AFRL-RI-RS-TR-2020-158

DEEP LEARNING COLLABORATIVE RADIOS - TEAM ZYLINIUM PHASE 3

EMBEDDED INTELLIGENCE RESEARCH LLC

SEPTEMBER 2020

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2020-158 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
FRANCES A. ROSE
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
SEPTEMBER 2020	FINAL TECHNICAL REPORT	JAN 2017 – JAN 2020

4. TITLE AND SUBTITLE

DEEP LEARNING COLLABORATIVE RADIOS -
TEAM ZYLINIUM PHASE 3

5a. CONTRACT NUMBER
FA8750-17-C-0071

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62303E

6. AUTHOR(S)

Robert J. Baxley
Roy S. Thompson

5d. PROJECT NUMBER
SCC2

5e. TASK NUMBER
ZY

5f. WORK UNIT NUMBER
LI

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

EMBEDDED INTELLIGENCE RESEARCH LLC
Team Zylinium
267 Kentlands Blvd., STE 5051
Gaithersburg MD 20878

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITE
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2020-158

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. PA# 88ABW-2020-2694
Date Cleared: 8/31/20

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Refined both the core radio features in our radio with more granular erasure coding and a more flexible MAC with dynamic slot reallocation and FDMA, as well as the reasoning components of the radio that control when we change our operating state to attempt more points and hog more spectrum. Demonstrated flawless co-existence with both active and passive incumbents. It is encouraging to be part of the scientific process while also working to build technology that is truly useful. We plan to participate in the POWDER follow on exercises to further flush out the applicability of our technology.

15. SUBJECT TERMS

Spectrum Collaboration Challenge, Adaptive Wireless Networks, Spectrally Efficient Communication

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**FRANCES A. ROSE**
U	U	U	UU	53	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

Table of Contents

List of Figures

1 SUMMARY

Team Zylinium is pleased to present the following technical report to DARPA describing our Collaborative Intelligent Radio Network (CIRN) design. We worked very hard throughout Phase 3 to refine both the core radio features in our radio (more granular erasure coding and a more flexible MAC with dynamic slot reallocation and FDMA) as well as the reasoning components of our radio that control when we change our operating state to attempt more points and hog more spectrum. While we did well in implementing our Phase 3 technical roadmap, including features that were not originally on the roadmap, we still came up short with a 3rd place finish. A main cause of our underperformance was not realizing until relatively late in the competition how aggressive other teams would be in the spectrum.

As we discuss below, one of the most exciting aspects of our Phase 3 work is that we demonstrate flawless co-existence with both active and passive incumbents. This is a notable milestone because it means that there are many practical use cases for the radio technology we developed. It is encouraging to be part of the scientific process while also working to build technology that is truly useful. We plan to participate in the POWDER follow on exercises to further flush out the applicability of our technology. We also have other commercialization ideas that involve leveraging the best parts of the collaboration protocol.

2 INTRODUCTION

2.1 Competition Overview

The Spectrum Collaboration Challenge (SC2) is an ambitious DARPA grand challenge that seeks to spur innovation in spectrum sharing technology [1]. SC2 was announced in 2016 and ran to 2019 in 3 one-year phases. A goal of SC2 is to increase innovation in RF spectrum access schemes and protocols. Demand for spectrum from commercial and military users is ever-increasing and the hypothesis behind SC2 is that existing spectrum deconfliction mechanisms are suboptimal.

The existing paradigm for spectrum management for civilian radio networks is one of two extremes: exclusive licensed operation or unlicensed operation. Exclusive licenses are inefficient because licensees rarely use the spectrum all of the time. In contrast, the wild-west of spectrum access in unlicensed bands is both suboptimal and not consistent enough for certain wireless communications use cases that demand a high quality of service (QoS). A third path, the path explored by SC2, is to assume that a low-rate "collaborative" communication channel exists between spectrum users. Over this channel, spectrum users can transmit simple performance, observation, and request information. This information, coupled with machine learning and advanced optimization techniques can allow the network of radios to robustly optimize for any desired ensemble state. For instance, if a certain radio network has a high QoS requirement, then the other radios can take in this information and optimize their usage accordingly.

We think that SC2 has been hugely successful in spurring innovation in this area. It is fair to say that little practical work and no real deployments have been created around the use of automated collaborative spectrum usage and deconfliction. SC2 is easily the most comprehensive display of the "art of the possible" in this space. It is clear from SC2 that as 5G standards evolve, spectrum collaboration will be a critical component to achieve vastly higher spectrum efficiencies.

2.1.1 SC2 Hardware & Radio Deployment

To support SC2, DARPA created a massive channel emulation and radio development environment called "Colosseum". The Colosseum has 128 two-channel Ettus X310 SDRs that are wired together through a 256 x 256 channel emulation RF multiplexer. The emulator can be configured to replicate channel environments that are representative of the many-to-many RF connections present in the real world. Each SDR is connected to a 24-core Xeon-class server computer with 128GB of RAM and a 10 GigE network connection to the radio.

To deploy a radio design to Colosseum, a team creates an LXC [2] image and then specifies how many nodes that image should be deployed to. Once the LXC image is provisioned on Colosseum nodes, DARPA starts and stops the radios through a radio control API that each team implemented in their image. After a match completes, all log file information is copied over to persistent storage on the team's gateway and the image is removed from the radio node to make way for the next reservation.

2.1.2 SC2 Competition Overview

For the competition, 5 teams are in a match at a time and each team is composed of 10 radio nodes. Each node is fed traffic from an IP traffic generator called MGEN [3]. The nodes see this traffic on their wired network interface and are then responsible for transmitting the traffic over their RF interface to other nodes in the network. The receiving node will then route the traffic through its wired interface back to the MGEN so that the number of correct packets can be recorded and tracked. Each traffic packet is associated with a data "flow" and different flows have different traffic volumes and packet latency requirements. Flows also have a point value associated with them. Harder flows tend to be worth more points. Teams are graded on how well they can maintain the throughput and latency requirements for each flow.

2.1.3 SC2 Match Scoring

Specifically, a match is broken into a series of 1 second measurement periods (MPs). After a flow's requirements are met for Q consecutive MPs, the team starts accumulating the points for that flow in every subsequent MP. If an MP passes where the requirements are not met, then the team stops earning points until the flow has been held for Q consecutive MPs again. So there is a scoring premium given to teams that consistently achieve flow requirements.

The above-described score is the Measurement Period Score (MPS). This MPS is tracked for each team in each match but is just an intermediate scoring metric. A team's Match Score is the end-result score that teams are graded on and the Match Score is the worst team's MPS for each MP unless all teams achieve a MPS above the pre-defined threshold. If all teams are above this threshold, then the Match Score for a team in a given MP is their individual MPS.

While this may be slightly confusing, the resulting scoring dynamics are straight forward. Teams have to be good spectral citizens, because if the worst team in the match does not score well, then all teams receive that worst-team score. However, if a baseline of performance is achieved, i.e. the MPS threshold is met for all teams, then a team needs to score well as its performance is solely decided by the number flow-points it achieves in each MP. So there is a tension where teams need to adaptively switch between being polite in the spectrum when at least one team is below the threshold, to being aggressive in score seeking when all teams are above threshold.

2.1.4 SC2 Tournament Ranking

The tournament configuration is also a driving factor in the radio design. Ten teams were admitted to the final tournament. The tournament consists of 5 knockout rounds and then culminates in a final round. Up to the final round, in each round of the tournament, the last-place team is eliminated. A team's score in each non-finals round is the sum of the match score for that team in that round.

For the final round, the scoring dynamic changes considerably. While a team is still given a match score, the tournament score in the final round is a function of the team's rank in each match. With only 5 teams admitted to the final round and 5 teams in each match, there is no

dynamic where overly aggressive score seeking is detrimental. That is, if a team is very aggressive with score seeking such that they jam all of the other teams, the worst-case outcome is that the match will end in a draw. On the other hand, if a team under-scores in a finals match, then its rank will be low and ultimately that team will score fewer points than other teams in the finals. When combining all of these aspects, there is much more incentive to be aggressive with spectrum usage and score seeking in the final round even if that detrimentally affects other teams.

2.2 Overall Strategy and Approach to the SC2 Problem

Our strategy for SC2 was to create a configurable radio that had many degrees of operational flexibility and then layer on various expert system and optimization routines in order to maximize our score. To do this, as we describe in the following sections, we built a physical layer based on channelized OFDM with both polar coding and erasure coding error correction capabilities as well as mechanisms for each radio to unilaterally move in the spectrum without having to preconfigure channel hopping across the network.

We overlaid a decision-making engine on top of this core radio technology called the Mandate Optimization Engine (MOE). The MOE is the collection of decision making routines that our radio network uses to determine which mandates to attempt, where to operate in the spectrum, and what posture to assume in the spectrum. The MOE has both centralized decision-making components as well as distributed components. The general separation of decision-making concerns is split so that the centralized decision limits the action space of each node. Then each node could unilaterally make decisions in the limited space in order to optimize their local performance. We describe the details of how this works in the following sections.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Software Design and Architecture

Our radio design uses both the custom processing blocks in the FPGA and the CPU. The radio processing is orchestrated by GNU Radio [5], while a few separate Python processes handle CIL messages and collaboration reasoning.

We used the Ettus RFNoC infrastructure [6] to connect IQ streams between processing blocks internally in the FPGA, and to connect the FPGA to the host processing by leveraging the UHD driver. In addition to using some of the standard blocks provided by RFNoC, we also implemented 3 new RFNoC blocks by writing custom Verilog code:

1. **Channelizer**
    - The Channelizer implements a polyphase anti-aliasing FIR filter and an FFT to convert a wideband (sampled at Fs) IQ stream into a set of equally spaced channels. The FFT size (N) can be set at run-time to be 8, 16, 32, 64, or 128. Each channel that comes out of the channelizer is spaced by Fs/N, and sampled at (Fs*2)/N (i.e. 2x oversampled). The outer half of the channels are dropped such that the aggregate rates going into and out of the channelizer are the same.
2. **Complex Correlator**
    - The Complex Correlator performs the complex correlation of the input IQ sequence and a 256 sample replica sequence that can be loaded at run time. In our radio implementation, this block is used to search for the start of each burst preamble (which is a complex pseudo-random sequence). The complex correlation is implemented using 3 256-tap FIR filters. Because these filters must be able to support a high input rate, this is by far the most resource (DSP48 block) consuming unit in our design.
3. **Correlation Peak Detector**
    - The Correlation Peak Detector implements a CFAR detection algorithm to search for peaks in a real-valued input stream above a configurable threshold. All samples that are not determined to be peaks are set to 0 in the output stream. Our implementation uses this block to find the exact sample that each received burst starts on.

On the RX side, the X310 is configured to sample both antennas at 160MHz when running a scenario with a 40 MHz maximum RF bandwidth. Because 160MHz is not an officially supported sample rate of the X310, we perform a custom tuning of the clock NCO and calibrate digital sample path between the ADC and FPGA at startup. Each antenna's digital IQ stream is then shifted by 42MHz to remove the LO offset and downsampled by 2x to 80MHz by the DDC in the FPGA. The 80MHz IQ stream is sent to the RFNoC channelizer block which channelizes the 80MHz wide IQ stream into 32 channels which are each offset by 1.25MHz and sampled at 2.5MHz.

The channelized IQ from both streams is sent directly to the host processor, and the IQ from the first antenna is sent to the Complex Correlator RFNoC block which performs a complex correlation with the waveform preamble sequence to create a real-valued correlation surface. The correlation surface is passed to the Correlation Peak Detector, which finds peaks and zeroes out all non-peak samples. The peaks for each channel are then passed on to the host to indicate the start of received packets.

The IQ from both antennas, along with the peak detection locations are sent to the Zylinium Demodulator, which performs the OFDM equalization and demodulation, polar decoding, and CRC checking. Bursts that pass the CRC check are forwarded on to the Zylinium Link Layer control unit, which reconstructs the bursts into packets and passes on to the Traffic Generator. The processing blocks involved in this process are shown in the Figure below.

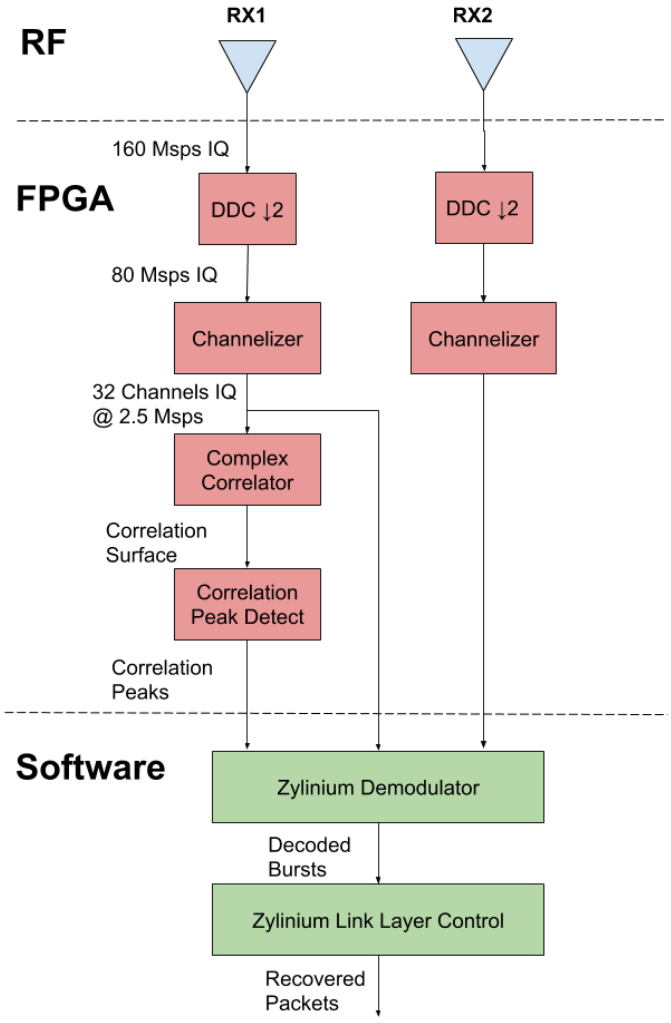


Figure 1 Receiver processing chain

On the TX processing side, we perform all of the modulation processing in software. Specifically, packets are broken up into fragments that fit into our fixed chunk size. These fragments are then allocated among the available channels by the Zylinium Link Layer Control unit. The Zylinium Modulator performs the OFDM modulation of each channel, and then combines all active channels into a wideband (40Mhz) IQ burst, which corresponds to the length of our TDM slot. The burst IQ samples are then sent into the FPGA with a precise timestamp to schedule the transmission so that it starts at the slot time assigned to this node. Within the FPGA, this goes through an RFNoC DMA FIFO to a DUC that shifts digitally by 42MHz and performs a 4x upsampling to the DAC rate of 160MHz. The TX processing chain is shown in the figure below.

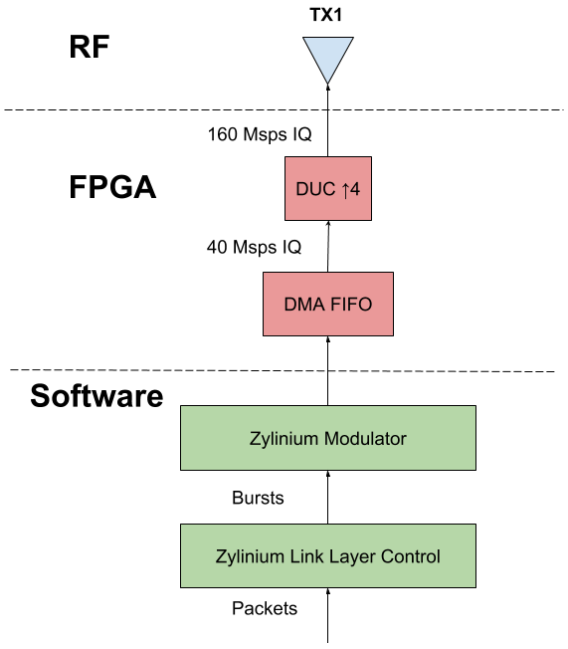


Figure 2 Transitter processing chain

All other radio stream processing is handled by software GNU Radio blocks. The figure below shows the top-level GNU Radio flowgraph for our radio design. The flowgraph includes both the RFNoC (FPGA) blocks and custom GNU Radio software blocks.

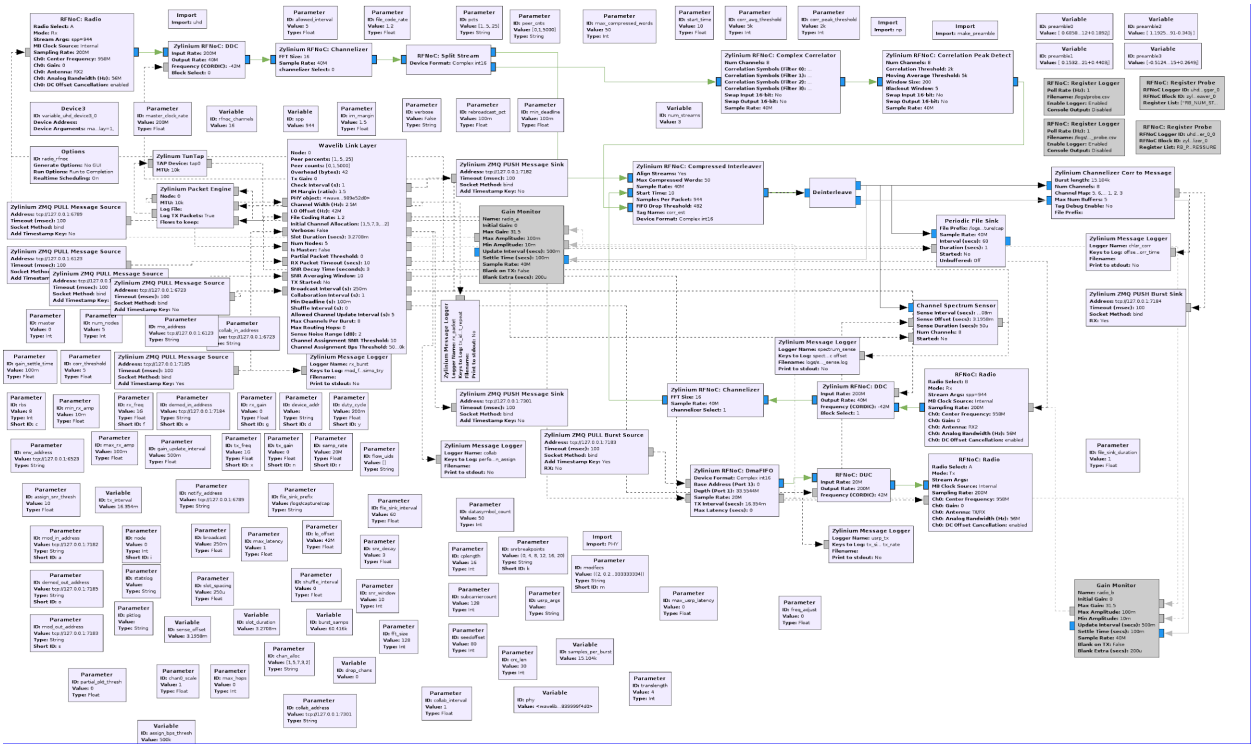


Figure 3 Main radio flow graph

In order to optimize processing efficiency and CPU core utilization, we run many modulation and demodulation worker flowgraphs in parallel. Whenever the main flowgraph gets a correlation hit on a new burst or needs to modulate a new transmit burst, it sends the relevant IQ samples to one of the worker threads through a ZMQ socket message.

