period. Note that the y-axis is now logarithmic, in order to accommodate the large dynamic range over which this signal varies.

Based on this 'nominal behavior' dataset and another dataset consisting of subband statistics taken when the system was purposefully subjected to different kinds of hardware failures, the following classification algorithm was devised:

ZEROES - This classification is given to a subband spectrum when all but the DC component (first subband) in the spectrum consist of zero values. This happens when there is no transfer from digital data from the RCU to the RSP board. The RSP board will see a constant (DC) signal and faithfully show the subband statistics that go with it.

This happens in case a RCU is not powered, which in turn can happen in case a RCU makes a bad connection with the back-plane, or in case all RCUs are unpowered. In the former case, both the RCU that makes a bad contact and its sibling connected to the same Antenna Processor show a 'ZEROES'-type subband statistics spectrum.

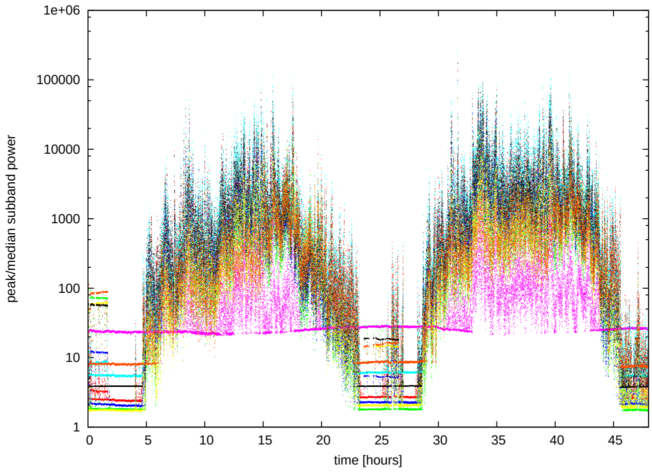

Figure 3: Peak/Mean subband power over 48 hour period

ABNORMAL - This classification is given to a subband spectrum if the signal detected is not 'ZEROES', and when the median power is less than $8 \cdot 10^5$ or exceeds 10^7, or peak power is less than 10^6 or exceeds 10^{13}. These boundaries establish a rather wide window of values that are deemed 'reasonable'; if a subband spectrum has median/peak power outside these ranges, a variety of things may be wrong. See Figure 7 for an example subband spectrum that is classified like this.

For example, 'ABNORMAL' conditions have been observed when subband statistics were captured during RCU power removal (indicating transient behaviors between two otherwise classifiable behavioral modes), and when a noise source was connected to the RCU input.

NO SIGNAL - This classification is given if the signal detected is not 'ZEROES' or 'ABNORMAL', and if the peak/median ratio is less than 1.5. This situation is encountered when the connection between the LBA antenna and the RCU is severed. See Figure 5 for an example subband spectrum that is classified like this.

LOW SIGNAL - This classification is given if the signal detected is not 'ZEROES' or 'ABNORMAL' or 'NO SIGNAL', and if the peak/median ratio is less than 10. This situation is encountered at night-time (when the radio frequency environment is quiet) but also at day-time, in case the Low-Noise Amplifier is not powered. See Figure 6 for an example subband spectrum that is classified like this.

MEDIUM SIGNAL - This classification is given if the signal detected is not 'ZEROES' or 'ABNORMAL' or 'NO SIGNAL' or 'LOW SIGNAL', and if the peak/median ratio is less than 50. This classification was introduced as an intermediate between a 'LOW SIGNAL' and ' 'HIGH SIGNAL' classification, and is used in the model as an 'ambiguous' situation, i.e., both a situation where one would expect a 'low' signal or a situation where one would expect a 'high' signal could also give rise to a 'medium' signal classification: the signal is *ambiguous*.

HIGH SIGNAL - This classification is given if the signal detected is not 'ZEROES' or 'ABNORMAL' or 'NO SIGNAL' or 'LOW SIGNAL' or 'MEDIUM SIGNAL'. This is the usual classification at day-time. See Figure 4 for an example subband spectrum that is classified like this.

The subband spectrum classifications described above are used as inputs to the diagnostic health model used for Fault Detection and Fault Identification. They can be found in the FTS-1 model reproduced in Appendix A (type 'ReceiverUnitOutputSignal').

