University of Massachusetts Boston

Presented by : **Josh Vo, Austin Trinh, Johnny Daou, Osiris Germain, Simon Pham**

Distributed Spectrum Sensing Research Tool

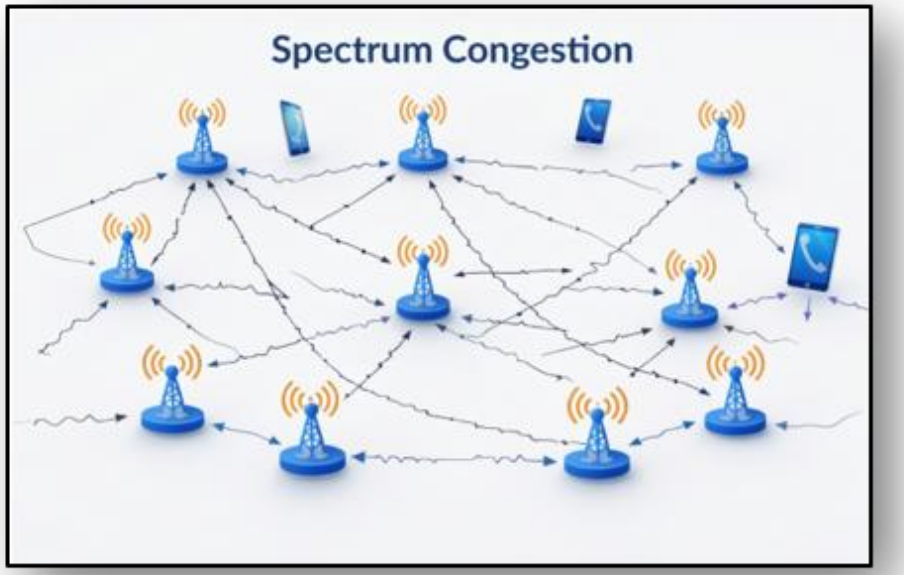

Joshua Vo

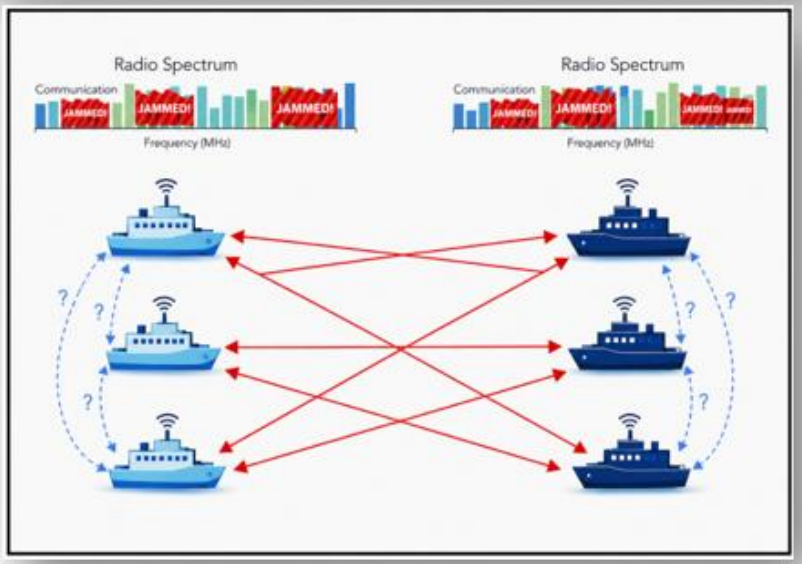

Austin Trinh


Johnny Daou


Osiris Germain


Simon Pham





Introduction & Motivation

Background

The radio spectrum is divided into licensed channels assigned to **Primary Users (PUs)**. Spectrum Congestion becomes a problem when many users are using a channel. To combat this, the **Dynamic Spectrum Access (DSA)** framework is introduced to help with efficient management of the frequency spectrum. Dynamic Spectrum Access (DSA) allows **Secondary Users (SUs)** to use idle channels when PUs are inactive. This research focuses on **Distributed Spectrum Sensing (DSS)**, where nodes make sensing decisions without relying on a central controller.

Problem

Spectral awareness is essential, and as technologies advance, issues such as spectrum congestion and interference are more likely to arise.

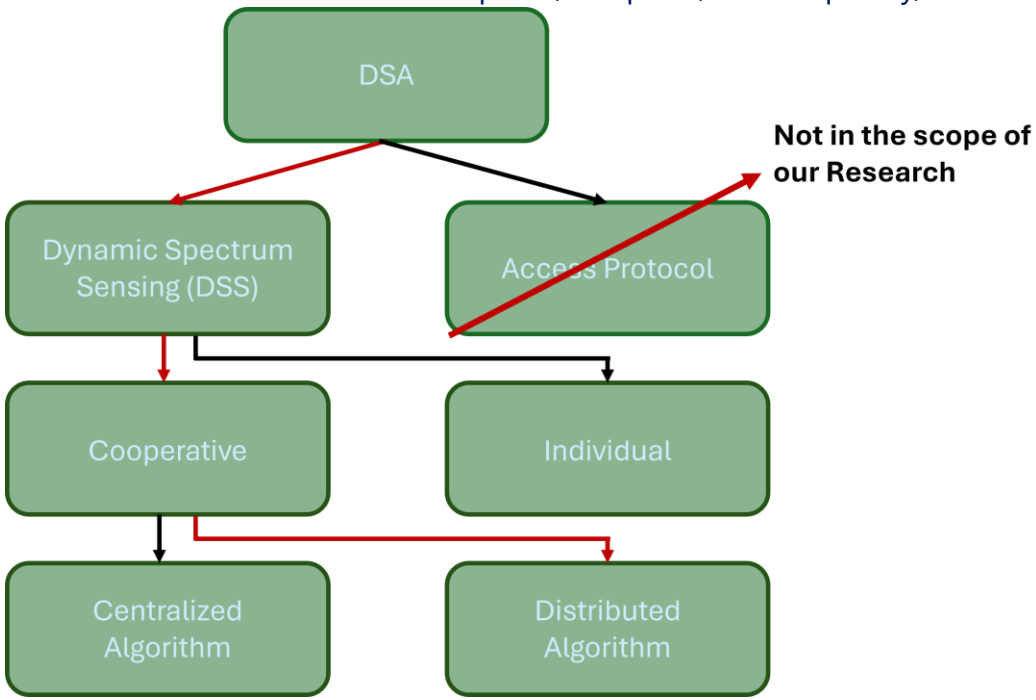
Solution

Through offline simulations and experiments with the distributed algorithm from DSS, we aim to develop a testing system that allows researchers to utilize the documentation and framework we have established.

Dynamic Spectrum Access (DSA)

What is DSA, Why is it important?

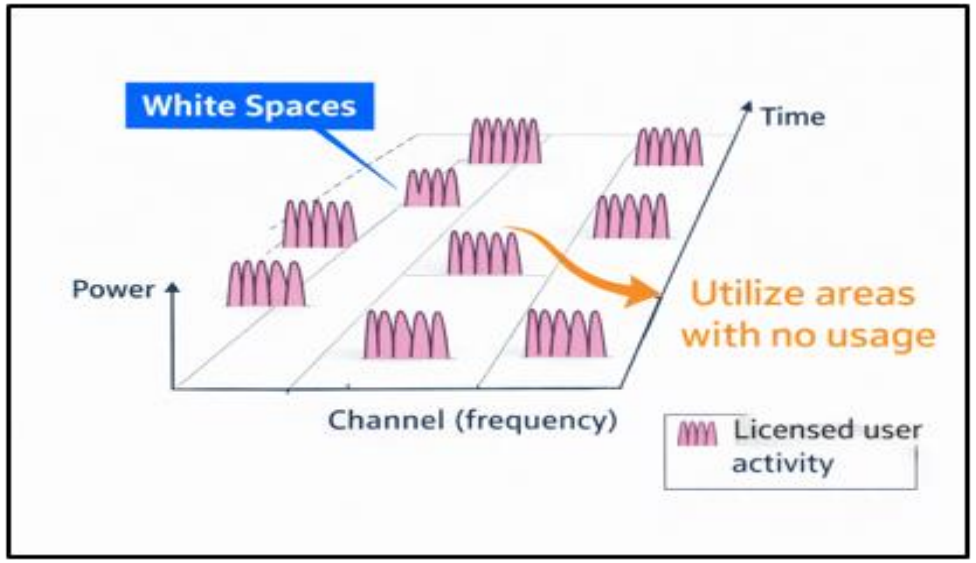
DSA is a framework that enables a secondary user (SU) to actively detect and access any vacant channels while avoiding occupied ones. This capability allows for more efficient utilization of the frequency spectrum. The system can be understood across three dimensions: spatial, temporal, and frequency, all of which are dynamic in our environment.



Time, Frequency, and Space Dimensions

How the Time and Frequency Dimensions affect our System

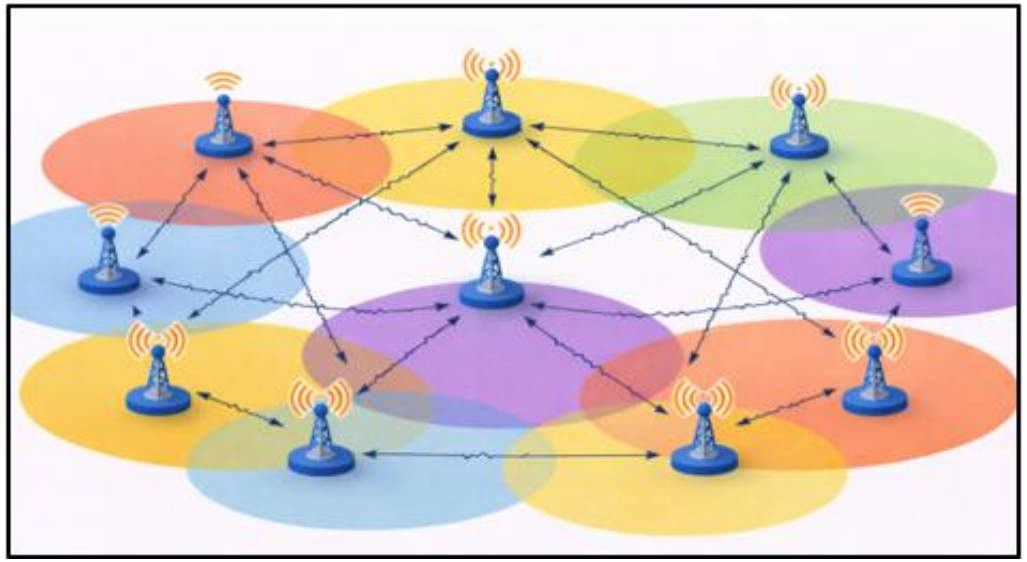
The frequency dimension matters because different channels have different activity levels. The time dimension matters because spectrum conditions change as users, interference, or jamming appear and disappear within a channel. Because of this, the system must keep sensing and quickly adapt its channel decisions.



Time and Frequency

How the Spatial Dimension affects our System

Because nodes are in different locations, they experience different signal conditions. Distance, obstacles, and interference can cause one node to detect activity while another misses it. Spreading sensors across space gives the network better coverage and more reliable channel decisions.



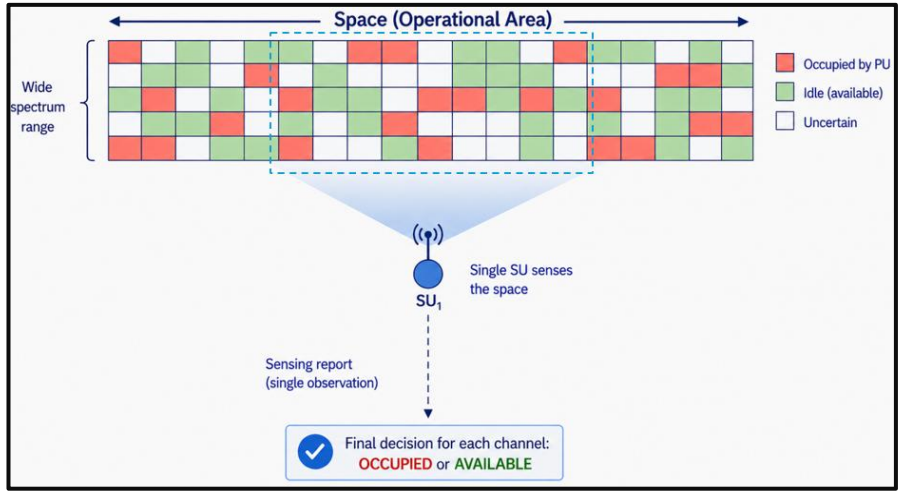
Spatial

Connection between Spectrum Sensing and Dynamic Spectrum Access

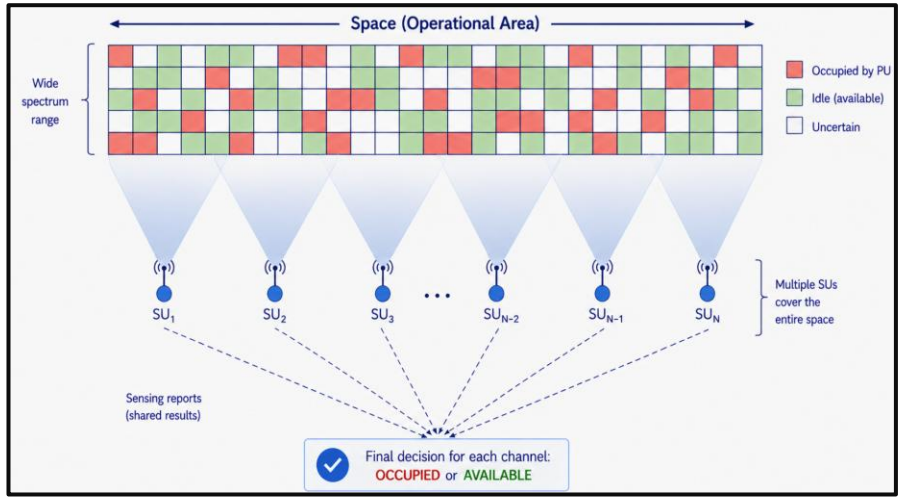
Why does spectrum sensing matter for DSA?