We also use ZMQ sockets to pass information from our main radio flow graph to our auxiliary reasoning and collaboration processes.

3.1.1 Computational Performance

FPGA Utilization

The table below shows the utilization of the major processing blocks in our FPGA design. We are currently using approximately 67% of the logic resources, 46% of the DSP resources, and 84% of the BRAM resources. Even though these numbers indicate that we have some margin to add additional processing the FPGA, we have found in practice that it is difficult to be able to route all signals and still meet the timing constraints even with this design.

Unit	LUTs	LUT %	DSPs	DSP %	BRAM	BRAM %
DDC (x2)	15064	5.93%	74	4.81%	22	2.77%
Channelizer (x2)	12550	4.94%	128	8.31%	150	18.87%
Complex Correlator	25326	9.96%	387	25.13%	105	13.21%
Correlation Peak Detect	13838	5.44%	1	0.06%	42	5.28%
DMA FIFO	3616	1.42%	0	0.00%	39	4.91%
DUC	6683	2.63%	55	3.57%	12	1.51%
Other/Infrastructure	94495	37.17%	71	4.61%	300	37.74%
Total	171572	67.49%	716	46.49%	670	84.28%

Figure 4 FPGA resource utiization

CPU Utilization

The CPU utilization is highly dependent on the scenario bandwidth and aggregate traffic level of our radio. We have found that under high loading scenarios when running at 40 MHz, our static CPU utilization is approximately 550% (or 5.5 of the 24 available CPU cores). The remaining 18.5 cores are available to run worker threads that perform modulation and demodulation processing in parallel. The number of worker threads is configurable, but our default configuration launches 10 modulator and 10 demodulator threads.

With the SCE configuration, our radio was rarely under strain in matches. However, we are not confident we could accommodate more than 40MHz of processing. Because the post-channelized IQ is 2x oversampled, and because we process the RX streams from both antennas, the raw data rate going from the X310 to the SRN over the 10 Gb Ethernet link is approximately 5.12 Gbps (40MHz * 2x oversampled * 2 antennas * 32 bits/sample) + overhead. We have found in practice that the UHD driver will occasionally drop samples from the X310 when running at 40MHz, and if we run much higher than this, it will drop significant samples, which can have major performance impacts. Because even occasional dropped samples can cause us to miss a flow for a single MP and severely impact our match score, we configured the radio to use a maximum sampling rate of 37.5 MHz when the scenario RF bandwidth is over 27.5 MHz, a sampling rate of 30 MHz when then RF bandwidth is between 16.25 and 27.5 MHz, and a sampling rate of 20 MHz when below 16.25 MHz.

3.2 Radio Design

3.2.1 SCE Improvement Overview

In SCE, we have extended our PE2 implementation in a number of ways including:

- Refactored our codebase so that the MOE is a stand-alone module that can be independently tested and exercised.
- Built a simulation framework to simulate MOE decision making. The framework used colosseum logs to emulate the SNR and packet drop rates from real scenarios.

- Changed the MAC so that the TDMA slots did not have to be uniformly distributed across nodes.
- Decreased the per channel bandwidth and increased the number of channels available to make it easier to find holes in the spectrum without interfering with other teams.
- Created an optimization procedure to allocate slots to maximize scoring given the network capacity constraints and rate requirements.
- Changed MAC so that FDMA/TDMA frequency allocations are supported.
- Added support for streaming and file-based erasure coding.
- Changed packet structure to support robust erasure coding and low-latency links. Also added ability to support data transmission to multiple receiving nodes in one packet.
- Implemented support for changing our aggressiveness posture based on match inputs.
- Increased our accuracy in our CIL reporting for voxel usage and score reporting.
- Overhauled our MOE logic to support the new rules and to be more effective as incorporating CIL reports from other teams

Just as in the previous phases, we have implemented a version of channelized OFDM where channels are allocated in time and frequency to different users. Previously, we implemented a multi-hop routing protocol that reasons about the optimal hop path through the network based on pairwise link signal-to-noise ratios broadcast by each node. It turned out that multi-hop routing was only required to pass broadcast messages on links that did not have any flows, so we removed this feature in SCE to simplify the control logic. It seems that in some scenarios, when a link did not contain any data, the SNR of the radios in that link was extremely low. This is despite the fact that the geometry of the nodes should have meant that the SNR was high.

3.2.2 Physical Layer

The physical layer of our radio is based on channelized OFDM. That is, we transmit using N "channels" of OFDM. Each of these channels could be deactivated unilaterally by the transmitter. At the receiver side, each receiver constantly looks for transmissions on all channels, which means that we didn't have to pass information to the receiver to instruct it to tune to a certain channel. The channel width is 585.9kHz and the slot length is 3.4ms.

Each channel contains OFDM waveforms with K subcarriers per symbol and P symbols per slot. In our competition design, we set K=128 and P=15. In PE2, P was 50; we reduced the burst size in order to reduce the latency of packets so that we could meet the 120ms flow requirements. Of the P symbols, the first two symbols were used for synchronization and for signaling the modulation and error control rate of the error correcting code for that slot. Specifically, the first symbol is a randomly modulated preamble symbol that is used for correlation. The second symbol contains 60 digitally modulated BPSK subcarriers at ½ rate coding. These 60 bits convey the TxId, intended RxId, the modulation coding scheme, and the number of sub-slots that are populated in this slot.

The “payload” symbols in the slot contain a mix of data and pilot subcarriers [7]. Our competition design contained 16 pilot carriers per symbols that were evenly spaced throughout the frequency support of the symbol.

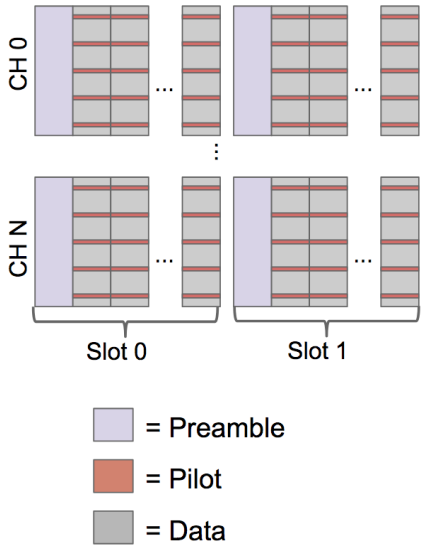


Figure 5 Waveform pilot, payload, and preamble structure

The receiver processing consists of the following steps:

1. Channelization into *N* channels,
2. Correlation with the preamble sequence to determine if signal was present and gather coarse time synchronization,
3. Carrier frequency offset estimation using the preamble symbol,
4. Coarse channel frequency response estimation using the pilot symbol,
5. Determination of the payload modulation coding scheme from the second preamble symbol,
6. Fine channel state information estimation using the symbol pilots,
7. Soft demodulation of the constellations in each subcarrier on each antenna,
8. Bit SIMO bit estimation, and
9. Erasure coding recovery of data chunks that did not pass CRC.

As part of the receiver-side processing, our radios implement SIMO diversity decoding. The SIMO decoder uses a combination of selection diversity and combining diversity to leverage multiple antenna streams. SIMO processing was only possible due to our FPGA channelization processing. Without the FPGA handling the channelization load, we would not have been able to handle decoding two receive streams.

Our SIMO decode processing works as follows: demodulate and try to polar decode one antenna, if that fails, try to decode the other antenna stream (selection diversity), if those both fail, average the subcarrier soft symbols between the two antennae and then decode these averaged values (combination diversity). As expected, we achieved a 3dB performance improvement with SIMO processing.

In correlating with the preamble, we found that vanilla correlation was not very robust to the constantly changing noise floor and channel loss environments in Colosseum. Instead, we

implemented a constant false alarm rate (CFAR) correlation detector that used "training cells" near the peak or "cell" under test location to estimate the noise power. We then used this estimate of the noise power to scale the peak value. This peak detector has the dual benefits that 1) its threshold does not need to change with the background noise level and 2) that it can filter out "near peaks" that might cross threshold with the vanilla correlation.

To estimate the channel state information, we use a minimum mean squared error estimation procedure that implicitly estimates the delay spread of the channel response. The mechanics of this estimation are that we stack the pilot subcarrier channel estimates in a vector and then apply a linear operation (matrix multiplication) to the pilot estimates in order to interpolate the channel estimate across the data subcarriers. The matrix we use is equivalent to a sinc-shaped interpolation of the pilot channel estimates across the data subcarriers.

After reception and equalization, the soft constellation outputs are passed to our error correction decoder that uses Polar codes. Polar codes were invented in 2009 and were shown to be capacity achieving. The beauty of Polar codes is that they are the first capacity-achieving block code with a deterministic construction. This is as opposed to LDPC, for instance, that rely on random sparse encoding matrices. In addition to being deterministic, the block construction also has a butterfly-like nature to it which makes it amenable to radix-2 "fast" matrix multiplication implementations. In the case of Polar codes, this means that the encode and decode complexity is on the order of $n \log(n)$, which makes polar codes extremely fast.

Another component of the physical layer was adapting the modulation and gain to the channel condition. Specifically, we adaptively supported BPSK, 4-QAM, 16-QAM, and 64-QAM depending on current channel conditions (i.e. measured SNR between 2 nodes). For each modulation, we supported arbitrary Polar Code rates. We also experimented with 256-QAM, but were unable to get it to perform reliably in the Colosseum, even under ideal channel conditions.

3.2.3 Packetization

One large SCE improvement to our radio was the addition of erasure codes. To understand how those work, we first explain how we convert packets from the traffic generator into "bursts" which are composed of "chunks" of bit. A "chunk" is the smallest atomic grouping of bits that we can transmit and is 900 bits.

A radio will get various time slots and in each of those time slots, it will be able to use a subset of the channels. The unit of spectrum that is in one time slot in one channel is called a "burst". The number of chunks that fit into one burst depends on the modulation coding rate that the link supports. At the lowest modulation/coding, only one chunk fits in each burst, but at higher rates, we may fit as many as 10 chunks in one burst. There may be situations where the modulation/rate support 10 chunks, but we only send 7 because that is all of the data that needs to be sent. The various components of our packetization architectures are illustrated in the figure below.

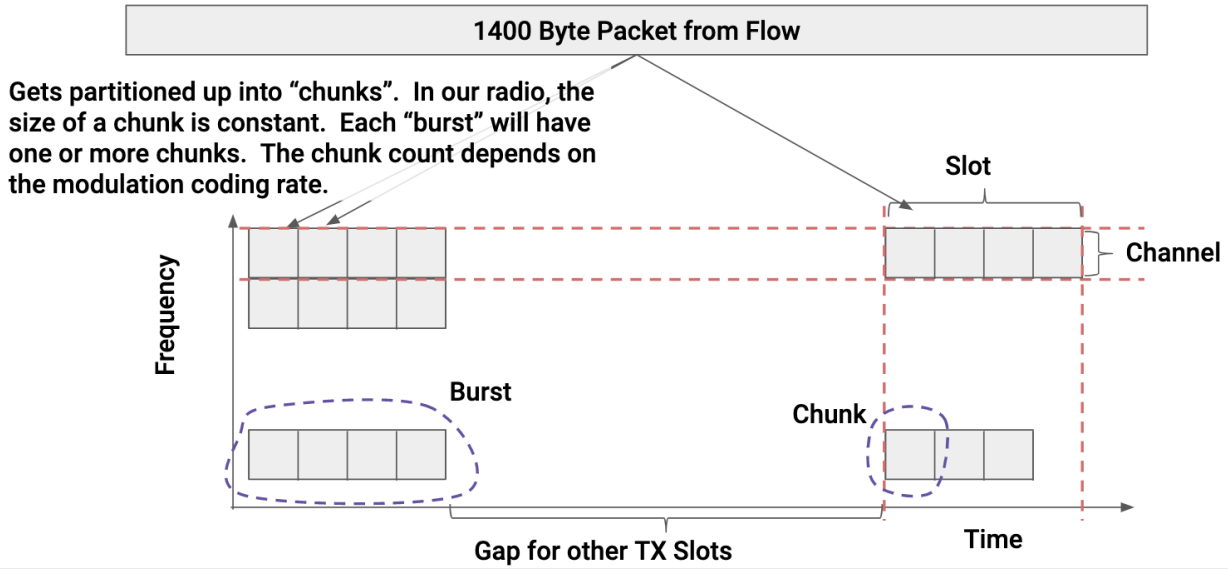


Figure 6 Visual illustration of a "burst", "chunk", and "slot"

With the packetization explained, we can now explain how we modified our erasure coding scheme to support correction in bursty noise environments.

Earlier in SCE, we implement erasure coding on the packet stream that came from the traffic generator. This was useful for high-rate streams that had many packets per second, but many streams were very low rate such that there was only one to three packets in each MP. Erasure coding as we had it set up was of limited utility in this case because there was only 1 to 3 packets that we were trying to recover. Moreover, a packet would span many chunks, so if a burst of noise knocked out one chunk, we would lose an entire packet's worth of data. The challenge is illustrated in the figure below.

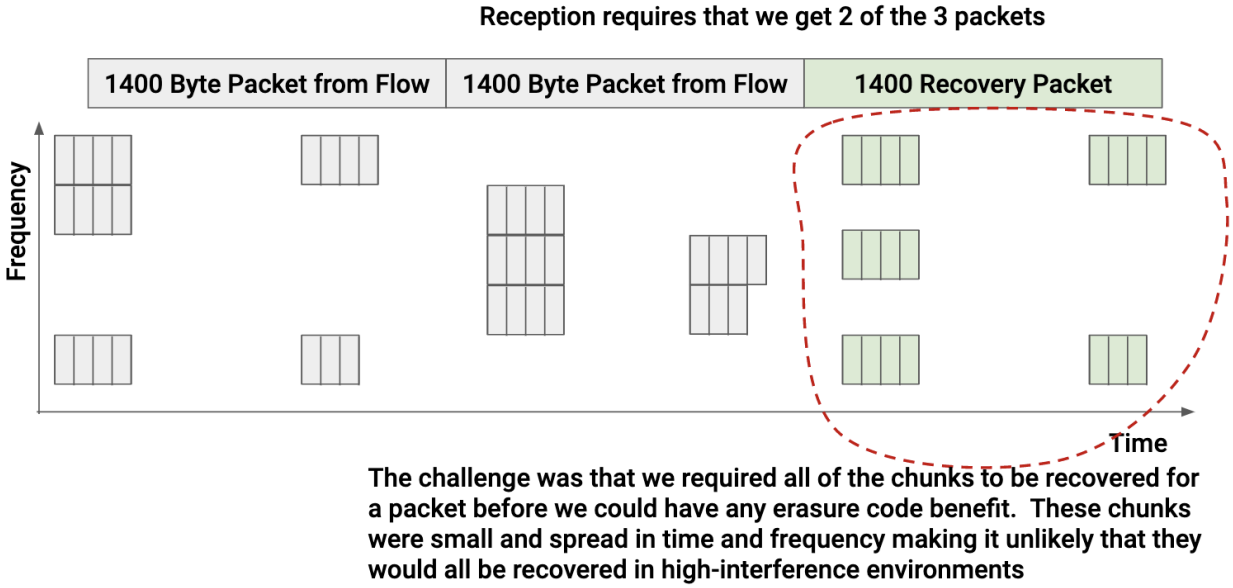


Figure 7 Notional example of our original approach to erasure coding packet data

In the last month of the competition, we completely rewrote our packetization routine so that we could erasure code at the chunk level, thus enabling burst noise bit loss at the much more granular chunk level. We proved that this led to an increase in performance in every scenario. This also allowed us the flexibility to transmit an arbitrary number of erasure-coded chunks in order to fill up the spectrum in more aggressive radio modes.

