Wireless Software and Hardware platforms for Flexible and Unified radio and network control

Project Deliverable D10.1

Design of software architecture for intelligent control and showcases

Contractual date of delivery:	31-12-2015
Actual date of delivery:	23-12-2015
Beneficiaries:	IMINDS, TCD, CNIT, TUB, NCENTRIC, RUTGERS
Lead beneficiary:	IMINDS
Authors:	Merima Kulin (IMINDS), Ingrid Moerman (IMINDS), Spilios Giannoulis (IMINDS), Ilenia Tinnirello (CNIT), Perluigi Gallo (CNIT), Anatolij Zubow (TUB), Mikolaj Chwalisz (TUB), Piotr Gawłowicz (TUB), Nicholas Kaminski (TCD), Luiz DaSilva (TCD), Robin Leblon (NCENTRIC), Ivan Seskar (RUTGERS)
Reviewers:	Ivan Seskar (RUTGERS)
Work package:	WP10 – Intelligent control of radio and network operation
Estimated person months:	4
Nature:	R
Dissemination level:	PU
Version:	1.15

Abstract:

This public deliverable provides a detailed description on how the showcases defined in D2.2 will be augmented with intelligent adaptations and how these showcases drive the functionalities and role of the WiSHFUL Intelligence Framework. This deliverable extends the original WiSHFUL software architecture (as presented in D2.2) with the WiSHFUL Intelligence Framework, which is further called the generalized software architecture for intelligent radio and network control.

This document will serve as a guideline for the implementation of the basic intelligence modules that will be offered for the Open Calls 3, 4 and 5.

Keywords:

Intelligent radio control, intelligent network control, intelligence framework, software architecture, showcases

Executive Summary

This public deliverable provides a detailed description on how the **showcases** earlier defined in D2.2 are augmented with intelligent adaptations. Three showcases have been identified:

- ***Intelligent load and interference aware MAC adaptation***: this showcase aims to demonstrate how to increase network performance by adding intelligent mechanisms for deciding on two possible adaptation mechanisms: (1) optimization of parameters for the current MAC protocol, or (2) selection of the most suitable MAC protocol for the current network conditions while taking into account the application requirements. The goal is to implement (1) a general model based on experimental collection of data, that is responsible for intelligent MAC protocol selection, and (2) a protocol-specific model based on an analytical expression of the link performance for intelligent MAC parameter tuning.
- ***Intelligent slot allocation in a hybrid TDMA MAC***: this showcase aims to demonstrate that an explicit hidden-node detection phase can be avoided by adding an intelligent mechanism for analysis and optimisation of the slot allocation for a hybrid TDMA MAC. The goal is to show that by collecting data about the interference condition within each time slot at each wireless node and performing data fusion at central device in the network, it is possible to implicitly detect the wireless links suffering from the hidden node problem without performing an explicit detection.
- ***Learning about primary user behaviour and best MCS scheme selection for a secondary user:*** this showcase will examine an approach for supporting channel occupancy sensing through intelligence and learning and involves the following steps: (1) accurate primary user detection with low overhead from channel power estimates employing a variable threshold, (2) learning the primary user characteristics through channel occupancy analysis in multiple channels with multiple nodes, (3) online learning of the best modulation and coding scheme for secondary user transmissions.

The intelligent showcases have driven the requirements for and the design of the *generalized software architecture for intelligent radio and network control*. The original software architecture, as presented in deliverable D2.2, will be extended with the following architectural components:

- ***Data Collection Component***: this component is responsible for data acquisition of the network status and the application requirements. With respect to the network status, the experimenter can specify the radio and/or network parameters he wants to monitor by choosing the parameters of interest from a predefined set of possible options (offered by the UPI interfaces) and the collection time window. With respect to the application requirements, a new interface (Application API) is needed to feed the application requirements to the intelligence framework. Collected data can further be summarized or compressed using specific aggregation methods limiting transmission of (redundant) data and enhancing network lifetime.
- ***Intelligence Composition Component:*** this component offers different approaches that can be selected by the experimenter for finding optimal radio and network settings. The intelligence modules will be offered as a collection of algorithms (e.g. optimisation and machine learning techniques) that can be applied for user-specific scenarios. The Intelligence Composition Component also offers modules for pre-processing data such as data cleaning, normalization, and data transformation.
- ***Action Component***: this component represents an interface between the outputs of the intelligence algorithm and the UPI functions that enable the control of the behavior of wireless nodes. This component translates the intelligence decisions taken by the Intelligence Composition Component in a sequence of UPI calls.
- ***Intelligence framework user interface***: this user interface is responsible for the interaction of the user with the WiSHFUL intelligence framework. The user interface offers different

interaction modules for selecting network parameters, for selecting the duration of the monitoring, for selecting the type of aggregation, for selecting the intelligence modules (with the possibility to use an existing algorithms or adding custom algorithms) and for selecting the actions to be executed at the wireless nodes. The user interface allows the experimenter to compose a pipeline of the different data processing tasks needed for the intelligent control of his wireless network.

This document will serve as a guideline for the implementation of the basic intelligence modules that will be offered in the 3rd, 4th and 5th open calls for experimentation. First intelligent modules and hierarchical control software prototype and first set of showcases will be implemented by month 20 (end of August 2016).

List of Acronyms and Abbreviations

API	Application Programming Interface
CSMA	Carrier Sense Multiple Access
CSV	Comma-Separated Values
FFT	Fast Fourier Transform
KDP	Knowledge Discovery Process
IQ	In Phase and Quadrature components of a signal
LQI	Link Quality indicator
MAC	Medium Access Control
MCS	Modulation and Coding Scheme
PRR	Packet Reception Rate
RSSI	Received Signal Strength Indicator
SQL	Structured Query Language
TDMA	Time Division Multiple Access
UML	Unified Modelling Language
UPI	Unified Programming Interface
UPI_G	Unified Programming Interface - Global
UPI_HC	Unified Programming Interface - Hierarchical
UPI_R	Unified Programming Interface - Radio
UPI_N	Unified Programming Interface - Network

Table of contents

1 Introduction

Intelligence is most widely studied in humans, but has also been observed in other biological systems, including microorganisms and plants. It generally refers to the capability of intelligent organisms to process information (i.e. stimuli), reason, learn and make decisions. Artificial intelligence aims to recreate biological intelligence in machines enabling them to automatically process information and perform cognitive functions such as reasoning and learning. Typical approaches for intelligent solutions are: optimization algorithms, machine learning techniques, data science methods, etc.

In WiSHFUL we aim to apply similar intelligence approaches to solve problems arising from the wireless networks domain. To this end, the overall WiSHFUL software architecture for radio and network control (see deliverables D2.1, D3.1 and D4.1) needs to be extended with additional modules that provide support for algorithms that can make intelligent decisions based on current network status and application requirements. Instead of using simple intuitive algorithms or static rules for adapting radio and network settings, the control of radio and network will be automated by applying advanced algorithms that capture the wireless network behaviour and learn the best strategy for taking decisions on radio and network configurations in a certain network context for optimally supporting wireless applications. The proposed ***intelligence framework*** is based on decomposing the cognitive radio network operation into individual data processing tasks.

This deliverable provides the design of a generalized framework for supporting intelligence in wireless networking. The basic requirements for this framework are driven from the showcases, earlier presented in deliverable D2.2, that are now augmented with intelligent adaptations. The ***generalized software architecture for intelligent radio and network*** control consists of different intelligence modules, the composition of the modules and the information flow between the modules.

This deliverable is structured as follows. Section 2 describes three intelligence showcases that apply intelligent mechanisms for increasing wireless network performance through intelligent radio and network control, and also discusses the functionality that needs to be supported by the WiSHFUL intelligence framework. Section 3 defines the general requirements of the WiSHFUL intelligence framework, starting from the specific requirements from the different intelligence showcases in section 2. Section 4 explains how the WiSHFUL intelligence framework extends the original WiSHFUL software architecture for radio and network control. It also details the main new architectural components, and further explains the workflow how an experimenter can compose his own intelligence solutions for solving his wireless problem. The main conclusions are summarized in section 5.

2 Intelligence showcases

2.1 Intelligent load and interference aware MAC adaptation

2.1.1 General description

In this showcase we aim to demonstrate how to increase network performance by adding intelligent mechanisms for deciding on two adaptation mechanisms: optimisation of parameters for the current MAC protocol or the selection of the different (i.e. most suitable) MAC protocol for the current network conditions (for example, number of nodes, density of nodes, interference conditions) and with respect to the application requirements (for example data rate, latency, reliability). In the earlier showcase (see D2.2) the decision algorithm was based on domain specific knowledge and experience by setting a threshold on the number of dropped frames.