Spectrum sensing provides the awareness needed for DSA.

- An agent (SU) observes a local part of the operational area across a spectrum range, providing a local statistical value, x_i, (signal strength) when a PU occupies a channel.



Individual Spectrum Sensing



Cooperative Spectrum Sensing

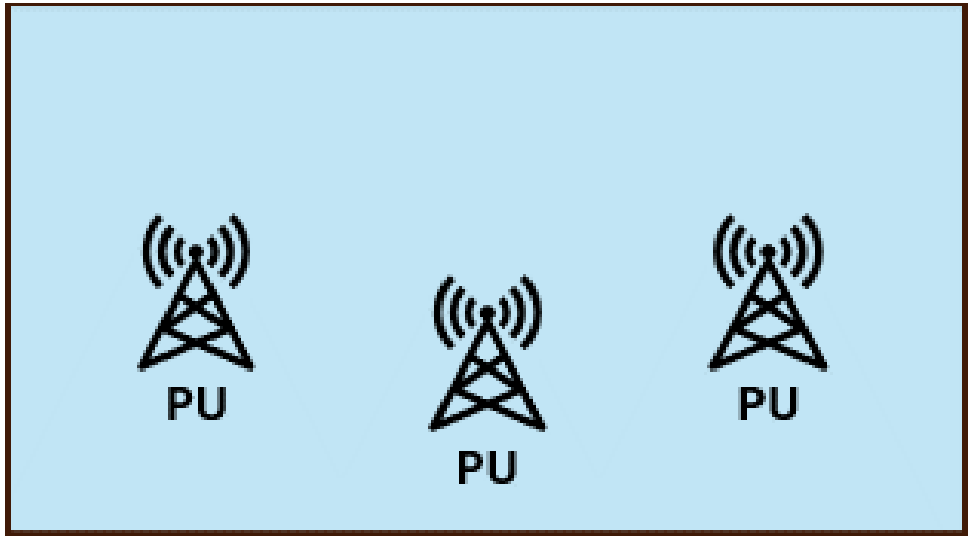
Spectrum Sensing

What is Spectrum Sensing?

Spectrum sensing involves observing the spectral environment to detect the presence or absence of a Primary User (PU) on specific frequency channels in a frequency bandwidth.

Individual

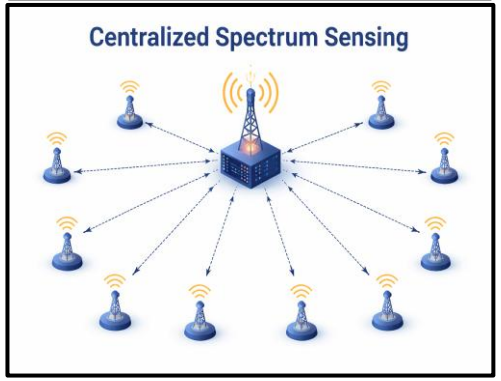
- Individual sensing involves a single agent observing the spectral environment and deciding an action.
- Small coverage of the spectral environment



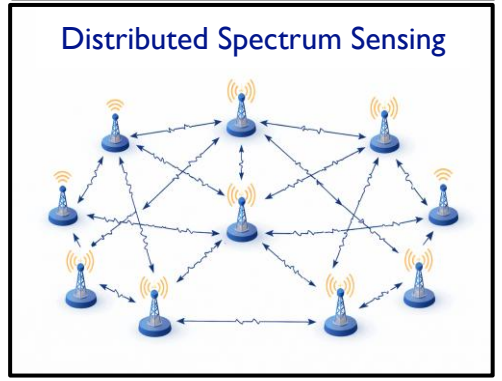
Cooperative

- Cooperative sensing involves multiple agents observing the spectral environment and collecting information to share in the communication environment to reach a global consensus
- Large coverage of the spectral environment

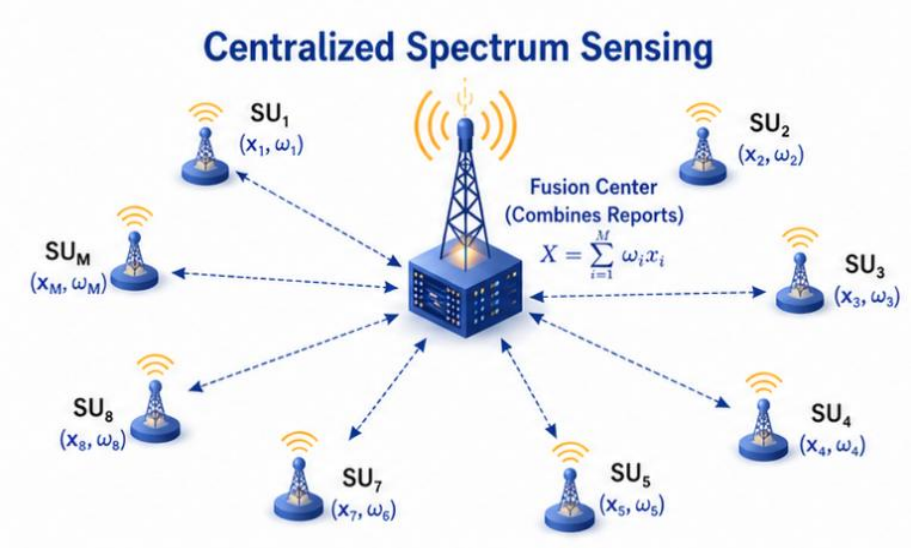
Centralized



Distributed



Cooperative Spectrum Sensing Architectures



In a **Centralized Spectrum Sensing** (CSS), each node gathers a local statistical value about the spectral environment and transmits that information to a fusion center, which aggregates the data and makes a unified decision (consensus).

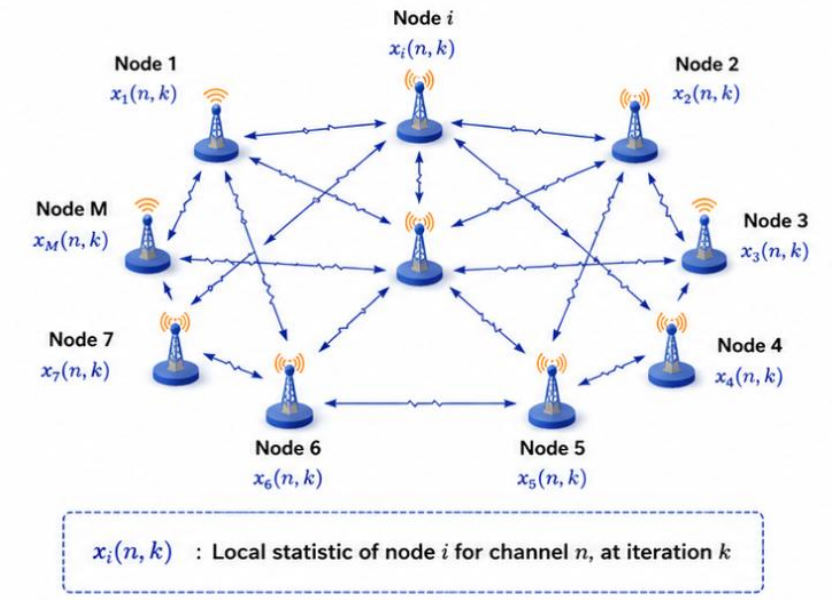
$$X = \sum_{i=1}^{M} \omega_i x_i$$

x_i = Statistical Value of an agent
w_i = Node Specific Weight

Pros
- Low complexity
- Faster Consensus Time

Cons
- Single Failure Point at fusion center

Distributed Spectrum Sensing



Cooperative Spectrum Sensing Architectures

In a **Distributed Spectrum Sensing** (DSS) each node gathers data about the spectral environment and share that information to their neighbors and iteratively updates their decision through local communication without relying on a central fusion center.

$$x_i\left(n,k+1\right)=x_i\left(n,k\right)+\frac{\alpha}{\delta_i}\sum_{j\in N_i(k)}\left(x_j\left(n,k\right)-x_i\left(n,k\right)\right)$$

$$\frac{\alpha}{\delta_i}=\frac{\text{global weight}}{individual\ weight}=\text{weighting conditions}$$

$$\sum_{j\in N_i(k)}\left(x_j(n,k)-x_i(n,k)\right)=\text{Neighborhood Difference}$$

Pros

- Network Security/Resilience
- Strong Network Infrastructure
- Scalability

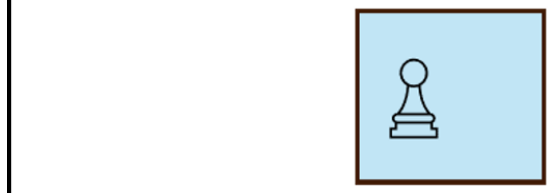
Cons

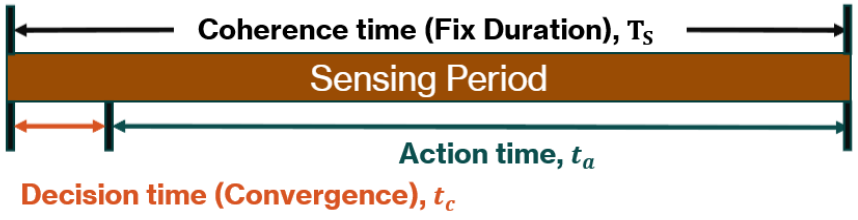
- Complex
- Slow Global consensus time

Timing Characteristics

Centralized Spectrum Sensing

- Less decision time (Fusion Center)
- More action time

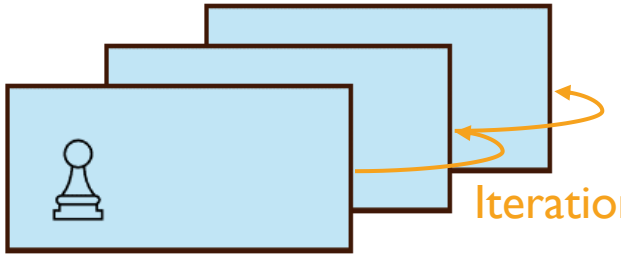


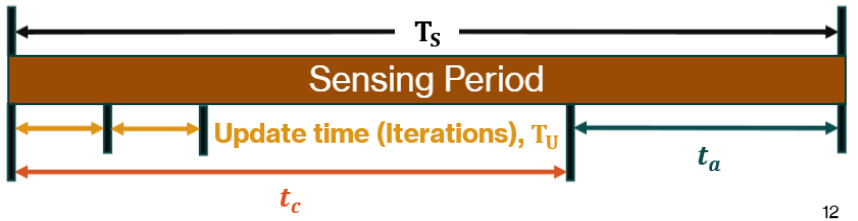


Distributed Spectrum Sensing

Slow Timing:

- More decision time (Multiple Iterations and nodes to pass through)
- Less action time





12

DSS sensing period must be:

- short enough to allow meaningful channel usage
- long enough to ensure accurate detection within the channel coherence time

Distributed Algorithm Model

Primary Users (PU): located in the Spectral Environment.

Spectral Environment: PU communicate with Agents, and SUs observe the power transmitted by the PUs.