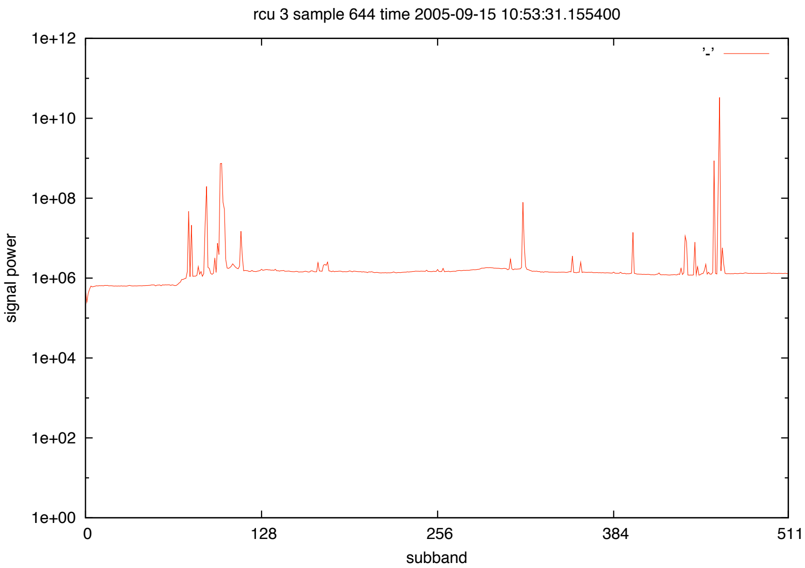

Figure 4: Typical 'High Signal' subband statistics

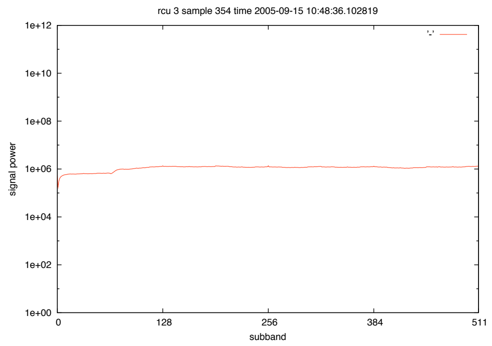

Figure 5: Typical 'No Signal' subband statistics

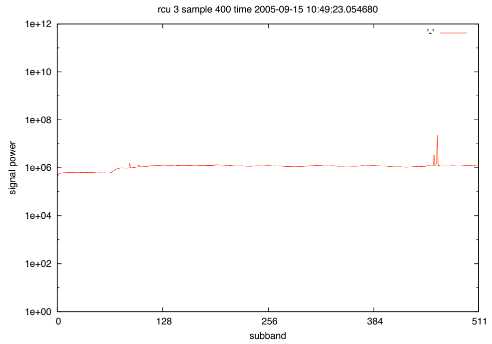

Figure 6: Typical 'Low Signal' subband statistics

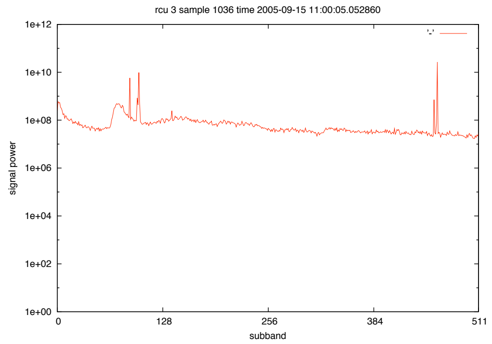

Figure 7: Example 'Abnormal' subband statistics

It is noted here that the method described above has been validated for FTS-1 data only. More experience with antenna data from different sites will need to become available to assess if this method will be robust enough to use for all stations.

One specific point of critique on the method described above is that it depends on absolute boundary values, which have been determined in a rather arbitrary fashion. This is less of a problem than it may seem because of the inherent robustness against outliers, and the definition of the 'MEDIUM_SIGNAL' subband statistics signal level, which introduces resistance against wrong diagnoses.

However, a signal classification method that also incorporates Antenna Correlation Matrix data (see below) could probably provide a significantly more robust classification algorithm. Experience with more real-life antenna data, coming from geographically different locations, will help to determine if such refinements are in fact necessary or not.

3.1.2 Antenna Correlation Matrix Ingestion

Preliminary analysis (based on ITS data) suggests that antenna correlation matrices can be very useful especially for the 'fault detection' phase of diagnosis: faulty antennas tend to stick out like a sore thumb when a graphical representation of the ACM is shown. Some effort has been expended to develop a 'partitioning' algorithm that tries to rearrange antennas in way way that groups together antennas that behave 'similarly'.

Figure 8 shows a 'raw' Antenna Correlation Matrix as measured at the Initial Test Site (ITS). Figure 9 shows the same data, re-order to show the grouping of different antennas that behave similarly.

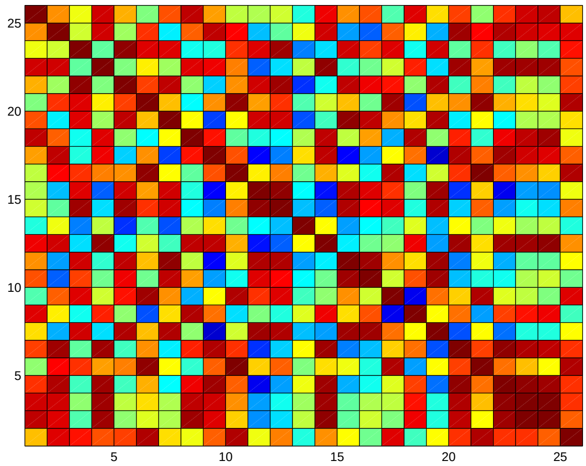

Figure 8: ITS Antenna Correlation matrix (ordered by antenna number)

This component is not yet implemented the FTS-1 test system. At the time of writing, no infrastructure was in place to routinely get antenna correlation data and to check how failures manifest themselves in

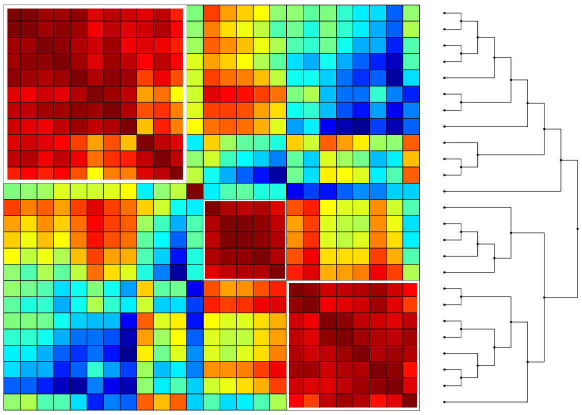

Figure 9: ITS Antenna Correlation matrix (reordered; similar antennas clustered)

such data.

