Communications Research Centre Canada
An Agency of Industry Canada

Centre de recherches sur les communications Canada
Un organisme d'Industrie Canada

# Policy-Based Spectrum Management

Prepared by :
Sarah Dumoulin, Communications Research Centre, 3701 Carling Ave., Ottawa, K2H 8S2

Prepared for:
Tricia Willink , Contract Technical Authority
DRDC - Ottawa Research Centre

**The scientific or technical validity of this Contract Report is entirely the responsibility of the Contractor and the contents do not necessarily have the approval or endorsement of the Department of National Defence of Canada.**

**Contract Report**
**DRDC-RDDC-2016-C124**
**April 2016**

Canada

CRC

# 1.Table of Contents

# 1. Introduction

Communication using wireless links requires the use of radio spectrum, which can be licensed or unlicensed depending on the frequencies selected. For licensed spectrum, users of wireless communication devices can expect a certain level of protection from interference from other wireless devices, but this protection can sometimes break down in areas near national borders, for instance, where licenses from one country conflict with those issued by those in its neighbour. Unlicensed operation offers no protection from interference, and radio operators may find their communications interrupted at any time by other wireless devices operating in their vicinity.

A possible way to solve this problem is to allow radio operators to jump to a new frequency when they experience interference. The question then becomes to which frequency the operator should switch in order to obtain the best Quality of Service (QoS) for the application they are currently using. Randomly selecting a frequency (frequency-hopping) is a sub-optimal solution as it does not guarantee that the new frequency is any better than the frequency that was just vacated. Frequently switching frequencies also needs to be avoided as each time the frequency is changed there is a cost in terms of connection disruption resulting in possible lost packets.

The Policy-Based Spectrum Management system attempts to show how a policy-based system could be used to dynamically select spectrum based on application QoS requirements. This system monitors both the quality of the spectrum (SINR) and network environmental information (network latency) in order to rank available frequencies and choose the best one for user application traffic. Users can enter policies on a GUI screen in order to determine how the data collected by the system is combined to rank available frequencies. Once a policy has been enacted, the system will switch traffic from frequency to frequency dynamically without the user having to intervene.

This system was built and successfully demonstrated at CRC in 2015. Demonstrations included streaming voice and video applications subjected to simulated network latency and generated interference.

# 2. System Architecture

The system is comprised of several pieces of software and hardware that work together in order to accomplish the goal of dynamically choosing the best frequency based on the enacted policy. Three USRPs are used together with software defined radios and Tap interface drivers in order to create an emulated cognitive radio, a monitoring agent is used to report on current link and spectrum status, a rules-based decision-making module is used to determine which link should be used for current traffic, and a GUI is used to display output to the user. Each of these is described in more detail below.

## 2.1 Emulated Cognitive Radio

A cognitive radio is a radio capable of monitoring its own performance and dynamically adjusting its settings in order to obtain better performance. Cognitive radio technology is a key enabling technology for dynamic spectrum switching and dynamic spectrum access systems.

Cognitive radios were not available for use in this project. Therefore, an emulated cognitive radio was constructed by using three USRP (Universal Software Radio Peripheral) platforms in parallel.

Each USRP was attached to a computer running a software modem. These modems communicated with the machine running the policy engine code via CORBA. The policy engine itself acted as a gateway to the WAN by using a Tap device interface driver to intercept incoming Ethernet frames at the link layer. Figure 1 shows how the emulated cognitive radio was constructed from its component parts.