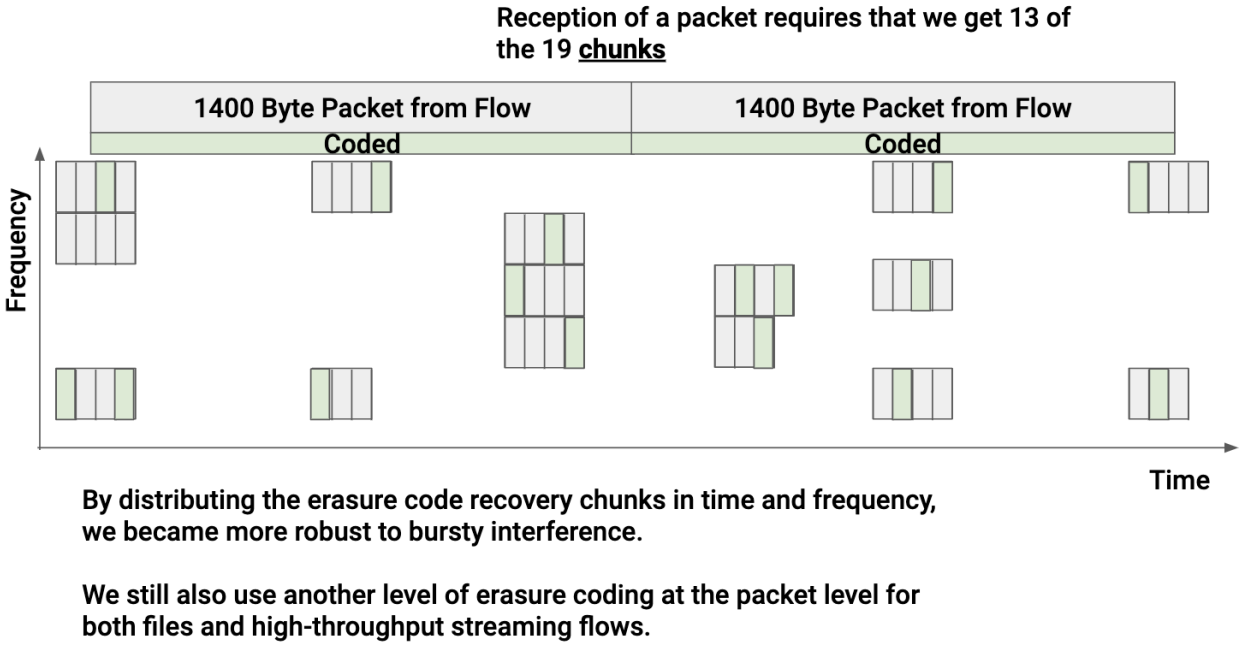


Figure 8 Example of more granular erasure coding at the chunk level

3.2.4 Medium Access Control (MAC) Layer: Mechanics

We implemented a hybrid TDD/FDD MAC for medium access control. To achieve this we needed tight time synchronization across all radios in the network. At the start of a match, there is no time synchronization among the nodes. To establish the time grid, the master node (i.e. the collaboration gateway) is always assigned slot 0. The master node will start transmitting a burst 4 times per second containing its node number and all other nodes will be listening to hear this. Once a node successfully receives and demodulates a burst from the master, it can determine the exact time offset from node 0 that it should transmit on based on the precise time that it received the burst from the master node and its relative node number.

This process continues until all nodes have received at least one burst from one other node and the grid has been established. In practice, as long as each node can "hear" at least one other node, the grid will typically be established properly within 1 second, and keep sync for the duration of the match. One nice result of solely using our radio signal for synchronization is that we do not require each node to have access to a universal time source such as GPS or NTP.

When it is each node's slot to transmit, it can select which of the frequency channels to transmit on. Each channel/slot is independently packetized and modulated, with a field in the burst header set to indicate the set of destination SRN numbers or a special broadcast SRN number.

The receiving nodes process this header to determine if there are any chunks in the burst that are addressed to them or that are addressed as broadcast packets.

The fixed TDMA-only structure we had in PE2 provides many desirable features such as preventing multiple nodes from transmitting at the same time and allowing for clean sensing of other networks, but it also introduces challenges, specifically that we have no mechanism to dynamically re-allocate more spectrum to nodes that have the most data to transmit, and it is difficult to achieve mandates with very low latency requirements since each node has to wait until it is its turn to transmit.

In SCE, we expanded the radio degrees of freedom so that 1) each node can occupy more than 1/10th of the time slots, and 2) so that a given time slot can support multiple transmissions from multiple radios on different frequency channels. The net result is that our SCE radio supports dynamic time slot allocation and FDMA. The top figure below shows a TDMA-only scheme where each node gets one timeslot. The bottom figure shows the same spectrum usage, but with the bursts allocated in an FDMA/TMDA grid.

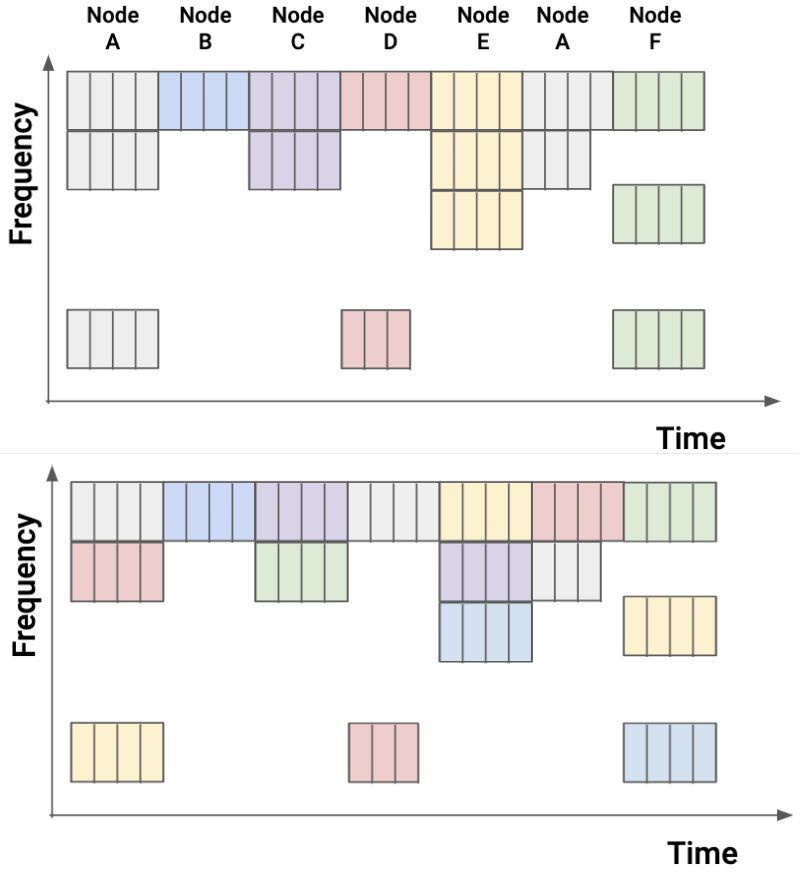


Figure 9 Top: notional TDMA spectrum allocation, Bottom: notional TMDA/FDMA allocation

The following two figure panes show spectrum captures of the TDMA-only operation (top) compared to TDMA/FDMA operation (bottom)

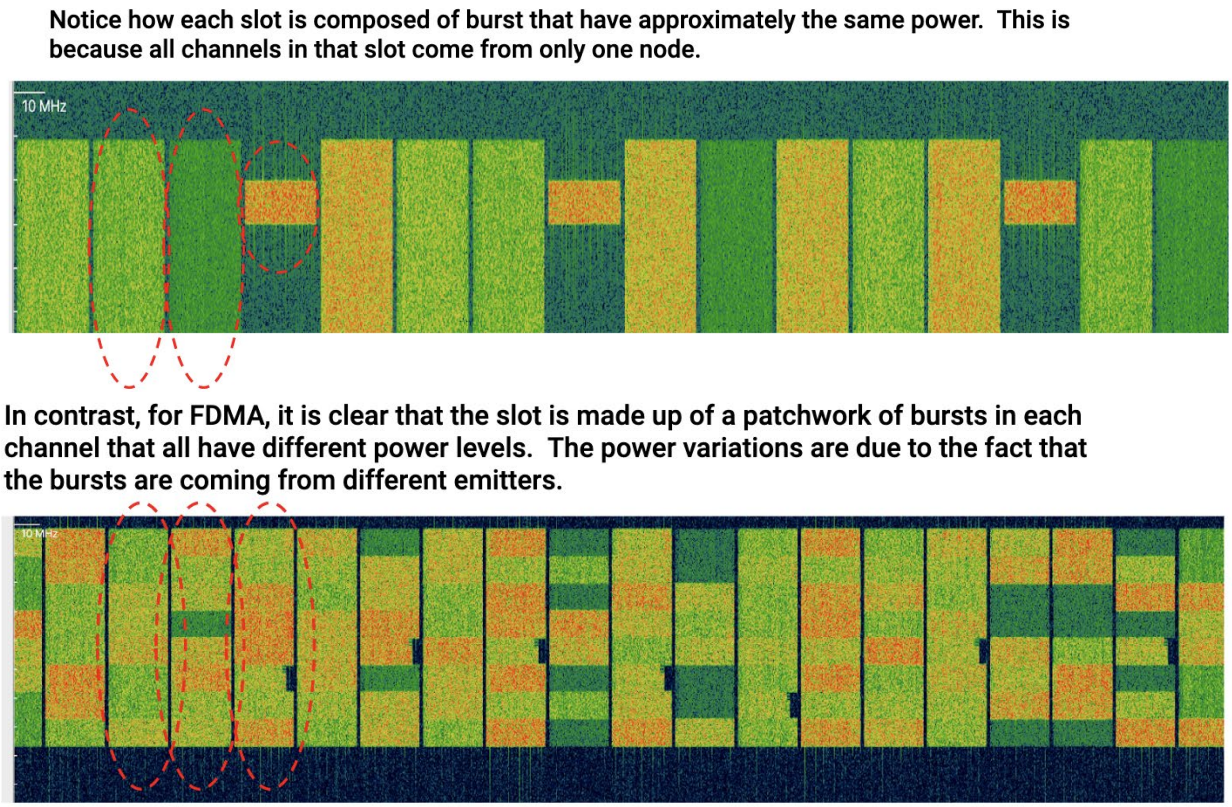


Figure 10 Top: spectrogram of a TDMA spectrum allocation, Bottom: spectrogram of a TDMA/FDMA allocation

To make the dynamic slot reallocation work, we created the concept of a "frame" which is 50 slots. In each frame, every node needs at least one slot so that it can transmit broadcast and timing information. That leaves the other 40 slots available for reallocation among the nodes. The details of the optimization are discussed below in the MOE section.

To make FDMA work at the radio level, the radios had to converge on a common frequency reference. If they were not precisely synchronized in frequency, then inter-carrier interference would result, which severely degrades link performance. Converging on a common frequency grid required that each radio estimate its carrier frequency offset (CFO) and then adjust its tuner to compensate so that it was on the same frequency grid as the master node. The plot below shows the CFO of two nodes in a link as they converge on a common frequency reference point. The CFO units on the y axis are in terms of a proportion of the channel width. The x-axis is time.

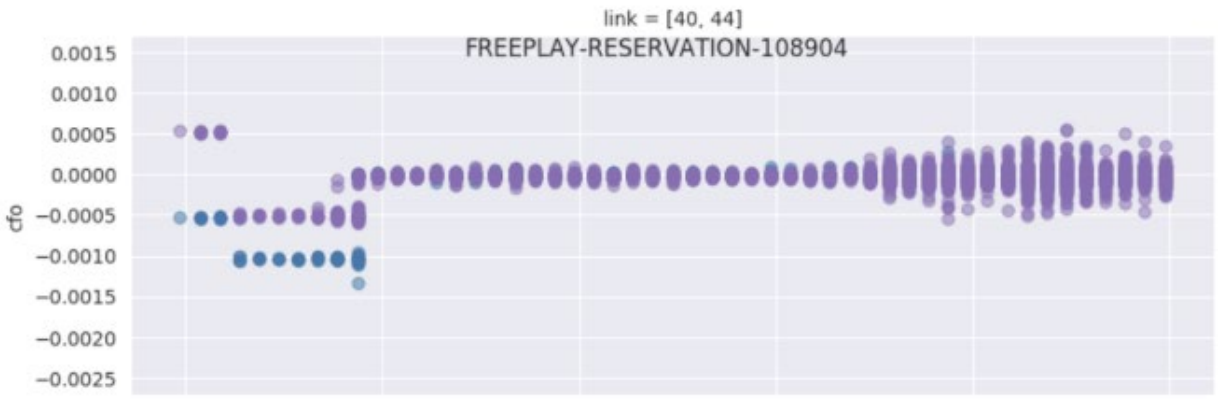


Figure 11 Trajectory of the carrier frequency offset over time

3.2.5 Medium Access Control (MAC) Layer: Spectrum Selection

In PE1, our radio simply did unilateral spectrum sensing to determine where to transmit. This was moderately effective but had a couple of disadvantages. The first was that unilateral spectrum sensing across 10 TDD nodes meant that our team's collective spectrum usage was a hodge-podge of time-frequency tiles spread all over the spectrum. This made it difficult for other teams to avoid being jammed by our transmissions. In PE2, we fixed this by 1) having all nodes on our team globally converge on a list of channels that they could select from and 2) using CIL spectrum request information as part of our channel selection process, which tends to be much more stable than spectrum sensing readings. In SCE, we added modifications to the reasoning to take into account our relative score position in the match when choosing how much spectrum to use.

The mechanics of the channel selection process include both centralized and distributed reasoning components. The first step in the process is that each radio transmits their channel spectrum measurements to the collaboration gateway node. The gateway node then collects this information, and voxel request information from competitors and from incumbents, to generate a master list of the subset of channels a radio on our network is allowed to use. The size of the subset is dictated by stage bandwidth and by the number of other teams that checked in on the CIL and the aggression level of our network. We will describe the details of how that works below in the MOE Flow Selection section.

Once the subset of "allowed channels" is established by the gateway node, it is broadcast to all other nodes in the network. Each individual node then uses spectrum sensing to determine which of the channels on the list it will use in each of its transmission slots. This procedure has the benefit that it kept the spectrum usage of our nodes confined to a relatively small subset of the available frequencies while also allowing each radio the local flexibility to avoid interference.

The figure below shows a hypothetical channel selection configuration without centralized control. In that case, almost all channels are used even though less than half of the spectrum is

occupied. Having a transmit pattern like this makes it very difficult for other teams to find open spectrum where they can transmit. Once we added centralized control, the channel allocation on the right is more representative. In the plot on the right, there are red outlines that represent the "allowed channels". The same amount of spectrum is used in both cases, but on the right, our team's spectrum usage leaves more consistent holes where other teams can more easily slot in.

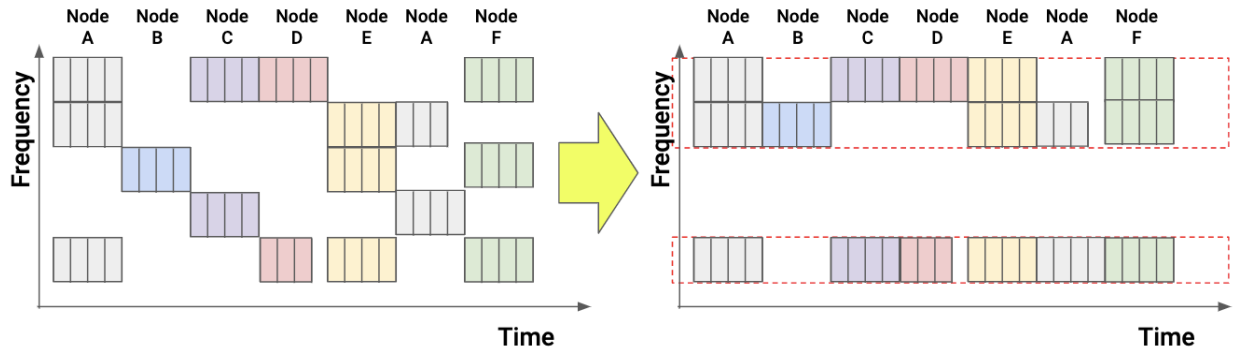


Figure 12 Left: unilateral channel selection by each node. Right: centralize channel selection with channel selection refinement by each node.

3.2.6 Network Layer

The main functions of the Network Layer of our radio design are to 1) maintain a prioritized queue of packets to be transmitted, 2) keep track of the current network state (i.e. SNR map between all nodes in the network), and 3) route packets as necessary based on current conditions.

In PE1, our network packet queue structure was pretty simple: packets with the highest priority went to the top of the queue. In PE2 and SCE, the scoring was more nuanced which made the computation of priority much more complicated, but the basic idea of having a queue of packets where the packets were organized in priority order was still in place. Just as in PE1, our PE2/SCE queueing logic kept track of each packet's latency and the latency requirements in order to drop packets whose latency exceeded the requirement. But the PE2/SCE design had the additional feature that we are continually refining and pruning the queue based on the link SNR states and the observed latency of packets that had just been sent in previous slots. In this sense, our PE2/SCE queue was less of a queue and more of a holding pin of packets.

In order to share information about the current performance of the network, each node transmits a broadcast packet 2 times per second. This broadcast packet contains the pairwise link-snr measurements, the spectrum sensing measurements for each channel, and CIL information (MOs met, progress towards MOs, etc.). With this information, we can build up a map of the current SNR between all nodes in the network. Additionally, this information is used by the collaboration gateway node to populate its performance collaboration messages.

Broadcasts are randomly rebroadcast by receiving nodes 10% of the time. This is to ensure that even if there is no direct link between two nodes, the broadcast packets will permeate the network and reach all nodes.

3.3 Data Sources, Decision Making, & Flow Selection

A core component of our success is our mandate optimization engine (MOE). Since each radio in our network is assigned non-overlapping time/frequency slots, each radio can act and optimize its score without any effect on the other radios. This allows straightforward distributed mandated reasoning optimization and means that each node can act unilaterally. The input-output of the MOE and the core components of our radio reasoning are illustrated in the diagram below.

As inputs, at each node, the MOE takes into account the 9 link SNRs to other radios, the list of available mandates and their requirements, and feedback from the radio PHY as to whether packets in a particular mandate are getting transmitted before the latency deadline.