3.2 SHM Database (DB)

The System Health Management database stores all SHM-related data and supplies it to all SHM components. It is used to store incoming sensory information (scalar values coming in via the OTDB or MIS), subband statistics, antenna correlation matrices, behavioral component models, structural component data as obtained by querying MIS, and combined structural/behavioral models as synthesized by the Model Synthesis (MS) component of SHM.

The most important data residing in the SHM database is contained in the *Event* table. Here, timestamped value changes for all observables in LOFAR (sensor data, settings, subband statistics classifications, etc.) are recorded. This table allows the FD and FI modules to establish the status (value) of a given component's observable variables at a certain time in the past, which (combined with the model) provides all necessary input to the diagnostic process.

One important function provided by the SHM-DB's 'Event' table is that it provides signal time ordering. Due to transport delays, it may happen that changes of values of observable quantities arrive at SHM with a lag (perhaps a few seconds). In this way, it could happen, for example, that the *effect* of a certain reconfiguration is observed prior to its *cause*.

To preserve the important property of causality, the SHM-DB functions to buffer all incoming observations and impose a time-ordering on them. The diagnostic processes (FD and FI modules) will slightly 'lag behind' the present (perhaps 10—30 seconds, TBD) in order to make sure that a consistent model of the instant for which diagnosis is done is available.

In the final system, analysis of archived data in the SHM database will be used to analyze and optimize system performance. The SHM database can always be accessed through an ODBC interface (for data analysis programs that support that). Alternatively, it is of course always possible to perform command line queries on the data of interest, write that data to a local file, and then proceed analysis on the locally stored data.

3.3 SHM Model Synthesis (MS)

The Model Synthesis component serves to synthesize up-to-date behavioral models of LOFAR subsystems, that correspond to the actual hardware available in the system.

While SHM provides behavioral models of all components of interest from a system health perspective, the actual structure of LOFAR systems that will be diagnosed is stored within MAC. For example: remote stations could differ in the exact number of high-band/low-band antennas that are connected. Also, it is expected that during nominal operation of LOFAR, new hardware revisions of certain hardware components may become available, and therefore SHM must be able to cope with different configurations of the same basic system.

SHM Model Synthesis brings together the structural LOFAR hierarchy (as stored in MAC's PVSS database) and the behavioral models that are provided by SHM. The end result of this operation are Lydia models that correspond (both structurally and behaviorally) to the actual setup that is present at any given time.

In order to obtain structural information from MAC, the SHM Model Synthesis component will talk to the MAC Information server to inquire about the layout of a station or other modelled LOFAR subsystem. The details of this protocol have not been worked out yet; for LDR 1.7, a simple hand-synthesized model of FTS-1 was constructed in order to bypass the Model Synthesis step.

3.4 SHM Fault Detection (FD)

The Fault Detection subsystem continuously monitors LOFAR subsystems for which a model is available, to decide if the status information coming from the components that are being monitored arouse any suspicion of a System Health Management related issue. If this is the case, the monitored system is reported to the 'Fault Isolation' module (described in the next section) that is responsible for isolating the problem cause by means of reconfiguration.

It is expected that the Fault Detection stage will be performed on each LOFAR station with the same frequency with which the Subband Statistics will be queried (about once every ten minutes per station).

In principle, the Fault Detection step can be skipped; Fault Isolation should itself be capable of concluding that a system-under-consideration is, in fact, healthy. However, the process of Fault Isolation can be computationally expensive, and may involve reconfiguration request (from SHM to MAC) that interfere with the use of components to produce scientifically valid data.

For these reasons, the Fault Detection phase was introduced, which is completely non-intrusive and computationally cheap. Either Fault Detection or a human operator can flag a certain LOFAR subsystem to be subjected to the Fault Identification phase.

It is noted that, in the FTS-1 demo-system, there is no real distinction between Fault Detection and Fault Isolation. Effectively, the Fault Detection step is used both for detection and identification of problems, using a Lydia model. This will eventually evolve into a distinction where FD uses cheap heuristics, and FI uses a Lydia-based diagnostic finite-domain solver.

3.5 SHM Fault Isolation (FI)

The Fault Isolation component takes over the diagnostic process when SHM-FD signals that a problem has been detected. SHM-FI will suggest *reconfigurations* which (when accepted by MAC) should yield information that helps to isolate a fault. Ideally, this process should result in the identification of the Root Cause Of Failure (RCOF) in as few steps as possible.

In the current, basic model of the FTS-1 system, this step is superfluous. All model-based diagnostics for this test system is performed by SHM-FD, described above.

