

An Automated and Optimized Framework for 6 GHz Wi-Fi AFC Compliance Verification

Yong-Xiang Zhan¹, Chin-Ya Huang², Chien-Chao Tseng¹, Zheng-Han Wu³, Jie-Long Lin³, Jia-Ru Li³, Yu-Ming Chen³

¹Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan

²Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taiwan

³Telecom Technology Center, Taiwan

¹{loki.cs14, cctseng}@nycu.edu.tw, ²chinya@mail.ntust.edu.tw

³{tarma.wu, jielong, steve.lee, ymchen}@ttc.org.tw

Abstract—To enable Wi-Fi operation in the 6 GHz band while protecting incumbent services, Automatic Frequency Coordination (AFC) serves as the key interference-mitigation mechanism. To certify AFC compliance, the Wi-Fi Alliance (WFA) defined the AFC Device Compliance Test Plan and released the AFC DUT (device under test) Test Harness as its reference implementation. The harness, however, exhibits vendor lock-in on both ends: the Control App must be cross-compiled and installed on each DUT, and the spectrum-analyzer driver supplied with the harness targets only one instrument vendor—an instrument that not every compliance laboratory can afford. In this paper, we propose a single-host automation architecture that opens both ends to extension without rebuilding the harness. On the DUT side, we relocate the Control App from the DUT to the Test Harness host and drive each DUT through Secure Shell (SSH) with an externally configurable command mapping table that decouples abstract control intents from device-specific commands. On the instrument side, we introduce a vendor-neutral spectrum-analyzer module behind the existing test-script interface so that any commercial analyzer can be supported by adding a single driver file. Our evaluation across three vendor DUTs confirms that the proposed architecture preserves the existing Test Harness and Test Scripts while removing per-DUT cross-compilation and per-instrument rebuilds, reducing each new DUT or analyzer to a configuration edit or a single driver file.

Index Terms—Automatic Frequency Coordination (AFC), dynamic spectrum management, remote Control App, test automation, test harness, Wi-Fi 6E, Wi-Fi 7.

I. INTRODUCTION

The rapid growth of smartphones, IoT sensing devices, and high-resolution streaming media has made the 2.4 GHz and 5 GHz Wi-Fi channels increasingly congested. To relieve this congestion, regulators have progressively opened the 6 GHz band (5.925–7.125 GHz) for Wi-Fi 6E and Wi-Fi 7 use [1]. The 6 GHz band, however, is not entirely free for new unlicensed use; sub-bands are already allocated to licensed incumbent services, including the Fixed Service for terrestrial point-to-point microwave links, the Fixed Satellite Service for ground-to-satellite communication, and the Radio Astronomy Service for sensitive astronomical observation. To protect these incumbents while permitting 6 GHz Wi-Fi, the Wi-Fi Alliance (WFA) and the Wireless Innovation Forum jointly proposed the Automatic Frequency Coordination (AFC) mechanism: a Standard Power Access Point (AP) must query a cloud

AFC system before transmitting, in order to obtain the spectrum range and maximum equivalent isotropic radiated power (EIRP) permitted at its current geographic location [2], [3]. To verify that AFC devices (e.g., APs; the devices under test, DUTs) meet the corresponding regulatory requirements [2], [4], the WFA published the *AFC Device Compliance Test Plan* [5] and open-sourced the AFC DUT Test Harness as a reference implementation for compliance laboratories [6]. While recent academic work has begun to examine 6 GHz AFC system deployment in practice [7], the implementation-side automation of running this reference harness across heterogeneous DUTs and instruments remains under-explored.