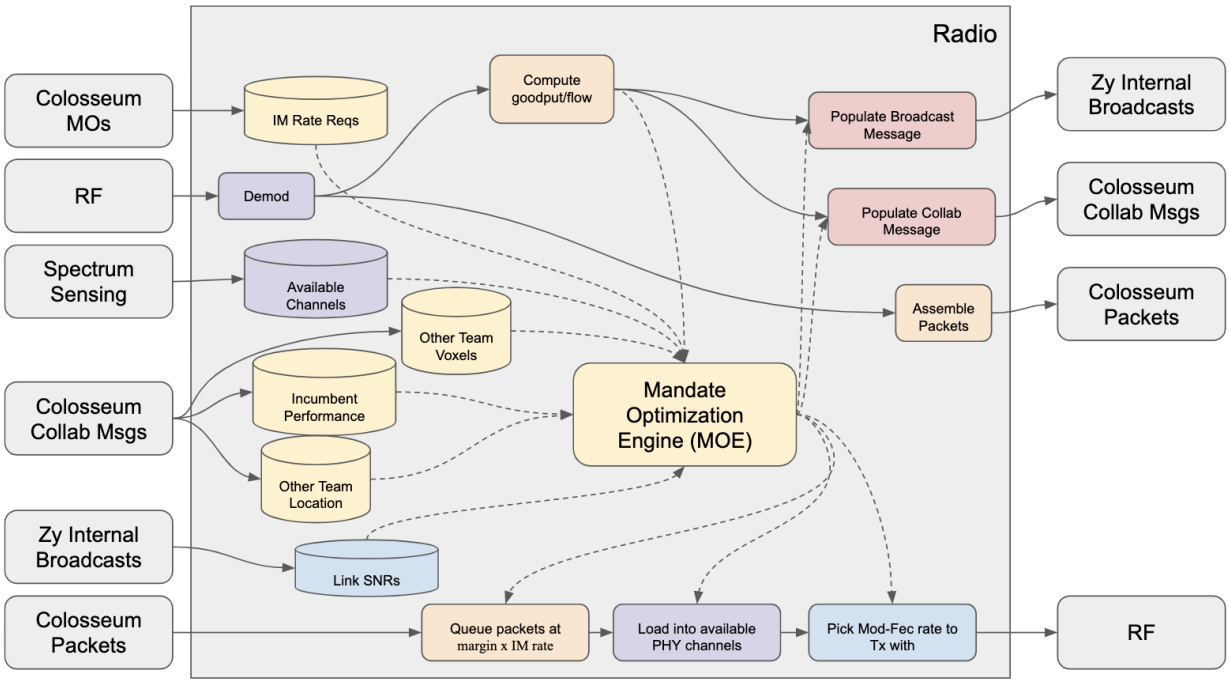


Figure 13 Input and output relationship for MOE

The first step in MOE processing is to take the list of mandates and rank order them by difficulty. The MOE does this by computing the fractional channel capacity that each mandate requires. That is, the MOE can use the link SNRs to determine how much rate each link can support. The MOE also knows the rate load that each mandate requires. The fractional capacity is simply the mandate rate divided by available rate. The MOE takes these fractional capacity numbers along with the mandate point values and does an iterative knapsack optimization to determine which mandates fit in our capacity budget to maximize the number of points being attempted.

In addition to picking which mandates to attempt, the MOE also keeps track of when a packet has to be transmitted. The MOE takes advantage of latency requirements that are very high by only transmitting those packets when less time-sensitive packets are not in the queue. This

works by simply choosing to transmit the packet with the shortest latency window first. Since files have a huge number of packets and long latency windows, the radio can't just procrastinate on sending file packets until they become most pressing. Instead, we split the file up into parts and assign a pseudo-latency requirement to each part. This effectively means that we attempt to transmit the file at a constant rate.

The MOE gets feedback about which packets are being dropped due to our latency gate. If we are consistently dropping packets, then the radio will start pruning the list of mandates that it is attempting. The end result is that our radios are only attempting the combination of mandates that are achievable.

3.3.1 Radio Node Spectrum Resource Allocation

At the network level, the master node performs mixed integer programming to determine how many time/frequency slots to give each node. The optimization is given as

$$\begin{aligned} \max_{\{\mathcal{S}_i\}_i} \quad & \sum_k p_k \mathcal{I}_k + \delta m \\ \text{s.t.} \quad & \sum_k r_{k,i} \mathcal{I}_k + m \leq \mathcal{S}_i c_i \ \forall \ i \\ & \sum_i \mathcal{S}_i = T \\ & \mathcal{S}_i \geq 1 \\ & l_k \geq L \text{ if } \mathcal{I}_k = 1 \\ & 3.4\text{ms } T/\mathcal{S}_i \geq \alpha L, \end{aligned} \tag{1}$$

where S_i is the slot count for the node i, T is the total slots available, m is a capacity margin that we want to be big, δ is the weight parameter we give to the margin in the maximization, $r_{k,i}$ is the rate required to achieve the kth flow that comes from the ith node, $I_k \in [0,1]$ is an indicator of whether the kth flow is active, c_i is the capacity out of the ith node, l_k is the maximum latency of the kth flow, a is the margin for the flow requirement, and p_k is the point value of the kth flow. The radio solves this optimization using Google's OR-Tools library [8].

The basic idea is to maximize the points while ensuring that each node has enough capacity to achieve its active flows. The optimization will result in the flow indicator, I_k, being zero for inactive flows. As part of the optimization, we also seek to maximize the capacity margin of each flow--that is the amount of excess capacity given each node should be consistent. This prevents the optimizer from doing a lopsided allocation for situations where all flows are easily achieved with the available rate. That is, if all nodes only require one slot to achieve their flows, then, without the slack variable, the optimizer will allocate 1 slot to 9 nodes and 41 slots to the

10th node. With the slack variable, the optimizer will allocated an equal number of slots to each node in this example.

The last two terms handle the latency constraint. The idea is that we need the spacing between slots to be less than the latency requirement and the spacing is approximately the total number of slots divided by the slot count for the node under consideration times the slot interval, (3.4 ms for this example but it can vary from 2-5 ms depending on the radio configuration).

Once the slot count is resolved, we perform a second optimization to evenly distribute the slots among the nodes. For instance, we may have a slot count of S = [4,1,5,5,5,10,1,2,2,12].

We must translate this slot count per node to an actual slot schedule and we need the schedule to evenly distribute the slots for a given node across time. To accomplish this goal, we solve

$$\begin{aligned} \max_{\{\mathcal{N}_{i,q}\}_{i,q}} \quad & |\mathcal{N}_{i,(q+1)\%T} - \mathcal{N}_{i,q}| \\ \text{s.t.} \quad & |\mathcal{N}_{i,:}| = \mathcal{S}_i, \end{aligned} \tag{2}$$

where $N_{i,q}$ is the slot index of the qth slot of the ith node. The solution to this problem provides the slot schedule which is then distributed to all nodes via broadcast transmissions.

At the individual radio level, each radio in our network is assigned non-overlapping time/frequency slots so each radio can act and optimize its score without any effect on the other radios. While the slot optimization at the master node produces a list of active and inactive flows per node, each node must still decide whether to pursue all active flows as the channel conditions evolve. To decide which flows to transmit, each node is continuously solving its local knapsack optimization to attempt the set of flows that maximize its score. If a flow that should be feasible continues to under-perform, it gets permanently blacklisted and excluded from the optimization in all future time steps.

3.3.2 Posture

Towards the end of year three, it became apparent that we had crafted a samurai sword for what was shaping up to be a gunfight. At that point our radio design was mostly optimized for precise, low-impact, collaborative operation in the ensemble. This meant using the minimum amount of spectrum required to achieve our flows, and erring on the side of settling for a lower score when greedier teams were in the spectrum. It became clear from practice runs that other teams were starting to use more than their "fair share" of the spectrum. This was a logical outcome of the year three scoring function and tournament ranking system which rewarded aggressive score seeking and even outright jamming of other teams in the spectrum.

Accordingly, we added two modes to our radio: one that would use much more spectrum than we thought was needed in order to disrupt other teams and a second that used a FDMA allocation so that we had multiple nodes transmitting in each time slot thereby increasing the transmit power

of each individual node. The master node assesses the performance of the other teams based on their score reports. If the score of the lowest team exceeds the threshold score (75% of the ensemble threshold score), then our radios all start using much more spectrum. If, in using much more spectrum, we are still not achieving the score target established by our slot optimization (80% of the score we think we should be achieving), we then transition into FDMA mode which significantly increases our transmit power per node and has the dual effect of increasing our SNR and increasing the interference we impart on the other teams' nodes.

3.3.3 Channel Selection

Our default behavior is to select the channels that are least occupied as we describe in the latter part of this section. However, when we increase our aggressiveness level, we have to be mindful of that the least robust team does not fall below the score threshold. So, when we are in higher-level aggression modes, we choose the allowed channel list simply by avoiding voxels that the lowest-scoring teams are transmitting in. This may mean that we have more collisions with high-scoring teams, but that tradeoff is worth it to lower the impact on the lowest-scoring teams.

In the low-aggression mode, our radio makes extensive use of the collaboration channel. In reasoning about which channels we choose to transmit on, we take into account spectrum voxel requests from all teams. Specifically, we use the free-space path loss model to predict the effect of a voxel request on each of our nodes. This leads to a 40 x 10 matrix per channel of how much power each of our nodes can expect to see from each of the other teams' nodes.

In order to build that power matrix, our radio first builds a distance matrix that has the pairwise distance from our 10 nodes to all other nodes in the match (e.g. 40 nodes for a 5-team match). An example distance matrix is displayed below. Some teams incorrectly report their position which can cause extremely large distance measurements. To deal with this, we clip all distances to 5000 m. It is clear in this run that one team is incorrectly reporting distances since we have 10 rows of 5000 m readings.

```
Sat Nov 24 09:47:52 2018 Distance matrix:
[[  664.02863056   676.93785172   731.15331597  3929.93897584  3785.3269746   3867.65498737  2203.37021537  2224.24606822  2028.40067877  2180.44015177]
 [  629.18605404   589.74327775   755.2603244   3666.45479298  3525.97610186  3608.69633138  1950.68270671  1977.72608737  1778.73148221  1931.6083406 ]
 [  657.27565326   641.56106439   760.42263193  3802.66359412  3655.22224577  3740.48969941  2084.58833501  2111.50508102  1909.98576643  2065.02098803]
 [ 3591.07129722  3471.99901635  3750.99365949   339.56809678   196.00040992   135.184636    1690.70169408  1723.88720816  1873.97548375  1737.88619332]
 [ 3656.12532834  3538.32756821  3810.61301968   417.24996221   305.14489372   233.3690154   1760.96301235  1783.0517055   1944.32258732  1802.58531861]
 [ 3805.27316059  3685.19727296  3968.02409946   282.62054863   306.31904144   216.16935673  1901.70136921  1942.71435411  2085.39764404  1953.31278358]
 [ 2286.55677926  2163.31625257  2456.04437089  1408.39695029  1245.06165778  1324.6023055    387.38939247   506.94137639   570.35496787   472.05172316]
 [ 2236.30901997  2113.90154364  2403.10903467  1458.89716363  1291.50644205  1369.33887383   329.9436545    439.27048183   515.83230603   408.19564759]
 [ 2168.35564032  2042.55469164  2342.5071647   1544.46798681  1398.29826426  1481.5919873    347.24684675   525.25078473   499.92102477   457.64440353]
 [ 2246.32603392  2126.13528539  2409.44222023  1469.88121469  1283.36033432  1357.04884584   359.11594061   390.01718585   535.1119124    395.03151369]
 [ 2351.12892489  2225.55982764  2527.0775029   1361.65736593  1216.38108763  1300.67619025   497.49602855   648.59911306   665.9293385    598.20839582]
 [ 3784.47582421  3661.41003182  3954.06675191    94.5898619    296.8011935    264.73225286  1880.2247425   1946.94780363  2064.76965536  1945.02826232]
 [ 2027.73560883  1903.9599217   2195.42747025  1668.21308751  1509.13274606  1588.47202608   158.65044196   339.14058214   323.32307387   268.75992728]
 [ 3860.78833203  3737.75067939  4025.1163546    166.04709791   365.75325304   322.5751063   1953.65077777  2017.80413203  2140.32902149  2017.67729779]
 [ 2207.11976141  2079.75313897  2385.53756556  1544.88377297  1414.73221662  1500.88667347   468.48953089   656.30009987   597.83580382   581.60330576]
 [  232.06471692   304.02182245   283.2776277   3837.18739419  3672.61088037  3750.40771498  2060.29230765  2042.28630531  1874.93762592  2015.63501803]
 [ 2187.17297636  2064.11396144  2357.125638    1508.66323549  1337.43636865  1416.56869493   284.23071959   400.81610365   468.18461725   365.09018285]
 [  253.43495178   215.42723608   413.69016909  3634.74411251  3472.43360413  3551.85484287  1864.28057096  1855.90402818  1680.19900649  1824.58478441]
 [  288.76384117   304.19118725   394.15928348  3753.08616384  3595.98950767  3674.86574723  1985.5003643   1977.77248717  1803.83901581  1946.18758566]
 [ 3769.48250805  3648.44323461  3933.79849809   196.12328438   251.32371737   170.99879633  1863.22841668  1912.19406852  2047.70211929  1918.93650018]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.          5000.        ]
 [ 1574.64220319  1457.17896625  1733.9694057   2136.43349348  1957.78347212  2030.49426002   361.14537111   302.23003607   194.1128065    284.08200356]
 [ 1737.75486516  1615.42608568  1904.03716422  1957.23255938  1790.38788135  1867.19678801   175.07548194   257.19654899    41.60685442   174.32016729]
 [ 1819.55653223  1705.36506551  1968.58177749  1927.21912512  1742.93839224  1811.49725645   273.25969248    86.95064228   258.11742723   160.94466595]
 [ 1759.77430754  1643.90976438  1916.44754775  1974.64826612  1786.34590653  1857.70995985   270.89377072   121.24608012   212.46557965   159.29514153]]
```

Figure 14 Example matrix of pairwise distances

These distances and the reported duty cycle from the CIL are used to create another matrix that is the pairwise "power observed" transfer function for each combination of Zylinium and opponent nodes. By summing down each column in this incident power matrix, we get the total power we expect to see from all other nodes in the network. We use this estimate to provide a scalar weighting of how occupied (from our team's perspective) each channel is. The MOE combines this collaboration occupancy information with information spectrum sensing broadcast information in order to create an ordered list of channel occupancy across all nodes in our network. The master node truncates this list to a smaller subset of channels and these are the channels available to all nodes for transmission.

The diagram below shows how the various inputs are combined to create this "channel occupancy" metric, T_c, over time. There is a voxel request component V_c that depends on the duty cycle and the distance, there is a spectrum sensing component S_c that depends on the spectrum sensing readings from our nodes, and then there is an incumbent component I_c, which is weighted by a factor of 3. The resulting metric, T_c, is shown in the plot at the bottom of the diagram.

The master node sort orders T_c and selects the N channels with the lowest occupancy, where N is dictated by the aggression posture at that moment in time.

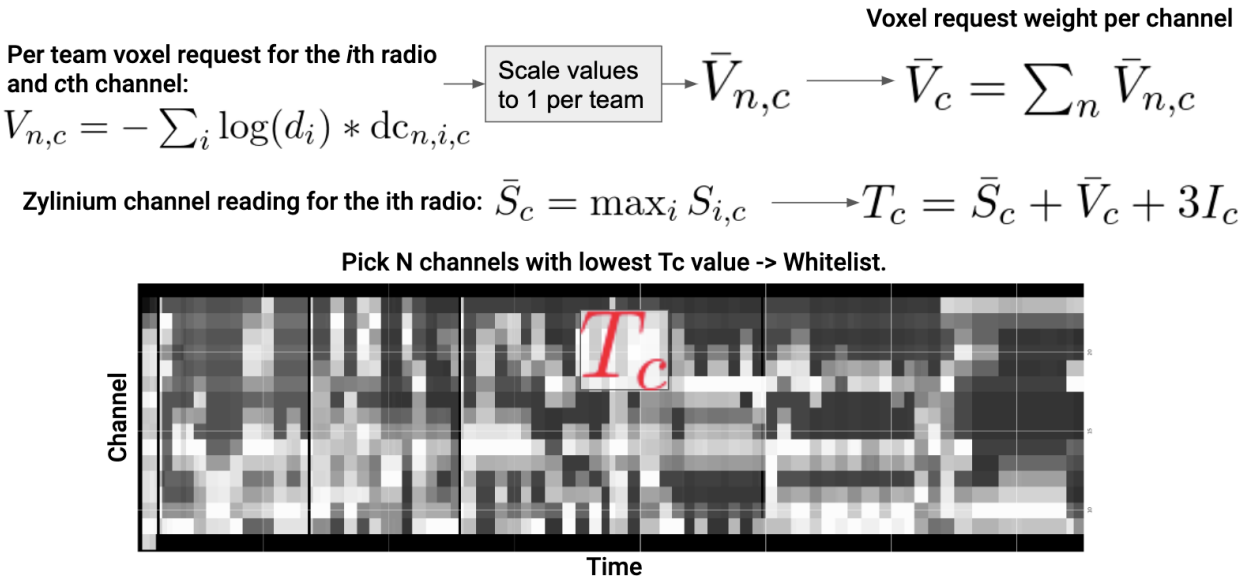


Figure 15 Description of how each radio creates the channel selection metric

The figure below shows our spectrum voxel requests plotted above the channel occupancy plot. It is clear from the voxel requests that our radio is going in and out of aggressive mode. It is also clear that there are stages where we use only a tiny amount of spectrum in the second half of the match. We can see from the two plots that the voxel request plot has puzzle-piece-like shapes that fit into the darker (lower-occupancy) portions of the spectrum occupancy plot. This provides some level of confirmation that the radio is selecting channels as intended.