When more elaborate models are used, this step will be implemented. Currently, a static ‘diagnostic procedure’ is foreseen: as soon as a problem is detected in LOFAR subsystem for which a model is available (e.g., a Remote Station), SHM will request a number of pre-defined reconfigurations that will be optimized for RCOF identification.

Fault Isolation will rely on application of Lydia-based models, and a diagnostic finite domain solver. The solver will generate diagnoses, which are basically explanations for observed external behavior, consistent with the system model. The introduction of Section 4 describes this in more detail.

3.6 SHM Web Services (WS)

System Health Management provides a limited Web-based interface, primarily intended to show the current status of the diagnostic components in the system. Web services provide ‘quick-look’ access to relevant diagnostic sensor information (e.g., current subband statistics for stations), that allow an operator to quickly assess the working of a station.

Furthermore, Web Services will be able to present an ‘explanation’ for a diagnosis, to support an operator while verifying if it was sensible. Lastly, a limited amount of control can be implemented via WS to tune System Health Management, for example in order to ’tune’ SHM parameters (failure probabilities, etc.).

Web Services are implemented using Apache-2 and 'mod-python', a Python-based templating system similar to ASP and PHP.

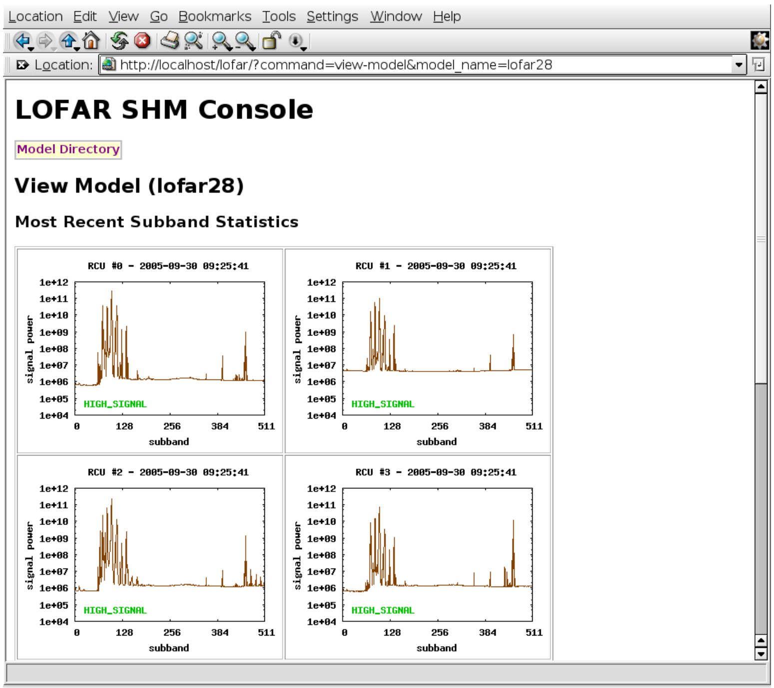

Figure 10: View of the healthy FTS-1 station via web-server

4 SHM Models

The LOFAR SHM system will do model-based diagnostics. In short, this means that a model of (parts of) LOFAR must be available that allows the core diagnostics algorithm to distinguish anomalous from nominal behavior. Such a model describes the system under consideration by expressing relations between externally observable variables, which can be inputs (e.g., RCU settings) or outputs (e.g., subband spectra).

The behavioral relationships between inputs and output (i.e., the *system behavior*) is parameterized by a class of variables that are specifically marked as *health variables*. In contrast to input and output variables, health variables are usually not directly observable. All health variables taken together form the *health vector* of the system.

The challenge of model-based diagnostics is this:

Given a system model (parameterized by a health vector that is not directly observable) and a number of observations of actual system behavior (modelled by assignments to input and output variables), find health vectors that are consistent with all observations.

In general, (too) many health vectors are possible, so an ordering on health vectors must be defined. Some health vectors are more probable than others, and we are looking for *parsimony of explanation*; for example, when both the assumption of a loose cable, and the assumption of a loose cable, combined with a broken low-noise amplifier, could explain a given set of observations, the former diagnosis is preferred.

For LOFAR, we use the Lydia modelling language that is capable of describing a great variety of system models. The Lydia tool-set contains diagnostic finite domain solvers that are able to 'reason' over a model and observations. By assigning probabilities to health variable states, these solvers list health states in descending order of probability (i.e., the most probable health states that explain a given set of observations come first).

4.1 Station Model

The LOFAR Station Model describes the behavior of LOFAR stations. A LOFAR station consists of the following components:

1. Up to 96 Low-Band Antennas (LBAs). These are organized as pyramid-shaped 'dual dipoles', each having two antennas and two Low-Noise Amplifiers (LNAs - one for each polarization), the power for which is delivered to the dual-dipole by coax cable. The amplified signal is returned to the central station facility via the same cable.
2. Up to 96 High-Band Antennas (HBAs). These are organized in 4x4 'tiles', for which the directional

sensitivity can be optimized via configurable delay lines. At the time of writing, details on the HBA components are still a bit sketchy.

3. 96 Receiver Units (RCUs), each of which is connected to both an LBA and a HBA. RCUs filter, amplify, and digitize the incoming analog signal.