Agents (SU): Measure the environment and come up with their own statistic of band occupancy at an iteration: $x_i(n, k+1)$

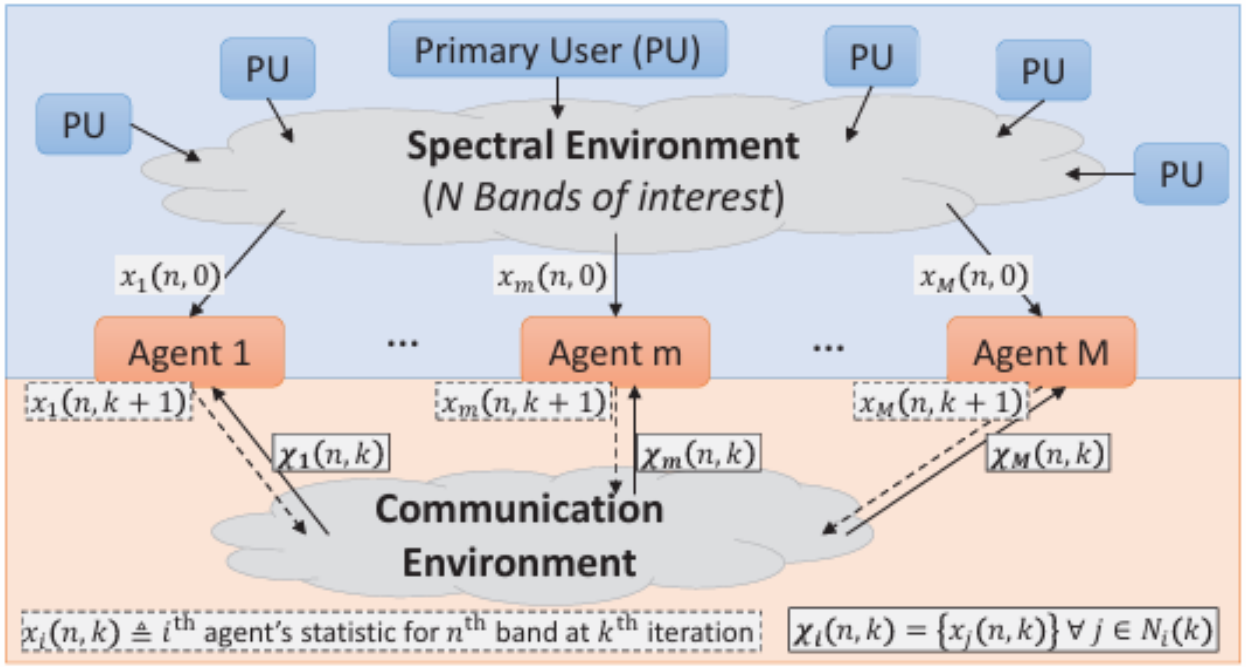
Communication Environment: Agents communicate and share their statics with their neighbors over a secured network: $(x_j(n,k) - x_i(n,k))$

Agents **receive the shared statics** and refine their local estimates: $x_i(n, k+1) = x_i(n,k) + (\text{Weight})(\text{Neighbors Contribution})$

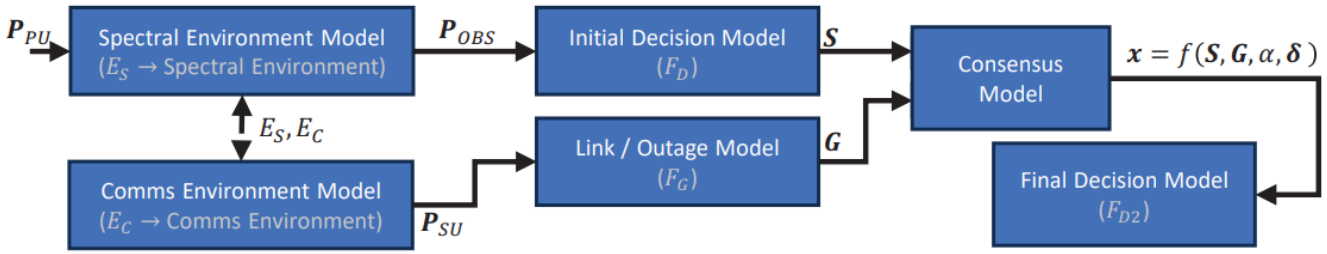
Agents rerun the same process but with the new local estimate until all agents reach a **consensus**.

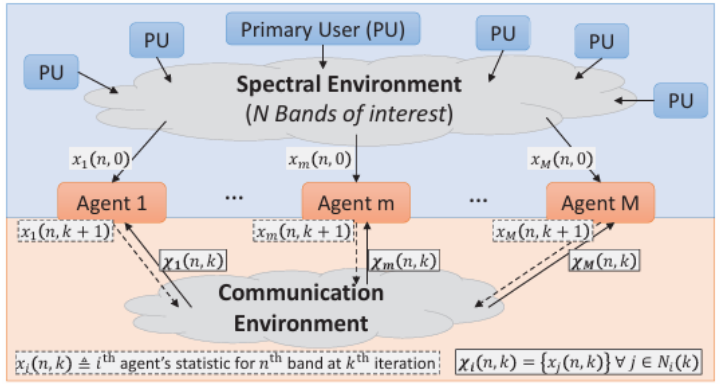
Number of channel n is extra variable so it can be ignored

$$x_i\,(n, k+1) = x_i\,(n,k) + \frac{\alpha}{\delta_i} \sum_{j \in N_i(k)} \left(x_j\,(n,k) - x_i\,(n,k)\right)$$



Model Framework Overview

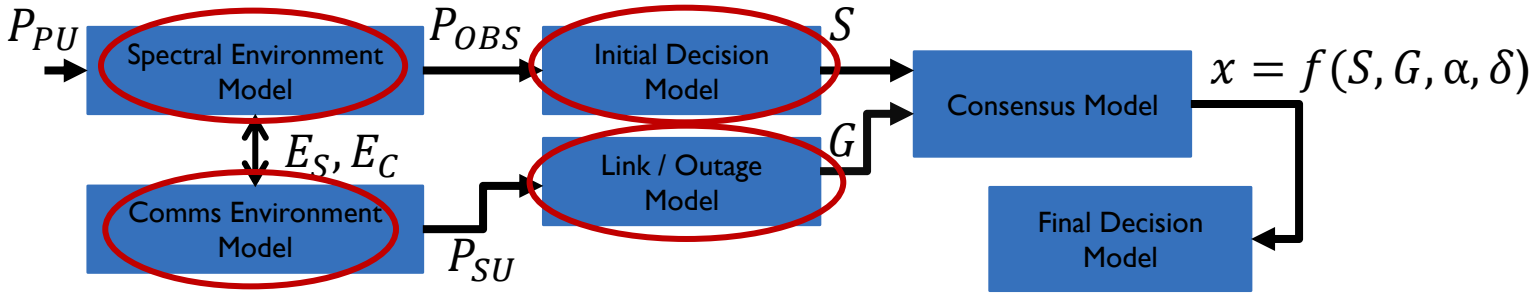




- **Spectral Environment Model**
  - Handles how primary user transmit power turns into observed power at the secondary users.
- **Communications Environment Model**
  - Handles how well the secondary users can communicate with each other over a controlled channel.
- **Initial Decision Model**
  - Takes the observed sensing data and turns it into each node's initial confidence [0,1].
- **Link / Outage Model**
  - Uses the communications results to determine which nodes are connected at each iteration {0,1}.
- **Consensus Model**
  - Takes the initial node statistics and the connectivity graph and updates each nodes statistic over iterations.
- **Final Decision Model**
  - Uses the consensus statistic at each node to decide on channel activity.

PS: Model Framework can be modeled in different ways

Distributed Spectral Sensing Model Framework



Input: Primary User Power (Secondary User (M x M x K)

Same Environment when Input Power Observed (M x M)

$$P_{PU} = \begin{bmatrix} P_{PU(1,1)} & P_{PU} \\ P_{PU(2,1)} & P_{PU} \\ P_{PU(N_{PU},1)} & P_{PU} \end{bmatrix} \quad P_{SU} = \begin{bmatrix} P_{SU(1,1)} & P_{SU(1,2)} & P_{SU(1,M)} \\ P_{SU(2,1)} & P_{SU(2,2)} & P_{SU(2,3)} \\ P_{SU(M,1)} & P_{SU(3,2)} & P_{SU(M,M)} \end{bmatrix}$$

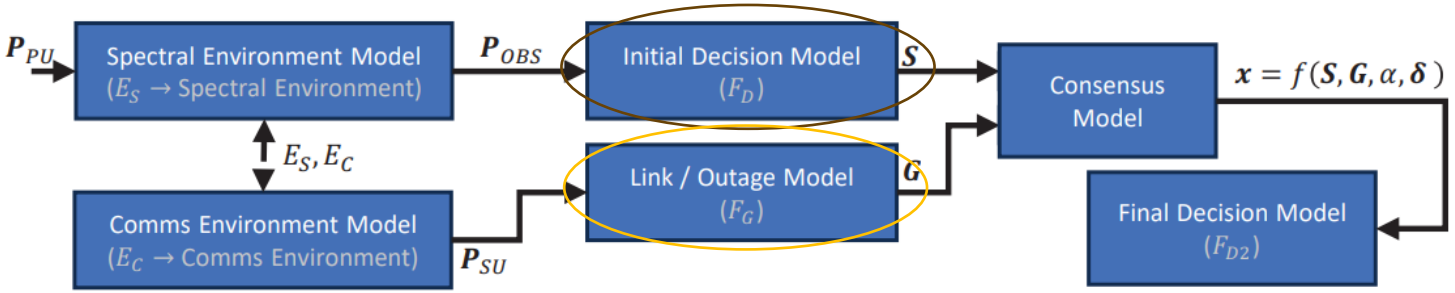
$$P_{SU} = \begin{bmatrix} P_{SU(1,1)} & P_{SU} \\ P_{SU(2,1)} & P_{SU} \\ P_{SU(M,1)} & P_{SU} \end{bmatrix}$$

Output: Power Observed (M x M x K)

$$P_{OBS} = \begin{bmatrix} P_{OBS(1,1)} & P_{OB} \\ P_{OBS(2,1)} & P_{OB} \\ P_{OBS(M,1)} & P_{OB} \end{bmatrix} \quad G = \begin{bmatrix} G_{(1,1)} & G_{(1,2)} & G_{(1,M)} \\ G_{(2,1)} & G_{(2,2)} & G_{(2,3)} \\ G_{(M,1)} & G_{(3,2)} & G_{(M,M)} \end{bmatrix}$$

$N_{PU} = \#\ of\ PU\ (Primary\ User)$
$M = \#$ of SU (Secondary User)
$N = Channels$
$K = Iterations$

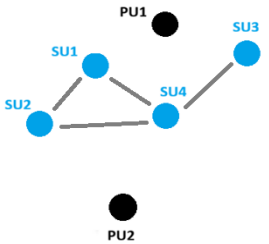
Data Structure



Example:

$$S = \overbrace{\begin{bmatrix} 0.95 & 0.70 \\ 0.72 & 0.82 \\ 0.90 & 0.51 \\ 0.89 & 0.86 \end{bmatrix}}^{\text{N}} \Big\}\text{M}$$

$$G = \overbrace{\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}}^{\text{M}} \Big\}\text{M}$$



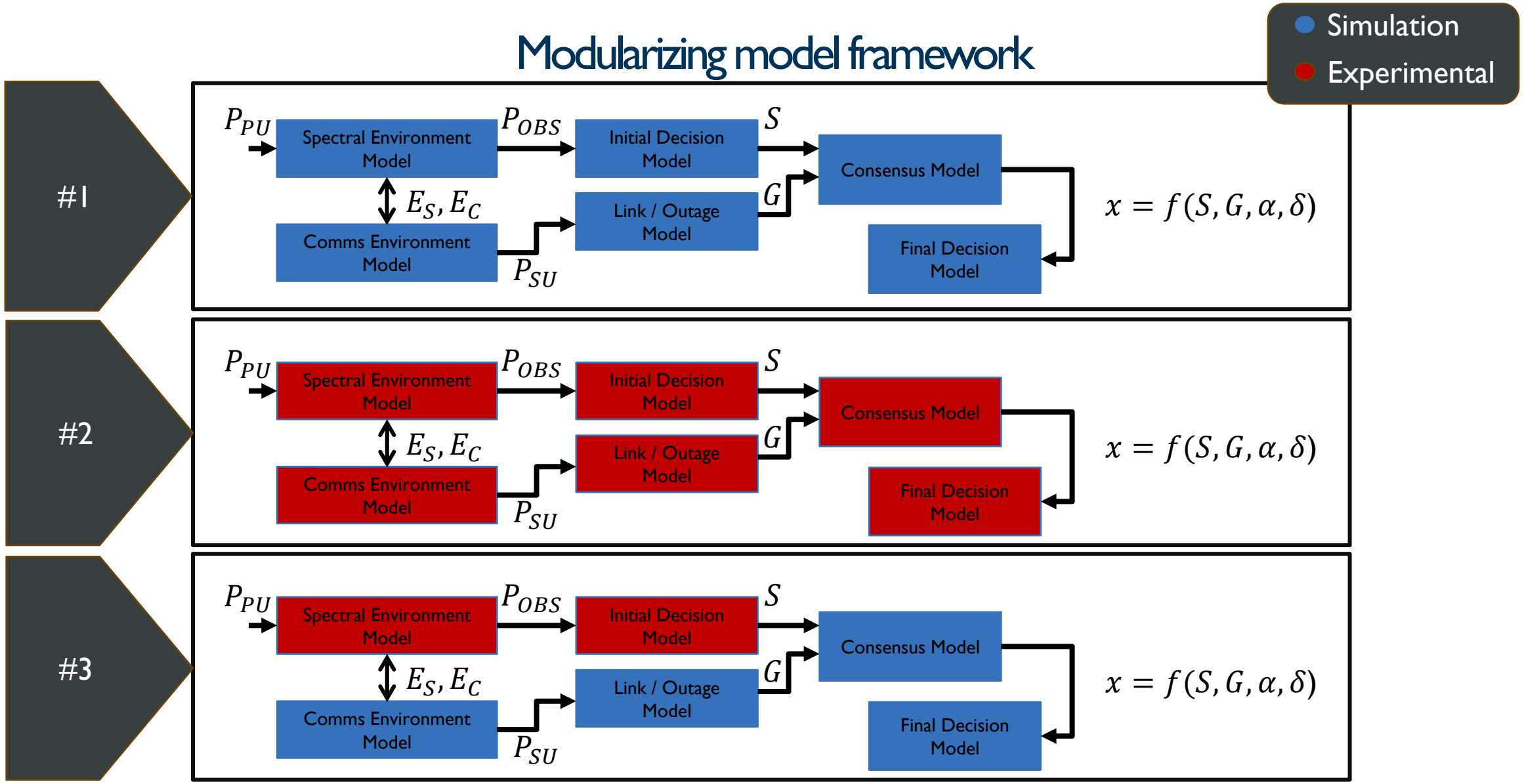
$$Example\ calculation\ for\ SU1 \rightarrow x_1(k=1) = \frac{S_{1:} + S_{2:} + S_{4:}}{3} = \frac{[0.95 \quad 0.70] + [0.72 \quad 0.82] + [0.89 \quad 0.86]}{3}$$

$$= [0.8533 \quad 0.8000]$$

$$x(1) = \begin{bmatrix} 0.8533 & 0.8000 \\ 0.8533 & 0.8000 \\ 0.895 & 0.685 \\ 0.865 & 0.7275 \end{bmatrix}$$