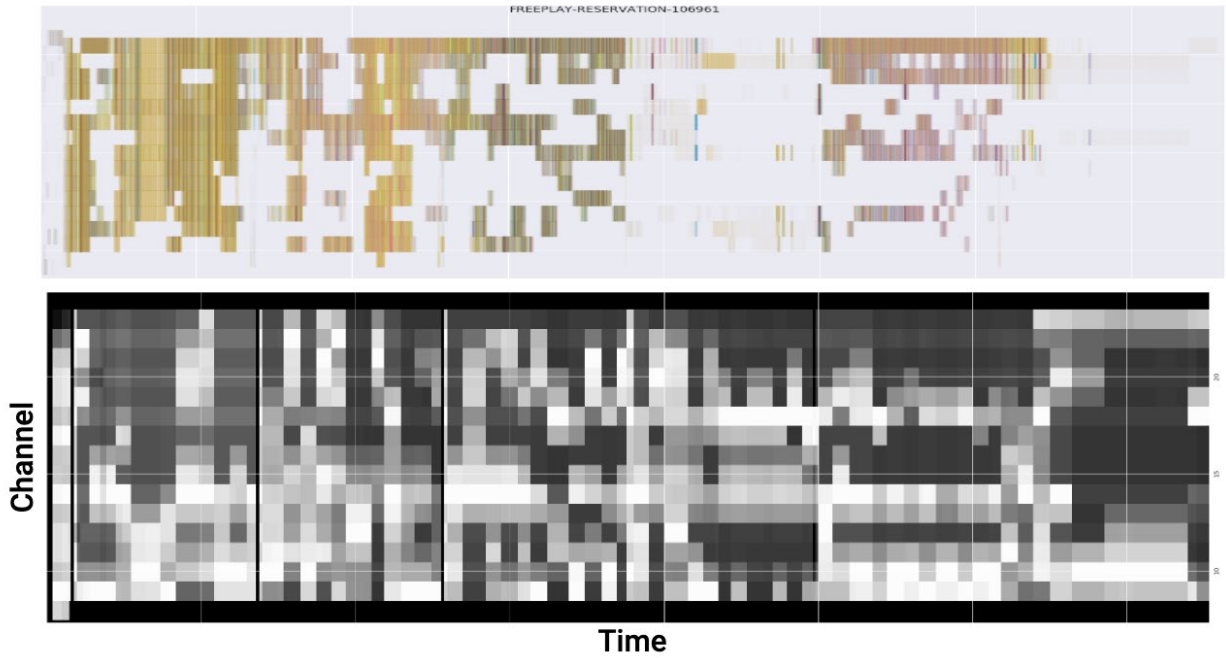


Figure 16 Illustration of the comparison between our spectrum usage (top) and the channel selection metric (bottom)

As an example, the plot below shows the sensed spectrum and the adjusted spectrum for a passive incumbent run. Think of this plot as one vertical cross-section of the T_c plot above. In this case, the passive incumbent occupied the upper half of the spectrum. It is clear that our adjusted spectrum measurements are taking this into account because the upper channels have a much higher "occupancy" value. It is also clear that there is another team in the match that is requesting voxels. This is clear because the lower channels also have an adjustment factor, though it is small. This adjustment factor is shown in the plot as the difference from the blue "x"'s and the orange circles. This indicates that the other teams are far away from our radios.

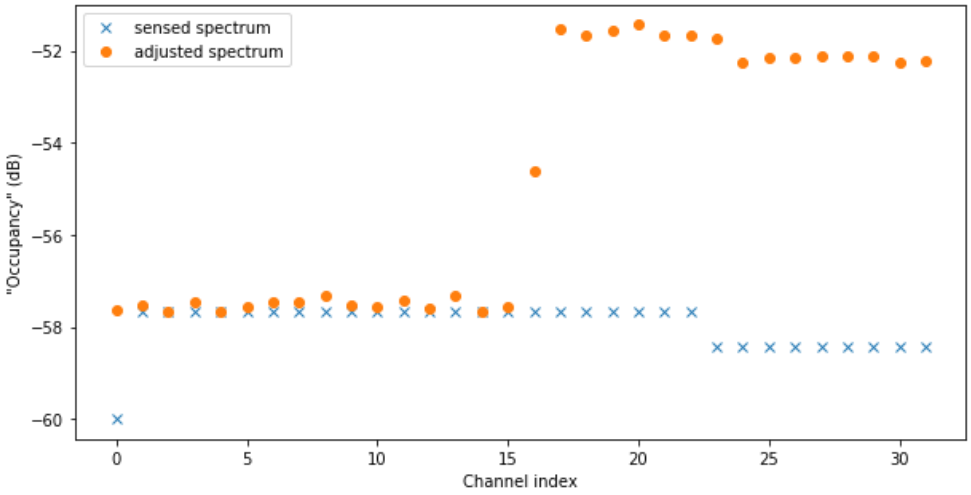


Figure 17 Spectrum occupancy as perceived by one of radios

3.4 DevOps

A significant factor in our (near) success had to do with how rigorously we logged and analyzed all of our Colosseum runs. We built a pipeline where we could kick off a batch job using any combination of branch or git-sha identifiers for each repo involved in our radio. With the branch specified, we had an Ansible [4] playbook that would build our LXC container, use jinja to template out the radio and batch configuration files, copy the files to the LZ, and kick off the job.

After a job finishes, we parse both our internal logs and the drc logs. The log information is stored in a postgres database. We also have a job that computes scores and metrics from each match. The metrics and scores are stored in postgres and can be viewed using our Tableau dashboards. In addition to Tableau, plots of the scores are sent to our team's messaging app (Slack) where they can be immediately viewed. In this process, we log the git-sha of all repos involved in our build so that we have traceability back to the code that produced a specific result. Storing the git-shas also allowed us to recreate any run by simply inputting those sha values into our Ansible playbook. The workflow from Ansible to log artifacts is shown in the figure below.

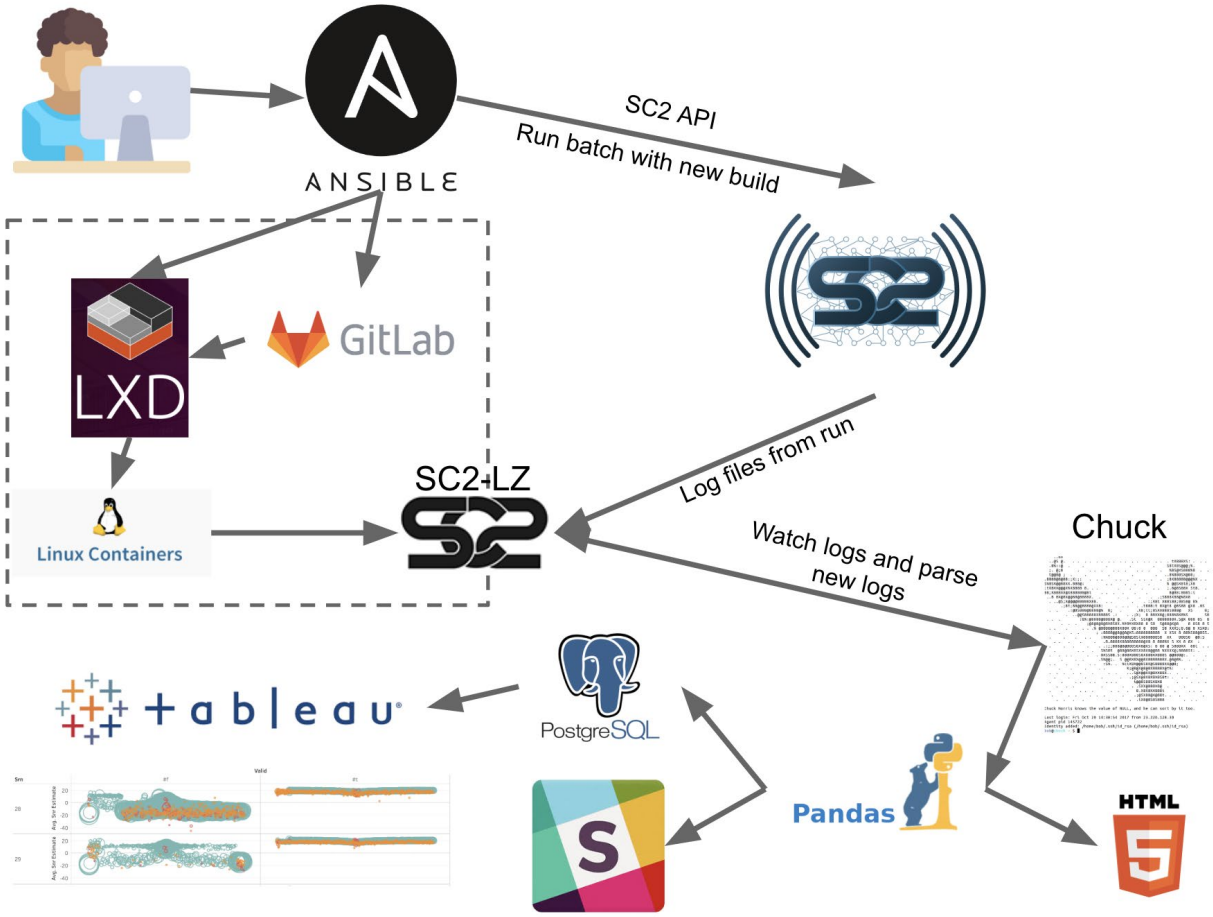


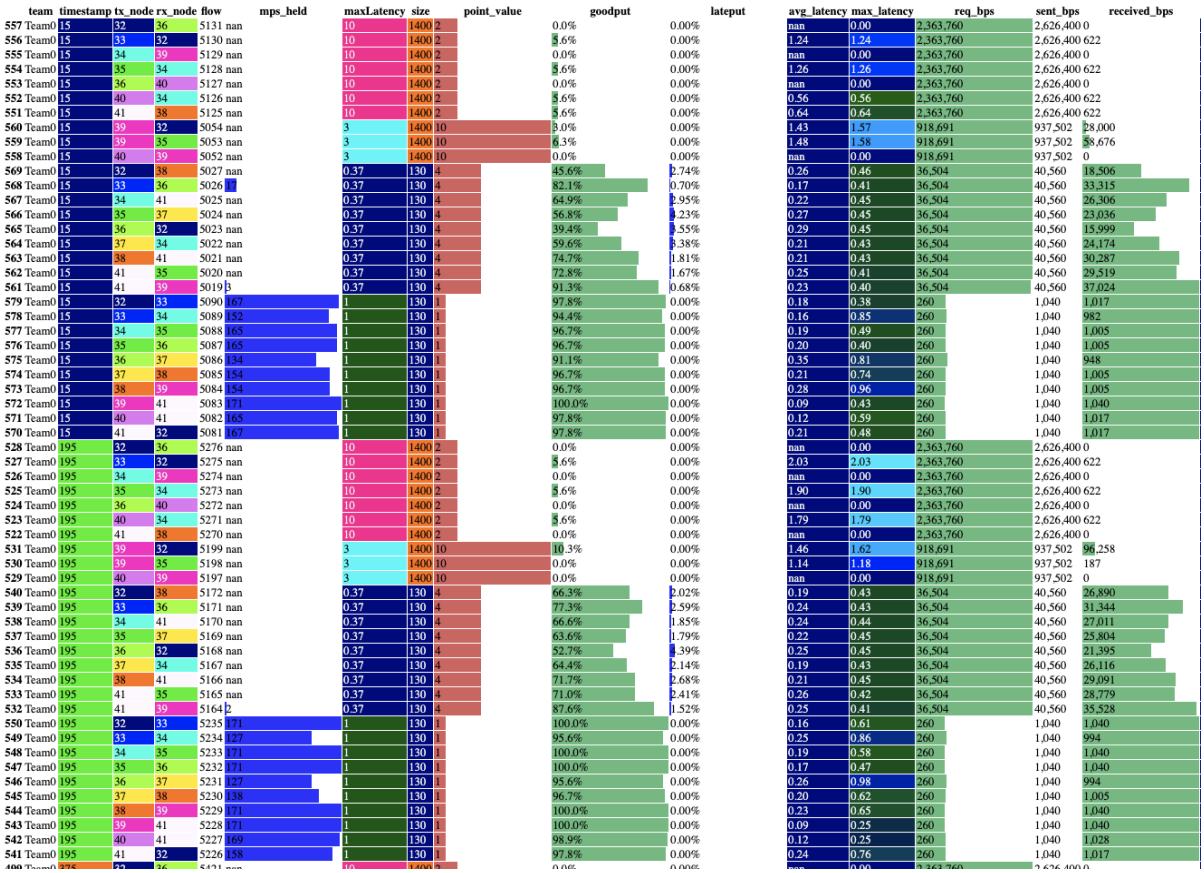
Figure 18 Job creation and data processing workflow

The figure below shows an example of the radio configuration for a given run as that configuration is printed in our Slack feed. You can see a dictionary of configuration parameters followed by a long list of the repositories that are part of our build. For each repository, we are printing out commit sha of that was used in the build.

```
RESERVATION-71284 | z-380-380;filerate1.2pct20_481_team1.conf | 4059
{"peer-pct-map": "{0: .2, 1: .2, 2: .2, 3: .2, 4: .2, 1000: .2}", "verbose": "True", "datasymbol-count": "50", "im-margin": "200000.0",
"num-nodes": "10", "max-latency": "0.7", "max-usrp-latency": "0.0", "assign-snr-thresh": "-10", "slot-spacing": "0.00015", "tx-amplitude":
"0.7", "tx-gain": "23", "cp-length": "8", "trans-length": "6", "file-sink-duration": "0.25", "file-sink-interval": "273", "shuffle-
interval": "0.95", "mod-fecs": "{ (1,0.875): 4.5, (2, 0.875): 7.5, (4, 0.65625):14, (4,0.875):18, (6, 0.7291666666666666):20.5, (6,
0.875):23}", "file-code-rate": "1.2", "seedoffset": "10", "rx-gain": "31.5", "num-mod-workers": "12", "num-demod-workers": "12", "logfile":
"/logs/radio.log", "pktlog": "/mnt/ramdisk/pkts.log", "statslog": "/logs/pkt_stats.log", "stdout": "/logs/stdout.log", "gain-update-
interval": "0.5", "max-rx-amp": "0.75", "min-rx-amp": "0.5", "file-sink-prefix": "/mnt/ramdisk/capture/cap", "corr-threshold": "7", "rx-
latency": "0.1", "chan0-scale": "2.0", "max-hops": "3", "partial-pkt-thresh": "0", "decoder-args": "--enc-fb-gen-method GA --dec-type SC --
dec-implem FAST --mdm-type QAM --mdm-max MAX --dec-simd INTRA"}
nlohmann-json: 359f98d14065bf4e53eeb274f5987fd08f16e5bf
googletest: de5be0eb28b74ecd6335e3bd61d9dc8914ce0e57
xtl: 4cc4a9d1028288435cce9d0188da5650651c98e1
xsimd: 6e7dbe48110b945b5f36be53b36bf898ff22ff02
xtensor: fae63bc381bf492b7f09d1336c7c0365f58d28d5
xtensor-fftw: 90c513c1228002b7823a0b9bd42fdaaaceace0b1
xtensor-blas: a02367e2c92583c95f34d2277afa3f5bf8a4c0e7
gpsd: 4e132a9d5be022aa570eba49f08b29797a0c139e
protobuf: a6189acd18b00611c1dc7042299ad75486f08a1a
uhd: 6db2e906652ba50a73e3a782ccce0be2583974f6
gnuradio: 14d2bfac73193bf36c32d5ae009b31786b1ed510
aff3ct: 2f116fec3db3e2b11e66f1679176fecfd02696f7
sc2-cm: 18d91be86923edebf21128c1ceca03edb41d3994
wirehair: 54ae467b53f32470cc5c726b8717e6d38d863784
wavelib: 87b234bc52126366302ca7db7b831d29f18bac86
gr-ettus: 4777fcb75bf2e15d08f87d7624202289b97140fa
rfnoc-zylinium: 0b65272e988e221f38ab524dda1e9fcf9bb3516a
gr-zylinium: 44229ab713ba07ec7ddbc2309c02597ceb49384a
collaboration-protocol: e9f8efc461a1f0e332ef2fb3a09c8efcaf5f84b8
fpga: 903987b4bda3a7bb8a6c9ea93e8547f247a040e9  ../fpga/x300.bit
```


Figure 19 Example log output produced by our log parsing pipeline

The figure below shows a snapshot of the flow-level metrics that are printed to Slack. We would scrutinize these reports in order to find deficiencies in our radio logic. Towards the end of SCE, we wanted to make sure that any flow our radio attempted led to an increase in our score. That is, we wanted to avoid the doubly-bad scenario where we use spectrum transmitting packets, but do not receive any points. We could do that check by ensuring if the "met_im" column was zero (meaning we didn't meet IMs for any MPs), then the percent should also be zero. If the percent column was a large number, but met_im was a small number, this would indicate that we are only getting a subset of the flow packets through and that, more importantly, we were not getting enough packets through to meet the mandate.