4. 24 Remote Station Processors (RSPs), each of which connects to 4 RCUs. The RSP is responsible for splitting the incoming signal into subbands, and beamforming (through phase shifting and adding the signals coming from its four antennas).

5. LAN and WAN connection: which is used to transfer the RSP-synthesized signals to the Central Core of LOFAR, and communicate within the station.

6. Clock generation and distribution, which provides a high-precision clock signal to all RCU boards.

7. Power generation and distribution: this provides power to all digital and analog components.

8. LCU: a general-purpose PC that is responsible for local station control.

9. Climate control: which is responsible for maintaining temperature and humidity conditions that are within the operating envelope for all station hardware.

10. Miscellaneous: e.g., equipment for non-astronomical LOFAR applications.

The current Station Model (as shown in Appendix A) is a simplified version of the Stations that will eventually be rolled out starting in 2006. The current model only incorporates the following components: LBAs, LNAs, RCUs, RSPs, and power generation (partially).

The current station model is specifically trimmed to represent the FTS-1 hardware. As only limited hands-on experience with the FTS-1 hardware is available, the model has only been validated with a subset of system failures. Currently, the model is able to correctly diagnose the following types of failures:

- antenna cable failure.
- unpowered low-noise amplifier.
- power-off for all RCUs.
- bad contact between an RCU and the back-plane.
- conditions that are persistent and ‘abnormal’.

One important feature of the FTS-1 ‘Station’ model is that it is able to diagnose over time, i.e., diagnosis is not necessarily performed over a single time instance. In this way, the model can be used to perform diagnostics by combining a number of observations that represent different settings and/or sensor readouts, which gives a tremendous boost in diagnostic capabilities.

Also, the model is written in such a way that the diagnostic reasoner can outright reject a complete observation, if (for example) it contradicts the observations made at other time instances. This ensures robustness against incidental sensor glitches: a component will never be diagnosed as 'faulty' based on a single anomalous observation.

4.2 CEP Model

The Central Processing model will represent the central computation and storage facilities that are available within LOFAR, and the associated failures. No modeling efforts have been expended so far in this area.

4.3 WAN Model

The Wide Area Network will represent the data and control transport network of LOFAR, which connects the Central Core to the Remote Stations. No modeling efforts have been expended so far in this area.

A The FTS-1 System Health Management Model

This section shows the model that is used to diagnose the FTS-1 system (October/November 2005). This model is written in the Lydia language, which superficially resembles C. The intention of this section is to convey at least some flavor of what a diagnostic model looks like.