The goal is to demonstrate how the data collection through the unified programming interfaces and aggregation can be exploited for the purpose of adding intelligence in the wireless network.

We propose to implement (1) a general model based on experimental collection of data, that is responsible for ***intelligent MAC protocol selection,*** and (2) a protocol-specific model based on an analytical expression of the link performance for ***intelligent MAC parameter tuning***. The output of the models will drive the decision to the wireless nodes through the unified programming interfaces.

The intelligent models will be implemented following the six steps of the KDP (Knowledge Discovery Process) used in the data science community:

- ***Step 1 - understanding the domain***: the goal of this step is to identify and state the wireless networking problems that need to be solved, and formulate them as data science problems. We assume here that under certain network conditions and application requirements, (1) one MAC protocol will outperform other MAC protocols and (2) certain MAC parameter settings will lead to better performance than other MAC parameter settings. For this showcase, we propose to translate the *MAC selection problem* into a data science *classification* problem with the different MAC protocols configured with different parameters as classes (for example, MAC1, MAC2 and MAC3). We further assume that the knowledge on the network conditions, acquired on the different nodes, will also be available globally at a central device in the network (via UPI_G). The final target is to build a model that can predict the most suitable MAC protocol (the output of the model), with the network conditions and the application requirements as input. For cases where an analytical model is available for a given protocol, we also propose to use *classical optimization* schemes for the *tuning of available MAC parameters.* In this case, by exploiting an expression relating the desired system performance to the network state (in terms of observable network parameters) and MAC parameters, it is possible to use a time-varying estimate of network state, for finding the optimal configurations of MAC parameters.
- ***Step 2 - understanding the data***: we have to define which data we have to collect through the unified programming interfaces: for example, which data is relevant to measure the network condition and/or performance (e.g. RSSI measurements can be relevant data for representing the link quality, counting the number of sent packets and successfully received packets can give an indication of the reliability of the network, the density of the neighbourhood can be retrieved from the number of nodes in the neighbour table, etc.). Using the domain knowledge (see step 1), we need to find the best representation of the data, i.e. we have to define the relevant features (the input data of the model) and the labels (the output of the model – in case of a classification problem the output of the model are the classes). We will run several experiments under different network conditions, using different MACs and with different application requirements and will collect a dataset. A huge number of individual RSSI measurements are probably not a good representation of a network condition. It may be

better to average the RSSI raw measurements over a certain time window, or to collect the minimum, maximum, median RSSI values over a time window. So we will need to aggregate raw data. This operation will reduce the amount of collected data, which may be also required because of limited storage capabilities in the wireless network (either locally at the node or globally in a central controller device). The output of this step consists of unprocessed training data. Training data (set of features and labels) is required to build the classification model (see step 4).

- ***Step 3 - data pre-processing***: this step can involve further operations on the data set, such as data cleaning (remove outliers), data transformation (scaling / normalization, aggregation), and data reduction (remove irrelevant information, reduce dimensionality of datasets).
- ***Step 4 - data mining***: this step utilizes one or more machine learning algorithms for data mining (for example decision trees). In this showcase each learning algorithm will build and tune the classification model using the (pre-processed) training data until all training data is classified with sufficiently high accuracy. The output of this step is one or more trained models.
- ***Step 5 - evaluation of the discovered knowledge***: this step provides the mechanisms for evaluation of the model(s) to check if the model(s) generalize(s) well for new test data. We will evaluate the learned model against a separate test set (different from the training data) using cross-validation and repeat the steps 1 to 5 until we find the best performing model.
- ***Step 6 - using the discovered knowledge***: while the machine learning approaches in steps 4 and 5 are executed in an offline manner, this step uses the gained insights for the design of a runtime deployment.
  For the *MAC selection*, we can implement the best performing model on a central controller device in the network, as part of the Global Control Program. Based on the input information it selects the most suitable MAC protocol and informs the wireless nodes about the selected MAC using the unified programming interface UPI_G.
  For the *tuning of the MAC parameters* by means of analytical functions of the network state, we can implement the tuning based on both local and global observations for estimating the network state. The decisions will then be enforced by using the unified programming interfaces UPI_R/UPI_N and/or UPI_G.

The process of model selection is presented in Figure 1.

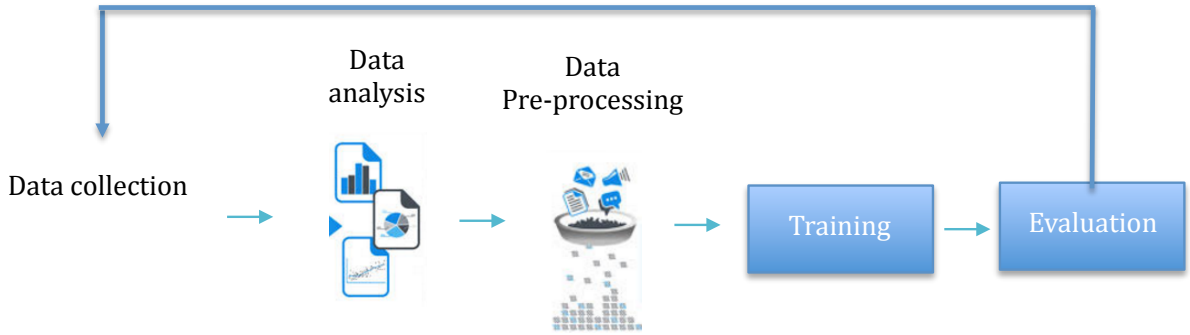

Figure 1: Process of model selection

Summarizing, for this showcase, we will evaluate two different approaches for optimizing MAC performance: the MAC selection approach, in which we will compare the performance of different MAC schemes, such as TDMA and CSMA, working with different operation parameters (e.g. different contention windows or different allocated slots) by performing learning on sets of previously collected experimental data; the MAC parameter tuning as a function of the network state, in which we will evaluate the effect of local or global estimates of the network state. The two approaches can

also be combined, for first identifying a MAC protocol among a finite set of possible configuration choices, and then improving the performance of the selected scheme in a wider configuration range.

2.1.2 Mapping to the intelligence framework

In order to realize this showcase, we expect the following functionality from the WiSHFUL intelligence framework:

- We need to collect raw data from the radio and the network via the local and global unified programming interfaces, UPI_R, UPI_N and UPI_G and provide these data to the data collection component. The data needs to be stored in a suitable format, so that the data can be used for further operations in the data collection component. For research purposes it is desired to collect the raw data with the finest possible granularity, which may lead to a huge raw data set. The data set will be used for further offline processing (see steps 2 to 5 of the knowledge discovery process).
- For this showcase we consider the following ***operations in the data collection component***
  - Reduction of data by simple averaging over a certain time window, which leads to a data set with a coarser granularity. Again the reduced data set needs to be stored in a suitable storage format that automatically adapts to the time window. It may be interesting for a researcher or wireless developer to explore different granularities for optimising his wireless system.
  - Aggregation operations in view of extracting relevant features, such as:
    - Determine minimum of (reduced or raw) data over a certain time window
    - Determine maximum of (reduced or raw) data over a certain time window
    - Determine quantiles of (reduced or raw) data over a certain time window
    - Determine percentiles of (reduced or raw) data over a certain time window
    - Build a histogram of (reduced or raw) data over a certain time window
  - The output of each aggregation operation will transform and possibly further reduce the data set. This new data set needs to be stored in a suitable format.
- There may be a need for further ***pre-processing operations*** to deliver the data set in the required format to the machine learning algorithms. Possible pre-processing steps are:
  - A data cleaning operation to remove outliers from the data set.
  - A scaling or normalization operation.
  - A transformation operation (for example from linear to logarithmic values).
- For this showcase we will consider a limited number of basic supervised learning algorithms for building a classification model, for example decision trees or neural networks.
- The data set needs to be able to ***split in a training set and a test set.*** While the training set is used to train/learn the parameters of the classification model, the test set is used to validate the performance of the model in terms of generality (i.e. on data previously un-seen by the model). A popular model evaluation method is k-fold cross-validation [1], where training and evaluation is repeated k times through k rounds, where in each round one of the folds is kept for testing, while the others are used for training. For splitting the data into k consecutive folds, existing tools will be considered, which are possibly part of the machine learning tools mentioned earlier.
- Once we have found a sufficiently performing classification model, this model together with all operations on the collected raw data (data reduction, aggregation, pre-processing) needs to be ***implemented in the Global and/or Local Control Programs***. While the learning phase is

performed on the whole dataset at once, we now have to perform all data operations each time new data instances arrive in real (or near-real) time.