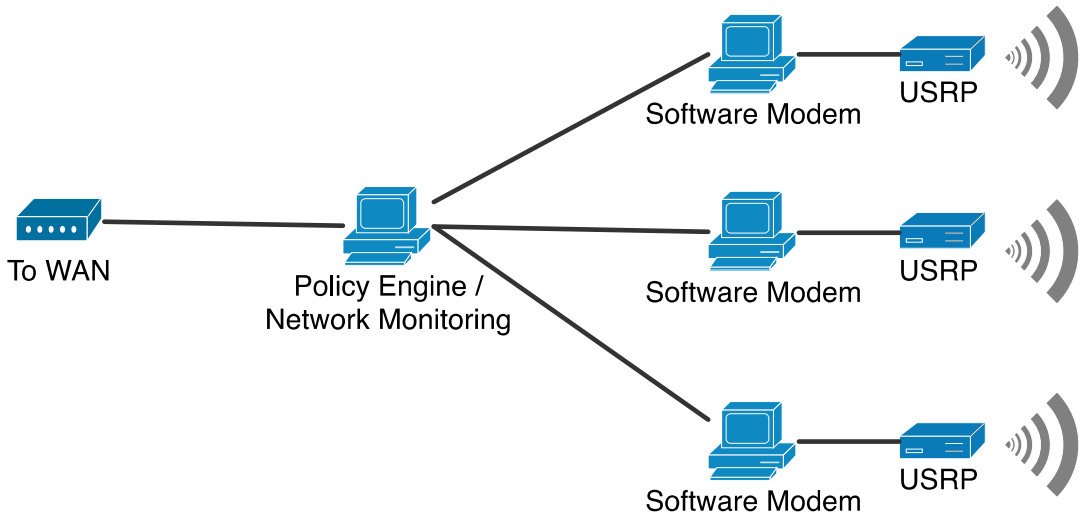


**Figure 1: Emulated Cognitive Radio Platform**

Thus, traffic flowing over the network would arrive at the policy engine or network-monitoring device at the edge of the emulated cognitive radio platform. At this point, it would be intercepted by the Tap driver operating at the device and, instead of being routed as an IP packet, would instead be redirected by the policy engine via CORBA to the USRP connected to the current active frequency for application traffic. Once the traffic was received by the paired USRP listening to the chosen frequency the packets would be re-injected into the IP network using the Tap driver and would then proceed to its destination in the normal matter.

Figure 2 shows the internal connections between the components within the policy engine. The Tap interface listens on the policy engine's WAN Ethernet interface and communicates directly with the decision-making module by passing along packets that need to be sent to the RF links (or receiving packets that need to be sent to the WAN). The Monitoring Agent communicates directly with the local decision-making module in order to inform it of local link status, which it receives by querying the CORBA interface directly and by sending packets through the CORBA interface onto the RF links. The decision-making module runs the decision-making process every interval and chooses the frequency on which to send application traffic. It then routes incoming packets taken from the Tap interface over the appropriate CORBA interface in order to route to the appropriate USRP device that will use the chosen frequency. The decision-making module also receives incoming packets from the CORBA interfaces, parses those that are needed for decision-making operations, and passes the remaining network traffic onto the Tap module for injection into the network. Note that the decision-making module also communicates with the policy-input GUI, but this GUI need not be co-located with the policy engine and is thus not show in Figure 2.