Listing ('fts.sys')

```
/* FTS-1 model */

type RadioFrequencyEnvironment = enum {
    NOMINAL_LOW,
    NOMINAL_MEDIUM,
    NOMINAL_HIGH
};

type DayOrNight = enum {
    NIGHT,
    TWILIGHT,
    DAY
};

type LowNoiseAmplifierPower = enum {
    POWERED,
    UNPOWERED
};

type LowNoiseAmplifierPowerSupplyHealth = enum {
    HEALTHY,
    FAULTY
};

type ReceiverUnitPower = enum {
    POWERED,
    UNPOWERED
};

type ReceiverUnitPowerSupplyHealth = enum {
    HEALTHY,
    FAULTY
};

type LowBandAntennaHealth = enum {
    HEALTHY,       /* connected. The LNA's output signal will be transported faithfully
                      to the RCU's input */
```

```
    CABLE_SEVERED, /* disconnected. This will manifest itself in the NO_SIGNAL sbstats. */
    ABNORMAL
};

type ReceiverUnitHealth = enum {
    HEALTHY, /* The RCU propagates the incoming signal (which /may/ be faulty) via its
                ADC to the RSP's AP */
    BAD_CONTACT  /* The RCU has a bad contact. This will look to the RSP as a stuck
                    signal (giving a spectrum with just a DC) */
};

type ReceiverUnitInputSignal = enum {
    NO_SIGNAL,
    LOW_SIGNAL,
    MEDIUM_SIGNAL,
    HIGH_SIGNAL,
    ABNORMAL
};

type ReceiverUnitOutputSignal = enum {
    ZEROES,
    NO_SIGNAL,
    LOW_SIGNAL,
    MEDIUM_SIGNAL,
    HIGH_SIGNAL,
    ABNORMAL
};

system RadioFrequencyEnvironmentSimulator(RadioFrequencyEnvironment rf_env)
{
    DayOrNight day_or_night;

    attribute observable(day_or_night) = true;

    switch (day_or_night)
    {
        DayOrNight.NIGHT ->
        {
            /* at night-time, we suppose there is a LOW signal */
            rf_env = RadioFrequencyEnvironment.NOMINAL_LOW;
        }
        DayOrNight.TWILIGHT ->
        {
            /* at twilight, we may have either low, high signals */
            rf_env = RadioFrequencyEnvironment.NOMINAL_MEDIUM;
        }
        DayOrNight.DAY ->
        {
```

```
            /* during day-time, it is guaranteed that we will have strong signals */
            rf_env = RadioFrequencyEnvironment.NOMINAL_HIGH;
        }
    }
}

system LowBandAntenna(
        RadioFrequencyEnvironment rf_env,
        LowNoiseAmplifierPower lna_power,
        ReceiverUnitInputSignal rcu_input,
        LowBandAntennaHealth health
    )
{
    /* This describes the cable, Low-Band antenna, plus Low-Noise Amplifier behavior */
    /* The LNA is supposed to be flawless, but it can be de-powered. */

    attribute health(health) = true;
    attribute probability(health) =
        cond (health) (
            LowBandAntennaHealth.HEALTHY       -> 0.999999;
            LowBandAntennaHealth.CABLE_SEVERED -> 0.0000005;
            LowBandAntennaHealth.ABNORMAL      -> 0.0000005;
        );

    switch (health)
    {
        LowBandAntennaHealth.HEALTHY ->
        {
            switch (lna_power)
            {
                LowNoiseAmplifierPower.POWERED ->
                {
                    switch (rf_env)
                    {
                        RadioFrequencyEnvironment.NOMINAL_LOW ->
                        {
                            (rcu_input = ReceiverUnitInputSignal.LOW_SIGNAL)
                         or (rcu_input = ReceiverUnitInputSignal.MEDIUM_SIGNAL);
                        }
                        RadioFrequencyEnvironment.NOMINAL_MEDIUM ->
                        {
                            (rcu_input = ReceiverUnitInputSignal.LOW_SIGNAL)
                         or (rcu_input = ReceiverUnitInputSignal.MEDIUM_SIGNAL)
                         or (rcu_input = ReceiverUnitInputSignal.HIGH_SIGNAL);
                        }
                        RadioFrequencyEnvironment.NOMINAL_HIGH ->
                        {
                            (rcu_input = ReceiverUnitInputSignal.MEDIUM_SIGNAL)
```

```
                         or (rcu_input = ReceiverUnitInputSignal.HIGH_SIGNAL);
                        }
                    }
                }
                LowNoiseAmplifierPower.UNPOWERED ->
                {
                    switch (rf_env)
                    {
                        RadioFrequencyEnvironment.NOMINAL_HIGH ->
                        {
                            rcu_input = ReceiverUnitInputSignal.LOW_SIGNAL;
                        }
                        RadioFrequencyEnvironment.NOMINAL_LOW ->
                        {
                            rcu_input = ReceiverUnitInputSignal.NO_SIGNAL;
                        }
                    }
                }
            }
        }
        LowBandAntennaHealth.CABLE_SEVERED ->
        {
            rcu_input = ReceiverUnitInputSignal.NO_SIGNAL;
        }
        LowBandAntennaHealth.ABNORMAL ->
        {
            rcu_input = ReceiverUnitInputSignal.ABNORMAL;
        }
    }
}

system ReceiverUnit(ReceiverUnitInputSignal rcu_input,
                    ReceiverUnitPower rcu_power,
                    ReceiverUnitOutputSignal rcu_output,
                    ReceiverUnitHealth health)
{
    attribute health(health) = true;
    attribute probability(health) =
        cond (health) (
            ReceiverUnitHealth.HEALTHY     -> 0.999999;
            ReceiverUnitHealth.BAD_CONTACT -> 0.000001;
        );

    switch (health)
    {
        ReceiverUnitHealth.HEALTHY ->
        {
            switch (rcu_power)
```

```
            {
                ReceiverUnitPower.POWERED ->
                {
                    switch (rcu_input)
                    {
                        ReceiverUnitInputSignal.NO_SIGNAL ->
                        {
                            rcu_output = ReceiverUnitOutputSignal.NO_SIGNAL;