A practical AFC compliance laboratory must accommodate two layers of diversity. On the regulatory side, the band plan, power limits, and query semantics differ between regulators. On the implementation side, the firmware interfaces, operating systems, and instrument-control conventions differ from vendor to vendor. Regulatory variation, however, is not the dominant challenge; rather, implementation-side diversity is the bottleneck that drives the cost of automated test deployment. In the existing WFA reference implementation, vendor lock-in appears on two ends. On the DUT side, the Control App is designed to be installed on each DUT, which forces engineers to obtain the corresponding software development kit, study the Control App source structure, identify device-specific patches, and cross-compile and deploy a custom binary [6]. On the instrument side, the spectrum-analyzer driver shipped with the harness targets only one analyzer vendor; compliance laboratories that cannot afford that specific instrument have no in-harness fallback, since the graphical user interface (GUI) exposes no extension hook for replacing the driver. When the test campaign spans multiple AP vendors and the laboratory cannot replicate the WFA reference instrument, the cumulative integration cost dominates the engineering effort.

To overcome both forms of vendor lock-in together, this paper proposes a single-host automation architecture with two coordinated changes. On the DUT side, the Control App is relocated from each DUT to the Test Harness host and drives devices through Secure Shell (SSH), with an externally configurable command mapping table that separates abstract control intents from device-specific commands. On the

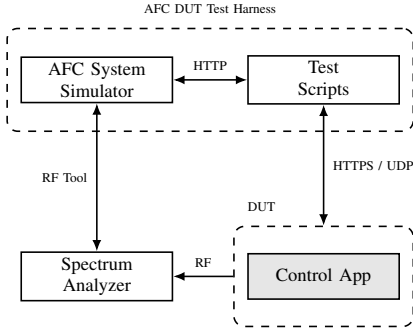


Fig. 1: Existing AFC DUT Test Harness architecture.

instrument side, we add a vendor-neutral spectrum-analyzer module behind the same interface that the Test Scripts already invoke and bind it to an Anritsu MS2720T over PyVISA/SCPI as a concrete, non-default driver. Together, these changes preserve the existing Test Harness and Test Scripts while making both ends extensible by editing data or adding a single file—no rebuild of the harness required. We do not claim novelty in DUT compliance findings; rather, the contribution is the automation architecture itself, evaluated by whether it dispatches the WFA Compliance Test Plan end-to-end with the same harness verdicts as a reference run—*automation parity*, a pseudo-oracle pattern [8] adapted to test-harness regression.

The remainder of this paper is organized as follows. Section II reviews the AFC DUT Test Harness and the Compliance Test Plan. Section III presents the proposed single-host automation architecture. Section IV reports the automation-parity evaluation across three DUTs. Section V concludes.

II. BACKGROUND

This section reviews the existing open-source WFA AFC DUT Test Harness [6]—a third-party reference tool, not the system this paper proposes—which layers AFC-specific modules on top of the WFA QuickTrack Tool framework [9]. We review these modules and the Compliance Test Plan as the existing baseline for the changes introduced in Section III. Within the harness, the Control App and the spectrum-analyzer driver are the two extension points reworked in Section III; the remaining components are reused as-is.

As illustrated in Fig. 1, the AFC DUT Test Harness has three core software components. The *AFC System Simulator* substitutes for the cloud AFC system; the Test Scripts can configure precise responses to exercise nominal and edge behaviors. The *AFC DUT Test Scripts* are Python scripts authored against the Compliance Test Plan [5]; they drive the simulator, the DUT, and the external spectrum analyzer. The *AFC Control App* is a lightweight per-DUT agent that translates standardized commands from the Test Scripts into the local operations required on that DUT [6].

The Compliance Test Plan defines five categories of test cases that the harness automates. *Successful Registration and Spectrum Access (RSA)* drives the DUT through registration and two consecutive queries, then checks that the actual radi-

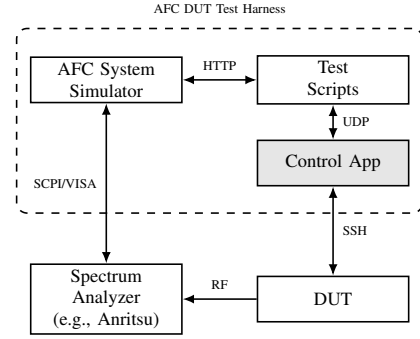


Fig. 2: Proposed single-host automation architecture.