- The intelligent ***decision***, which is the output of the classification model (or the MAC class) needs to be mapped into a sequence of control and configuration commands via the Unified UPI_R, UPI_N and UPI_G interfaces.

2.2 Intelligent slot allocation in a hybrid TDMA MAC

2.2.1 General description

In the earlier showcase (see D2.2) the Global Control Program performed an explicit detection of the wireless links suffering from the hidden node problem before setting-up and configuring the hybrid TDMA MAC protocol. Unfortunately, such an approach results in high overhead, as for each new wireless link all co-located nodes need to be tested. During the hidden-node detection phase no data traffic can be transmitted.

Therefore, in this showcase we demonstrate that an explicit hidden-node detection phase can be avoided if we add an intelligent mechanism for analysis and optimisation of the slot allocation for the hybrid TDMA MAC.

The goal is to show that by collecting data about the interference condition (e.g. packet loss) within each time slot at each wireless node and performing data fusion at central device in the network that runs the Global Control Program, it is possible to implicitly detect the wireless links suffering from the hidden node problem without performing an explicit detection.

The envisioned system requires different software architecture for radio and network control, as running the Local and Global Control Programs while using only UPG_R, UP_N and UPI_G interfaces, is not suitable. Instead we propose to design and implement a ***software architecture for hierarchical radio and network control*** by providing an ***inter Control Program interface***, ***UPI_HC***, allowing the simultaneous use of Local and Global Control Programs (see Figure 2. In the envisioned showcase the Local Control Programs would autonomously analyse the interference condition in each time slot using the raw data provided by the UPI_R/N interfaces, whereas the derived data (i.e. packet error rate in each time slot) is sent to the Global Control Program. Then the Global Control Program may decide on a new slot allocation to be used. If this is the case, it instructs the Local Control Programs to use the adapted slot allocation.

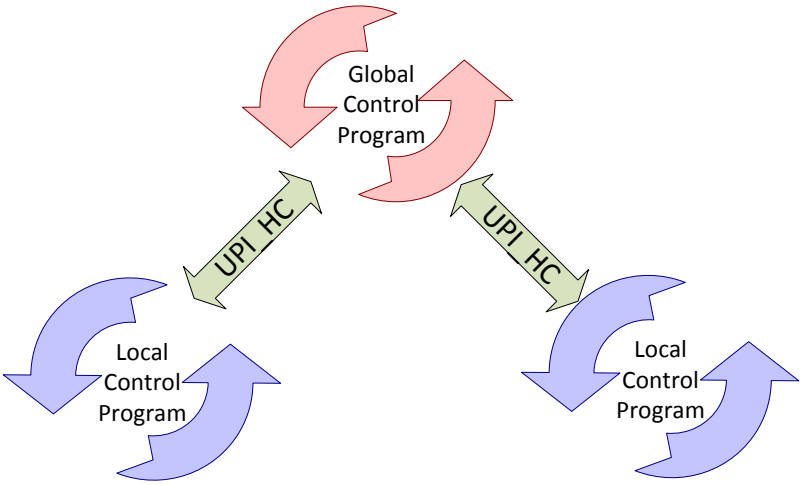

Figure 2. Software architecture for hierarchical radio and network control.

2.2.2 Mapping to the intelligence framework

In order to realize this showcase, we expect the following functionality from the WiSHFUL intelligence framework:

- Support of ***hierarchical control*** where the Local Control Programs independently analyse the raw data provided by the UPI_R/N interfaces and report only aggregated or derived data to the Global Control Program using the UPI_HC inter-controller interface. The Global Control Program performs data fusion techniques in order to make a decision and further informs the wireless nodes about his decision again using the UPI_HC interface.
- In order to avoid inconsistencies, changes to the slot allocation must be executed either by each node or by none of the nodes. Hence we will extend the Action Component of the Intelligence Framework to be able to execute a sequence of control commands in a (distributed) transactional scope. There will be two levels of such ***transactional execution***: (1) *global – execution* of a sequence of control commands on a set of nodes in the network, if at least one of them fails, the execution of all commands have to repeated or rolled back; (2) *local – transactional execution* of a set of functions on a single node, whereas the global controller is notified about success/failure.

2.3 Learning about primary user behaviour and best MCS scheme selection for a secondary user

2.3.1 General description

Constraining the sensing bandwidth allows accurate sensing with limited demand on specialized hardware or signal processing support. As such, these approaches are suitable for using in general purpose reconfigurable radio systems or even low power devices, as proposed for the Internet of Things. We therefore examine an approach for supporting sensing through intelligence and learning, utilizing the steps show in Figure 3.

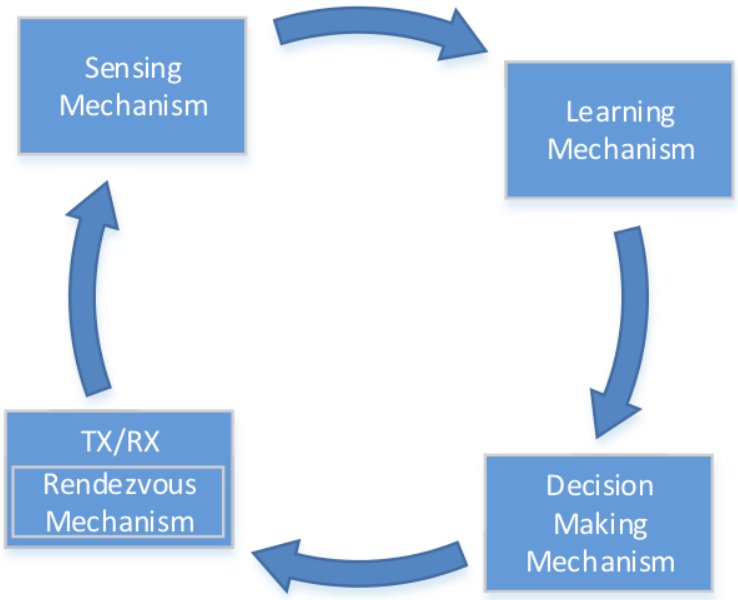

Figure 3: Cycle of Learning

Sensing here collects IQ samples and transforms them into frequency domain through an FFT to obtain a power measurement via the magnitude square operation over the FFT bins. The resulting bins are then separated into channels, with each channel averaged across frequency and time using moving averages. In this way, the system is able to consistently estimate channel power for later use in the system. Primary user frequency may be detected from these channel power estimates through thresholding. Since the radio environment is dynamic, we employ a variable threshold based on the estimated noise floor in each channel. Noise floor estimation completes the detection approach by providing an online view of the noise environment faced by the radio systems. This method of primary user detection provides an accurate mechanism with sufficiently low overhead when performed over limited bandwidth.

Learning is then employed to allow the effective use of the above detection mechanism in multiple channels. Here, learning has two goals: learning the statistical characteristics of a primary user and learning the best modulation and coding scheme for secondary user transmissions. To capture the primary user's characteristics, we will perform channel occupancy analysis, which assumes that the primary user channel access pattern is a stochastic process. To aid this process, we will build a channel occupancy distribution from collected detection data. This distribution allows the creation of a Markov chain model, which collects the set of probabilities for transitions between channels. This model may be updated during operation to track variable primary user behaviour. The secondary user will then select the channel least likely to be occupied by the primary user at any given time step. To learning the best modulation and coding, a Q-Learning based algorithm will be employed to track the success of each configuration. This lightweight approach to learning is suitable for online usage.

2.3.2 Mapping to the intelligence framework

Realization of this showcase will require the following functionality:

- We need to collect IQ samples from the radio via the local unified programming interfaces UPI_R, and provide these data to the data collection component. The data collection component may then perform the detection process discussed above to provide input into the Markov model of primary user activity.
- We need to collect performance information, in the form of the number of bits successfully transmitted, for each transmission that occurs. This information may then be entered into the Q-Learning scheme to determine suitable modulation and coding scheme for use.
- We need to select a channel, and modulation and coding scheme for use. This selection will be made on the basis of the output of the learning mechanisms discussed above.