                        }
                        ReceiverUnitInputSignal.LOW_SIGNAL ->
                        {
                            rcu_output = ReceiverUnitOutputSignal.LOW_SIGNAL;
                        }
                        ReceiverUnitInputSignal.MEDIUM_SIGNAL ->
                        {
                            rcu_output = ReceiverUnitOutputSignal.MEDIUM_SIGNAL;
                        }
                        ReceiverUnitInputSignal.HIGH_SIGNAL ->
                        {
                            rcu_output = ReceiverUnitOutputSignal.HIGH_SIGNAL;
                        }
                        ReceiverUnitInputSignal.ABNORMAL ->
                        {
                            rcu_output = ReceiverUnitOutputSignal.ABNORMAL;
                        }
                    }
                }
                ReceiverUnitPower.UNPOWERED ->
                {
                    rcu_output = ReceiverUnitOutputSignal.ZEROES;
                }
            }
        }
        ReceiverUnitHealth.BAD_CONTACT ->
        {
            rcu_output = ReceiverUnitOutputSignal.ZEROES;
        }
    }
}

system ReceiverUnitPowerSupply(ReceiverUnitPower rcu_power, ReceiverUnitPowerSupplyHealth h)
{
    attribute health(h) = true;
    attribute probability(h) =
        cond (h) (
            ReceiverUnitPowerSupplyHealth.HEALTHY -> 0.999999;
            ReceiverUnitPowerSupplyHealth.FAULTY  -> 0.000001;
        );
```

```
    switch (h)
    {
        ReceiverUnitPowerSupplyHealth.HEALTHY ->
        {
            rcu_power = ReceiverUnitPower.POWERED;
        }
        ReceiverUnitPowerSupplyHealth.FAULTY ->
        {
            rcu_power = ReceiverUnitPower.UNPOWERED;
        }
    }
}

system LowNoiseAmplifierPowerSupply(LowNoiseAmplifierPower lna_power,
       LowNoiseAmplifierPowerSupplyHealth lna_powersupply_health)
{
    attribute health(lna_powersupply_health) = true;
    attribute probability(lna_powersupply_health) =
        cond (lna_powersupply_health) (
            LowNoiseAmplifierPowerSupplyHealth.HEALTHY -> 0.999999;
            LowNoiseAmplifierPowerSupplyHealth.FAULTY  -> 0.000001;
        );

    switch (lna_powersupply_health)
    {
        LowNoiseAmplifierPowerSupplyHealth.HEALTHY ->
        {
            lna_power = LowNoiseAmplifierPower.POWERED;
        }
        LowNoiseAmplifierPowerSupplyHealth.FAULTY ->
        {
            lna_power = LowNoiseAmplifierPower.UNPOWERED;
        }
    }
}

system FTS_RemoteStationProcessor(
           ReceiverUnitPowerSupplyHealth rcu_powersupply_health,
           LowNoiseAmplifierPowerSupplyHealth lna_0_powersupply_health,
                                              lna_1_powersupply_health,
                                              lna_2_powersupply_health,
                                              lna_3_powersupply_health,
                                              lna_4_powersupply_health,
                                              lna_5_powersupply_health,
                                              lna_6_powersupply_health,
                                              lna_7_powersupply_health,
           LowBandAntennaHealth lba_0_health, lba_1_health, lba_2_health, lba_3_health,
```

```
                                lba_4_health, lba_5_health, lba_6_health, lba_7_health,
          ReceiverUnitHealth rcu_01_health, rcu_23_health, rcu_45_health, rcu_67_health
       )
{
    RadioFrequencyEnvironment rf_env;
    ReceiverUnitPower rcu_power;
    LowNoiseAmplifierPower lna_0_power, lna_1_power, lna_2_power, lna_3_power,
                           lna_4_power, lna_5_power, lna_6_power, lna_7_power;
    ReceiverUnitInputSignal rcu_0_input_signal, rcu_1_input_signal,
                            rcu_2_input_signal, rcu_3_input_signal,
                            rcu_4_input_signal, rcu_5_input_signal,
                            rcu_6_input_signal, rcu_7_input_signal;
    bool accept_observation;

    attribute health(accept_observation) = true;
    attribute probability(accept_observation) = accept_observation ? 0.9999999 : 0.000001;

    ReceiverUnitOutputSignal rcu_0_predicted_output_signal;
    ReceiverUnitOutputSignal rcu_1_predicted_output_signal;
    ReceiverUnitOutputSignal rcu_2_predicted_output_signal;
    ReceiverUnitOutputSignal rcu_3_predicted_output_signal;
    ReceiverUnitOutputSignal rcu_4_predicted_output_signal;
    ReceiverUnitOutputSignal rcu_5_predicted_output_signal;
    ReceiverUnitOutputSignal rcu_6_predicted_output_signal;
    ReceiverUnitOutputSignal rcu_7_predicted_output_signal;

    ReceiverUnitOutputSignal rcu_0_output_signal;
    ReceiverUnitOutputSignal rcu_1_output_signal;
    ReceiverUnitOutputSignal rcu_2_output_signal;
    ReceiverUnitOutputSignal rcu_3_output_signal;
    ReceiverUnitOutputSignal rcu_4_output_signal;
    ReceiverUnitOutputSignal rcu_5_output_signal;
    ReceiverUnitOutputSignal rcu_6_output_signal;
    ReceiverUnitOutputSignal rcu_7_output_signal;

    attribute observable(rcu_0_output_signal) = true;
    attribute observable(rcu_1_output_signal) = true;
    attribute observable(rcu_2_output_signal) = true;
    attribute observable(rcu_3_output_signal) = true;
    attribute observable(rcu_4_output_signal) = true;
    attribute observable(rcu_5_output_signal) = true;
    attribute observable(rcu_6_output_signal) = true;
    attribute observable(rcu_7_output_signal) = true;

    system RadioFrequencyEnvironmentSimulator rf_env_sim(rf_env);
    system ReceiverUnitPowerSupply rcu_pow_supply(rcu_power, rcu_powersupply_health);