	team	timestamp	tx_node	rx_node	flow	mps_held	maxLatency	size	point_value	goodput	lateput	avg_latency	max_latency	req_bps	sent_bps	received_bps
557	Team0	15	32	36	5131	nan	10	1400	2	0.0%	0.00%	nan	0.00	2,363,760	2,626,400	0
556	Team0	15	33	32	5130	nan	10	1400	2	5.6%	0.00%	1.24	1.24	2,363,760	2,626,400	622
555	Team0	15	34	39	5129	nan	10	1400	2	0.0%	0.00%	nan	0.00	2,363,760	2,626,400	0
554	Team0	15	35	34	5128	nan	10	1400	2	5.6%	0.00%	1.26	1.26	2,363,760	2,626,400	622
553	Team0	15	36	40	5127	nan	10	1400	2	0.0%	0.00%	nan	0.00	2,363,760	2,626,400	0
552	Team0	15	40	34	5126	nan	10	1400	2	5.6%	0.00%	0.56	0.56	2,363,760	2,626,400	622
551	Team0	15	41	38	5125	nan	10	1400	2	5.6%	0.00%	0.64	0.64	2,363,760	2,626,400	622
560	Team0	15	39	32	5054	nan	3	1400	10	3.0%	0.00%	1.43	1.57	918,691	937,502	28,000
559	Team0	15	39	35	5053	nan	3	1400	10	6.3%	0.00%	1.48	1.58	918,691	937,502	58,676
558	Team0	15	40	39	5052	nan	3	1400	10	0.0%	0.00%	nan	0.00	918,691	937,502	0
569	Team0	15	32	38	5027	nan	0.37	130	4	45.6%	2.74%	0.26	0.46	36,504	40,560	18,506
568	Team0	15	33	36	5026	87	0.37	130	4	82.1%	0.70%	0.17	0.41	36,504	40,560	33,315
567	Team0	15	34	41	5025	nan	0.37	130	4	64.9%	2.95%	0.22	0.45	36,504	40,560	26,306
566	Team0	15	35	37	5024	nan	0.37	130	4	56.8%	4.23%	0.27	0.45	36,504	40,560	23,036
565	Team0	15	36	32	5023	nan	0.37	130	4	39.4%	3.55%	0.29	0.45	36,504	40,560	15,999
564	Team0	15	37	34	5022	nan	0.37	130	4	59.6%	3.38%	0.21	0.43	36,504	40,560	24,174
563	Team0	15	38	41	5021	nan	0.37	130	4	74.7%	1.81%	0.21	0.43	36,504	40,560	30,287
562	Team0	15	41	35	5020	nan	0.37	130	4	72.8%	1.67%	0.25	0.41	36,504	40,560	29,519
561	Team0	15	41	39	5019	3	0.37	130	4	91.3%	0.68%	0.23	0.40	36,504	40,560	37,024
579	Team0	15	32	33	5090	167	1	130	1	97.8%	0.00%	0.18	0.38	260	1,040	1,017
578	Team0	15	33	34	5089	152	1	130	1	94.4%	0.00%	0.16	0.85	260	1,040	982
577	Team0	15	34	35	5088	165	1	130	1	96.7%	0.00%	0.19	0.49	260	1,040	1,005
576	Team0	15	35	36	5087	165	1	130	1	96.7%	0.00%	0.20	0.40	260	1,040	1,005
575	Team0	15	36	37	5086	134	1	130	1	91.1%	0.00%	0.35	0.81	260	1,040	948
574	Team0	15	37	38	5085	154	1	130	1	96.7%	0.00%	0.21	0.74	260	1,040	1,005
573	Team0	15	38	39	5084	154	1	130	1	96.7%	0.00%	0.28	0.96	260	1,040	1,005
572	Team0	15	39	41	5083	171	1	130	1	100.0%	0.00%	0.09	0.43	260	1,040	1,040
571	Team0	15	40	41	5082	165	1	130	1	97.8%	0.00%	0.12	0.59	260	1,040	1,017
570	Team0	15	41	32	5081	167	1	130	1	97.8%	0.00%	0.21	0.48	260	1,040	1,017
528	Team0	195	32	36	5276	nan	10	1400	2	0.0%	0.00%	nan	0.00	2,363,760	2,626,400	0
527	Team0	195	33	32	5275	nan	10	1400	2	5.6%	0.00%	2.03	2.03	2,363,760	2,626,400	622
526	Team0	195	34	39	5274	nan	10	1400	2	0.0%	0.00%	nan	0.00	2,363,760	2,626,400	0
525	Team0	195	35	34	5273	nan	10	1400	2	5.6%	0.00%	1.90	1.90	2,363,760	2,626,400	622
524	Team0	195	36	40	5272	nan	10	1400	2	0.0%	0.00%	nan	0.00	2,363,760	2,626,400	0
523	Team0	195	40	34	5271	nan	10	1400	2	5.6%	0.00%	1.79	1.79	2,363,760	2,626,400	622
522	Team0	195	41	38	5270	nan	10	1400	2	0.0%	0.00%	nan	0.00	2,363,760	2,626,400	0
531	Team0	195	39	32	5199	nan	3	1400	10	10.3%	0.00%	1.46	1.62	918,691	937,502	96,258
530	Team0	195	39	35	5198	nan	3	1400	10	0.0%	0.00%	1.14	1.18	918,691	937,502	187
529	Team0	195	40	39	5197	nan	3	1400	10	0.0%	0.00%	nan	0.00	918,691	937,502	0
540	Team0	195	32	38	5172	nan	0.37	130	4	66.3%	2.02%	0.19	0.43	36,504	40,560	26,890
539	Team0	195	33	36	5171	nan	0.37	130	4	77.3%	2.59%	0.24	0.43	36,504	40,560	31,344
538	Team0	195	34	41	5170	nan	0.37	130	4	66.6%	1.85%	0.24	0.44	36,504	40,560	27,011
537	Team0	195	35	37	5169	nan	0.37	130	4	63.6%	1.79%	0.22	0.45	36,504	40,560	25,804
536	Team0	195	36	32	5168	nan	0.37	130	4	52.7%	4.39%	0.25	0.45	36,504	40,560	21,395
535	Team0	195	37	34	5167	nan	0.37	130	4	64.4%	2.14%	0.19	0.43	36,504	40,560	26,116
534	Team0	195	38	41	5166	nan	0.37	130	4	71.7%	2.68%	0.21	0.45	36,504	40,560	29,091
533	Team0	195	41	35	5165	nan	0.37	130	4	71.0%	2.41%	0.26	0.42	36,504	40,560	28,779
532	Team0	195	41	39	5164	2	0.37	130	4	87.6%	1.52%	0.25	0.41	36,504	40,560	35,528
550	Team0	195	32	33	5235	171	1	130	1	100.0%	0.00%	0.16	0.61	260	1,040	1,040
549	Team0	195	33	34	5234	127	1	130	1	95.6%	0.00%	0.25	0.86	260	1,040	994
548	Team0	195	34	35	5233	171	1	130	1	100.0%	0.00%	0.19	0.58	260	1,040	1,040
547	Team0	195	35	36	5232	171	1	130	1	100.0%	0.00%	0.17	0.47	260	1,040	1,040
546	Team0	195	36	37	5231	127	1	130	1	95.6%	0.00%	0.26	0.98	260	1,040	994
545	Team0	195	37	38	5230	138	1	130	1	96.7%	0.00%	0.20	0.62	260	1,040	1,005
544	Team0	195	38	39	5229	171	1	130	1	100.0%	0.00%	0.23	0.65	260	1,040	1,040
543	Team0	195	39	41	5228	171	1	130	1	100.0%	0.00%	0.09	0.25	260	1,040	1,040
542	Team0	195	40	41	5227	169	1	130	1	98.9%	0.00%	0.12	0.25	260	1,040	1,028
541	Team0	195	41	32	5226	158	1	130	1	97.8%	0.00%	0.24	0.76	260	1,040	1,017

Figure 20 Example table of flow information that is printed as html after each run

4 CIRN Decision Making

To ensure that we are responsive to the specific questions posed by DARPA we have addressed each question as a subsection below. Some of this information is also discussed in the preceding sections.

4.1 If you notice another team struggling to pass the Ensemble Threshold, how will your CIRN react?

The master node in our network is continuously monitoring the performance reports from other teams. In parallel, the master node has a decision loop callback that gets called once per second. In this callback, we examine the worst-performing team's score. If it is less than 75% of the ensemble threshold and we are in aggressive mode, then we step down through our aggressiveness levels at a rate of one step per 3 seconds. The aggressiveness levels are described in the posture section above.

If we find that we are the worse-performing team, then we immediately jump to the most aggressive posture regardless of whether other teams are struggling.

The master node keeps track of the network posture and transmits this to all other nodes in the network every 500ms.

A second mechanism we implemented to protect weaker teams is that we switch our spectrum allocation algorithm when we are in an aggressive posture mode. As background, when we are at the lowest level of aggression and are using minimal spectrum, we choose spectrum based on an algorithm that factors in the spectrum requests from all teams. This is described in the Channel Selection section above. However, when we are in aggressive mode, we stop considering the voxel requests from teams that are above threshold and instead only use voxels from teams that are reporting being under ensemble threshold.

4.2 How does your CIRN determine the appropriate number of flows that the spectrum resources can carry?

We have a two-level process for selecting which flows to attempt. There is a centralized component that performs a global flow selection for all nodes. Then, there is a radio-local component where more granular flow optimization decisions are made. At the centralized level, the master node performs the optimization described in Channel Selection. The result of this optimization is a "global flow blacklist" which is a list of the flows that should never be attempted in the local flow optimization.

At the local level, flows are selected by solving a local knapsack optimization problem where each node solves to maximize its points based on the capacity it has to each other node and the rate requirement for each flow. In order to avoid "flow jitter", we smooth the capacity estimates over time so that the optimization is stable.

To support both levels of optimization, each node senses and tracks the SNR of the packets it receives from every other node in the network. These packet-level SNRs get aggregated into a

link SNR between the node and all of the other nodes in the network. So each node is actively estimating 9 link SNR values for all of the links into itself. In parallel, as part of the broadcast thread, each node is transmitting these link SNR estimates.

The master node collects all of the link SNR estimates to create the full link SNR matrix for each pairwise link in the radio network. The SNR matrix is converted to capacity and these link capacities are what is used to perform the flow optimization.

One tricky element in the flow selection is that we do not know the rate required for the file flows until the file is received from the traffic engine by the node meant to transmit that file. The first file flow might not come in until well into the match. We had to implement two elements to handle this: 1) each node maintains a rate estimate of its file flows and transmits these as part of the broadcast messages, and 2) the flow optimization ignores file flows in the centralized channel allocation before the flow's rate is known.

4.3 How does your CIRN handle its own high point value (i.e., priority) traffic? How does your CIRN handle other CIRN's high point value traffic?

High-point-value traffic is treated just like any other traffic in the sense that the global and local flow selection optimization processes take into account the required rate and the point value of each flow. These parts of the MOE then use that information to select the subset of flows that are active. If the high-point flows have a rate requirement that is below the link capacity, then it is likely that they will be made active.

A high-point flow will not be attempted if 1) the rate requirement for that flow is higher than the capacity of the link or, 2) if the aggregate points of the other flows are higher and those flow can all be executed with a lower capacity requirement. That is, if the high-point flow is very hard relative to the number of points on offer, then the optimization may still not make that flow active.

Even when a flow is active, if it requires a very high rate that is near capacity, we may not be able to consistently close the link. If we sense that flow isn't consistently being achieved, we may deactivate it in favor of lower-rate flows even if those flows offer a lower point-per-capacity rate.

We do not take into account the flow point values attempted by other teams at all.

4.4 How does your CIRN handle active incumbents? What is the procedure/algorithm for learning its pattern?

We started with two assumptions: 1) the transmission schedule of the active incumbent is periodic, and 2) that we can infer when the active incumbent is not active by noting when the INR readings were exactly -12dB. Assumption 2 seemed to always hold, but assumption 1 occasionally fell apart due to the incumbent radio flowgraph dropping samples. It seems that the intention was for the incumbent to be periodic, but there were occasional glitches that caused that

not to be the case. Nevertheless, the incumbent did usually maintain periodicity for long periods of time.

Our approach was to:

1) First estimate the period of the incumbent transmission sequence by simply finding the peak of the autocorrelation of the INR reports.
2) Estimate the "mask" of when the incumbent was off in a given period by marking each 100ms-interval where INR was not equal to -12 as a "1", and other values as "0".
3) Pass three pieces of information to all nodes using the broadcasts: a) the timestamp of the beginning of a period, b) the length of a period, and c) a boolean mask of when it was safe to transmit in a period.
4) Each node uses the mask information to only transmit in time intervals where the incumbent was off.

As an example, here is a section of the broadcast message for RESERVATION-108854, which is scenario 8352: `"ai_prediction": {"ai_period": 6.000000, "ai_start_time": 1569470610.888695, "ai_mask": [0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1,1]}`. This indicates that the period is 6 seconds. The mask elements represent 100ms intervals. When the mask is "0", nodes are allowed to transmit. The plot below shows the active incumbent reports and our score. We were able to achieve all of the points without tripping the incumbent's threshold.

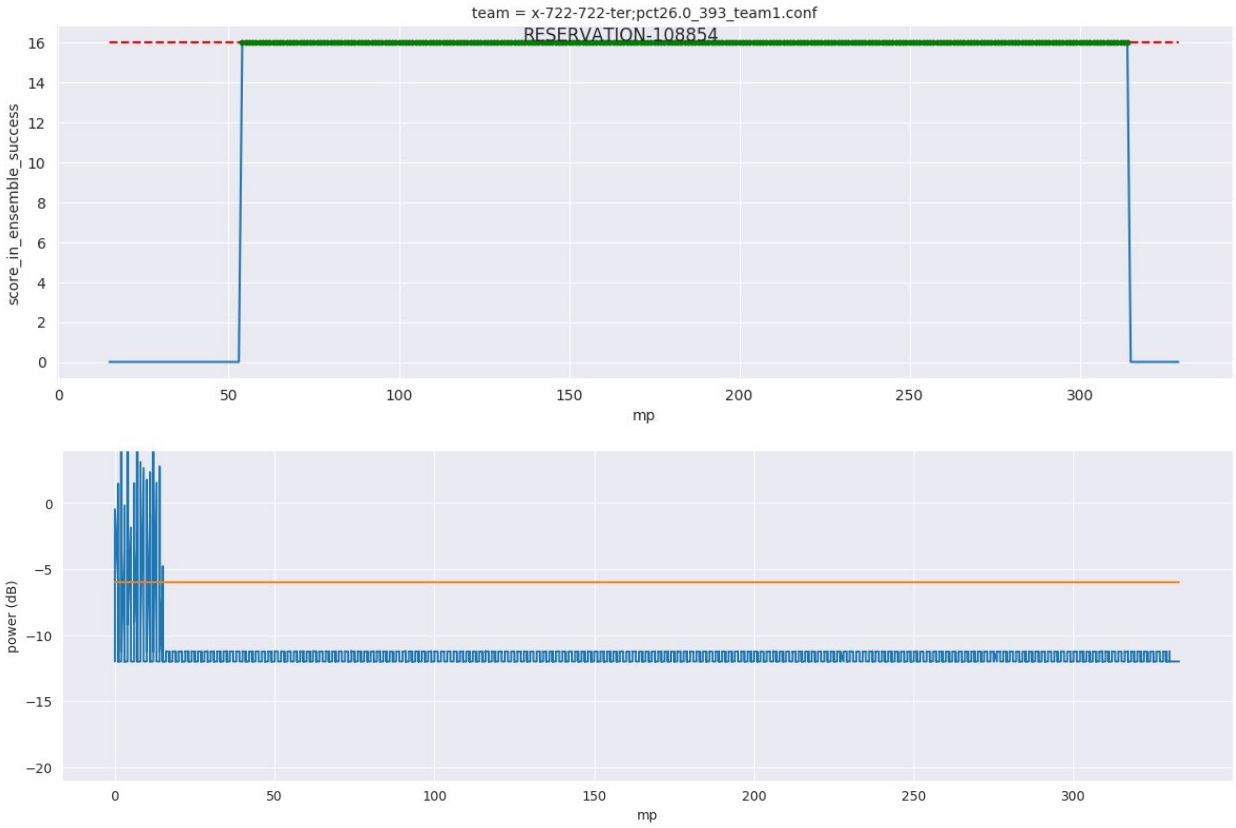


Figure 21 Top: score of active incumbent match. Bottom: Incumbent INR reports over time

Since our radio system relied on tight time synchronization across all nodes in order to achieve TDMA, we already had a broadcast infrastructure to keep the nodes time synchronized to sub-ms levels. To do this, each radio maintained a transmission event loop. The mechanics of that loop were that a radio would sleep for short periods of time in a loop and then transmit if it was its slot time. So we already had the mechanics of "blanking" transmissions while radios waited for their TDMA slot.

Once we had the active incumbent mask distributed to all nodes, this mask essentially acted as a TDMA slot mask such that nodes would only transmit if 1) it was their TDMA slot and 2) the current time was not prohibited by the active incumbent mask.

This ended up being very effective. Using this approach, we were able to have 5 Zylinium networks in the hardest active incumbent scenario at once, with all of the networks achieving the ensemble score threshold.

4.5 How does your CIRN handle passive incumbents? What is the procedure/algorithm for adapting aggregate interference?

There are two parts to how we handle passive incumbents: one part deals with the spectrum occupancy of the incumbent and how we allocate our channels to stay out of the incumbent's way, and the second part deals with incumbent violation reports.

The incumbent channel reservations get incorporated into the rest of our channel allocation logic in a similar way to voxel requests from other teams. That is, in our channel selection, we accumulate occupancy weights for each of channels into the metric T_c. As outlined above, the occupancy metric is a function of the distance of each team, their voxel requests, our spectrum sensing, and the incumbent channel reservations. The incumbent reservation is weighted the same as if three teams where requesting voxels. The net result of this scheme is that we will choose to transmit in passive incumbent voxels if all the other voxels are requested by 3 teams.

In reaction to violation reports, our master sends out a message to all other nodes to go to "silent mode". In that mode, only the master radio transmits packets and the only packets it transmits are broadcast messages twice per second. When the incumbent is no longer in violation, our radios all resume their operation with no modifications.

4.6 How does your CIRN handle jammers? How are they detected? What type of reaction do you expect from the CIRN?

We do not explicitly sense jammers. But each node is performing local channel selection based on spectrum measurements. A jammer will cause the spectrum power measurements to be very high in the channels used by the jammer. As a result, the radio will choose to use these channels with a lower prioritization compared to open channels.

4.7 When insufficient spectral resources are available, how does your CIRN decide which competitor's spectrum to attempt to use?

We have two modes for channel selection. When we are in "low-aggression mode", we just select the least occupied spectrum as perceived through the T_c metric. That is, we take into account voxel requests, node distances, and power sensed data to find the least occupied spectrum. In this mode we do not take into account the other teams' score.

In "aggressive mode", we perform the same channel selection process, but we exclude teams that are above threshold from the calculation. That is, we do not consider any voxel requests from teams above threshold in the calculation.

4.8 Does your CIRN estimate whether it's winning or losing a match? How does your radio react when it's winning? How does your radio react when it's losing?

We estimate our performance in a match by tracking our score estimate and the score reports from other teams. If our radio is in last place, we immediately go into aggressive mode. If all teams are above 75% of the ensemble threshold, we go into aggressive mode. If we are scoring less than 80% of our expected score, we go into aggressive mode with FDMA. We do not have any other rules that incorporate our relative match score position.