3 General requirements for intelligence framework

The requirements of the intelligence framework can be represented by four main **functional components** as shown in Figure 4, namely:

- ***data collection***: retrieving selected radio and network data;
- ***aggregation***: performing data operations to reduce the amount of data or to change the representation of data;
- ***data analysis***: intelligence approach that is selected by the experimenter for finding optimal radio and network settings, often starting with looking into the statistical characteristics of the collected and aggregated data;
- ***action***: adapting radio and network settings.

The basic requirements for the four functional components are described in the next subsections.

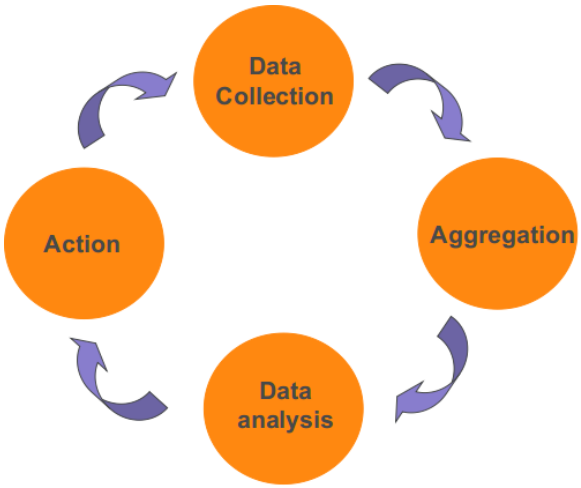

Figure 4: Functional requirements for intelligence support

3.1 Data collection

The basic requirements for data collection functional component can be logically split into two parts: (1) data about radio/network operation, and (2) application requirements. To gather application requirements, we introduce a new interface, the Application API.

- Relevant ***data about radio operation*** (RRSI, LQI, IQ samples, etc.), ***and network operation*** (number of sent packets, successfully received packets, number of nodes in the neighbour table, etc.) needs to be retrieved through the WiSHFUL UPIs, UPI_R, UPI_N and UPI_G. Unprocessed data directly collected via UPIs are further called raw data.
  - Raw data needs to be stored in a suitable format (for example comma-separated values (CSV), Structured Query Language (SQL) table, excel file, .dat file, etc.) so that further data operations can be easily executed. For offline processing of data (for instance for data science approaches) or for research purposes, it may be desired to collect the raw data with the finest possible granularity, which may lead to a huge raw data set. For online processing of data, raw data may be collected with a coarser granularity.
  - It must be possible to specify the format of the data storage
- ***Application requirements*** need to be collected, such as minimal data latency, jitter reliability, maximal throughput, etc. A new interface (Application API) needs to be defined to feed the application requirements to the intelligence framework.
  - The Application API can either be used statically for network initialization, or can be used to dynamically alter the behaviour of the action component, and as such will define the behaviour of the entire network.
  - Multiple requirements can be combined, and distinct requirements can be defined for simultaneously running data streams
  - Application requirements can be defined at the network level, or can be specific to one or more nodes.

3.2 Aggregation

The basic requirements for Aggregation functional component are listed below:

- Several ***data operations*** must be supported to reduce the amount of data, to extract relevant features, or to change the representation of data. Operations identified so far are:

- simple averaging over a certain time window
- determine minimum of (reduced or raw) data over a certain time window
- determine maximum of (reduced or raw) data over a certain time window
- determine quantiles of (reduced or raw) data over a certain time window
- determine percentiles of (reduced or raw) data over a certain time window
- build a histogram of (reduced or raw) data over a certain time window
- perform a Fast Fourier Transform (FFT) algorithm
- perform a magnitude square operation (for example over FFT bins)
- combining different collected raw data (e.g. determine packet loss from the number of sent packets and successfully received packets)

- Where possible, existing tools for performing data operations will be used like SciPy [2] and Pandas [3]**.** In this case the experimenter needs to be informed about the supported input and output data format, and the parameters of the data operation (for example the time window, the size of a bin, the number of bins, the function for combing different data). The experimenter should be able to select input and output data formats and specify the parameters of a specific data operation.
- In case a desired input and/or output format is not supported, it must be possible to perform an appropriate data format conversion in order to ensure that a supported input format is used for performing the desired data operation, and in order to ensure the results of the data operation is stored in the desired output format.
- It must be possible to chain data operations, meaning that multiple data operations can be executed sequentially (for example a FFT followed by magnitude square operation).

3.3 Data analysis

The basic requirements for the Data Analysis functional component are listed below:

- ***Pre-processing operations*** must be supported to deliver the data set in the required format to the machine learning algorithms. As these pre-processing operations may depend on the machine-learning algorithm used, they are considered as part of the data analysis. Although some pre-processing steps may also be performed as part of the aggregation step.
  - So far the following pre-processing steps have been identified:
    - A *data cleaning operation* to remove outliers from the data set. It should be possible to add criteria for detecting outliers (e.g. values larger or smaller than a certain boundary are not realistic and should be deleted). Data cleaning will further reduce the data set.
    - A *scaling or normalization* operation. This operation will not further reduce the data set.
    - A *transformation* operation (for example from linear to logarithmic values). This operation will not further reduce the data set.
  - Similarly as for data-aggregation, existing tools for performing pre-processing operations will be used, where possible. The experimenter needs to be informed about the supported input and output data formats, the parameters, and the criteria of the pre-processing operation. The experimenter should further be able to select input and output data formats and specify the parameters and/or criteria of a pre-processing operation.
  - In case a desired input and/or output format is not supported, it must be possible to perform appropriate data format conversion in order to ensure that a supported input format is used for performing the desired pre-processing operation, as well as to ensure that the results of the pre-processing operation is stored in the desired output format.

- It must be possible to chain pre-processing operations, meaning that multiple pre-processing operations can be executed sequentially (for a data cleaning operation followed by a scaling operation).

- Different ***intelligent techniques*** need to be supported. The WiSHFUL intelligence framework will support a limited number of basic techniques. Other and more sophisticated techniques can be added later by Third Parties in Open Calls (3, 4 and 5), or may be developed in the eWINE project [4].
  - The following intelligent techniques will be supported:
    - *Classical optimisation,* more specifically based on analytical models expressing the network performance as a function of protocol tuneable parameters and network observed statistics. By simply aggregating the network statistics of interest, the optimal protocol tunings can be determined in a closed form (in case the analytical model is provided by an explicit invertible function) or in a tabular form. A module will be implemented for easily specifying the optimization table, providing a function to be optimized with/without constraints, integrating dedicated optimization tools and libraries [5, 6, 7].
    - *Data science approach*, more specifically supervised learning algorithms for *offline learning* of a classification model that can predict the most suitable class out of a limited number of classes, with as input the network conditions and the application requirements. In WiSHFUL existing implementations of machine learning algorithms will be used (such as for example the scikit-learn [8] tools or the Weka tools [9]). These software tools already support many supervised learning algorithms for training the model, like Nearest Neighbours [10], Decision Trees [11], Logistic regression [12, 13], Neural Networks [14]. These tools also provide the algorithms for pre-processing and for model evaluation (e.g. k-fold cross validation).
    - *Online learning approaches* like reinforcement learning, more specifically Q-learning a model-free method of determining the optimal action-value function for providing evaluative feedback to an adaptive process. WiSHFUL will make use of existing implementations of this algorithm, such as found within the PyBrain toolkit [15].
    - *Statistical approaches*, more specifically the creation of a Markov chain model, a representation of the adaptation of a primary user, here applied in an online manner. Within WiSHFUL we will examine the feasibility of applying a simple Markov model, which is easily implementable as states are observable, versus the necessity of a more complex hidden Markov Model, which implies hidden operation, through existing toolkits such as GHMM [16].
  - The experiments should be able to ***select*** one of the supported intelligent ***approaches***. Each approach will require a specific ***composition*** of modules (for pre-processing, learning, model validation, etc.). Each module will further require a specific ***configuration*** that can be specified by the experimenter and will be dependent on the wireless problem that will be solved.

3.4 Action

The basic requirements for the Action functional component are listed below:

- The ***decisions*** (or predictions in machine learning terminology) that are the output of the learned model or the optimisation algorithm, need to be ***mapped onto a sequence of control commands and configuration settings*** either by directly using the already implemented UPI calls, or by further abstracting the UPI calls.

