Enabling CBRS Experimentation through an OpenSAS and SDR-based CBSD

Oren Rodney Collaco*, Mayukh Roy Chowdhury*, Aloizio Pereira da Silva*, Luiz DaSilva*
* Commonwealth Cyber Initiative, Virginia Tech, USA, {orenrc, mayukhrc, aloiziops, ldasilva}@vt.edu

***Abstract*—The increased demand for spectrum has motivated the Federal Communications Commission (FCC) to open band 48, also known as the Citizen Broadband Radio Service (CBRS) band, spanning 3.55 to 3.7 GHz, for shared wireless broadband use. Access and operations in the CBRS band is managed by a dynamic Spectrum Access System (SAS) that enables seamless spectrum sharing between incumbent and secondary users. In this paper, we demonstrate an enhanced version of an open source SAS [1], OpenSAS, with added functionality to showcase interaction between General Authorized Access (GAA) and Priority Access License (PAL) user tiers. We further showcase the use of the Google SAS Test Environment [2] with OpenSAS and a Software-defined Radio (SDR)-based CBRS Base Station Device (CBSD) developed by our team. Spectrum sharing is achieved by using a client on the CBSD that is in continuous communication with the SAS to conform to its spectrum usage and transmission power rules at a given location. Our demo steps through the entire SAS-CBSD cycle which includes registration, spectrum inquiry, grant request, heartbeat request, grant relinquishment, and de-registration.**

***Index Terms*—CBRS, Experimentation, Open source, Spectrum Access System (SAS), Spectrum Sharing, Testbed**

I. Introduction

The US Federal Communications Commission (FCC) decided in 2015 to allow shared commercial use of the 3550-3700 MHz band, the Citizen Broadband Radio Service (CBRS) band. A system to protect these users from damaging interference is required, which led to the definition of a Spectrum Access System (SAS) responsible for granting authorization and preventing damaging interference from CBRS Base Station Devices (CBSDs) transmitting on the same channel as the incumbents. Each CBSD has to receive a grant from the SAS before transmitting and to receive valid heartbeat responses to continue transmitting.

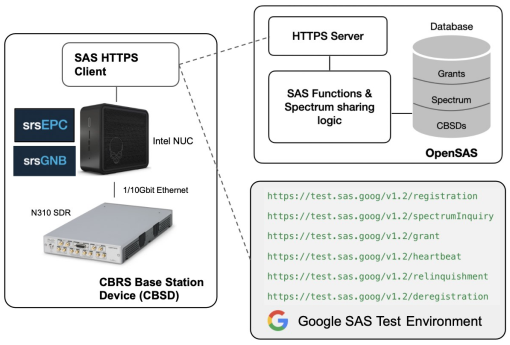
Our demo for the first time shows a Software-defined Radio (SDR)-based CBSD communicating with both an open-source SAS, *OpenSAS*, and the Google SAS test environment. *OpenSAS* is an extended version of the SAS developed at Virginia Tech and described in [1]. OpenSAS has been built to the same specifications as used by a commercial SAS, which enables replicating a CBRS ecosystem in a testbed environment.

Using OpenSAS we showcase two use cases. The first involves communication between a SDR-based CBSD prototype developed by us with OpenSAS and the Google SAS. The second shows two tiers of users (General Authorized Access (GAA) and Priority Access License (PAL)) in the CBRS ecosystem coexisting in the same frequency band. To the best of our knowledge, this is the first demonstration of a Wireless Innovation Forum (WINNF) specification-compliant open source SAS verified using the Google SAS Test Environment.

The contribution of our work is twofold. First we extend the open source SAS developed at Virginia Tech [1] to enable connectivity with other compliant commercial SAS. To achieve this, we replaced the socketIO connection between the SAS server and the CBSD client with an Hypertext Transfer Protocol Secure (HTTPS) connection. Second, we connect our CBSD client to the Google SAS test environment and to OpenSAS to validate the interface compliance with WINNF specifications for commercial SAS.

II. Demo Architecture

The demo architecture, depicted in Figure 1, consists of three main components: The OpenSAS server, the Google SAS Test Environment, and the SDR-based CBSD prototype.



```
https://test.sas.goog/v1.2/registration
https://test.sas.goog/v1.2/spectrumInquiry
https://test.sas.goog/v1.2/grant
https://test.sas.goog/v1.2/heartbeat
https://test.sas.goog/v1.2/relinquishment
https://test.sas.goog/v1.2/deregistration
```

Fig. 1: Demo architecture.

a) OpenSAS: OpenSAS implements an HTTPS server where the Uniform Resource Locator (URL) endpoints for the requests can be accessed. OpenSAS contains modular functions such as registration, spectrum inquiry, and grant request for WINNF-specified requests from a CBSD. The grant algorithm works as follows: The HTTPS server calls the corresponding modular functions such as *register()* based on the URL in the HTTPS GET request. All the required logic for the implementation of the registration, such as verifying the format of the message and checking for errors or values out of bounds in the request message, are carried out. A database entry is then created for the CBSD based on its request message. For the purpose of this demo, we assume that the CBSD antenna transmits in a spherical pattern. When making the decision whether another CBSD interferes with an existing CBSD, we check the distance between the CBSDs and if the distance is above a threshold, we assume there is no interference.

b) Google SAS Test Environment: Google provides a test environment for interoperability testing between a CBSD and the Google SAS. In our demo, we connect our prototype CBSD to verify that the SAS-CBSD interface is compliant with WINNF specifications and thus interoperable with a commercial SAS.

c) SDR-based CBSD Prototype: Our CBSD prototype consists of a SAS Client and the srsRAN software stack for a 5G base station. The SAS Client is responsible for communicating with the SAS and controlling the base station based on the response. Our SDR-based CBSD prototype is implemented in one small-factor computer (Intel NUC) and an N310 SDR.