Reruns the same process with next iteration of G and previous iteration of x to reach convergence if possible

Modularizing model framework



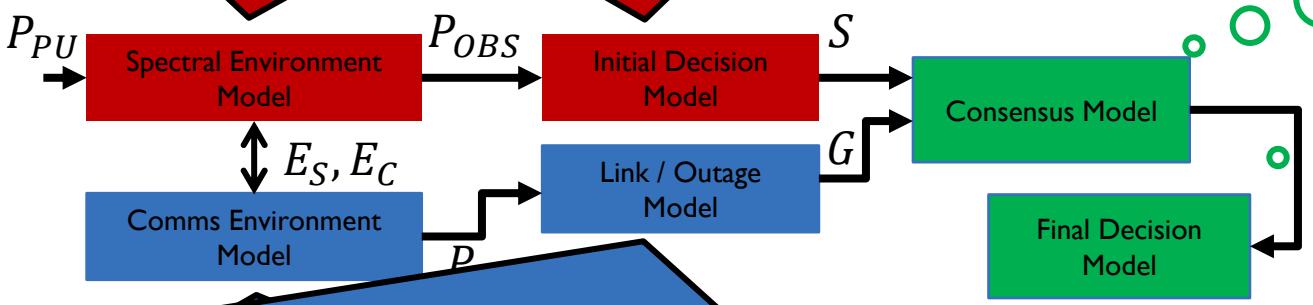
Modularizing model framework

Experimental

Small scale scenario at UMB using SDR and measuring power of TX from RX

Use Obtained data to find the best weighing scale to limit Amp to [1,0]

Can be defined however not the research focus area



$$x = f(S, G, \alpha, \delta)$$

Simulation

Bernoulli (Probabilistic Model)

Signal strength simulated → Sigmoid function

$$p(x) = \frac{1}{1 + e^{-(\frac{ss - threshold}{\sigma})}}$$

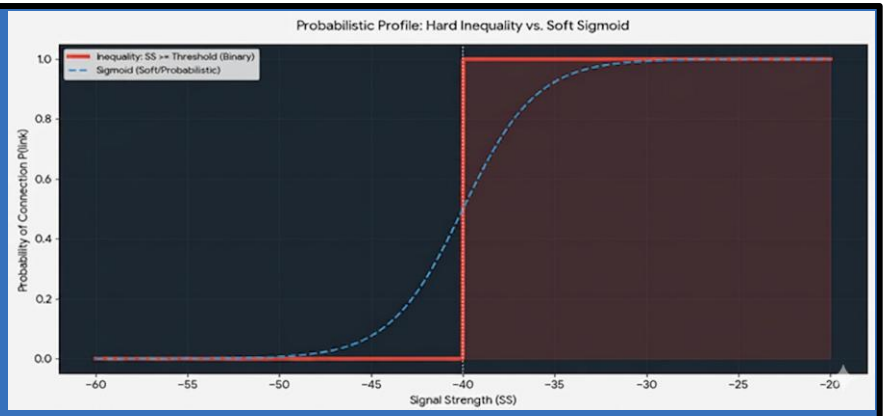
Sigmoid centers at threshold
- ss = threshold (50%)
- ss > threshold (higher probability)
- ss < threshold (lower probability

Sigma
- Increasing σ → Softens threshold (Gradual Increase)
- Decreasing σ → Hardens threshold (Steep Increase)

Binary (Hard threshold Model)

Signal strength (SS) inequality

Connectivity = SS ≥ Threshold



Experimental & Simulation System

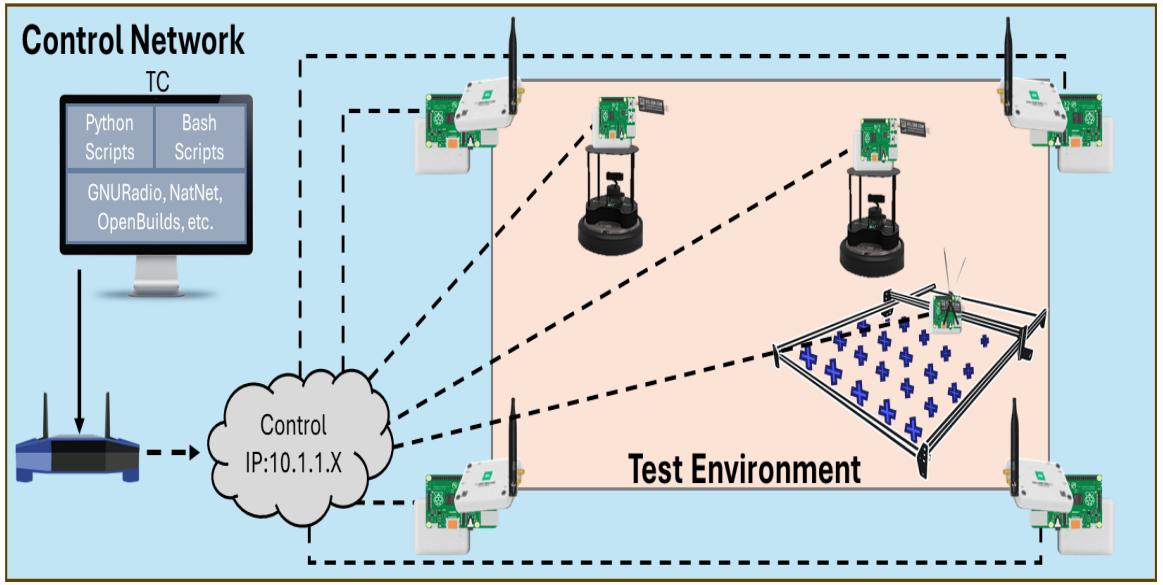
Experimental

Pros

- Understand how the system works with real life effects

Cons

- Requires more time and effort
- Scalability limitations



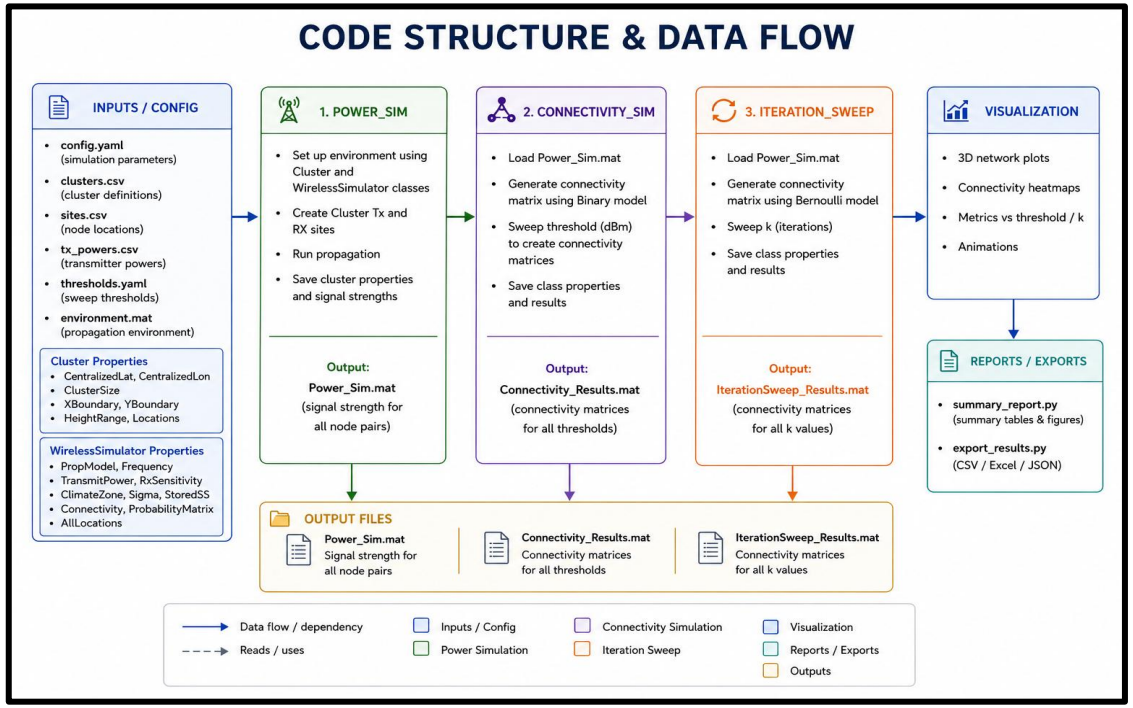
Simulation

Pros

- Scenarios isolation to learn characteristics
- Easier repeatability
- Scalable

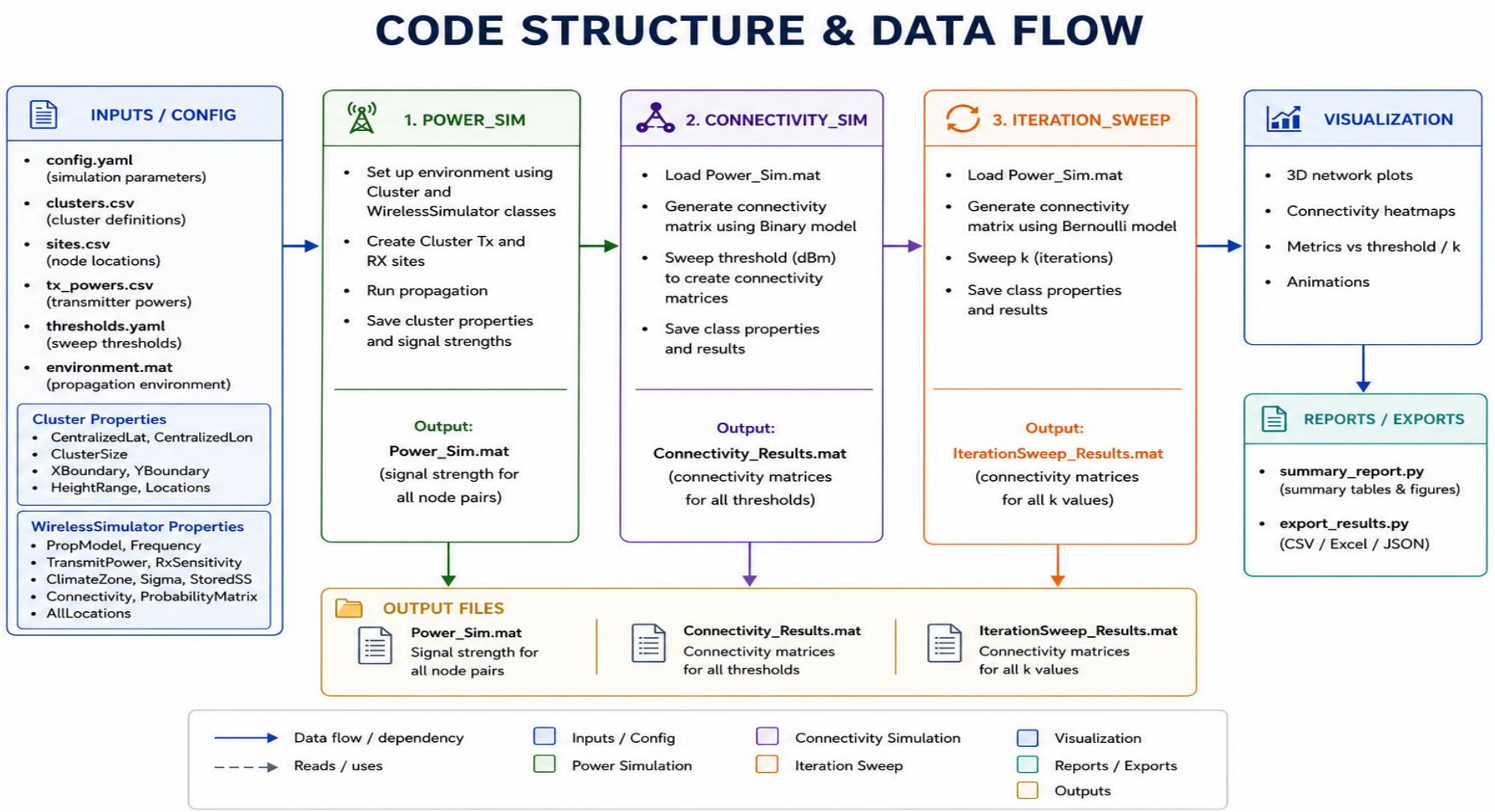
Cons

- Cannot emulate real life effects in scenarios
- Results depend heavily on how accurate the model is