- There is a need for a ***control interface for hierarchical control***, UPI_HC, between Local and Global Control Programs, to exchange information about decisions or constraints about decisions:
  - UPI_HC provides a flexible interface for a wide range of Global and Local Control Programs
  - UPI_HC supports separation of concerns of local and global control programs, namely helps in ensuring that both will not try to change the same parameter, e.g. change channel, which can lead to inconsistencies in the system. But the final result will be mainly dependant on the careful design and use of Global and Local Control Programs and the utilization of the UPI_HC to avoid such inconsistencies.
- There is a need for a ***Transactional Execution Component*** in the Local and Global Control Programs to ensure that:
  - changes done through UPIs can be confirmed;
  - changes already made can be rolled back in case of an error;
  - execution time requirements are met.

4 General software architecture of the intelligence framework

4.1 Conceptual intelligence framework

The connection between the WiSHFUL software architecture for radio and network control and the intelligence framework is made by the Unified Programming Interfaces. The generic functional view from Figure 4 can hence be mapped to the conceptual framework for enabling intelligence shown in Figure 5.

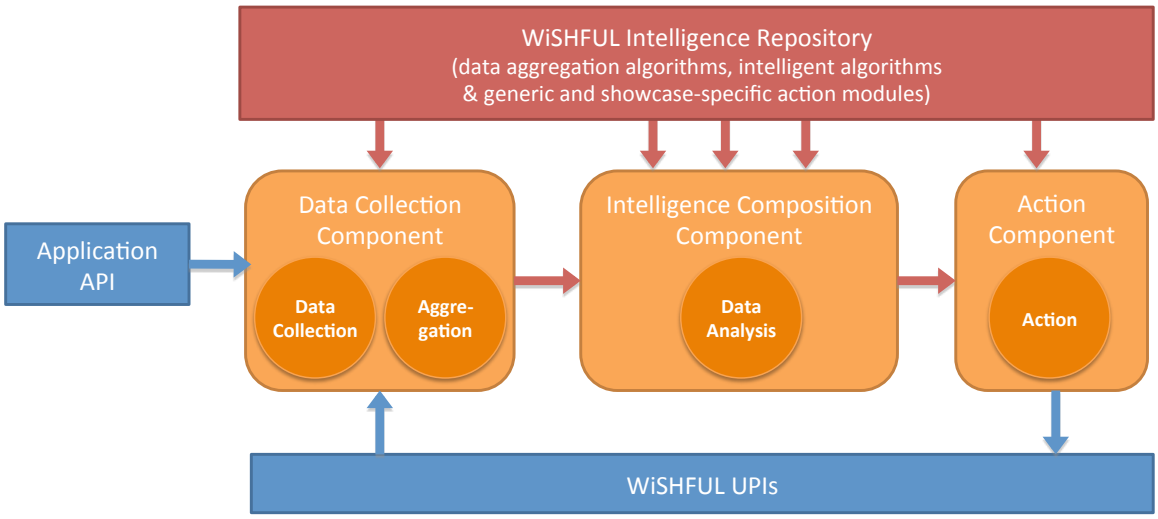

Figure 5: Conceptual framework for enabling intelligence

As the UPIs are unified abstractions that span several wireless technology platforms, the components of the intelligence framework are generic. The Data Collection Component is a generic software module that interacts with the WiSHFUL UPIs, UPI_R, UPI_N and UPI_G to retrieve data about radio and network operation (i.e. channel occupancy, LQI, RSSI, PRR, etc.), and with the Application API to retrieve information about the application requirements. The Data Collection Component also implements aggregation functionality, as described in section 3. The Intelligence Composition Module offers support for composing and configuring several algorithms available in the WiSHFUL

Intelligence Repository into a self-contained intelligence engine that uses the data provided by the Data Collection Component and triggers configuration through the Action Component. The Action Component uses the WiSHFUL UPIs to adjust the configuration of radio and network. The radio and network configuration should be viewed as the output of the intelligence process. Such a configuration can deal with individual parameters (e.g. center frequency, backoff delay, etc.), radio processing elements (e.g. filter swapping), a waveform (e.g. a modulation and coding scheme) or a protocol (e.g. new MAC scheme).

The WiSHFUL intelligence framework offers a common set of tools that enable the realization of intelligent approaches using the algorithms from the repository. Together with the UPIs the WiSHFUL software architecture of the intelligence framework enables reasoning about the current network state and applying actions to change the configuration of radio and network.

4.2 Components of the intelligence framework

4.2.1 Data Collection Component

The data collection component is responsible for data acquisition of the network status and the application requirements. With respect to the network status, the user has only to specify the radio and/or network parameters he wants to monitor by choosing the parameters of interest from a predefined list of possible options (offered by implementations UPI_R, UPI_N and UPI_G) and the time window (period) of collection. With respect to the application requirements, the experimenter has to provide the application requirements in a suitable format to the Data Collection Component. There are several ways of providing the application requirements: one way is specifying the application requirements in the experiment description (e.g. by defining specific properties in the OEDL description of the defApplication code block), another way is a custom API developed by the experimenter. The advantage of a custom API is that it can also be used outside the experimentation environment in a production environment. In addition, the user can specify if the data has to be summarized or compressed using a specific aggregation method, as defined in section 3.2. The aim of data aggregation in wireless networks is typically done for the purpose of limiting transmission of (redundant) data and enhancing network lifetime. Data aggregation functions can be applied at the wireless nodes locally (using UPI_R and UPI_N) or remotely (using UPI_G).

4.2.2 Intelligence Composition Component

The intelligence modules will be offered as a collection of algorithms (e.g. optimization and machine learning techniques) that can be applied to user-specific scenarios. In addition, the Intelligence Composition Component will be expanded with modules for pre-processing data (e.g. data cleaning, normalization, data transformation). The required functionality of the Intelligence Composition Component is described in section 3.3.

4.2.3 Action Component

The Action Component represents an interface between the outputs of the intelligence algorithm and the UPI functions that enable the control of the behaviour of wireless nodes. To this end, intelligence decisions need to be translated in a sequence of UPI calls. Most actions will be quite generic, as the UPIs are technology-agnostic. However some actions may be showcase specific and may require a showcase specific implementation. The required functionality of the Intelligence Composition Component is described in section 3.4.

4.2.4 WiSHFUL Intelligence Repository

The intelligence repository offers several aggregation algorithms and state-of-the-art algorithms for performing pre-processing of data, optimization and machine learning to solve wireless network related problems. The repository provides access to either existing toolboxes or newly implemented

algorithms (by WiSHFUL partners or external experimenters, such as Third Parties participating in the WiSHFUL open calls).

4.2.5 Intelligence framework user interface

Finally, the Intelligence framework also has to offer a user interface to interact with the WiSHFUL intelligence framework. Figure 6 shows a generic UML diagram with interaction modules describing how the user/experimenter can interact with the WiSHFUL intelligence framework.

The interaction module 'Select network parameters' describes a sequence of actions that are executed based on the selected radio and network data that have to be monitored according to the user definition. For each selected parameter the user may specify the duration of monitoring. This interaction module is related to the services of the data collection components. The interaction module may be extended by selecting the type of aggregation that has to be performed, either locally at the wireless nodes (using UPI_R and UPI_N) or globally at a central controller device (using UPI_G). For instance, to create an intelligent MAC protocol selection control program, the experimenter uses the 'Select network parameters' interaction module for (i) selecting appropriate radio and network parameters that will best reflect the current state of the radio environment (e.g. RSSI, LQI), (ii) obtaining information about the performance that is achieved under operation of a set of MAC protocols (e.g. latency, energy consumption), and (iii) selecting each MAC protocol under whose operation the parameters are to be collected. In case of the intelligent control program being deployed at runtime, the user may specify the duration of monitoring for each selected parameter, i.e. the time window during which a sample of data will periodically be reported (e.g. collect RSSI every second). In addition the user may use an aggregation method for each type of data that has to be collected (e.g. the mean value of a set of RSSI values instead of each particular measurement).