ated bandwidth and power match the AFC grant. *Unsuccessful Registration and Spectrum Access (USA)* returns an empty spectrum list and verifies that the DUT stops transmitting in the band. *Successful Spectrum Access Update (SAU)* changes operational parameters and checks the re-registration flow. *Unsuccessful Spectrum Access Update (UAU)* repeats SAU but with the AFC returning no spectrum after the update. *Unsuccessful Server Validation (USV)* alone exercises eight Transport Layer Security anomalies—two Subject Alternative Name mismatches (badafc.com and wfatestorg.org), certificate signed by a different root, revoked certificate via stapled Online Certificate Status Protocol (OCSP), OCSP stapling disabled, expired stapled OCSP response, eNULL cipher enabled, and missing root certificate—and confirms that the DUT refuses to send its AFC query.

In a typical run, the operator selects a test case from the harness GUI; the framework launches the Python script, which configures the simulator response, sends a control command via the Control App to trigger the DUT's AFC query, and finally commands the spectrum analyzer to measure the DUT's transmitted spectrum and power. The Test Harness evaluates the result against the test plan's PASS/FAIL criteria.

III. SINGLE-HOST AUTOMATION ARCHITECTURE

In contrast to the existing harness reviewed in Section II, we propose the single-host automation architecture to remove vendor lock-in on both ends of the WFA test harness without rebuilding it. The architecture comprises three coordinated changes: (i) Control App relocation from each DUT to the host; (ii) command-mapping externalization; (iii) spectrum-analyzer restructuring into an abstract interface plus a per-instrument driver file. Fig. 2 shows the proposed architecture.

A. Control App Relocation

In the existing architecture, the Control App must be built against each DUT's operating system and architecture, so every new DUT incurs cross-compilation and installation effort, and any control-logic update forces redeployment to all target devices. As shown in Fig. 2, the proposed architecture centralizes the Control App on the Test Harness host (fixed to Ubuntu) and drives each DUT over SSH, leaving the DUT firmware unchanged.

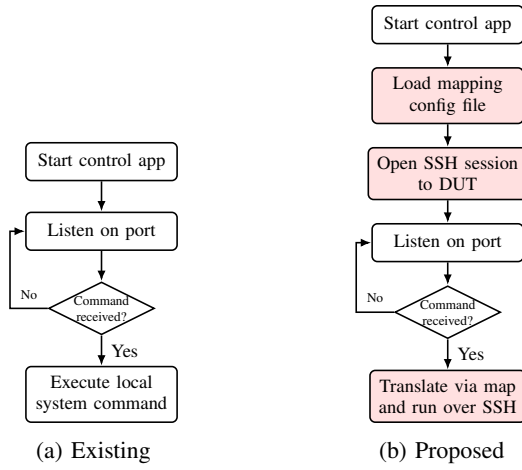


Fig. 3: Comparison of Control App internal flow.

TABLE I: Existing vs. proposed DUT automation control.

Item	Existing	Proposed
Deployment	Control App on every DUT	Centralized on the host
Build	Cross-compile per DUT OS	Single fixed host environment
Command mapping	Hard-coded; update needs rebuild	External file; update without rebuild
Control path	Local syscall on DUT	Remote SSH execution

This design assumes that each DUT exposes an SSH server—common on engineering builds, but possibly disabled on production-locked commercial devices. Because the mapping table (Section III-C) resolves abstract intents to plain command strings, the SSH dispatch layer is replaceable by an alternative control channel—e.g., a serial console over a universal asynchronous receiver-transmitter (UART) or a vendor’s representational state transfer (REST) management interface—without altering the mapping mechanism or the harness-facing interface; validating such transports is left to future work.

B. Control App Workflow Redesign

Fig. 3(a) and 3(b) contrast the internal flow of the existing and the proposed Control App, respectively. The existing flow listens on a local port after install on the DUT, then runs the received system command in place. The proposed flow starts on the host, reads an external mapping configuration file at startup, opens an SSH session to the DUT, listens on a port, and on every command looks up the mapping table and dispatches one or more concrete commands to the DUT over SSH. Table I summarizes the differences.