SIMULATION: CODE STRUCTURE

Offline Simulation:



SIMULATION INSTANCE SCENARIO

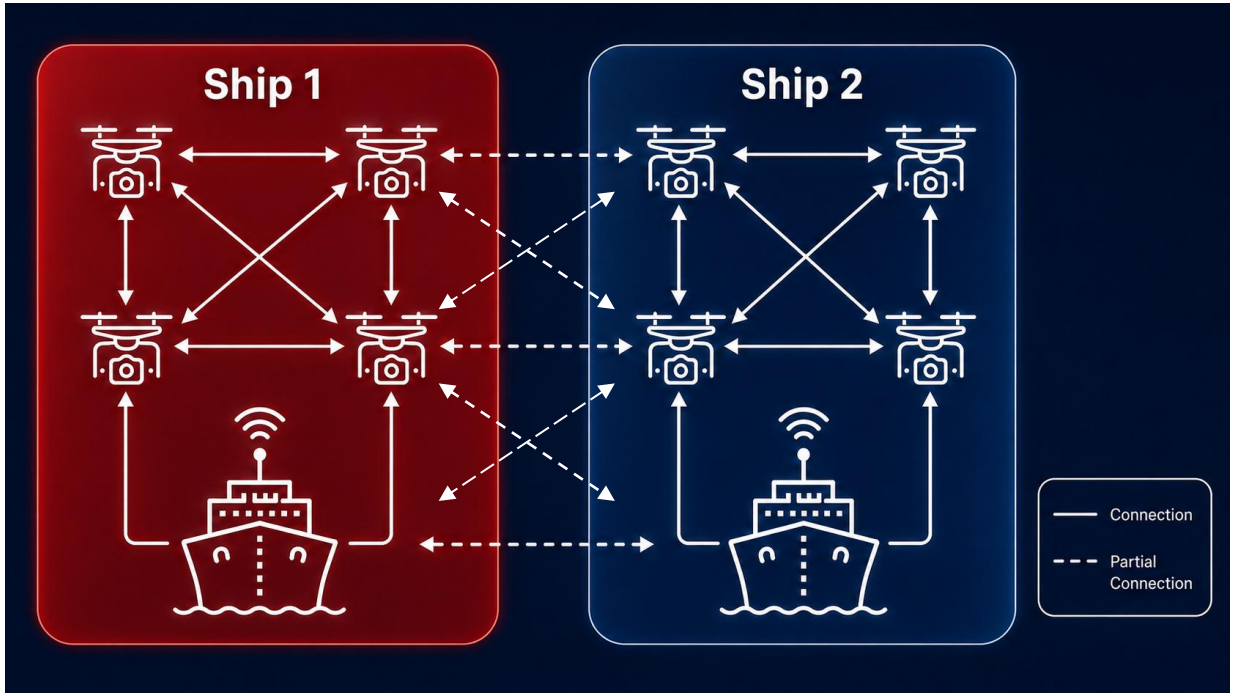
Maritime clustering scenario

- 2 Ships with a cluster of nodes at the ships
- 2 overhead drone clusters above the ships

What are important characteristics that establishes link connectivity between node to node?

Hypothesis

- Distance of the nodes within the cluster and cluster to cluster
- Power threshold



SCENARIO 1: FAR SHIP DISTANCE, SMALL OVERHEAD SPREAD

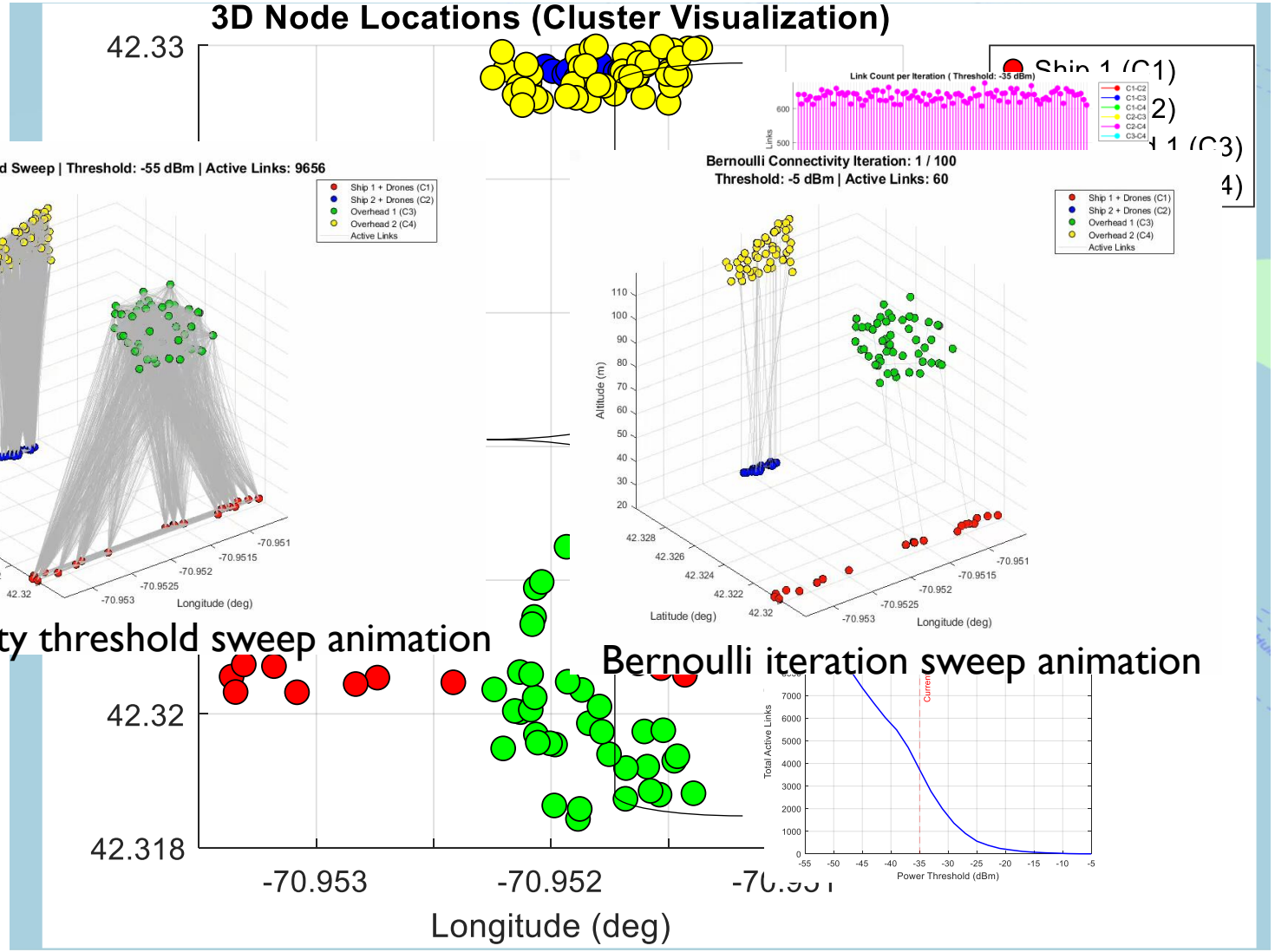
Parameters:

- Number of nodes per cluster:
  - C1, C2 = 20, 20 ← Ships
  - C3, C4 = 50, 50 ← Drones
- Height range per cluster:
  - Low_alt = [20, 20] ← Ships (e.g, ground)
  - High_alt = [100, 120] ← Drones
- Distance between ships:
  - d1 = 1000m

[OPTIONAL: boundary for rectangle/square clusters]

- C1 = 350m x 50m
- C2 = 50m x 50m
- C3 = 100m x 500m
- C4 = 100m x 100m

Simulation



SCENARIO 2: FAR SHIP DISTANCE, LARGE HEMISPHERICAL OVERHEAD SPREAD

Parameters:

- Number of nodes per cluster:
  - C1, C2 = 20, 20 ⟵ Ships
  - C3, C4 = 100, 100 ⟵ Drones
- Height range per cluster:
  - Low_alt = [20, 20] ⟵ Ships (e.g, ground)
  - High_alt = [20, 120] ⟵ Drones
- Distance between ships:
  - d1 = 1000m

[OPTIONAL: boundary for rectangle/square clusters]

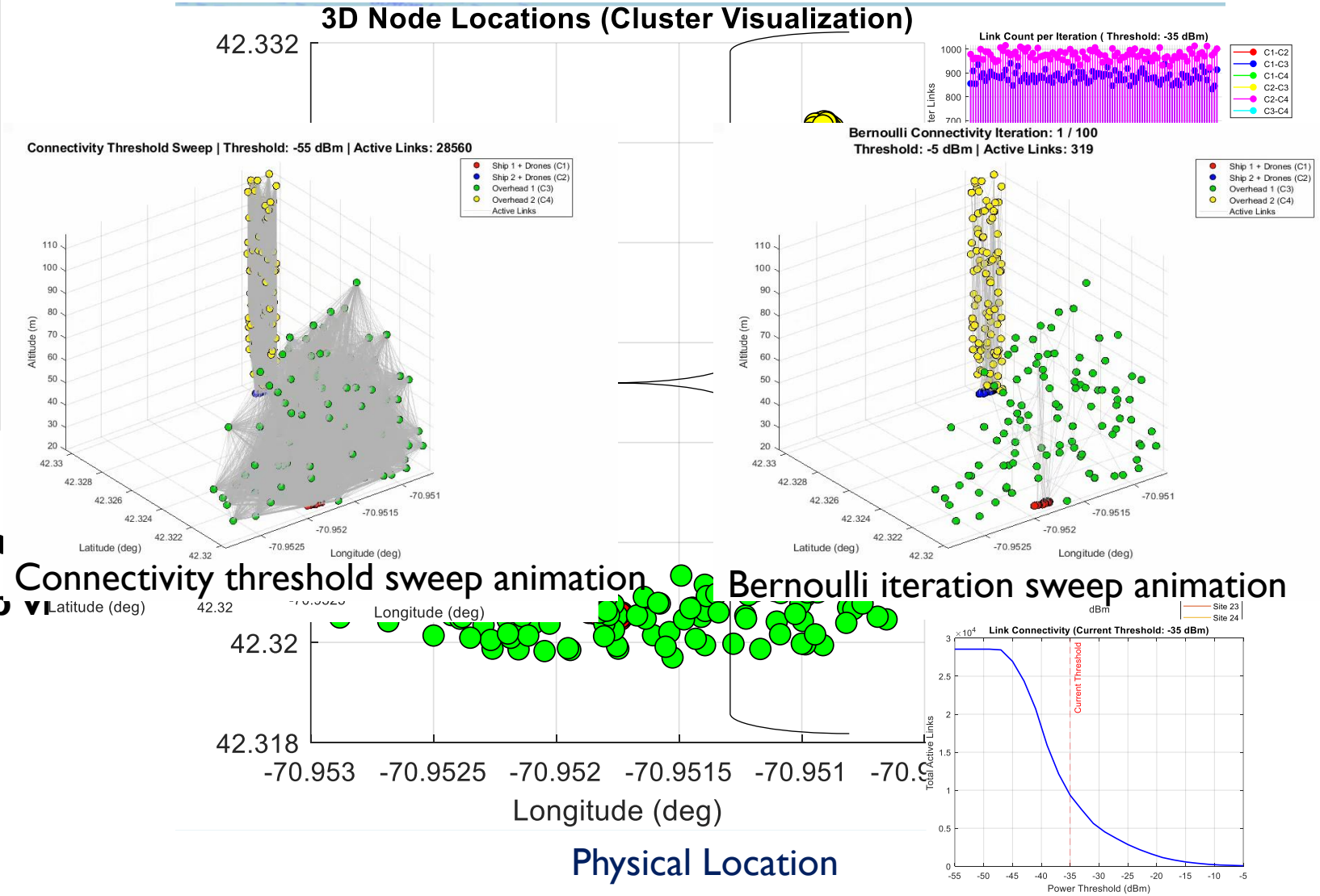
- C1 = 50m x 50m
- C2 = 50m x 50m

} Fixed square cluster

[OPTIONAL: boundary for cross-sectional/hemisphere clusters]

- Radius C3, C4 = High_alts(2) – High_alts(1) = 100

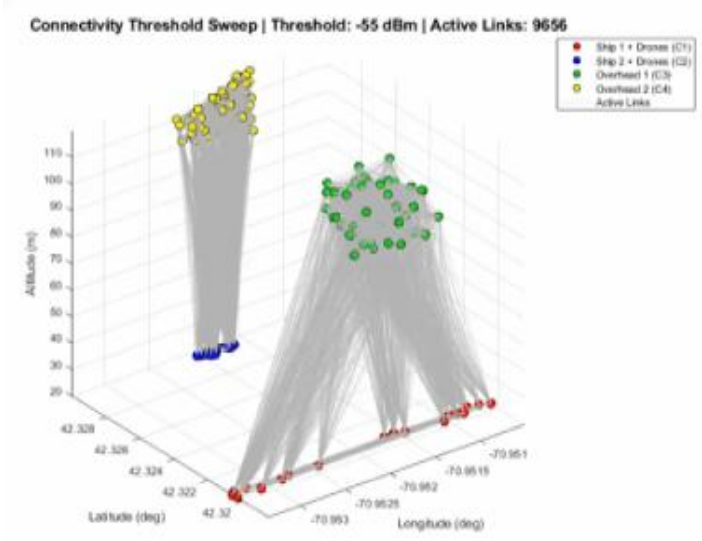
Simulation



COMPARATIVE ANALYSIS OF SCENARIOS

Scenario Highlight #1

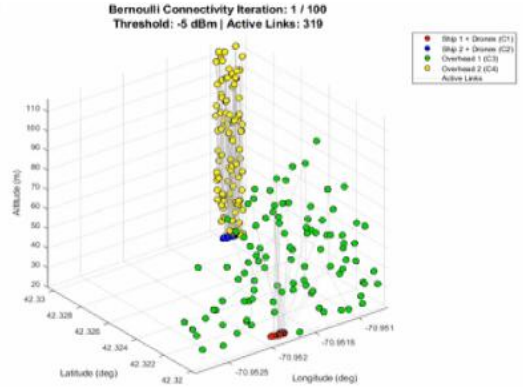
For a binary model, the large ship-to-ship distance meant no connections from ship 1 cluster + overhead to ship 2 cluster+ overhead.