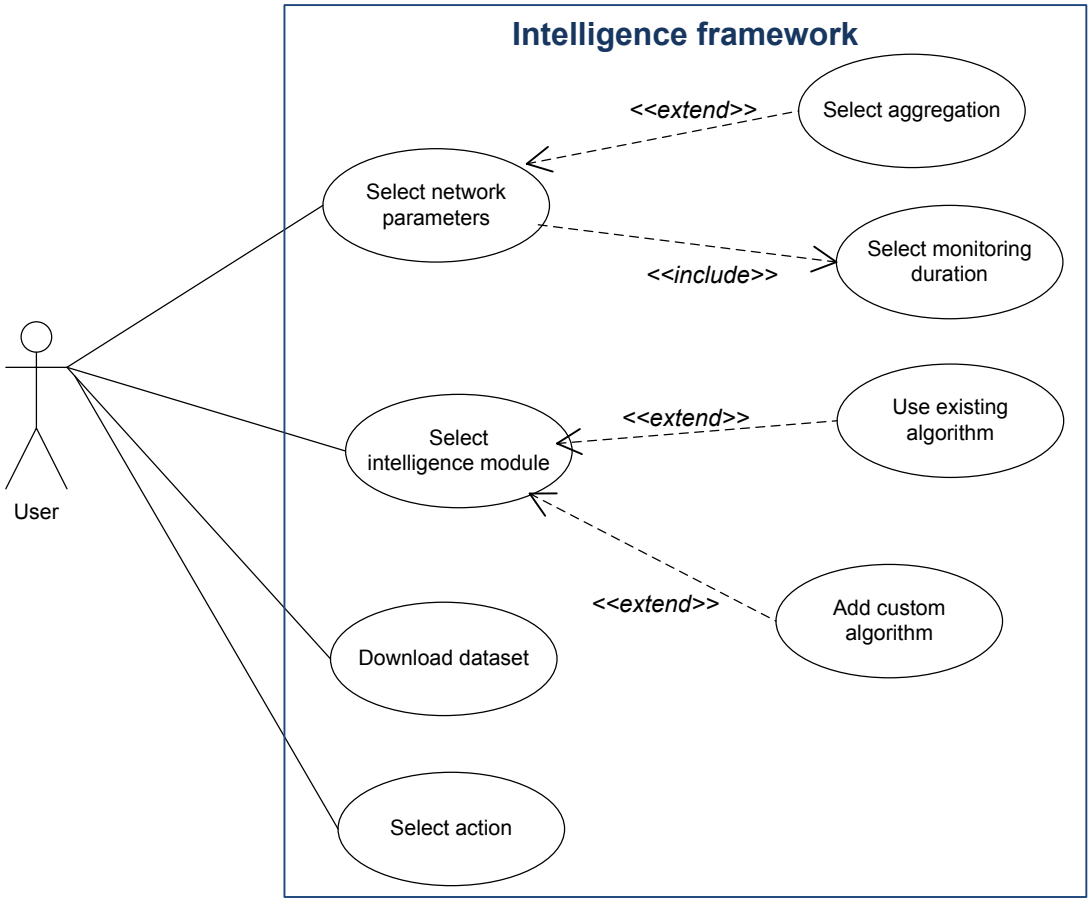

Figure 6: Interaction modules describing the interaction of the experimenter with the intelligence framework

The interaction module 'Download dataset' describes the actions that need to be performed to successfully download all data specified in the previous interaction module in a selected data format. For instance, after collecting the selected radio and network parameters the overall data may be downloaded simply in .csv format and as such used for further data analysis and training an offline model. This interaction module is not used for the case of a runtime intelligent control program.

The interaction module 'Select intelligence module' describes the intelligence actions that are performed based on monitored and aggregated data selected in interaction module 'Select network parameters'. These actions represent a pipeline (see section 4.3) composed of pre-processing and machine learning intelligence algorithms (learning algorithms and model evaluation algorithms) used for offline learning, or algorithms for an online learning approach, or an offline-trained model deployed at runtime. This interaction module is related to the services offered by the intelligence composition component. For instance, to create an intelligent MAC protocol selection control program, the user uses this interaction module to select (i) pre-processing techniques for cleaning, transforming and reducing the selected data that is being monitored (either at runtime or for offline training), (ii) learning algorithms (e.g. neural networks, decision trees) that are being used (at runtime or for offline training), and, in case of also performing an offline learning phase, (iii) the model evaluation approach (e.g. k-fold cross-validation). The user may always add a custom implementation for any of the mentioned algorithm types.

The interaction module 'Select action' refers to the actions that are being executed at the wireless nodes as a result of the decisions made by the Intelligence Composition Component. This interaction module is related to the service of the Action Component. An example of action may be a command sent to a wireless node for switching to a particular MAC protocol as decided by a trained learning algorithm.

4.3 Workflow for defining intelligence solutions

Figure 7 depicts the five main operations and their order the user needs to respect to add intelligence control to the general WiSHFUL architecture. Each operation in the intelligence workflow may be simply realized by importing the appropriate module (e.g. collect RSSI measurements as the radio parameter, use the average aggregator to collect only average values of RSSI, import decision trees as the machine learning algorithm, etc.). Each main operation in the workflow may consist of multiple processes.

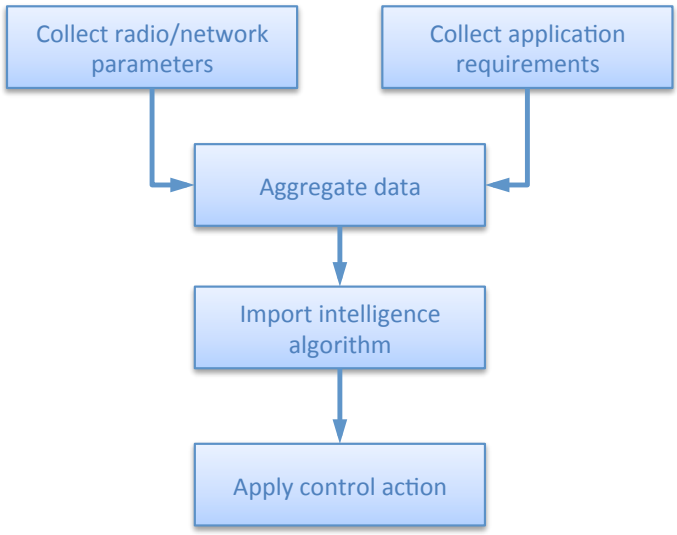

Figure 7: Intelligence workflow

While several basic intelligence algorithms will be offered by the WiSHFUL consortium, the intelligence framework enables custom implementation of intelligence algorithms designed by the experimenter. The framework provides all prerequisites to support intelligence services, through data collection, aggregation and action components, so that the experimenter can focus on the design of his intelligence algorithm and plug in his algorithm into the workflow.

While the workflow in Figure 7 gives a high level view on how to use the intelligence framework, a **pipeline** gives a detailed view on how the intelligent control of a wireless network is composed of individual data processing tasks.

An example of a pipeline for the offline learning of a model, for example a classification model that predicts the most suitable network configuration (out of a limited number of possible network configurations) with as input the network conditions and the application requirements is shown in Figure 8. Raw radio and network data is collected via the UPI_R, UP_N and/or UPI_C unified programming interfaces and stored in a huge data storage component. In this example two types of parameters are collected. The first parameter needs to be aggregated according to 'Aggregation algorithm 1', but the raw data format as collected from the UPIs, is not supported by this algorithm. Therefore a format conversion to one of the supported input data formats by Algorithm 1 is required. The second parameter that is collected needs two Aggregation operations according to 'Aggregation algorithm 2' and 'Aggregation algorithm 3'. For these data processing operations no data format conversion is needed, as the data format of the raw data and the output format of Aggregation Algorithm 2 are both supported by the Aggregation module. The experimenter has selected 'offline learning of a classification model', but before implementing the required intelligence algorithms for this approach, he wants to clean the collected data from the network and to transform the application requirements. He therefore selects the appropriate 'Pre-processing algorithms' and applies the algorithms on the aggregated data and application requirements. All pre-processed data are now stored in a single data storage component that has a suitable format for the Intelligence algorithms that will be applied next. In case of a data science approach, the data could be split in training data and test data using the cross-validation algorithm. The training data is further fed to a machine learning algorithm for training a model, the test data is used for validating the trained model and the training process is repeated to find the most suitable model. The offline learning approach stops with the selection of the best model. There is no need for the Action Component up to this point.

The learned model can now be used in a runtime implementation of the intelligent approach (see Figure 9). Similar steps for format conversion, aggregation and pre-processing are used as for the offline intelligent approach (see Figure 8). However, while the offline learning approach is performed on the whole dataset at once, we now have to perform all data operations each time new data instances arrive, leading to different data formats and operations on small data storage components. Pre-processed data is now directly fed to the selected learned model. The output of the model is sent to the action component, where the configuration decided by the model is translated in the appropriate UPI function calls using different action modules for applying different configuration settings.