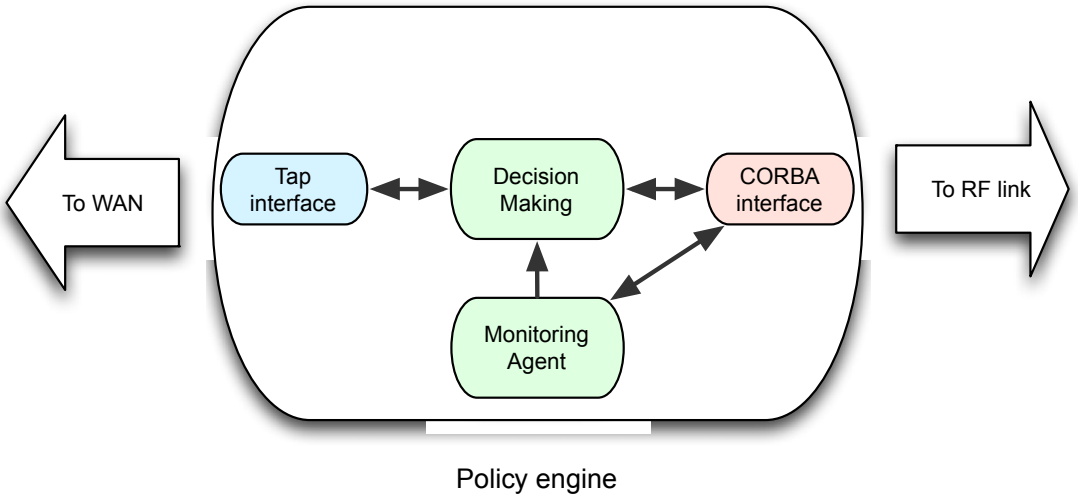


**Figure 2: Internal diagram of policy engine components**

It is important to note that routing via Tap and CORBA is not necessary in order to make a system such as this function. This method of forwarding packets was chosen due to limitations in the available software modem package. If a software modem package was chosen that could read packets directly from an Ethernet port, instead of from a CORBA object, routing of packets to the correct modem could have been done using a router. In such a configuration, the policy engine would need to modify the routing table rather than switch between different CORBA objects in order to route the traffic over the correct frequency.

Earlier, it was stated that cognitive radios are distinguished by their ability to sense and adapt to their environment. The emulated cognitive radio platform accomplishes this by sensing the Received Signal Strength Indicator (RSSI) and Noise Floor of a frequency whenever it receives a packet on that frequency, and by periodically measuring the latency of each radio link. The emulated cognitive radio platform is then able to adapt based on this input as it can dynamically switch its transmission and reception frequencies by routing application traffic over any one of the three USRP devices (where each device is tuned to a different transmission and reception frequency pair). The policy engine is set to periodically assess the suitability of the available spectrum for the chosen application and will switch frequencies when necessary by moving application traffic to a different USRP radio as needed.

## 2.2 Monitoring Agent

In order to make decisions, it is necessary to collect data regarding current network conditions. This is done by a monitoring agent that runs on a computer on each side of the network. This agent is tasked with monitoring and reporting to the decision-making module both the RSSI and noise floor of all frequencies that may be used in the system, and the round trip latency of each of the possible frequency pairs.

The monitoring agent gathers information in two ways. Firstly, it queries the software defined modems directly via their CORBA interfaces in order to obtain the latest SINR and noise floor measurements reported by each of them. Secondly, it sends out periodic small packets (20 byte pings, answered by 10 byte replies) on all links. These periodic packets serve a dual purpose: they allow the monitoring agent to measure the round trip latency of the links, and they ensure that there is some small amount of traffic on each link during each test interval to allow for an up-to-date SINR and noise floor reading. Data are collected at each monitoring agent and are subjected to some

hysteresis in order to smooth out the operation of the system. The decision-making module is sent an update on status every three seconds (configurable). Update frequency may be increased, but it should be noted that sending updates from the remote end requires bandwidth, and thus it is not without cost. Furthermore, there is little use in increasing the update frequency to be faster than the interval at which the decision-making module itself runs, as previous updates will simply be overwritten by more current information.

In a larger or more complex system, it would be interesting to examine whether suppressing updates by examining if they are relevant (e.g. has this number changed more than a certain threshold?) would be successful in reducing the bandwidth required to keep the decision-making module informed of current network conditions. Or, whether a model wherein the decision-making module asks for information when it is needed, rather than is pushed information continuously, is more efficient without sacrificing system performance.

## 2.3 Decision making

The decision-making module is where the heart of the system resides. It is here where the system ranks the frequencies based on the data received from the monitoring agent and the bounds provided by the currently enacted policy and determines which frequency is currently the best for application traffic. The decision-making module runs the ranking code every three seconds (configurable interval). An optimal value for this interval has not yet been determined, however in the lab it was found that running the decision-making module at this frequency was sufficient to react quickly enough to changing network conditions that users were unable to notice degradation before they were switched to a new frequency.

The decision-making module receives inputs from two sources: the policy input interface, where the user can input the policies to be enforced in the system, and the monitoring agents, where data is collected from the network and the radios. A diagram showing the interaction between these three systems is shown in Figure 3.