III. Demonstration

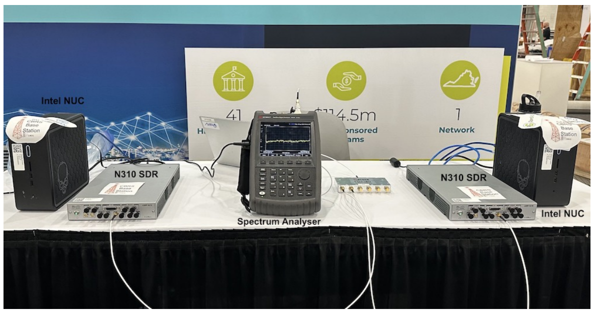
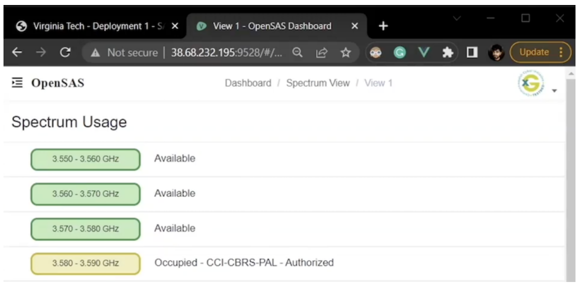


Fig. 2: Demo setup.

Part 1: Compliance testing using Google SAS Test Environment In this part of the demo, we step through all phases of the interaction between our SDR-based CBSD prototype and the Google SAS. These include registration, spectrum inquiry, grant request, heartbeat, grant relinquishment, and deregistration. We display this interaction in real-time using a web-based graphical interface provided by Google SAS. This part validates the compliance of the SAS-CBSD protocol with the WINNF specifications.

Part 2: GAA & PAL interaction using OpenSAS We use OpenSAS to demonstrate the interaction between two tiers of users in the CBRS ecosystem. We set up two CBSD prototypes, with one of them registered as a PAL user and the other as a GAA user in OpenSAS. WINNF specifies maintaining a list of CBSDs that have a PAL Protection Area and using that list to ascertain if a requesting CBSD is on the list [3]. For the purpose of this demo, we have used the FCC Identifier (ID) of the CBSD as the PAL identifier. We first register and request a grant for a certain frequency using the first CBSD and start heartbeating. Next, we register and request a grant from the second CBSD, which is registered as a PAL user in the OpenSAS, in the same frequency. As expected, the GAA CBSD receives a heartbeat response without a new transmit expiry time and is forced to stop transmitting. This is observed in real-time on the spectrum view of the OpenSAS dashboard as seen in Figure 3. The PAL CBSD, on the other hand, receives the grant for transmitting. We have run simulations using the


Virginia Tech - Deployment 1 - S
View 1 - OpenSAS Dashboard
Not secure
38.68.232.195:9528/#/...
Update
OpenSAS
Dashboard / Spectrum View / View 1

Spectrum Usage

3.550 - 3.560 GHz	Available
3.560 - 3.570 GHz	Available
3.570 - 3.580 GHz	Available
3.580 - 3.590 GHz	Occupied - CCI-CBRS-PAL - Authorized

Fig. 3: OpenSAS spectrum view.

OpenSAS to determine how the number of granted CBSDs affect the latency to acquire a new grant. The total grant latency, comprised of the roundtrip request latency and the OpenSAS processing latency, increases linearly with the number of granted CBSDs, as seen in Figure 4. This result can be explained by the grant lookup and distance calculation time. For every new grant request, the SAS calculates the distance of the requesting CBSD from every other granted CBSD. Thus, the number of granted CBSDs determines the number of distance computations the OpenSAS needs to perform, which directly corresponds to the total grant latency. To improve this, we propose creating zones and checking within the zone when a new CBSD is requesting a grant as future work. This should reduce the grant latency when a considerable amount of CBSDs spread across a region are connected since the lookup is limited to within a zone. Some other future work includes using free space path loss and antenna characteristics to calculate the CBSD interference area instead of a distance based model which should reduce the grant latency further and improve interference accuracy.

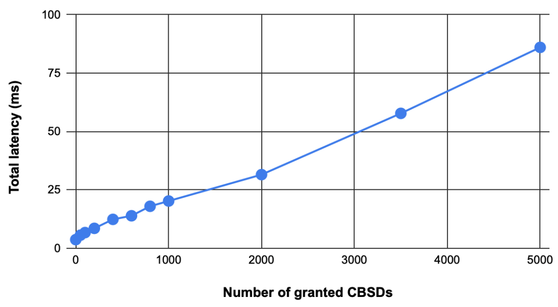


Fig. 4: Grant latency with increasing number of CBSDs.

Acknowledgements

This work was supported by Commonwealth Cyber Initiative (CCI), an investment in the advancement of cyber R&D, innovation, and workforce development. Visit CCI at: www.cyberinitiative.org.

References

[1] Wireless@VT. Spectrum access system. https://github.com/vtwireless/SAS Accessed 14 Nov 2022.

[2] Google. Understand the differences between the test & production SAS environments. https://support.google.com/sas/answer/9288077?hl=en&ref_topic=9350032 Accessed 6 Jan 2023.

[3] WInnForum. Signaling protocols and procedures for citizens broadband radio service (CBRS). https://winnf.memberclicks.net/assets/CBRS/WINNF-TS-0016.pdf Accessed Jan 2023.