C. Command Mapping Table

To absorb vendor differences in control commands, the proposed Control App reads an external mapping file, `afc.conf`, that describes the relation “abstract control behavior $\rightarrow$ concrete system command” as data. The Control App loads the file at startup and, during execution, looks up

```

# SSH
SSH_HOST=192.168.x.x; SSH_PORT=22
SSH_USERNAME=root; SSH_PASSWORD=

# Device query (with regex extractor)
CHANNEL_QUERY_CMD=iw dev wlan0 info
CHANNEL_QUERY_PATTERN=channel[[:space:]]+([0-9]+)

# Bandwidth (Wi-Fi 6E / Wi-Fi 7 widths)
BANDWIDTH_CONFIG_REQUIRES_REBOOT=0
BANDWIDTH_20=uci set wireless.<radio>.htmode\
='HE20'; uci commit wireless; wifi reload
BANDWIDTH_40=...
BANDWIDTH_80=...
BANDWIDTH_160=...

# Certificate (for USV anomaly tests)
CA_CERT_SET=cat /etc/controlapp/afc_ca.pem >> \
/etc/ssl/certs/ca-certificates.crt
CA_CERT_UNSET=cp /etc/ssl/certs/ca-certificates\
.crt.bak /etc/ssl/certs/ca-certificates.crt

# Operations
AFC_QUERY=afc_cli -c req

```

Fig. 4: Representative subset of `afc.conf`.

the abstract command issued by a Test Script and dispatches the resulting concrete command sequence to the DUT over SSH. The table is a key–value structure: the key names an abstract command and the value is a sequence of one or more concrete commands separated by semicolons.

Fig. 4 shows a representative subset of `afc.conf`. Entries cluster into five groups required by the Compliance Test Plan: *SSH connection* parameters; *device query* commands with a regular-expression pattern (`CHANNEL_QUERY_PATTERN`) used to extract the current channel from vendor-varying outputs; *bandwidth* commands (`BANDWIDTH_{20,40,80,160}`) covering the wider channel widths used by Wi-Fi 6E/7, plus a `BANDWIDTH_CONFIG_REQUIRES_REBOOT` flag for DUTs that must reboot after a width change; *certificate* commands (`CA_CERT_SET/CA_CERT_UNSET`) used by the USV anomaly cases; and *device operations* such as `AFC_QUERY` for triggering an AFC spectrum request.

D. Vendor-Neutral Spectrum Analyzer Module

The same form of vendor lock-in affects the RF measurement side: the WFA reference implementation ships only a LitePoint driver wired into the spectrum-analyzer module of the Test Scripts, with no GUI extension hook for other instruments. We separate the module into an *abstract interface* that the Test Scripts already call and a *per-instrument driver file* implementing that interface, following the layered abstraction pattern adopted by general-purpose laboratory measurement frameworks [10]; adding a new instrument requires no harness change beyond dropping one Python file alongside the existing module.

The interface exposes five primitives (spectrum_ prefix elided): `connect()`, `analyze_all(timeout)` (full-band sweep returning EIRP and power spectral density), `analyze(channel, bandwidth, timeout)` (targeted single-channel

sweep), `upload_support_data(step)`, and `generate_report(packets)`. The primitives are sized to what the Test Scripts actually call, not to raw Standard Commands for Programmable Instruments (SCPI), so each can bind to whichever SCPI subtree a given vendor uses for channel-power measurement. To validate end-to-end on a non-default instrument, we authored `spectrum_analyzer_anritsu.py` for the Anritsu MS2720T (the SCPI/VISA link in Fig. 2): it opens a PyVISA session, validates with `*IDN?`, and uses the Channel Power mode (`:CONFigure:CHPower`, `:FETCh:CHPower?`, `:FETCh:CHPower:DENSity?`) to implement the sweep primitives.