    system LowNoiseAmplifierPowerSupply lna_0_powersupply(lna_0_power, lna_0_powersupply_health);
```

```
    system LowNoiseAmplifierPowerSupply lna_1_powersupply(lna_1_power, lna_1_powersupply_health);
    system LowNoiseAmplifierPowerSupply lna_2_powersupply(lna_2_power, lna_2_powersupply_health);
    system LowNoiseAmplifierPowerSupply lna_3_powersupply(lna_3_power, lna_3_powersupply_health);
    system LowNoiseAmplifierPowerSupply lna_4_powersupply(lna_4_power, lna_4_powersupply_health);
    system LowNoiseAmplifierPowerSupply lna_5_powersupply(lna_5_power, lna_5_powersupply_health);
    system LowNoiseAmplifierPowerSupply lna_6_powersupply(lna_6_power, lna_6_powersupply_health);
    system LowNoiseAmplifierPowerSupply lna_7_powersupply(lna_7_power, lna_7_powersupply_health);

    system LowBandAntenna lba_0(rf_env, lna_0_power, rcu_0_input_signal, lba_0_health);
    system LowBandAntenna lba_1(rf_env, lna_1_power, rcu_1_input_signal, lba_1_health);
    system LowBandAntenna lba_2(rf_env, lna_2_power, rcu_2_input_signal, lba_2_health);
    system LowBandAntenna lba_3(rf_env, lna_3_power, rcu_3_input_signal, lba_3_health);
    system LowBandAntenna lba_4(rf_env, lna_4_power, rcu_4_input_signal, lba_4_health);
    system LowBandAntenna lba_5(rf_env, lna_5_power, rcu_5_input_signal, lba_5_health);
    system LowBandAntenna lba_6(rf_env, lna_6_power, rcu_6_input_signal, lba_6_health);
    system LowBandAntenna lba_7(rf_env, lna_7_power, rcu_7_input_signal, lba_7_health);

    system ReceiverUnit rcu_0(rcu_0_input_signal, rcu_power,
                              rcu_0_predicted_output_signal, rcu_01_health);
    system ReceiverUnit rcu_1(rcu_1_input_signal, rcu_power,
                              rcu_1_predicted_output_signal, rcu_01_health);
    system ReceiverUnit rcu_2(rcu_2_input_signal, rcu_power,
                              rcu_2_predicted_output_signal, rcu_23_health);
    system ReceiverUnit rcu_3(rcu_3_input_signal, rcu_power,
                              rcu_3_predicted_output_signal, rcu_23_health);
    system ReceiverUnit rcu_4(rcu_4_input_signal, rcu_power,
                              rcu_4_predicted_output_signal, rcu_45_health);
    system ReceiverUnit rcu_5(rcu_5_input_signal, rcu_power,
                              rcu_5_predicted_output_signal, rcu_45_health);
    system ReceiverUnit rcu_6(rcu_6_input_signal, rcu_power,
                              rcu_6_predicted_output_signal, rcu_67_health);
    system ReceiverUnit rcu_7(rcu_7_input_signal, rcu_power,
                              rcu_7_predicted_output_signal, rcu_67_health);

    if (accept_observation)
    {
        rcu_0_output_signal = rcu_0_predicted_output_signal;
        rcu_1_output_signal = rcu_1_predicted_output_signal;
        rcu_2_output_signal = rcu_2_predicted_output_signal;
        rcu_3_output_signal = rcu_3_predicted_output_signal;
        rcu_4_output_signal = rcu_4_predicted_output_signal;
        rcu_5_output_signal = rcu_5_predicted_output_signal;
        rcu_6_output_signal = rcu_6_predicted_output_signal;
        rcu_7_output_signal = rcu_7_predicted_output_signal;
    }
}
```

```
system FTS_RemoteStationProcessor_TCD()
{
    ReceiverUnitPowerSupplyHealth h_rcu_power;
    LowNoiseAmplifierPowerSupplyHealth lna_0_powersupply_health,
                                       lna_1_powersupply_health,
                                       lna_2_powersupply_health,
                                       lna_3_powersupply_health,
                                       lna_4_powersupply_health,
                                       lna_5_powersupply_health,
                                       lna_6_powersupply_health,
                                       lna_7_powersupply_health;
    LowBandAntennaHealth lba_0_health, lba_1_health, lba_2_health, lba_3_health,
                         lba_4_health, lba_5_health, lba_6_health, lba_7_health;
    ReceiverUnitHealth rcu_01_health, rcu_23_health, rcu_45_health, rcu_67_health;

    system FTS_RemoteStationProcessor t1(
            h_rcu_power,
            lna_0_powersupply_health, lna_1_powersupply_health,
            lna_2_powersupply_health, lna_3_powersupply_health,
            lna_4_powersupply_health, lna_5_powersupply_health,
            lna_6_powersupply_health, lna_7_powersupply_health,
            lba_0_health, lba_1_health, lba_2_health, lba_3_health,
            lba_4_health, lba_5_health, lba_6_health, lba_7_health,
            rcu_01_health, rcu_23_health, rcu_45_health, rcu_67_health
        );

    system FTS_RemoteStationProcessor t2(
            h_rcu_power,
            lna_0_powersupply_health, lna_1_powersupply_health,
            lna_2_powersupply_health, lna_3_powersupply_health,
            lna_4_powersupply_health, lna_5_powersupply_health,
            lna_6_powersupply_health, lna_7_powersupply_health,
            lba_0_health, lba_1_health, lba_2_health, lba_3_health,
            lba_4_health, lba_5_health, lba_6_health, lba_7_health,
            rcu_01_health, rcu_23_health, rcu_45_health, rcu_67_health
        );

    system FTS_RemoteStationProcessor t3(
            h_rcu_power,
            lna_0_powersupply_health, lna_1_powersupply_health,
            lna_2_powersupply_health, lna_3_powersupply_health,
            lna_4_powersupply_health, lna_5_powersupply_health,
            lna_6_powersupply_health, lna_7_powersupply_health,
            lba_0_health, lba_1_health, lba_2_health, lba_3_health,
            lba_4_health, lba_5_health, lba_6_health, lba_7_health,
            rcu_01_health, rcu_23_health, rcu_45_health, rcu_67_health
        );
}
```