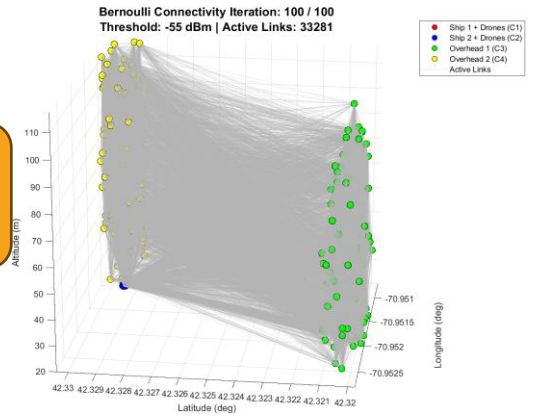
Scenario 1

Scenario Highlight #2

For a Bernoulli model, the large ship-to-ship distance had a lot of connections with a lower power threshold (at -55 dBm), but no connections with a higher power threshold (at -5 dBm)



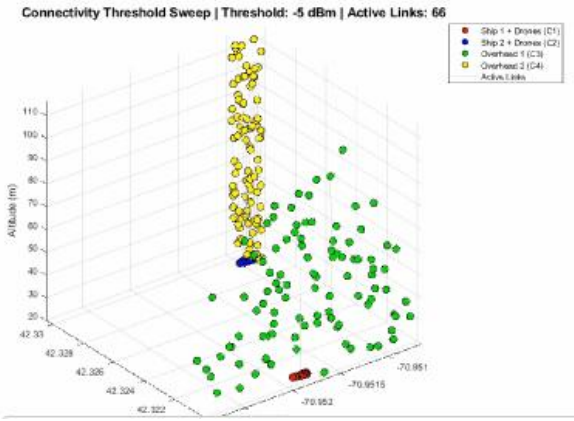
Scenario 2



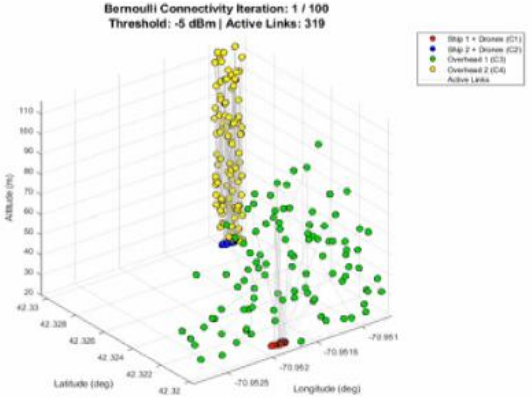
Softer connectivity

Scenario Highlight #3

Both models have a hard time establishing a connection even within clusters (relatively close node distance) at higher power thresholds. Also easier for Bernoulli to establish connections



Scenario 2



COMPARATIVE ANALYSIS OF SCENARIOS

Scenario Highlight #1

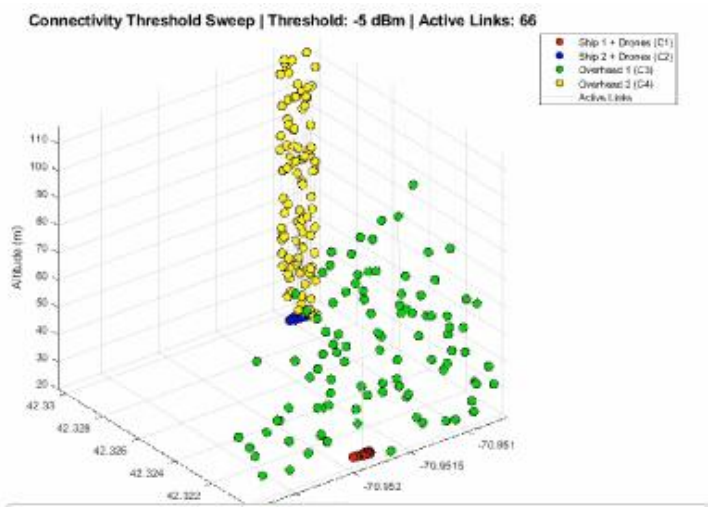
For a binary model, the large ship-to-ship distance meant no connections from ship 1 cluster + overhead to ship 2 cluster+ overhead.

Scenario Highlight #2

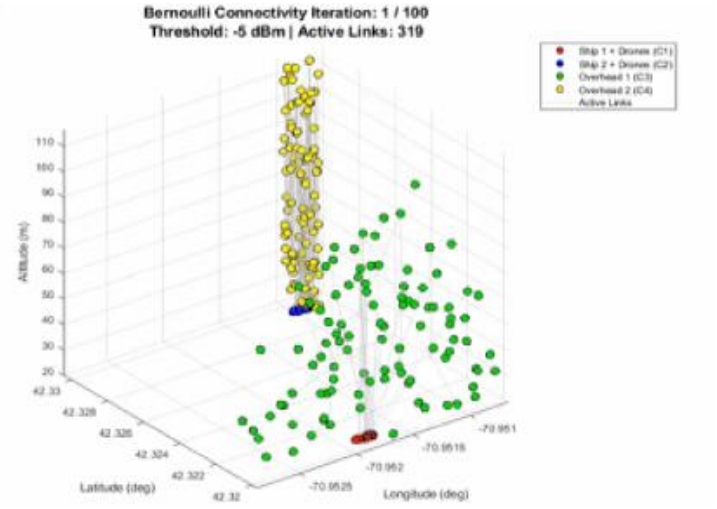
For a Bernoulli model, the large ship-to-ship distance had a lot of connections with a lower power threshold (at -55 dBm), but no connections with a higher power threshold (at -5 dBm)

Scenario Highlight #3

Both models have a hard time establishing a connection even within clusters (relatively close node distance) at higher power thresholds. Also easier for Bernoulli to establish connections



Scenario 2



Scenario 2

SYSTEM HIGHLIGHT

System parameters

Comms Environment Model

- Cluster shape (Rectangle, Square, Dome, Flat Circle)
  - Size of the shape (Boundaries in X, Y, Z)
- # of Nodes
- Positioning of the cluster (Latitude, Longitude, Height)
- Rx and Tx parameters
  - Frequency
  - Transmission Power
  - Propagation Model
  - Climate Zone
  - Receiver Sensitivity

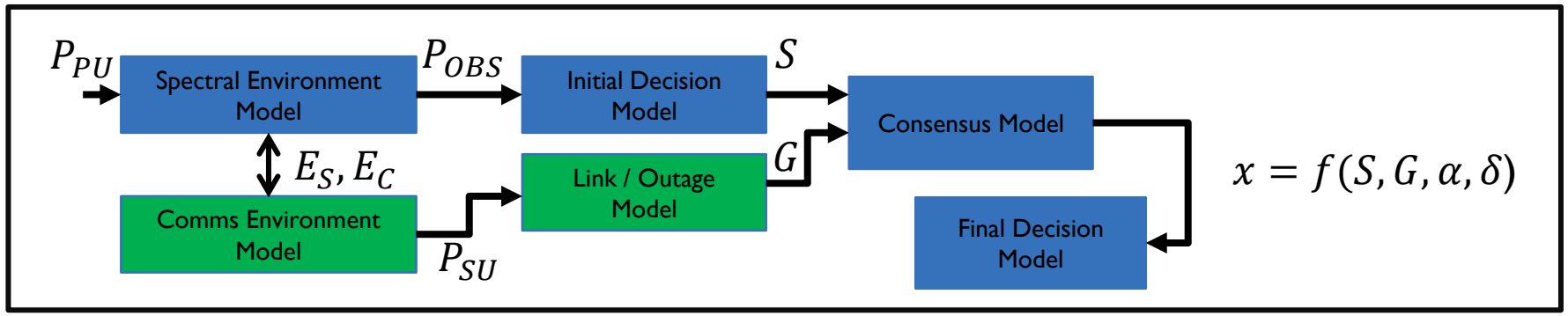
Link/Outage Model

- Link connectivity model
  - Power Threshold
    - Binary (Hard threshold)
    - Bernoulli (Probabilistic threshold)

Research Tool

- Comms environment can be manipulated to create any scenario, and different shape methods can be implemented (triangle/pyramid)
- Link/Outage Model can be manipulated to test thresholds and even implement further link/outage models other than Binary and Bernoulli

Modularized model framework



Hardware used / Experimental Setup

Raspberry Pi 4
Embedded computer used to run signal processing and control algorithms.

Wireless Access Point
Provides the connection between the Secondary Users

Portable Generator
Power for Wireless Access Point

Software Defined Radio
Receives and digitizes RF signals for spectrum sensing and analysis.

ADALM-PLUTO
Generates and transmits RF signals to emulate primary users in the experiment.

Experiments were done on the campus center lawn

Experimental setup

Testbed Controller:

- Runs GNU Radio TX flowgraph
- Controls testing variables
- Sends baseband data to ADALM-Pluto

Transmitter:

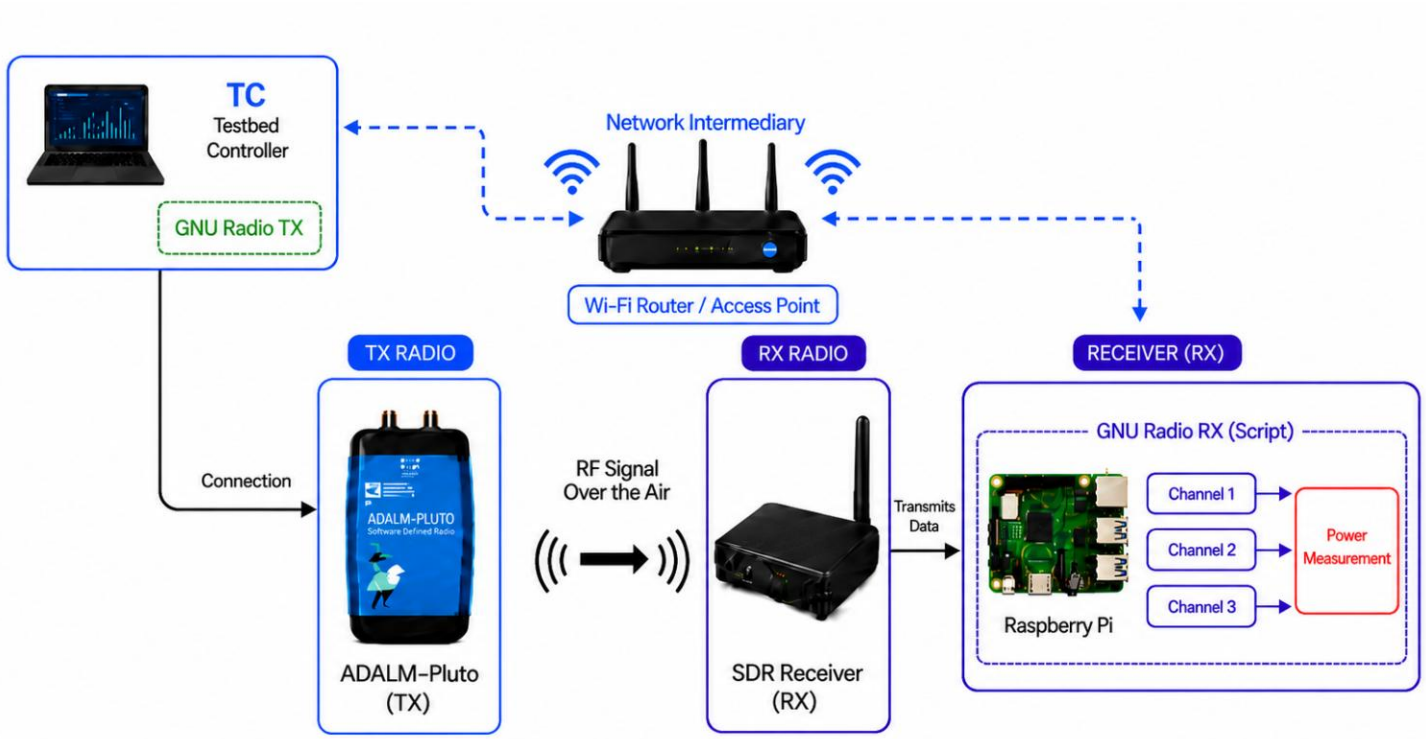
- Transmits predefined signals over the air