4.9 Is your CIRN able to detect which scenario it's in, and tune its performance accordingly? What scenario specific tuning does the CIRN do?

No, we make no attempt to fingerprint scenarios or to do scenario-specific tuning.

4.10 Do you have a strategy for ensuring you're not eliminated during the round robins?

We were generally pretty generous with spectrum when teams are below threshold, so we didn't expect to have any trouble in the round robins. Our biggest fear was that we would introduce a bug during our final development phases and that bug would cause a catastrophic failure that would lead to an early knockout in the round robin.

5 RESULTS AND DISCUSSION

5.1 Example: Collaboration Improving Performance

Towards the end of Phase 2, after we implemented the MOE, we were able to turn off the collaboration reasoning components in order to quantify how much collaboration helps the ensemble. We are pleased to report that collaboration is more effective at increasing an ensemble score than spectrum sensing alone. The plot below shows the outcome. It is clear that by accepting and reasoning about the spectrum voxel requests, we achieve a huge performance improvement.

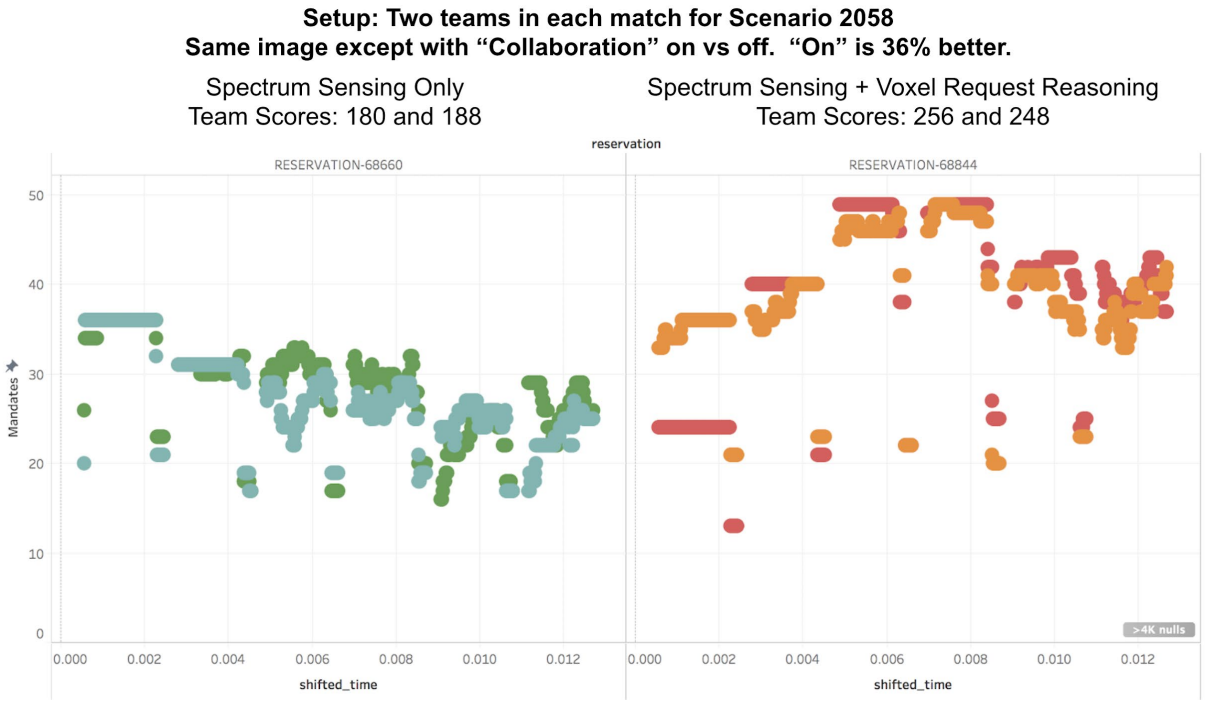


Figure 22 Comparison of performance with (right) and without (left) collaboration

5.2 Example: Protecting a Passive Incumbent from Aggregate Interference

In reasoning about passive incumbents, we used a fairly simple algorithm: if the incumbent power threshold is exceeded, stop transmitting everything but broadcast packets. The plot below shows a 3-team freeplay match where the ensemble is just barely breaching the incumbent threshold. Below the plot is a print out of our logs during this period. We can see both that we have tracked the power violation and that we have sent the "be silent" message to all our radios in an effort to reduce the ensemble power on the incumbent.

"peer_count": 3, "llc_is_silent": 1, "d_passiveViolation": 1}
"peer_count": 3, "llc_is_silent": 1, "d_passiveViolation": 1}
"peer_count": 3, "llc_is_silent": 1, "d_passiveViolation": 1}
"peer_count": 3, "llc_is_silent": 1, "d_passiveViolation": 1}
"peer_count": 3, "llc_is_silent": 1, "d_passiveViolation": 1}

Figure 23 Passive incumbent example

5.3 Example: Score Optimization with Weak Opponent & Strong Opponent

We can demonstrate our dynamic approach to both weak and strong opponents in one match, FREEPLAY-RESERVATION-109092. The following sequence of figures shows aspects of how our radio reacts to the other radios in the match.

The first plot below shows the spectrum usage for the 5 teams in a freeplay match. We are the team on the furthest left of the plot: Team 0. We have annotated the plot with periods of time where our radio enters and leaves “aggressive mode”. These periods of aggressive mode are clear from the plot because those are periods where our network is using most of the spectrum. Team 3 has a similar pattern of aggressive periods where it is also using most of the spectrum.

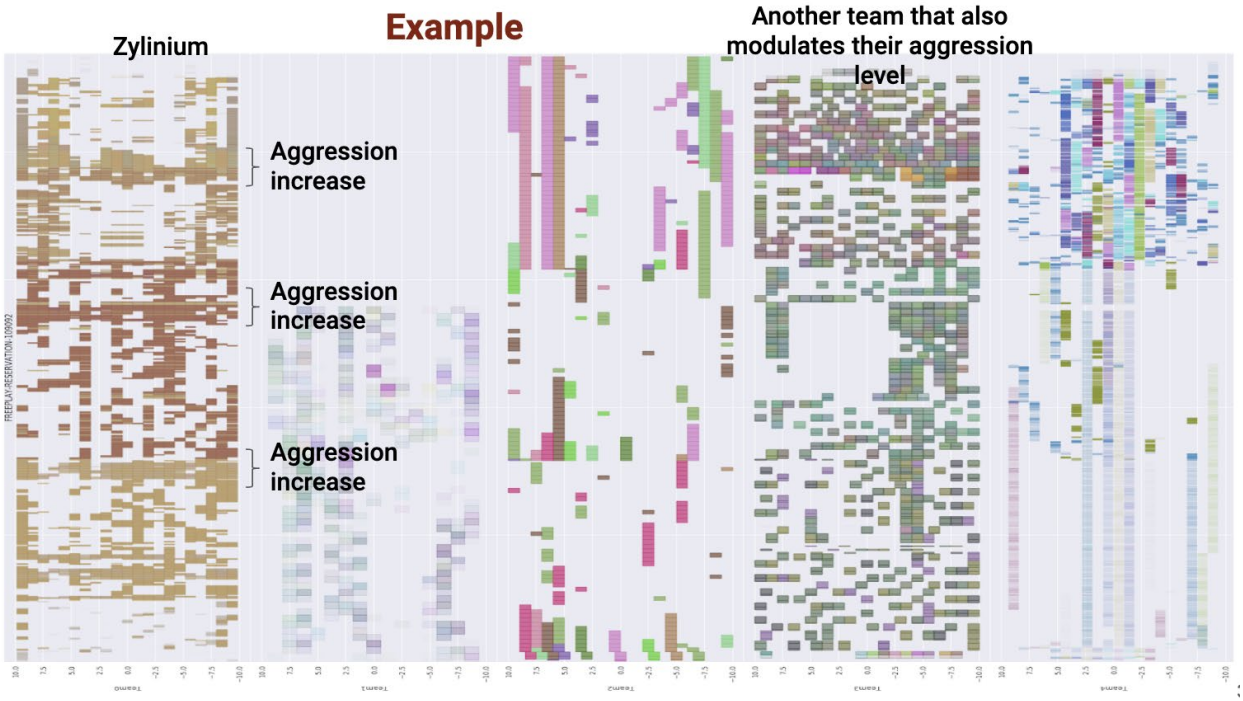


Figure 24 Spectrum voxel occupancy by team for a 5 team match

The figure below shows two plots: one is a plot of our spectrum usage over time and frequency as a spectrogram, and the other is a plot of the spectrum usage by team (y-axis) over time (x-axis). This is a 20MHz scenario and Zylinium is the blue line in the plot. It is clear from the plot, that there are periods of time where we choose to use 17MHz of the spectrum. These periods blip on and off in the first two stages. They are triggered by the worst team reporting a score over 75% of the ensemble threshold. Once the worst team drops below that level, our spectrum usage drops back down to 7MHz of spectrum, which is our neutral posture.

From the plot, we can see that in stage 3, after entering high-bandwidth mode, we had a different "fall off" response where we slower step our spectrum down back to the neutral posture level. This indicates two things 1) that the worst team has stopped scoring above 75% of the ensemble threshold and 2) that our network is not scoring at least 80% of the score our optimization algorithm predicts we should be scoring. Team 3 (red line), has a similar gradual response out of the top aggression level, although they take much longer to get back to their nominal frequency usage. That is, they are more aggressive than we are.

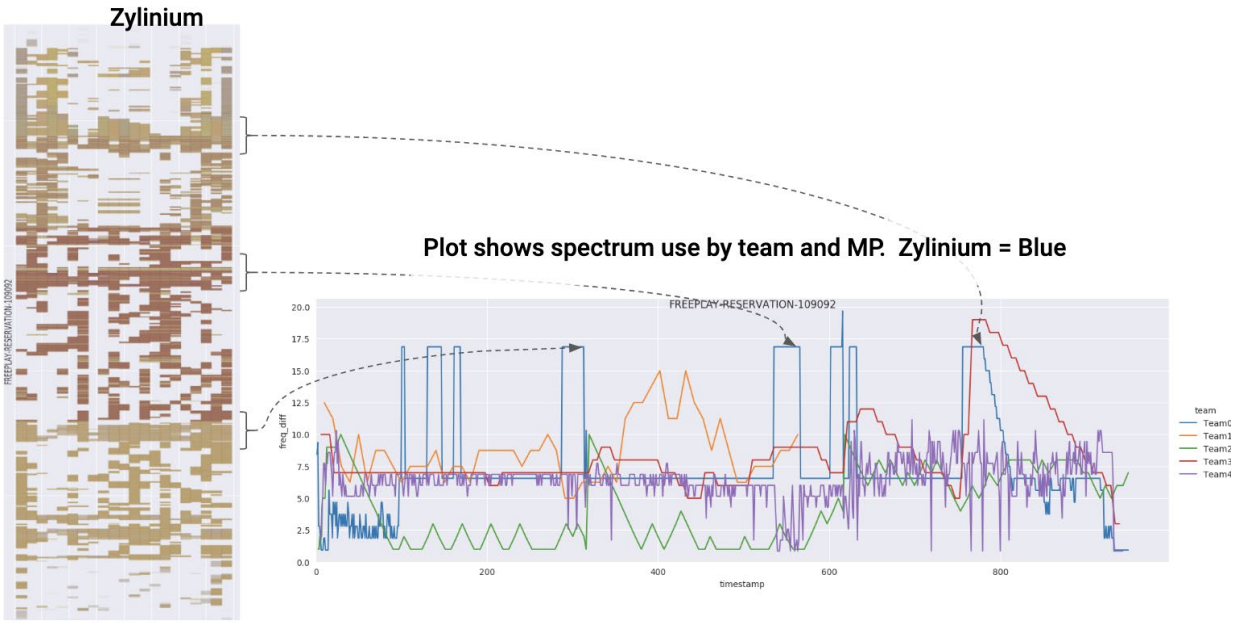


Figure 25 Spectrum voxel usage plotted in units of MHz used vs time

The figure below compares the team score (left), with the spectrum usage plot from the last figure. In the score plot, the red dashed line is the maximum score of that team at that MP, the magenta dots indicate that that team is above ensemble threshold, and the green dots indicate that the ensemble is above the threshold. Team 1 (second row) seems to be the performance-limiting team in the match, but it is clear that when they have bursts of score above threshold, that those small scoring bursts correspond to when our network uses the most spectrum, which verifies that our rule is working.

By stage 3, Team 1 has stopped reporting its score, and thus, we no longer consider them to be part of the match. Accordingly, our burst of aggression in stage 3 corresponds to the period of time when both Team 2 and Team 4 briefly exceed the ensemble threshold.

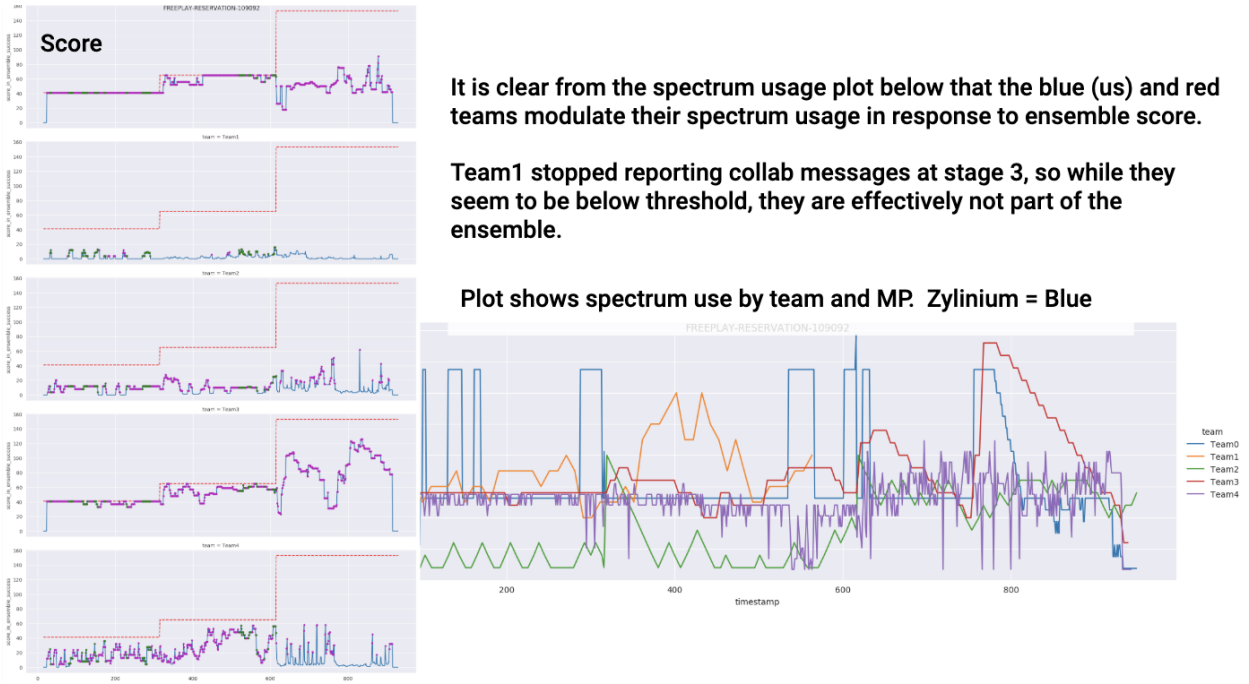


Figure 26 Score and spectrum usage for a match

5.4 Example: Identifying Spectrum to Reuse

We never reuse the same channel in the same timeslot across the network. That is, we never have two of our radios transmitting at the same time/frequency slot. Because of our very narrow time slots and because the nodes only operated in the allowed channel set specified by the central coordination component of our channel selection algorithm, it may appear at a macro level that we are doing frequency reuse across different portions of our radio network. But, in fact, we never reuse the same frequency in the same time. We only reuse frequency in an aggregate sense such that in a period of multiple seconds all radios are likely to confine their transmissions to the same set of "allowed channels".

5.5 Example: Flow Prioritization

A critical aspect of the competition was the ability for a radio network to judiciously select network flows to attempt. As explained in the preceding sections, we used variations on the knapsack problem to decide which flows to attempt. In this example, we use several plots from our analysis pipeline to show the behavior of our flow prioritization logic. These plots correspond to a 3-team freeplay match in scenario 7089, which is the scenario with very high-point-value flows. The first figure below is our score. It is clear from the plot that we are scoring nearly the maximum score in most of the match, which means that our radio is able to achieve the high-value flows.

Figure 27 Score for a 7089 match

The next figure shows two plots. The x-axis of the plots is time. Each dot corresponds to an MP in the match. The y-axis of the plot has rows of dots plots and each row corresponds to a flow. The annotation on the y-axis is "Tx ID - Flow ID - Point Value - Bps Requirement". So both of these plots show a subset of the flows for SRN 72. Some of the flows only require 260Bps (e.g. 5670-5674), while flow 5646 requires 918MBps.

The top plot has two dimensions of information encoded in each row: 1) the color encodes how much relative capacity the flow requires compared to how much capacity is available in that link: blue is nearly 0, while dark red indicates that the flow exceeds the capacity of the link, and 2) wider lines indicate MPs where our flow selection optimization has "blacklisted" the flow which means that, given the capacity, and our objective of maximizing the score, it is suboptimal to attempt that flow. As time passes, the link conditions may change and create a condition where it again makes sense to attempt this flow.

We can actually see this in the plot where the capacity requirement for the 5646 flow goes down (color becomes less red), thus leaving room for the 10MBps 5651 flow to be attempted. So in this example, the 5651 flow is the only flow that is avoided and it is only avoided for a short period of time.

The second plot in this figure shows the performance of each flow by MP. Green indicates that the flow requirements were met and the mandate was achieved in that MP, while red indicates that the mandate was not achieved. In this case, we are doing well in the low-rate flows, but we are not achieving the mandates for the two high-rate flows even though we are attempting those flows. That is the optimization routine thinks we should have the capacity to achieve these flows (i.e. they are not blacklisted), but we are actually not achieving these flows. In this case, bursty interference kept us from achieving the flows.