IV. EVALUATION

We evaluate the proposed architecture by *automation parity*: every test case dispatched through the single-host flow must complete end-to-end and yield the same Test Harness verdict as a reference run, so any FAIL outcome is a property of the DUT, not the automation. The three DUTs (DUT-A, DUT-B, DUT-C) are three different vendor devices, not test stages; they span three distinct firmware control surfaces—two OpenWrt-derived families with different radio-naming conventions and an nvram-based proprietary real-time operating system—so the mapping mechanism is exercised against unrelated command conventions. End-to-end runs against the Anritsu MS2720T cover DUT-A through the full Compliance Test Plan; DUT-B and DUT-C are exercised at the mapping-table level only, to confirm that the abstraction reaches additional firmware families. Vendor identities and firmware revisions are withheld under laboratory disclosure policy.

A. Startup and Map Load

On startup, the centralized Control App automatically loads `afc.conf` from the host—neither an install nor a build step is required on the DUTs. Fig. 5 excerpts the resulting log: bandwidth, channel, and certificate keys are loaded and resolved to DUT-specific commands; the `BANDWIDTH_20` entry binds to the `uci set` form used by the subsequent test case.

B. Command Execution and Mapping Validation

Table II lists the per-DUT mappings for the abstract command “set bandwidth to 20 MHz”; all three DUTs are correctly driven through the same mechanism despite different underlying commands. Fig. 6 excerpts the trace: the lookup resolves the abstract command to three concrete commands (`uci set`, `uci commit`, `wifi reload`), and each is dispatched over SSH in turn (configuration commands produce no standard output, so the extracted output stays empty unless an error occurs).

C. End-to-End Coverage and Wall-Clock Time

With Control App relocation, mapping table, and the Anritsu module in place, we run the full Compliance Test Plan against DUT-A and record per-case wall-clock time

```
14:14:41 info Loading AFC configuration from: ./afc.conf
14:14:41 debug Loaded config: SSH_HOST = 192.168.x.x
14:14:41 debug Loaded config: SSH_PORT = 22
14:14:41 debug Loaded config: CHANNEL_QUERY_CMD = iwconfig
<if>
14:14:41 debug Loaded config:
  BANDWIDTH_CONFIG_REQUIRES_REBOOT = 0
14:14:41 debug Loaded config: BANDWIDTH_20 = uci set
  wireless.<radio_A>.htmode='HE20';uci commit
  wireless;wifi reload
  ... (BANDWIDTH_40/80/160, CA_CERT_SET/UNSET,
  DEVICE_RESET, SPECTRUM_REQ elided)
14:14:41 info Successfully loaded 15 configuration entries
14:14:41 info Successfully initialized config factory
  from afc.conf
14:14:41 info SSH Config: 192.168.x.x:22 (user: root)
14:14:41 info Initializing SSH connection to
  192.168.x.x:22
```

Fig. 5: Abridged Control App startup log with the command mapping table loaded; device-identifying strings are anonymized as in Table II.

TABLE II: Per-DUT command mapping for the abstract intent “set bandwidth to 20 MHz”.

DUT	Mapped commands
DUT-A	<code>uci set wireless.<radio_A>.htmode='HE20';</code> <code>uci commit wireless;</code> <code>wifi reload</code>
DUT-B	<code>uci set wireless.<radio_B>.bandwidth='20';</code> <code>uci commit wireless;</code> <code>wifi reload</code>
DUT-C	<code>nvram set <bw_key>=1;</code> <code>nvram commit;</code> <code>restart_wireless</code>

and verdict. Table III summarizes the 29 single-run end-to-end executions covering all five categories. *All 29 cases completed end-to-end through the proposed automation*; the 5 FAIL verdicts reflect DUT side compliance behavior. Notably, the three USV FAIL cases—OCSPStaplingDisabled, ServerCertRevoked (revoked via stapled OCSP), and StapledOCSPRespExpired—all fall within OCSP stapling validation, while the other five USV anomalies pass on the same DUT and framework. Since the framework dispatches identically across all eight USV cases, the FAIL clustering localizes to the DUT’s OCSP processing rather than to the automation layer. Per-category time is dominated by spectrum-analyzer sweep and DUT settling, with Control App log timestamps showing SSH-dispatched device queries returning within the 1 s log resolution—negligible relative to per-case totals.