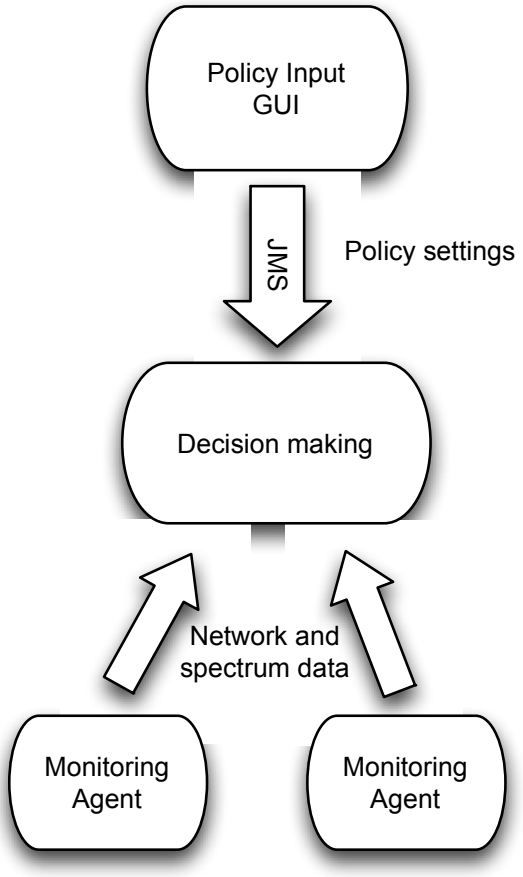


**Figure 3: Interaction between the Policy Engine, the GUI, and the monitoring agents**

The policy input interface communicates with the decision-making module via the Java Messaging Service (JMS). JMS was chosen as the communications protocol to allow the easy deployment of the GUI on a computer situated remotely from the policy engine. It is not necessary for operators who change policies to be co-located with the policy engine. In certain scenarios, it could be that the policy input would be required at a remote location such as at a NOC or other central command point. JMS allows the deployment of the policy input GUI at any location with a network connection to the policy engine. JMS further provides reliable messaging services that ensure that enacted policies will be received by the decision-making module even should it be disconnected at the time when the change in policy is pushed.

Monitoring agents communicate directly with the decision-making module via inter-process communication on the side where the monitoring agent is co-located with the policy engine, and via the active RF link from the remote side of the set up. It would be possible to use a lightweight messaging protocol (such as MQTT, CoAP, or even JMS) to send these updates to the policy engine, but in order to keep the bandwidth requirements on the RF link to a minimum it was decided to create a custom packet type that could be easily sent between the nodes. As all traffic passing over the RF links is handled by the decision-making module it was simple to add a custom packet type that could be recognized and parsed by the system in order to pass this data efficiently.

### 2.3.1 Ranking algorithm

Each time the decision-making module code is run, it takes the latest data from the monitoring agents and calculates a score for each frequency using the following formula:

```
s = bucketLatency(local_latency, max_lat) * weight_latency +
    bucketLatency(remote_latency, max_lat) * weight_latency +
    scoreSINR(local_snr, local_nf, min_sinr) * weight_snr +
    scoreSINR(remote_snr, remote_nf, min_sinr) * weight_snr;
```

Where weight_quality and weight_latency are the weights of link quality and link latency respectively on the slider in the GUI.

The function bucketLatency() takes the measured latency and normalizes it to a value between zero and one, except when the latency is outside of the maximum latency range specified. When the latency is higher than the maximum specified by the user the latency is set to -100 in order to ensure that the link is chosen only if all other links also have failed to fulfill at least one of the users' policy inputs. The latency is normalized in order that latency does not dominate the equation in cases where latency is high. In effect, latency values that are under the maximum latency value are separately scored on a scale between zero and one, with all latency values over a configurable maximum latency being assigned a zero. The function used for binning latency values is as follows:

```
if (latency > max_lat) {
      score = -100.0;
} else {
      bucketSize = (MAX_LATENCY/NUM_BUCKETS);
      bucketNum = (int) (latency / bucketSize);

      if (bucketNum > NUM_BUCKETS)
            bucketNum = NUM_BUCKETS;

      score = (1.0 - ((bucketSize * bucketNum) /
MAX_LATENCY));
}
```

Note that all latencies are measured in milliseconds. The default value used for NUM_BUCKETS is 8 and for MAX_LATENCY is 2000.

The number of latency buckets will determine the granularity of the latency score, while the maximum latency will determine the highest latency that will affect the latency score. Both of these values are configurable through the use of a java properties file.