Given more time, we would have developed more mechanisms to deal with missed mandates like this. It is not immediately clear why we were underperforming on these flows in this example.

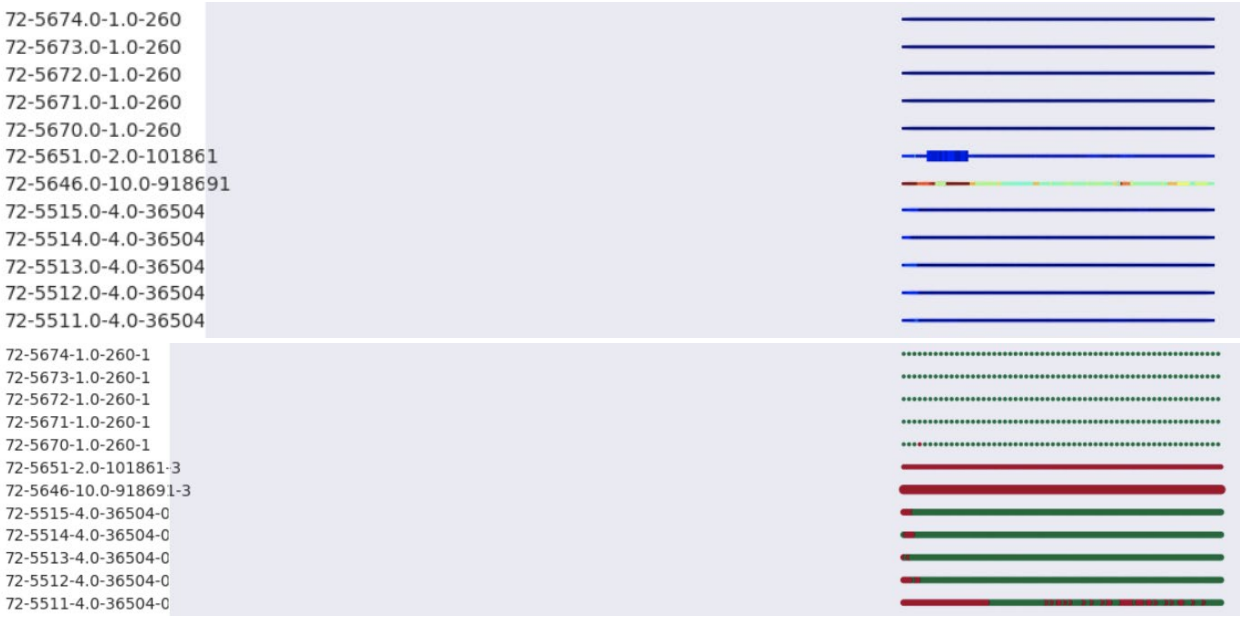


Figure 28 Top: visualization of the capacity required for each flow. Bottom: visualization of mandates were achieved by MP

5.6 Example: Meeting Qualification Criteria

There were various qualification criteria. CIL compliance for score reporting and general reporting is not that interesting. We simply verified that we met the criteria by running the CIL tool. The two more interesting requirements were the traffic rate requirement and the voxel reporting requirement.

For the traffic throughput requirement, we had a series of plots that showed the throughput of each link. The traffic criteria was based on the aggregate flow rate of all links in a network instead of the number of mandates achieve. Since this was the only situation where aggregate rate was relevant, we did not have a plot for aggregate rate. But we did track the rate per flow and we tracked the overall score, which is a function of the number of mandates met in each MP. If a network achieved all of the mandates in all of the MPs, then that was a sufficient but not necessary condition for passing the qualification hurdle.

The figure below shows a typical plot from our qualification run. In almost every MP, our score (green dots) equaled the maximum stage score (red dashed line), which indicates that we met the criteria.

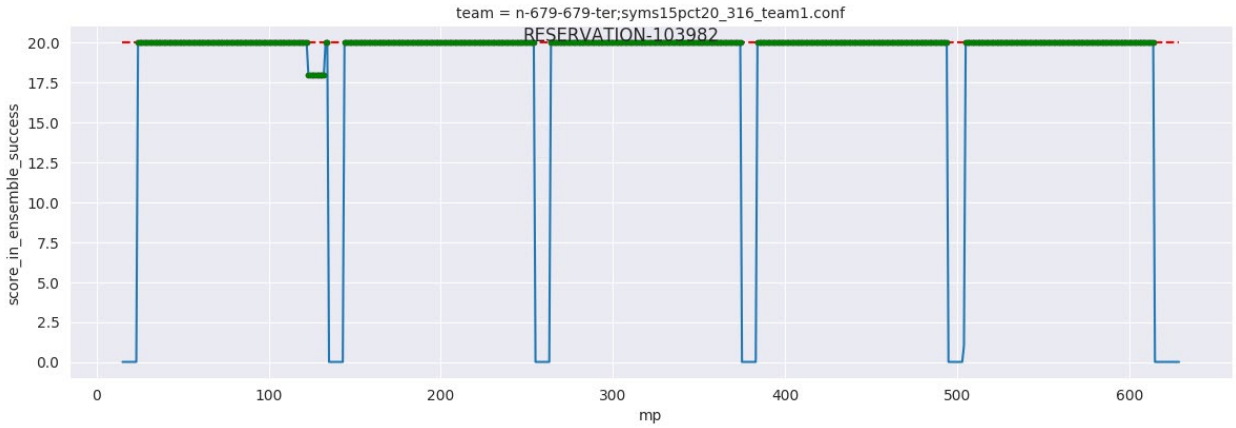


Figure 29 Score in a qualification run

A second confirmation that we met the traffic criteria is given in the table below which shows the flows from the first stage and part of the second stage for this match. The right-most column lists the goodput rate in bits per second for each flow. The third-from-the-right column, "req_bps" shows the average rate that we needed to achieve for all flows in order to meet the qualification criteria. It is clear from comparing those two columns that every flow has exceeded the average rate requirement and thus, that we have passed the qualification hurdle.

	team	timestamp	tx_node	rx_node	flow	mps_held	maxLatency	size	point_value	goodput	lateput	avg_latency	max_latency	req_bps	sent_bps	received_bps
99	team1.conf	15	38	42	5020	111	0.37	625	1	88.0%	0.00%	0.09	0.28	1,000,000	1,250,000	1,100,083
98	team1.conf	15	38	48	5019	111	0.37	625	1	88.0%	0.00%	0.09	0.27	1,000,000	1,250,000	1,100,042
97	team1.conf	15	39	38	5018	111	0.37	625	1	88.0%	0.00%	0.09	0.31	1,000,000	1,250,000	1,100,083
96	team1.conf	15	39	40	5017	111	0.37	625	1	88.0%	0.00%	0.09	0.25	1,000,000	1,250,000	1,100,083
95	team1.conf	15	40	39	5016	111	0.37	625	1	88.0%	0.00%	0.10	0.25	1,000,000	1,250,000	1,100,042
94	team1.conf	15	40	44	5015	111	0.37	625	1	88.0%	0.00%	0.10	0.24	1,000,000	1,250,000	1,100,292
93	team1.conf	15	41	40	5014	111	0.37	625	1	88.0%	0.00%	0.09	0.25	1,000,000	1,250,000	1,100,000
92	team1.conf	15	41	46	5013	111	0.37	625	1	88.0%	0.00%	0.09	0.25	1,000,000	1,250,000	1,100,000
91	team1.conf	15	42	41	5012	111	0.37	625	1	88.0%	0.00%	0.09	0.26	1,000,000	1,250,000	1,099,708
90	team1.conf	15	42	43	5011	111	0.37	625	1	88.0%	0.00%	0.10	0.36	1,000,000	1,250,000	1,099,708
89	team1.conf	15	43	42	5010	111	0.37	625	1	88.0%	0.00%	0.09	0.27	1,000,000	1,250,000	1,100,083
88	team1.conf	15	43	45	5009	111	0.37	625	1	88.0%	0.00%	0.09	0.27	1,000,000	1,250,000	1,100,042
87	team1.conf	15	44	43	5008	101	0.37	625	1	88.0%	0.00%	0.10	0.27	1,000,000	1,250,000	1,100,208
86	team1.conf	15	44	48	5007	111	0.37	625	1	88.0%	0.00%	0.10	0.27	1,000,000	1,250,000	1,100,000
85	team1.conf	15	45	39	5006	111	0.37	625	1	87.8%	0.00%	0.09	0.26	1,000,000	1,250,000	1,097,917
84	team1.conf	15	45	44	5005	111	0.37	625	1	88.0%	0.00%	0.09	0.27	1,000,000	1,250,000	1,099,792
83	team1.conf	15	46	41	5004	111	0.37	625	1	88.0%	0.03%	0.09	0.40	1,000,000	1,250,000	1,099,750
82	team1.conf	15	46	45	5003	111	0.37	625	1	88.0%	0.00%	0.09	0.25	1,000,000	1,250,000	1,100,083
81	team1.conf	15	48	38	5002	111	0.37	625	1	88.0%	0.00%	0.10	0.25	1,000,000	1,250,000	1,099,917
80	team1.conf	15	48	46	5001	101	0.37	625	1	88.0%	0.00%	0.10	0.25	1,000,000	1,250,000	1,100,250
79	team1.conf	135	38	42	5040	111	0.5	625	1	67.7%	0.00%	0.09	0.26	750,000	1,250,000	845,750
78	team1.conf	135	38	48	5039	111	0.5	625	1	67.7%	0.00%	0.09	0.29	750,000	1,250,000	845,667
77	team1.conf	135	39	38	5038	111	0.5	625	1	67.7%	0.00%	0.09	0.28	750,000	1,250,000	845,625
76	team1.conf	135	39	40	5037	111	0.5	625	1	67.7%	0.00%	0.09	0.28	750,000	1,250,000	845,708
75	team1.conf	135	40	39	5036	111	0.5	625	1	67.7%	0.00%	0.09	0.25	750,000	1,250,000	845,750
74	team1.conf	135	40	44	5035	111	0.5	625	1	67.7%	0.00%	0.09	0.23	750,000	1,250,000	845,667

Figure 30 flow data for a qualification run

The other interesting qualification criteria was the spectrum compliance criteria. The figure below shows our actual spectrum usage (red) overlaid on top of our CIL-reported spectrum usage (blue). The opacity of blue encoded the duty cycle of the voxels that we reported in the CIL. It was common for our voxel `'predicted_use': {'in_voxel_error': {'competitor_value': 0.17, 'threshold_value': 0.39, 'pass': True},`

`'out_of_voxel_error': {'competitor_value': 0.05, 'threshold_value': 0.17, 'pass': True}`. This shows that we passed the criteria by a large margin: 17% vs 39% threshold and 5% vs 17% threshold.

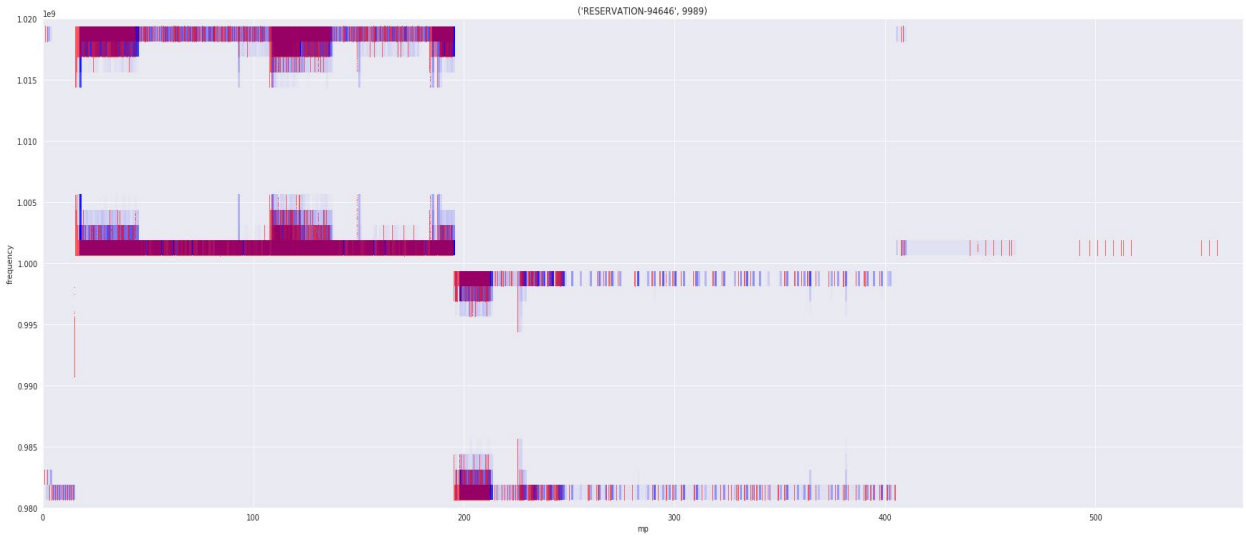


Figure 31 Spectrum usage vs CIL reported usage for a qualification run

6 CONCLUSIONS

The main conclusion from three years of intense work is that it is possible to create waveforms that can function in environments made of heterogeneous spectrum users. More importantly, we collectively demonstrated that a low-rate collaboration channel between the spectrum users can greatly enhance the collective ensemble radio performance. We intend to take our observations about the nuggets of success from SC2 and apply them to new commercial ventures in the spectrum space.

7 RECOMMENDATIONS/LESSONS LEARNED

The competition was very well run. We do not have any notable recommendations. Like any large scale competition where there are competing objectives and many teams with conflicting demands of the competition staff, some difficulties are impossible to avoid. Likewise, when there is only one winner, non-winning teams will feel aggrieved in myriad ways. The competition was certainly fair and all competitors had access to the rules and the Colosseum resources to succeed. The Colosseum is a truly amazing resource for spectrum research and development. Building and making it available to the community is a huge accomplishment for the DARPA SC2 team.

As we discussed in the previous section, we are excited to carry the momentum of the technological breakthroughs forward into larger-scale experiments and ultimately commercial applications. We recommend that DARPA and the Federal Government continue to fund research and development in these areas in order to keep the momentum going.

8 REFERENCES

[1] Tilghman, Paul. “If DARPA Has Its Way, AI Will Rule the Wireless Spectrum” *IEEE Spectrum* 56.6 (2019): 28-33.

[2] Bernstein, David. “Containers and cloud: From LXC to Docker to Kubernetes.” *IEEE Cloud Computing* 1.3 (2014): 81-84.

[3] Naval Research Laboratory, *Multi-Generator (MGEN) traffic generation tool*, 2019, Github Repository, github.com/USNavalResearchLaboratory/mgen.

[4] Hochstein, Lorin, and ReneMoser. *Ansible: Up and Running: Automating Configuration Management and Deployment the Easy Way*. O’Reilly Media, Inc., 2017.

[5] Eric Blossom, *GNU radio: tools for exploring the radio frequency spectrum*, Linux J. 2004.

[6] Braun, Martin, Jonathan Pendlum, and Matt Ettus.“RFNoC: RFnetwork- on-chip.” *Proceedings of the GNU Radio Conference*. Vol. 1. No. 1. 2016.

[7] Tang, Z., Cannizzaro, R. C., Leus, G., Banelli, P. (2007). Pilot-assisted time-varying channel estimation for OFDM systems. IEEE Transactions on Signal Processing, 55(5), 2226-2238.

[8] Perron, Laurent. “Operations research and constraint programming at Google.” *International Conference on Principles and Practice of Constraint Programming*. Springer, Berlin, Heidelberg, 2011.

9 List of Acronyms

ADC	Analog to Digital converter
ACS/GCP	Amazon Cloud Service/Google Cloud Platform
AP	Access Point
API	Application Programming Interface
BLE	Bluetooth Low Energy
BPSK	Binary Phase Shift Keying
BRAM	Block Random Access Memory
BT	Bluetooth
CFAR	Constant false alarm rate
CFO	carrier frequency offset
CIL	CIRN Interaction Language
CIRN	Collaborative Intelligent Radio Network
CPU	Central Processing Unit
CRC	Cyclic redundancy check
DAC	Digital to Analog Converter
DARPA	Defense Advanced Research Projects Agency
DDC	Digital Downconversion
DMA	Direct Memory Access
DSP	Digital Signal Processor
DUC	Dynamic Update Client
FCC	Federal Communications Commission
FDD	Frequency-driven development
FDMA	Frequency-division multiple access
FFT	Fast Fournier Transform
FIFO	First In First Out
FIR	Finite Impulse Response
FPGA	Field-programmable gate array
GCP	Google Cloud Platform
GNU	GNU'S not Unix
GPS	Global Positioning System
HTML	Hypertext Markup Language
IM	Individual Mandate
INR	International Normalized Ratio
IoT	Internet of Things
IQ	Quadrature signal
ISM	Industrial, Scientific, and Medical
LDPC	Low-density parity-check
LO	Local Oscillator
LTE	Long-term Evolution
LUT	Look up table
LXC	Linux containers
LZ	lzip archive
MAC	Medium Access Control
MGEN	Multi-Generator

MHz	Megahertz
MO	Mandated Outcome
MOE	Mandate Optimization Engine
MP	Measurement Period
MPS	Measurement Period Score
MVP	Minimum viable product
NCO	Numerically-controlled oscillator
NR	New Radio
NTP	Network Time Protocol
OFDM	Orthogonal Frequency-Division Multiplexing
PAWR	Platforms for Advanced Wireless Research
PE	Preliminary Event
PHY	Physical Layer
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RAM	Random Access Memory
RF	Radio Frequency
RFNoC	RF Network on Chip
RX	Receive
SC2	Spectrum Collaboration Challenge
SCE	SC2 Championship Event
SDR	Software Defined Radio
SIMO	Single-input multiple-output
SNR	Signal-to-noise ratio
SRN	Standard Radio Node
TDD	Test-driven development
TDM	Time-division multiplexing
TDMA	Time-division multiple access
TX	Transmit
UHD	USRP Hardware Driver
ZMQ	ZeroMQ
ZSE	Zylinium Spectrum Exchange