Receiver:

- Captures signal sent
- separates channels
- Measures received power per channel

Network Intermediary:

- Connects TC and receiver node through the same subnet



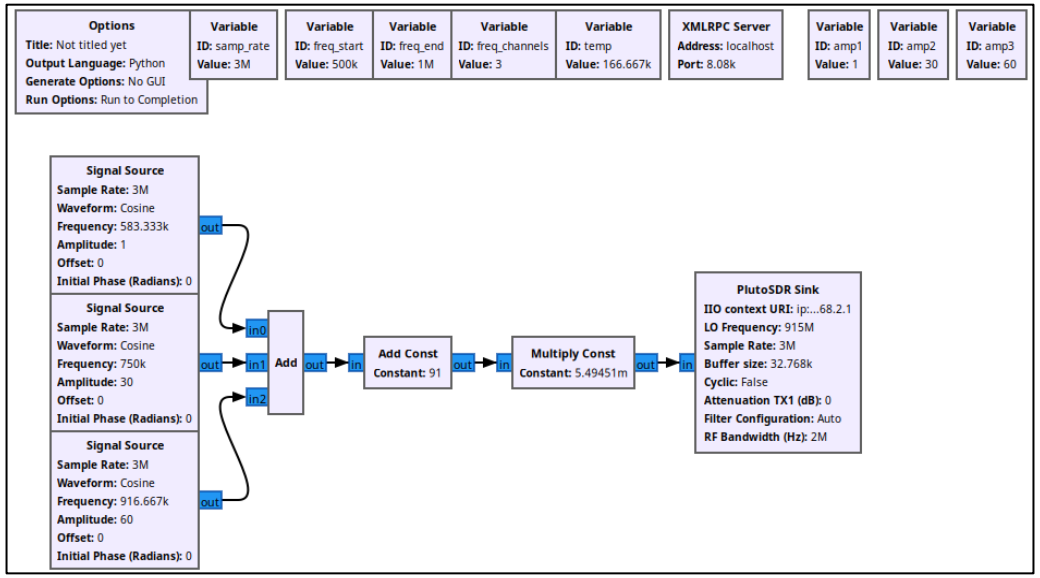
GNU Radio

GNU Radio lets us build a flexible software-defined radio system where we can generate signals, separate channels, and measure received power in real time.

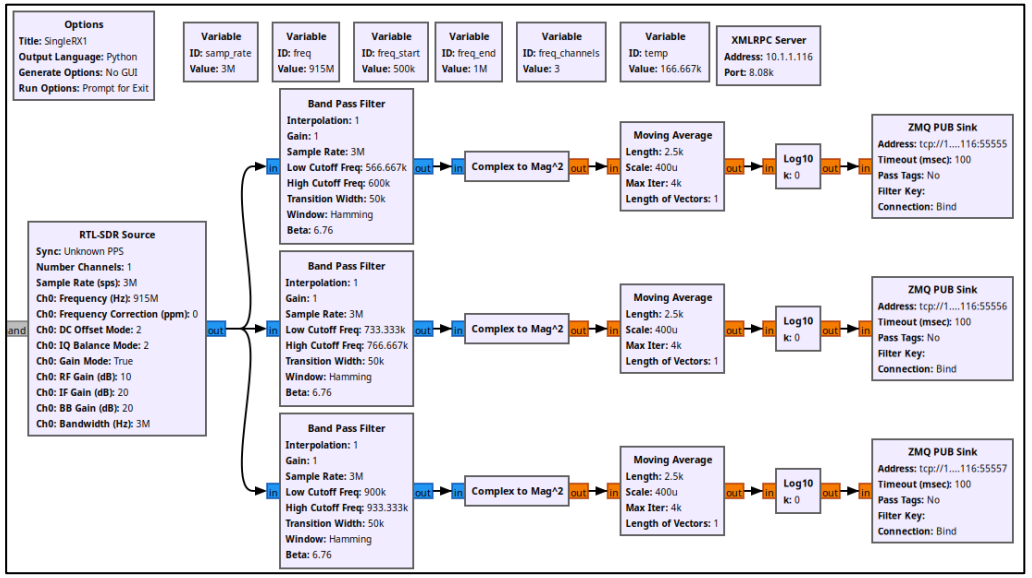
We also used XMLRPC and ZMQ

- **XMLRPC**: Allows us to control any variables passed along to GNU Radio
- **ZMQ**: Allows us to collect the power values received for further processing.

TRANSMITTER

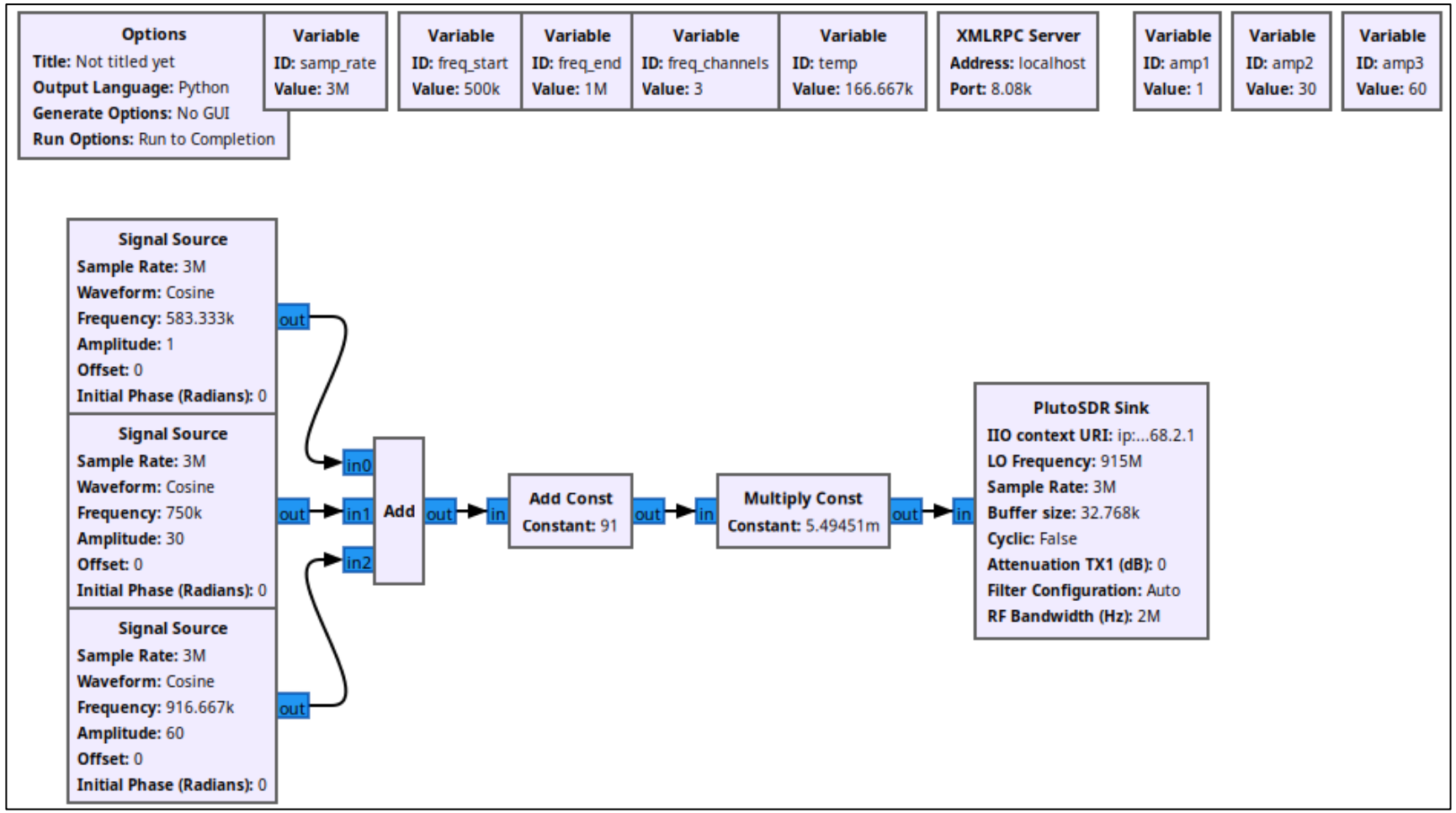


RECEIVER



GNU Radio TX

TRANSMITTER



VARIABLES

Sample Rate: 3MHz

Starting Frequency: 500kHz

Spacing Frequency: 166kHz

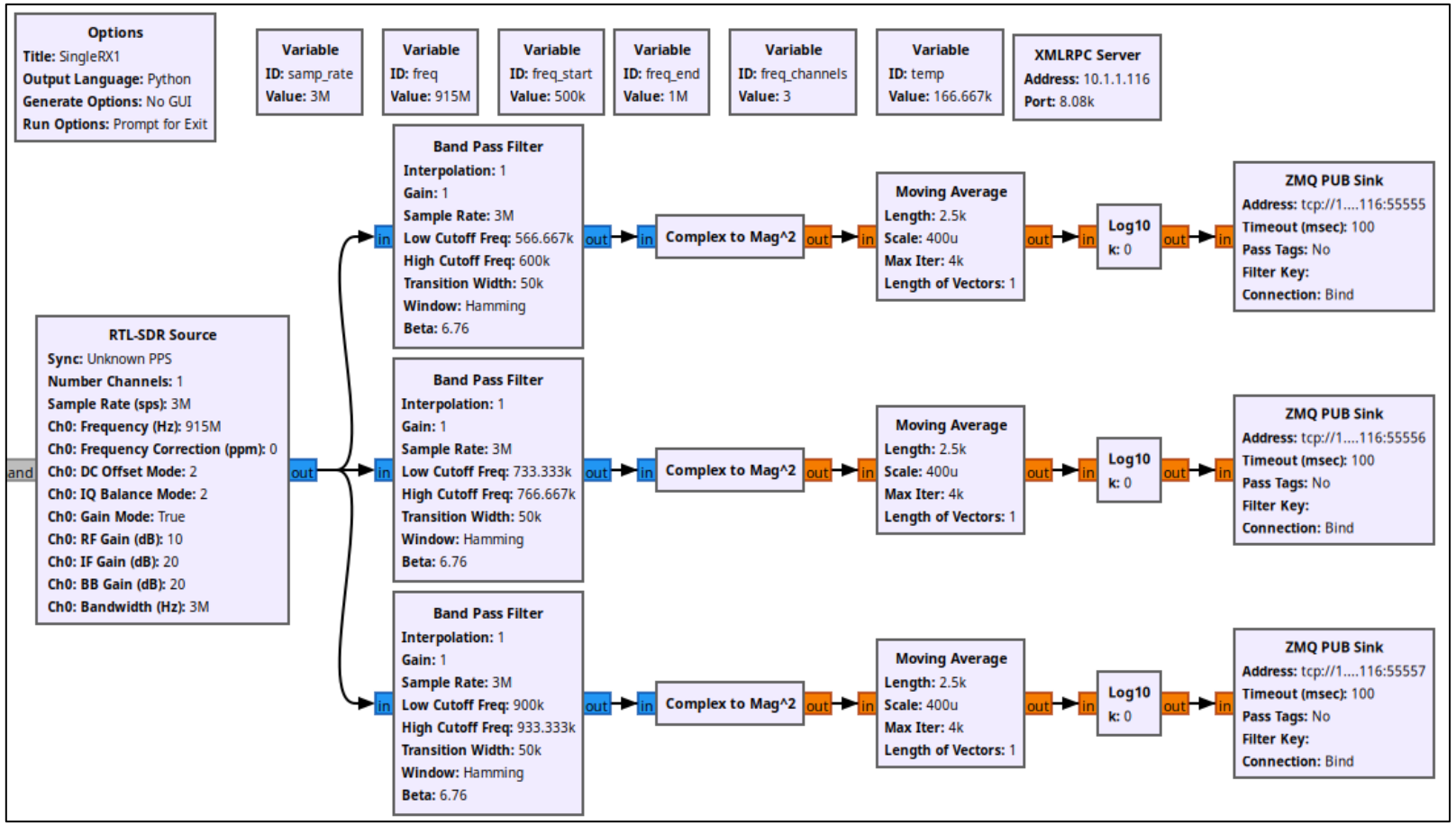
Stopping Frequency: 1MHz

Channels Amplitude: 1, 30, 60

Tx Attenuation: 0dB

GNU Radio RX

RECEIVER



VARIABLES

Sample Rate: 3MHz

Starting Frequency: 500kHz

Spacing Frequency: 166kHz

Stopping Frequency: 1MHz

Rx Gain: 50dB

Code / Scripts

Bash

```
echo "-----------------Running Remote Rx Scripts--------------------"
for PI in "${PIS[@]}"; do
    echo "Connecting to $PI..."
    ssh -tt "${USER}@${PI}" "cd ${REMOTE_DIR} && python3 ${REMOTE_CMD} -c $C >/dev/null 2>&1" &
done

echo "-----------------Running Local Tx Script--------------------"
echo "Connecting to localhost..."
python3 "${LOCAL_TX_SCRIPT}" -c $C >/dev/null 2>&1 &
echo "sleep 3"
sleep 3
echo "-----------------Running XMLRPC Script--------------------"
XMLRPC_ARGS="--pis ${PIS[@]} -f1 $F1 -f2 $F2 -c $C"
if [ "$Z_FLAG" = "Y" ]; then
    XMLRPC_ARGS="$XMLRPC_ARGS -z"
fi
python3 "${LOCAL_SCRIPT_XMLRPC}" $XMLRPC_ARGS
echo "-----------------Running ZMQ Script--------------------"
if [ "$N_FLAG" = "Y" ]; then
    python3 "${LOCAL_SCRIPT_ZMQ}" --pis "${PIS[@]}" -f1 "$F1" -f2 "$F2" -c "$C" -n
else
    python3 "${LOCAL_SCRIPT_ZMQ}" --pis "${PIS[@]}" -f1 "$F1" -f2 "$F2" -c "$C"
fi
```