These are just two examples illustrating the concept of how an experimenter can build his own pipeline for solving his wireless network problem using the intelligence framework user interface (see section 4.2.5). The WiSHFUL intelligence framework offers many different components in the pipeline together with the methods for interconnecting the different components. The experimenter can build his own pipeline using components already supported by WiSHFUL (available in the WiSHFUL intelligence Repository) and/or insert his own intelligence component(s) in the pipeline.

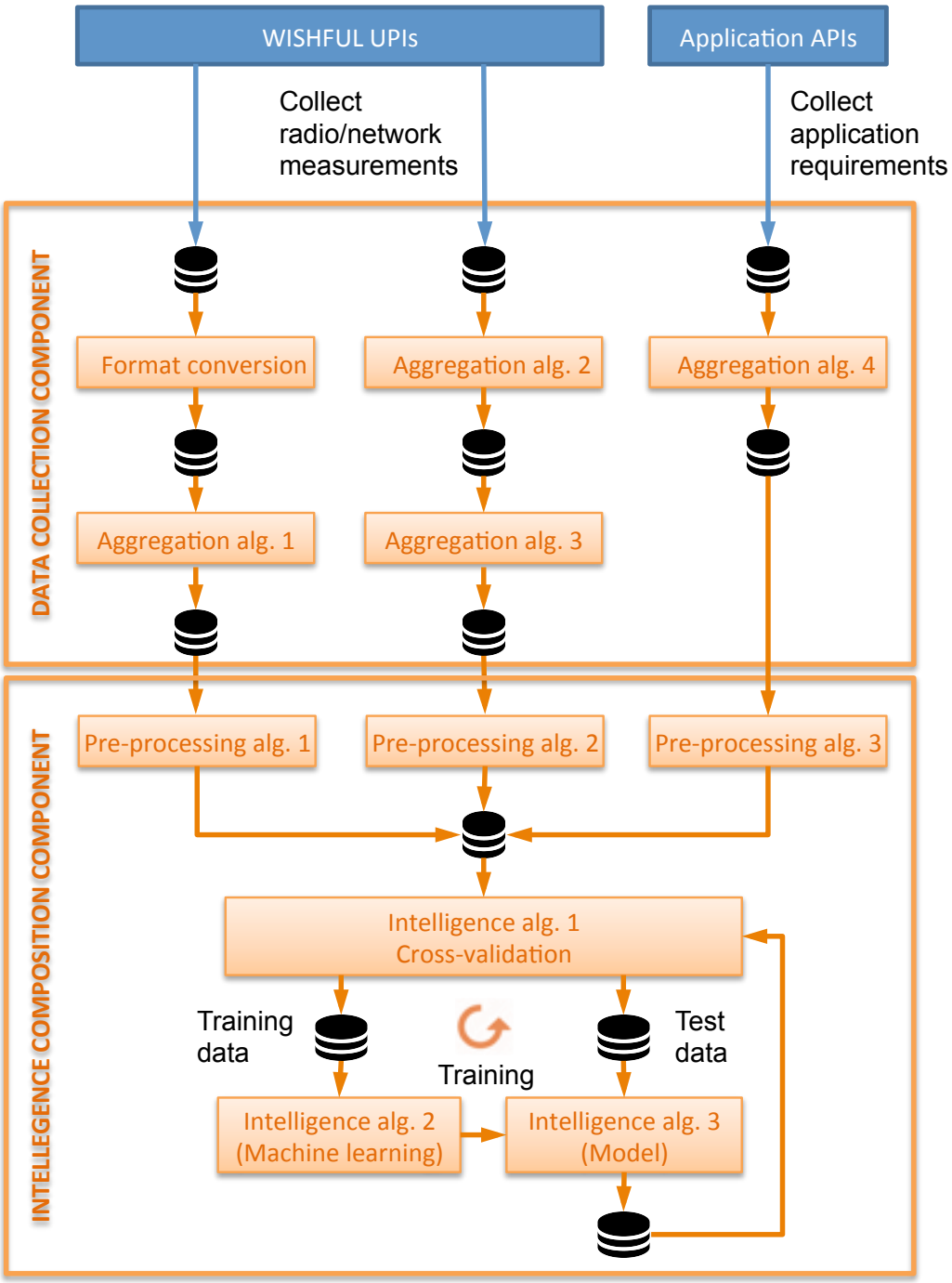

Figure 8: implementation of the pipeline for an offline intelligence approach

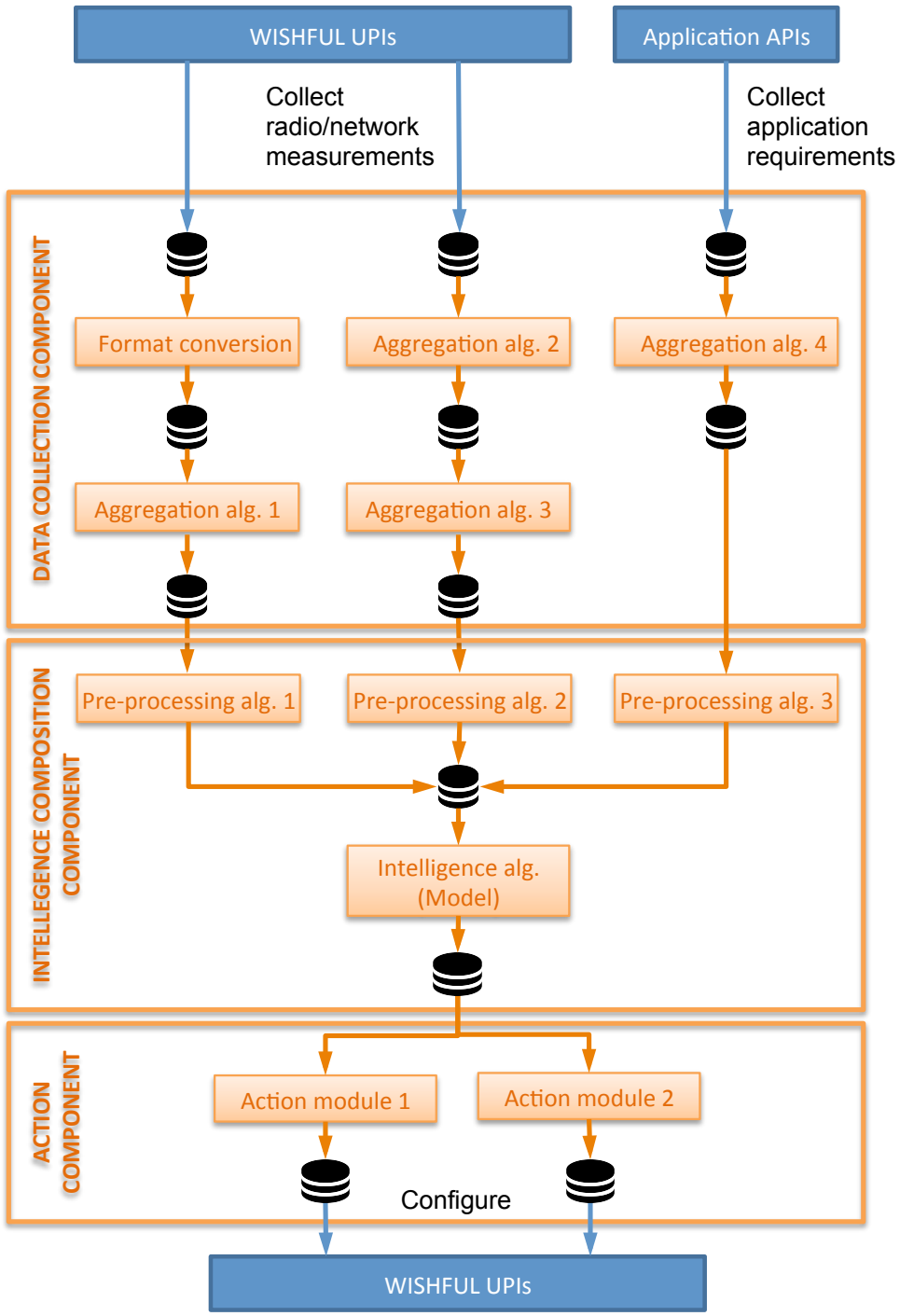

Figure 9: implementation of the pipeline for a runtime intelligent approach

4.4 General software architecture for intelligent radio and network control

The intelligence framework supports local or remote implementation of the intelligence pipeline in a semi-automatic way, where the human is only involved in the composition of the pipeline, as explained in previous section. In case of local configuration, the intelligence pipeline is implemented in the Local Control Program, and interacts with UPI_R and UPI_N. In case of remote configuration, the intelligence pipeline is implemented in the Global Control Program, and interacts with UPI_G, as illustrated in Figure 10. The framework also supports hierarchical control, using the control interface for hierarchical control, UPI_HC, between Local and Global Control Programs for exchanging information about decisions or constraints about decisions (for example a higher level Control Program specifying a parameter space to be used by a lower level Control Program). In the latter case, (different) intelligence pipelines are implemented in both Local and Global Control Programs.