The remote control path adds no per-command connection cost either: the SSH session is established once at startup and reused for every subsequent command (Fig. 3(b)), and the trace in Fig. 6 shows the mapping lookup and all three SSH round-trips completing within a single 1 s log timestamp. Millisecond-level profiling against local execution is left to future work; the residual overhead is bounded by the log resolution and immaterial next to sweep and settling times.

As a PASS example, Fig. 7 excerpts three UAU update-category cases on DUT-A dispatched through the single-host

```

14:16:11 debug Found command for key 'BANDWIDTH_20': uci
set wireless.<radio_A>.htmode='HE20';uci commit
wireless;wifi reload
14:16:11 debug Found 3 commands for bandwidth key:
BANDWIDTH_20
14:16:11 info Device supports live bandwidth
configuration (no reboot)
14:16:11 info Executing bandwidth command: uci set
wireless.<radio_A>.htmode='HE20'
14:16:11 info Execute remote SSH command: uci set
wireless.<radio_A>.htmode='HE20'
14:16:11 debug SSH command completed
14:16:11 debug Extracted SSH command output (0 chars): ''
... (uci commit wireless and wifi reload dispatched
identically)
14:16:11 info Successfully configured bandwidth without
reboot

```

Fig. 6: Abridged mapping-lookup and SSH dispatch trace for “set bandwidth to 20 MHz”; every entry shares the same 1 s log timestamp.

TABLE III: Per-category wall-clock and verdicts on DUT-A over the full Compliance Test Plan; DUT-B and DUT-C in Table II are mapping examples only.

Cat.	N (PASS)	avg [range] (s)
RSA	12 (11)	481 [477–492]
USA	3 (3)	205 [204–205]
SAU	3 (2)	537 [499–557]
UAU	3 (3)	559 [556–562]
USV	8 (5)	43 [35–54]
Total	29 (24)	167 min

flow and marked PASS against FCC 15.407(k)(8)(i)–(9)(i); coverage of the other categories is in Table III.

V. CONCLUSION

We proposed a single-host automation architecture for the WFA AFC DUT Test Harness that opens both ends to extension without rebuilding the harness. We validated the architecture by *automation parity*—each dispatch yields the same Test Harness verdict as a reference run, leaving the DUT as the only source of PASS/FAIL variance—along three axes. First, we ran the full AFC Device Compliance Test Plan on a primary DUT and confirmed that every dispatch reaches the Test Harness verdict, with PASS/FAIL outcomes tracking DUT side compliance behavior. Second, we exercised the mapping table on two additional DUTs, confirming that the abstraction reaches new vendors by editing data alone. Third, we bound the spectrum-analyzer interface to an Anritsu MS2720T as a working non-default driver, demonstrating that adding a new instrument requires only a single Python file. Known boundaries: DUTs must expose an SSH server, the mapping table targets stateless commands, and the spectrum-analyzer interface is validated on one instrument. We plan to extend the driver pool to additional analyzer vendors, to prototype non-SSH control transports (UART serial consoles and vendor REST interfaces) for production-locked DUTs, to quantify integration effort against the stock workflow, and to release the

The screenshot shows a web browser displaying the 'AFC DUT Compliance Test Report'. The report is titled 'AFC DUT Compliance Test Report' and includes sections for 'DUT Information', 'Test Result', and 'Test Measurements'. The 'DUT Information' section shows 'Standard Power AP' as the DUT System. The 'Test Result' section shows a table with columns 'FCC Requirements', 'Test Case Name', and 'Test Result'. The 'Test Measurements' section shows a table with columns 'Measurements Name', 'Description', 'Value', and 'Validation Result'.