The function scoreSINR() takes the RSSI and noise floor values, calculates the SINR value, and normalizes it to a value between 0 and 1, except if the SINR value is less than the minimum quality value requested. It does this by dividing the SINR value by the maximum effective SINR value measured for the system – i.e. the SINR value at which the BER is zero – and rounding down to one if the value is greater than one. This is done for two reasons, firstly in order to stop SINR values from dominating the equation, and secondly in order to stop the system from switching from a link with perfectly acceptable SINR performance to a link with "better" SINR values when the better SINR will not provide any improvement in terms of application performance. If the SINR value is less than the maximum quality value requested, the score is set to -100 in order to ensure that this link will be chosen only if all other links also have failed to match at least one of the user's policy inputs. The function used for scoring link quality values is as follows:

```
if ((snr-nf) < min_sinr) {
      score = -100.0;
}else {
      score = ((snr-nf)/25.0); // 25.0 is max effective SINR
      if (score > 1.0)
            score = 1.0;
}
```

### 2.3.2 Switching algorithm

Once scores have been calculated the system must decide whether to switch to a new frequency or stay where it is. A naïve switching algorithm would simply choose the link with the highest calculated score during each cycle and switch to that link. However, this would result in frequent switching in reaction to the continuous small fluctuations in link score. As frequency switching is not a zero cost operation this is not an optimal solution.

In order to combat link flapping behavior, the system has implemented hysteresis algorithms in an attempt to smooth the noisy data collected from the sensors as well as an algorithm that tries to balance the cost of spectrum switching with the potential gains offered by other available frequencies.

Hysteresis refers to the use of floating averages to smooth the collection of spiky data. Each of the data types collected for the decision-making module are subjected to hysteresis, as individual samples can vary significantly. However, it should be noted that samples cannot be smoothed too aggressively as to do so would negatively impact the reactivity of the system. The system must be able to react to increases in latency, or decreases in link quality, quickly enough that user experience is not gravely impacted. The number of samples used for running averages is a configurable variable in the current system that can be set using a java properties file.

After scores have been obtained using the smoothed data, caution must still be used before deciding to switch to a new frequency. Even with smoothed data, scores fluctuate somewhat with every iteration of the algorithm and when there are two or more similar links a situation wherein the algorithm floats between these links should be avoided. Therefore, a simple algorithm was implemented which examines the percentage change in the link score between any newly proposed "best" link and the currently chosen link. If the change in the score is less than five percent, the algorithm will not change its preferred link. If it is greater than ten percent it will move to the new preferred link right away. If it is between five and ten percent it will move to the new preferred link if and only if the link was not changed during the last two iterations of the algorithm.

## 2.4 GUI

The GUI serves both as the mechanism for users to input their policies and as the way in which the system displays its status. The GUI is written in Java and communicates with the policy engine using the Java Message Service (JMS). This permits the GUI to be operated on any machine that can be reached from the policy engine via a network connection for maximum flexibility.

### 2.4.1 Policy Input

The GUI screen allows users to input a policy to be used by the decision-making module when determining which frequencies to choose for application traffic. Users can set the parameters for the policy by using a slider which allows them to weight the application's needs for low latency versus high link quality, and may also set a maximum allowable link latency and a minimum allowable link quality. When the user updates the policy, it is pushed via JMS to the decision making module and will be used to rank the links during its next cycle.

Note that the policy input screen allows users to enact policies that will be too restrictive. That is to say, users may enact policies that will result in no current links present in the system matching their requirements. When this occurs, the policy engine will still choose the frequency that is the best match for the users parameters out of the frequencies available to it. However, it will also signal to the user that the chosen frequency does not meet expected requirements through the use of the GUI display. While active links are usually marked in green as "ACTIVE" when chosen as the best link by the policy engine, links that do not meet the minimum policy requirements of the user are marked in yellow and the user is warned by a dialogue pop-up window that the given link does not meet their requirements but is being chosen regardless due to the lack of other suitable alternatives.

### *2.4.2 System Status Display*

The GUI also serves to display the current status of the system to the users. The status of all of the RF links is displayed: their RSSI, noise floor, SINR, and latency as seen by the decision-making module along with a graphical indication of whether each link is currently active, available, or down. A link is marked "ACTIVE" if it is the link currently chosen as the best link for application traffic by the system. The "ACTIVE" link is the link over which application traffic is being routed. A link is marked "AVAILABLE" if the link is currently up but has not been chosen for application traffic. A link is marked "DOWN" if the link has failed to either receive a packet or respond to a ping during the previous three ping intervals (configurable). Links that are "DOWN" are not considered during the decision-making module's calculations as applications require an active link in order to transmit traffic.

