PAWS Review

A PAWS is an application layer protocol that allows use of white space spectrum for secondary users, and at the same time prevents interference effect for the primary user. Device intent to use white space spectrum must use PAWS to contact database for WS spectrum availability information via Internet. Database schedules time usage and spectrum allocation for the device based on devices' geo-location and applicable rule sets. It is natural to see that device works as client, and database is the server. From the client perspective, every time it wants to use WS spectrum, it needs to use a method which returns the availabilities of the spectrum. So the PAWS require JSON-RPC for this purpose. RPC stands for remote procedure call, and a RPC is a mechanism which does following:

A process on machine A calls a procedure on machine B. After the call machine A suspends itself, and machine B starts to execute. Then machine B returns result back to machine A, and machine resumes.

JSON stands for JavaScript object notation, and it is a text format for serialization of data. It is used here to convert structured data into a serialized text. JSON-RFC is transportation agnostic, which means it allows many forms of message transmission, via TCP, HTTP, HTTPS, etc.

PAWS requires JSON-RPC with HTTPS for implementation, and it can be viewed as three parts:

1. data format and its serialization tool ()
2. protocol - requests and responds (PAWS functionalities)
3. underlying data transmission (HTTPS)

In the transmission, the client should first initiate a HTTPS connection with server, and then create a data with HTTPS format, with method to be POST. Inside the body field of HTTPS format, there is one JSON-RPC object, and this object is created by the rules which must contain [“jsonrpc” "method" "params" "id"], and it is a request object defined in JSON-RPC. For JSON-RPC implementing PAWS, the RFC 7545 defines the details within object which is shown below, it is a request object:

```
{
   "jsonrpc": "2.0",
   "method": "spectrum.paws.methodName",
   "params": <PAWS_REQ>,
   "id": "idString"
 }
```

The **methodname** is the functionalities provided by PAWS. The parameters are the data model defined by PAWS (part 4 and 5). ID has to be unique for each device and is left for implementation choices.

The JSON-RPC object for response message looks like following for correct message:

```
{
        "jsonrpc": "2.0",
        "result": <PAWS_RESP>,
        "id": "idString"
}
```

<PAWS_RESP> above is another data model defined in RFC 7545 part 4, used for PAWS operation.

The JSON-RPC object for response message looks like following if something is wrong:

```
{
        "jsonrpc": "2.0",
        "error": {
```

```
        "code": -102,
        "message": "An appropriate error message.",
        "data": { ... }
      },
      "id": "idString"
}
```

PAWS defines additional error code besides those provided with JSON-RPC

PAWS protocol:

A basic work flow between device and a database is stated as following:

1. The device should know the URI of database either from pre-configured URI, or dynamically updated later.
2. The device establishes a HTTPS connection with database.
3. The device sends initialization request to database (may not be necessary for every TX)
   The database responses device (may not be necessary for every TX)
4. Registration (optional)
   Response (optional)
5. A device sends a get available spectrum request to database, and the device might be master device or a slave device. A slave device cannot sends request to database directly, but through master device.
   The database responds available spectrum responses to master device
6. If master device is sending request on behalf of a slave device, master device may need to verify with database if the slave device is permitted. (optional)
   Response (optional)
7. The master device may send a notification about what spectrum it intends to use to the database. (optional)
   Database should respond with acknowledge message. (optional)

Registration, verification and notification are optional depending on rule sets defined by different regulatory domains. The regulatory domain is usually a country. Each rule set can be represented by a rule set identifier.

Brief introduction to PAWS Functionalities: (**bolded parts are required,** others are optional)

1. **Database discovery at master device side**
2. **Initialization at database side**
3. Device registration at database side, but it is optional, and it can be implemented as separate component, or as part of Available spectrum query
4. **Available spectrum query for both master device and database**
5. Device validation
6. Spectrum Use notify at device and database, when database requires it, database lets device knows in its Available spectrum query responses

Purposes to have initialization:

The initialization is recommended for master device whenever it powers up or start to communicate with database. By using initialization, database can give device some rule sets information at a specific location. If initialization request contains rule set ID (inside device Descriptor), database must return the corresponding parameterized value for the rule set.

Implementation discussion:

To simply implementation, the required components are implemented first, they are Database discovery, initialization and available spectrum query.

Database discovery:

1. A device can be pre-configured with one or more URI of trusted databases, or the device can store a URI of one server which contains all available databases.
2. When database want to change its URI, it has to start notifying the device 2 weeks before the actual change. The change URI notification can be sent inside any response to device. The data structure of this notification should contain a list of DatabaseSepc which records a list of the name of database and the URI. After device receives the notification, it should replace the URI which it just contacted with the received list in the DatabaseSepc. If there is no response to a database, the device regards it as the case when there is no available spectrum.
3. The device should select different database when the selected database doesn't response for timeout or if the device receives an UNSUPPORTED ERROR.

Initialization:

1. The master device sends a message which contains the information about the device descriptor and its geo-location, inside the Device descriptor, it contains a list of rule set ID (number only), which is pre-configured in the device.
2. If the initialization request doesn't contain any rule set ID information, the database should use all the applicable rule sets at device's location, and respond all of them to the device. If the request has a list of rule set ID in the device descriptor, then the database only returns rule set that matches the ones in the device descriptor. If database doesn't work with the device type or none of the rule set which is described in the device descriptor, then it responds with UNSUPPORTED ERROR

Available spectrum query

The query can be a single or be grouped into a batch. To simplify implementation, the single query in each request is considered. Both device and database are involved in the WS spectrum selection. The device makes request to database to get available spectrum. The request data must contain location of the device. It is an option for device request to have a list of rule sets which the device supply and a union of parameter required for the rule sets. The database determines if there is parameter missing, i.e. geo-location. However, if rule set are supplied in the request the database will use this information to determine if parameters are missing. If the request is valid, the database sends a list of spectrum spec to device, and each spectrum spec is specific to one rule set. Inside a spectrum spec, there is a list a spectrum schedule, and it mainly deals with time. Spectrum schedule specifies: 1. what time is available for use; 2. how long it can be used; 3. What frequency location is allowed; 4. The power limit in a smallest resolution. When device receives response, it needs to check for rule set which it applies, and select one spectrum schedule it wants to use. There are many ways to make the selections, like longest time, maximal power, etc. Device needs to update if current time is greater than the time specified in the selected Spectrum schedule.

When the request is sent on behalf of one slave device, some information has to be replaced to the information of slave device. However, master device information is required.
When database responds, it contains multiple spectrumSpec and each spec is for the corresponding rule set. Inside the spec, it has allowed time for operation and the spectrum details. It has to define a bandwidth resolution and there is a limit on the maximal power for one resolution. The frequencies should be listed in increasing order.