Another extension to the original software architecture for radio and network control is the addition of an application API. This API can be a custom component developed by the experimenter, or could be defined in the experiment description using the testbed experimentation tools.

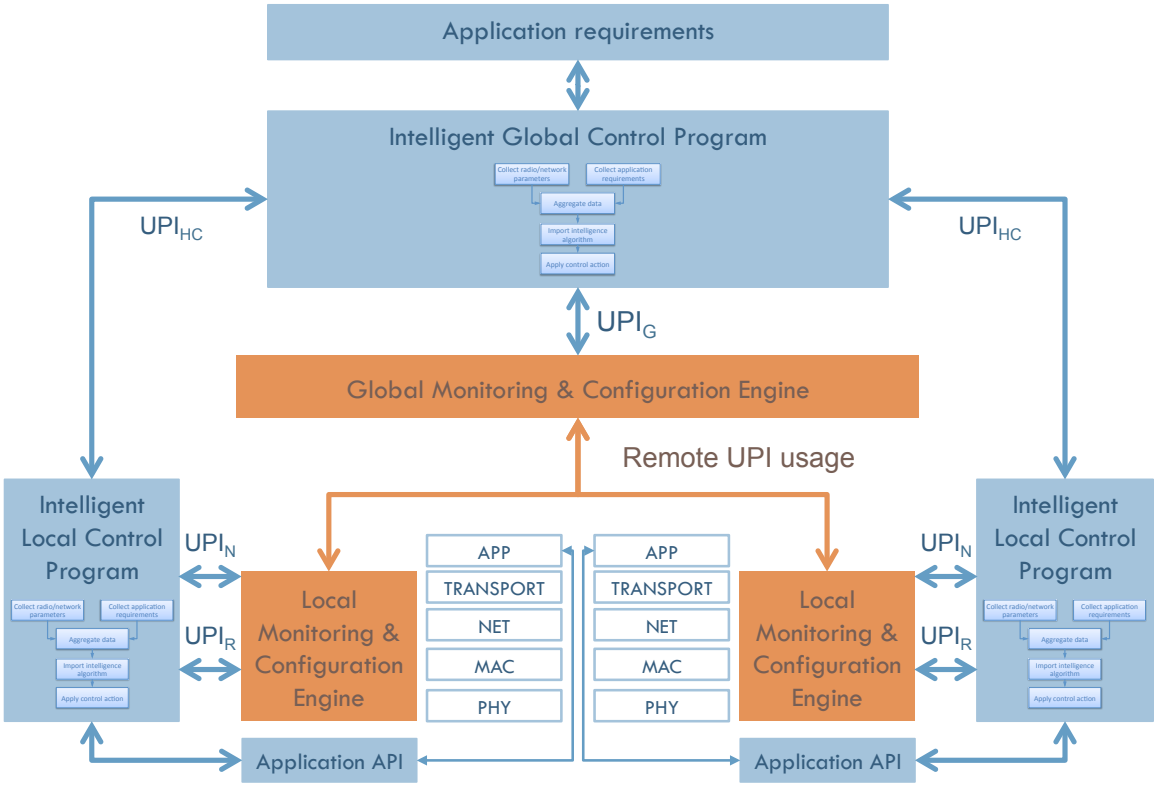

Figure 10: General software architecture for intelligent radio and network control

5 Conclusion

In this deliverable, three intelligence **showcases** have been defined:

- ***Intelligent load and interference aware MAC adaptation***: this showcase aims to demonstrate how to increase network performance by adding intelligent mechanisms for choosing one of the two strategies: (1) optimization of parameters for the current MAC protocol, or (2) selection of the most suitable MAC protocol based on the current network conditions and the application requirements.
- ***Intelligent slot allocation in a hybrid TDMA MAC***: this showcase aims to demonstrate that an explicit hidden-node detection phase can be avoided by adding an intelligent mechanism for analysing and optimizing the slot allocation of a hybrid TDMA MAC. The goal is to implicitly detect the wireless links suffering from the hidden node problem by collecting data at each wireless node and performing data fusion at central device in the network.
- ***Learning about primary user behaviour and selecting best MCS scheme for a secondary user***: this showcase will perform accurate primary user detection with low overhead from channel power estimates employing a variable threshold, followed by two learning phases. In the first phase, primary user characteristics will be learned through multiple channel occupancy analysis. In the second online learning phase, the best modulation and coding scheme for secondary user transmissions will be selected.

The showcases, were used to derive the requirements for the **intelligence framework** , and for the design of the **generalized software architecture for intelligent radio and network control**. The following extensions to the original software architecture, as presented in deliverable D2.2, have been defined:

- ***Data Collection Component***: this component is responsible for data acquisition of the network status and the application requirements. With respect to the network status, the experimenter can specify the radio and/or network parameters he wants to monitor by choosing the parameters of interest from a predefined list of possible options (offered by the UPI interfaces) and the time window of collection. With respect to the application requirements, a new interface (Application API) is needed to define and feed the application requirements to the intelligence framework. Collected data can further be summarized or compressed using specific aggregation methods limiting transmission of (redundant) data and overall data collection overhead and enhancing network lifetime.
- ***Intelligence Composition Component:*** this component offers a set of approaches that can be selected by an experimenter for finding optimal radio and network settings. The intelligence modules will be offered as a collection of algorithms (e.g. optimization and machine learning techniques) that can be applied for user-specific scenarios. In addition to actual machine learning algorithms, the Intelligence Composition Component also offers modules for pre-processing data such as data cleaning, normalization, and data transformation.
- ***Action Component***: this component represents an interface between the outputs of the intelligence algorithm and the UPI functions that enable the control of the behaviour of wireless nodes. This component translates the intelligence decisions taken by the Intelligence Composition Component in a sequence of UPI calls.
- ***Intelligence framework user interface***: this user interface enables interaction of the experimenter with the WiSHFUL intelligence framework. The user interface offers different interaction modules for selecting network parameters, duration of monitoring, type of aggregation, intelligence modules (with the possibility of using existing algorithms or adding custom ones) and the actions to be executed on the wireless nodes. The user interface allows the experimenter to compose a pipeline of the different data processing tasks needed for the intelligent control of the target wireless network.

6 References

[1] R. R. Bouckaert, “Choosing between two learning algorithms based on calibrated tests,” in Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 51–58.

[2] Python-based ecosystem of open-source software for mathematics, science, and engineering, http://www.scipy.org

[3] Python Data Analysis Library, http://pandas.pydata.org

[4] elastic Wireless Networking Experimentation (eWINE), H2020 project, GA No. 688116

[5] Scipy - Optimization and root finding, http://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html

[6] CVXOPT - Python Software for Convex Optimisation http://cvxopt.org

[7] APM - APMonitor Modeling Language http://apmonitor.com/wiki/index.php/Main/PythonApp

[8] scikit tools for data mining and data analysis, http://scikit-learn.github.io/stable

[9] Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/

[10] D. T. Larose, “k-nearest neighbor algorithm,” Discovering Knowledge in Data: An Introduction to Data Mining, pp. 90–106, 2005

[11] O. Maimon and L. Rokach, “Data mining with decision trees: theory and applications,” 2008

[12] D. A. Freedman, Statistical models: theory and practice. cambridge university press, 2009

[13] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip et al., “Top 10 algorithms in data mining,” Knowledge and Information Systems, vol. 14, no. 1, pp. 1–37, 2008.

[14] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin, Neural networks and learning machines. Pearson Education, Upper Saddle River, 2009, vol. 3.

[15] PyBrain toolkit for machine learning, http://pybrain.org

[16] GHMM library for general hidden markov models, http://ghmm.org