DUT Information		
AFC DUT System	Standard Power AP	
DUT Vendor Name		
DUT Product Model		

Test Result		
FCC Requirements	Test Case Name	Test Result
15.407(k)(8)(i), 15.407(k)(9)(i), 15.407(k)(9)(j)	CT_AFC_SP_AP_AFCDAU34_FrequencyChannel_10631_1 (Unsuccessful spectrum access update)	PASS
15.407(k)(8)(i), 15.407(k)(9)(i), 15.407(k)(9)(j)	CT_AFC_SP_AP_AFCDAU34_Channel_10624_1 (Unsuccessful spectrum access update)	PASS
15.407(k)(8)(i), 15.407(k)(9)(i), 15.407(k)(9)(j)	CT_AFC_SP_AP_AFCDAU34_FrequencyChannel_10617_1 (Unsuccessful spectrum access update)	PASS

Test Measurements			
Test Case Name: CT_AFC_SP_AP_AFCDAU34_FrequencyChannel_10631_1 (Unsuccessful spectrum access update)			
Test Result: PASS			
Band: 6GHz			
Measurements Name	Description	Value	Validation Result

Fig. 7: Compliance Test Report excerpt.

anonymized `afc.conf` mapping templates and the Anritsu MS2720T driver as open source to support replication.

ACKNOWLEDGMENT

This work was supported in part by the Center for Intelligent Team Robotics and Human-Robot Collaboration under the “Top Research Centers in Taiwan Key Fields Program” of the Ministry of Education (MOE), Taiwan, and in part by the National Science and Technology Council, Taiwan, under grants NSTC 114-2218-E-011-003 and 114-2221-E-A49-009. The authors used Claude Opus 4.8 (Anthropic) to translate and refine the English wording throughout this manuscript and take full responsibility for all content.

REFERENCES

- [1] Federal Communications Commission, “FCC adopts new rules for the 6 GHz band,” ET Docket No. 18-295, Apr. 23, 2020. [Online]. Available: <https://docs.fcc.gov/public/attachments/DOC-363945A1.pdf>
- [2] Federal Communications Commission, “Unlicensed use of the 6 GHz band,” Report and Order FCC 20-51, ET Docket No. 18-295, Apr. 23, 2020.
- [3] Telecom Infra Project, “Open AFC.” [Online]. Available: <https://www.telecominfraproject.com/graduated-project-groups/open-afc>
- [4] Innovation, Science and Economic Development Canada, “Decision on the Technical and Policy Framework for Licence-Exempt Use in the 6 GHz Band,” SMSE-006-21, May 2021. [Online]. Available: <https://ISED-Isde.canada.ca/site/spectrum-management-telecommunications/sites/default/files/attachments/2022/SMSE-006-21-2021-05EN.pdf>
- [5] Wi-Fi Alliance, “AFC Device (AFC DUT) Compliance Test Plan,” ver. 1.7, Jun. 16, 2024.
- [6] Wi-Fi Alliance QuickTrack, “AFC-DUT Test Harness Source Code,” GitHub repository. [Online]. Available: <https://github.com/Wi-FiTestSuite/AFC-DUT>
- [7] J. Park *et al.*, “Deploying automated frequency coordination system for Wi-Fi 6E in South Korea: Challenges and opportunities,” *IEEE Commun. Mag.*, 2024. [Online]. Available: <https://ieeexplore.ieee.org/document/10401897>
- [8] E. T. Barr *et al.*, “The oracle problem in software testing: A survey,” *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 507–525, 2015.
- [9] Wi-Fi Alliance, “QuickTrack speeds time to Wi-Fi CERTIFIED,” 2023. [Online]. Available: https://www.wi-fi.org/system/files/QuickTrack_Highlights_202306.pdf
- [10] S. Reinhardt *et al.*, “Lab::Measurement: a framework for controlling lab equipment,” *Comput. Phys. Commun.*, vol. 234, p. 216, 2019.