The application status display is updated at the same rate at which the monitoring agent updates the decision-making module. The GUI is informed of monitoring status updates via JMS messages, ensuring that the GUI can be located at any computer with a network connection to the policy engine.

# 3. System Demonstration Scenarios

In order to demonstrate the capabilities of the Policy-Based Spectrum Management system, a demonstration was developed that covered three different scenarios. These scenarios are described in more detail in the following sections.

## 3.1 Latency switching

For the first scenario, a voice call was established between two users on either side of the RF links. In order to demonstrate how the system could improve the status quo, the system was started with no policies enabled, meaning that no matter what occurred with the RF links no switching would occur, as the decision-making code was not operating. With the call in place, a latency injection tool created by CRC was used to add latency to the link that the call was using. Speaking with the

latency in place, it could be easily demonstrated that the quality of experience of the person using a phone application over such a delayed link was much reduced.

Once it was established that the quality of the link was degraded, the policy input screen was used to enact a "link delay" policy that prioritized the selection of a link with low link latency. Once this policy was enacted, the call nearly immediately switched over to a frequency with a better delay. The voice call was not interrupted, but the large latency disappeared.

## 3.2 Interference switching

For the second scenario, a video conferencing call was established between two users on either side of the RF links. In this case, a link quality policy was enacted, seeking to choose the frequency with the best SINR characteristics on which to route the call.

Once the video link was established, an RF signal generator was used to introduce interference on the link that was currently chosen to route the video call. When interference was introduced on the link, the call would quickly be moved to another link. The video call would not be dropped and the quality of the video was not affected by the switching of the video call.

## 3.3 Custom policies

For the third scenario, a voice stream was established between two users on either side of the RF links. In order to demonstrate the use of custom policies, a user-defined policy was entered that included a requirement for a link with a latency of less than 10 ms. As no link with such a latency was available, the link with the closest match to the requested requirements was chosen and the user was alerted to the failure to complete satisfy the policy requirements through the policy GUI: the chosen link was marked in yellow and a pop-up window was triggered indicating to the user that the chosen link was the "best fit" link.

# 4. Conclusions

The Policy-Based Spectrum Management system was developed as a proof-of-concept to demonstrate how policy-based systems could be extended to automate spectrum selection. This system was successful in that it showed that it is possible to design a policy-based system that will automatically choose the "best" frequency for communications between two radios given a user-defined criteria for what constitutes the "best" frequency. Furthermore, it was demonstrated that switching frequencies could be done in a manner transparent to the user – application traffic was uninterrupted by the change in frequency, and the frequency selection algorithm was most often able to detect problems and move away from frequencies experiencing performance issues before those issues became apparent to the user. This is promising in that it indicates that this type of technology could be used to maintain good quality of experience automatically by dynamically negotiating the best carrier frequency in a manner completely invisible to the user.

The demonstration system was built as a stand-alone system using narrow-band links. However, the principles demonstrated are more generally applicable. This type of system could be integrated into a more general policy-based traffic management system in order to dynamically control the spectrum chosen by the radio links within a network. Other types of radios with larger bandwidths, or

different frequency ranges, should also be able to use this type of system in order to manage which spectrum they should choose for communications.

Further work on this system needs to be done in order to determine how changing the sensing periods, amount of hysteresis, and triggering thresholds affect the system. Although the algorithm discussed above was found to be effective in reducing drift between frequencies, it was not shown to be optimal. Further testing is required to examine how the system functions under differing input conditions. Future work could also focus on which other possible sensed data could be used to characterize a radio frequency link's suitability for different applications. Should this type of system be integrated within a policy-based traffic management system, many more pieces of network and environmental data would become available. Determining which of these pieces of information could be useful to choosing the best frequency on which to communicate, and how to weight those pieces of information for differing applications could lead to a much more robust and useful algorithm for frequency selection in the future.