What we did for our test:

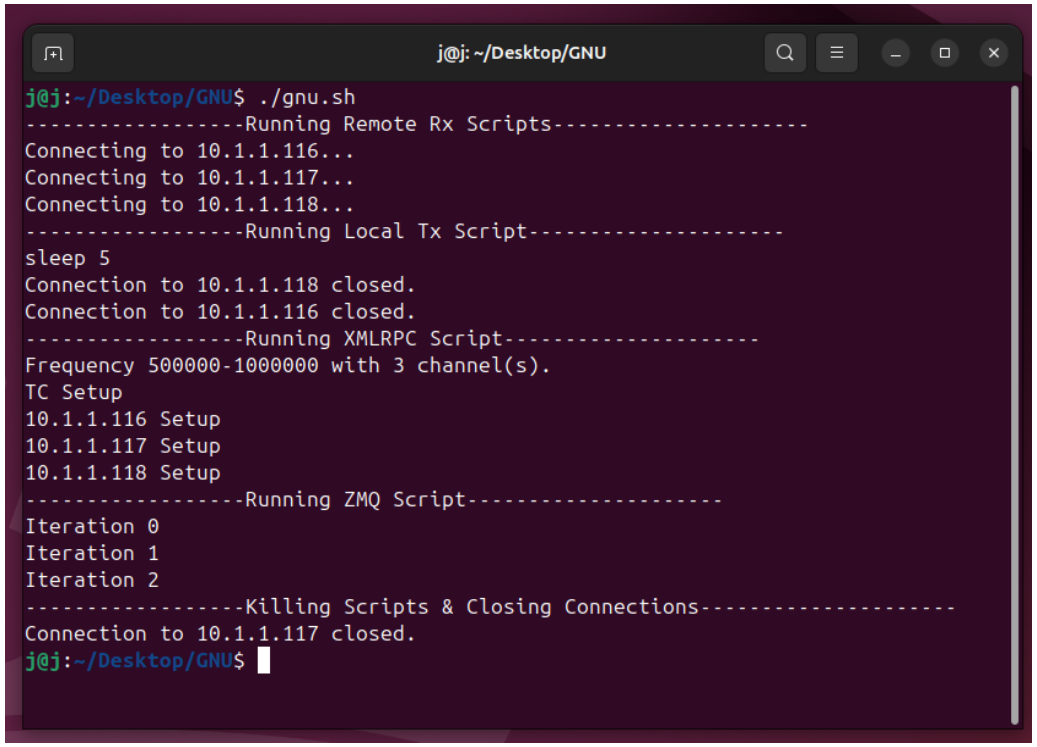
- 3 channels
  - Evenly spaced bandwidth
- 3 Iterations
  - 0.3s sleep per iteration
- 3 Addresses
  - 1.0s sleep per address
- Checks all channels -> Switches address -> Iterates
  - Outputs P_OBS into a csv file

XMLRPC Variables

```
parser.add_argument('--pis', nargs="+", help="Node Addresses")
parser.add_argument("-f1", "--freq_start", type=int, default=500000, help="Starting Frequency")
parser.add_argument("-f2", "--freq_end", type=int, default=1000000, help="Ending Frequency")
parser.add_argument("-c", "--freq_channels", type=int, default=3, help="Channel Amount")
parser.add_argument("-z", "--randomize", action="store_true", help="Randomize gains; otherwise use saved values")
```

Power Data Collection

```
-----------------Running ZMQ Script--------------------
tcp://10.1.1.116:55555 ||| Iteration 1 ||| Power: -34.57 dB
tcp://10.1.1.116:55556 ||| Iteration 1 ||| Power: -26.22 dB
tcp://10.1.1.116:55557 ||| Iteration 1 ||| Power: -16.80 dB
tcp://10.1.1.118:55555 ||| Iteration 1 ||| Power: -34.45 dB
tcp://10.1.1.118:55556 ||| Iteration 1 ||| Power: -29.91 dB
tcp://10.1.1.118:55557 ||| Iteration 1 ||| Power: -20.34 dB
tcp://10.1.1.116:55555 ||| Iteration 2 ||| Power: -34.31 dB
tcp://10.1.1.116:55556 ||| Iteration 2 ||| Power: -26.13 dB
tcp://10.1.1.116:55557 ||| Iteration 2 ||| Power: -16.72 dB
tcp://10.1.1.118:55555 ||| Iteration 2 ||| Power: -37.89 dB
tcp://10.1.1.118:55556 ||| Iteration 2 ||| Power: -29.32 dB
tcp://10.1.1.118:55557 ||| Iteration 2 ||| Power: -19.90 dB
tcp://10.1.1.116:55555 ||| Iteration 3 ||| Power: -34.22 dB
tcp://10.1.1.116:55556 ||| Iteration 3 ||| Power: -26.11 dB
tcp://10.1.1.116:55557 ||| Iteration 3 ||| Power: -16.70 dB
tcp://10.1.1.118:55555 ||| Iteration 3 ||| Power: -37.81 dB
tcp://10.1.1.118:55556 ||| Iteration 3 ||| Power: -29.53 dB
tcp://10.1.1.118:55557 ||| Iteration 3 ||| Power: -20.14 dB
```



What we did for our test:

3 channels
- Evenly spaced bandwidth

3 Iterations
- 0.3s sleep per iteration

3 Addresses
- 1.0s sleep per address

Flow:
- Checks all channels ->
- Switches address ->
- Iterates

Outputs P_OBS into a csv file

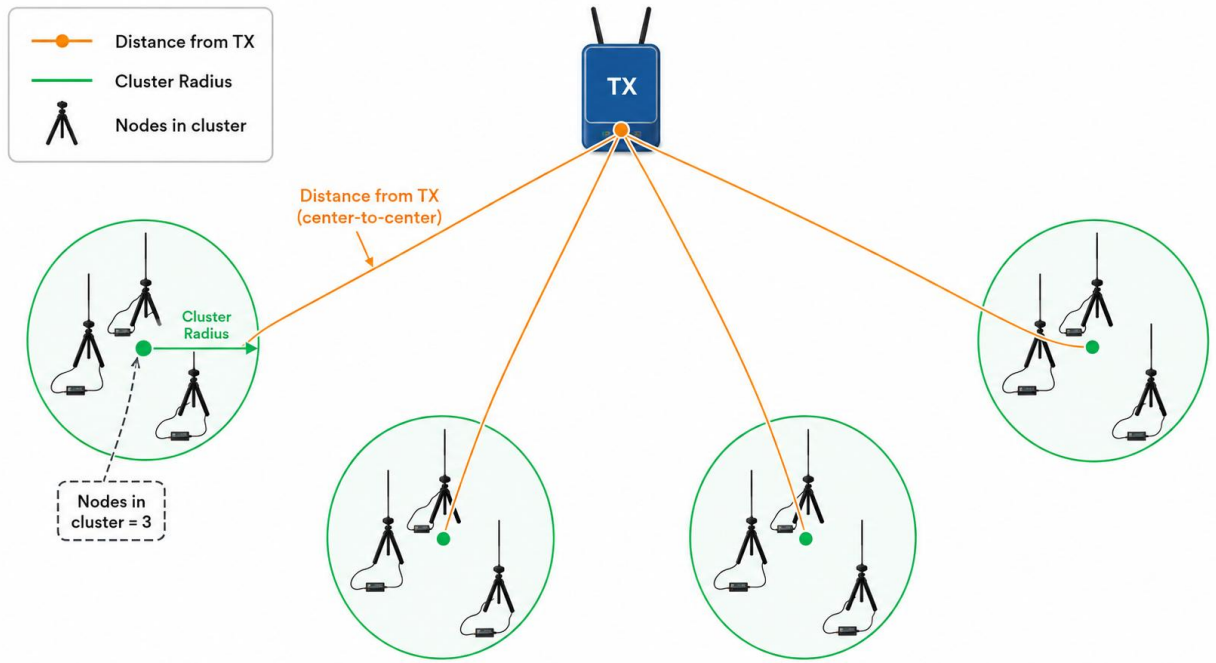
```
Channels,1,2,3
Pi.1,-41.73,-33.68,-24.43
Pi.2,-45.37,-37.94,-28.74
Pi.3,-39.31,-30.75,-21.39
```

Collecting multiple CSV files allows us to simulate multiple clusters/nodes.

Environmental Experimental Parameters

When testing for experimental scenarios we have a control variable that we vary and keep all other variables constant. The Parameters of our experiment is just the metadata for the experiments, so make sure we write down all the metadata when we take our data.

- Metadata to write down:
  - Distance from TX
  - Cluster Radius
  - Nodes In cluster
  - Number of Tests
  - Number of Channels
  - Number of Transmitters
  - Number of Receivers
  - Number of Iterations
  - Location (Water, Land, Trees, etc)
  - Physical Environment (Obstruction, Noisy Areas, etc)



Experiment TX On and Off Visual

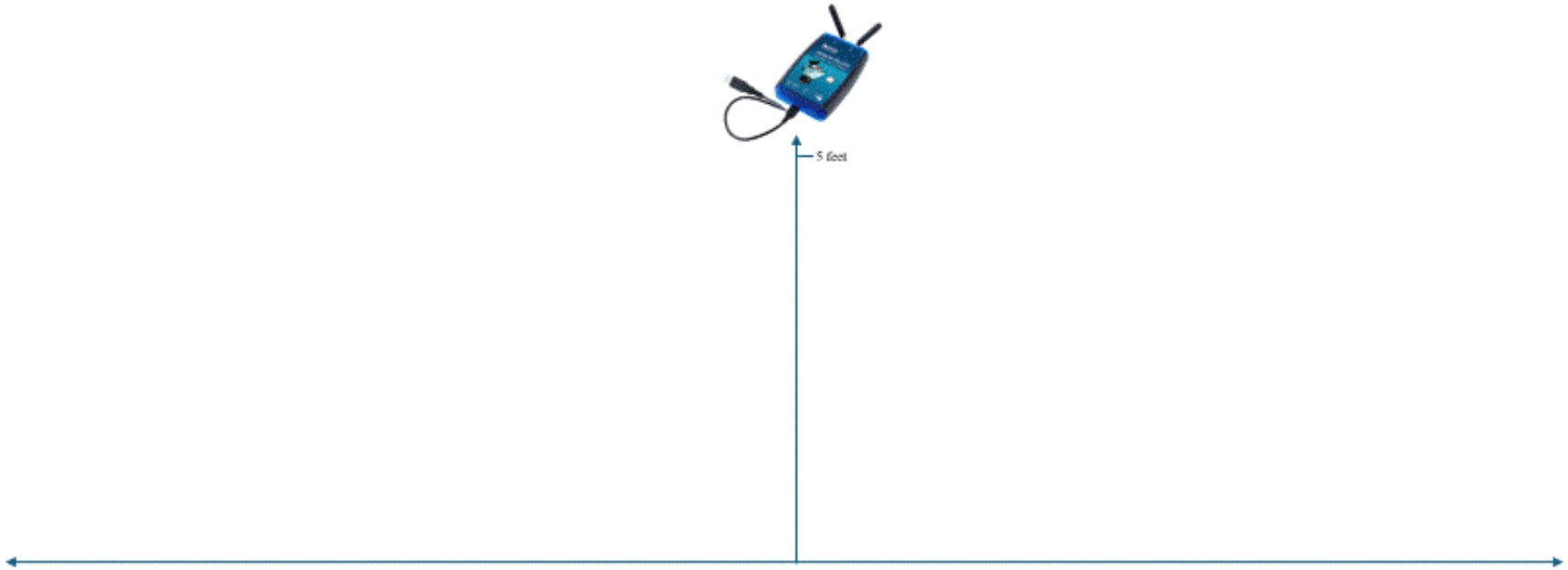
Meta Data

Test 1, Transmitter **On**:

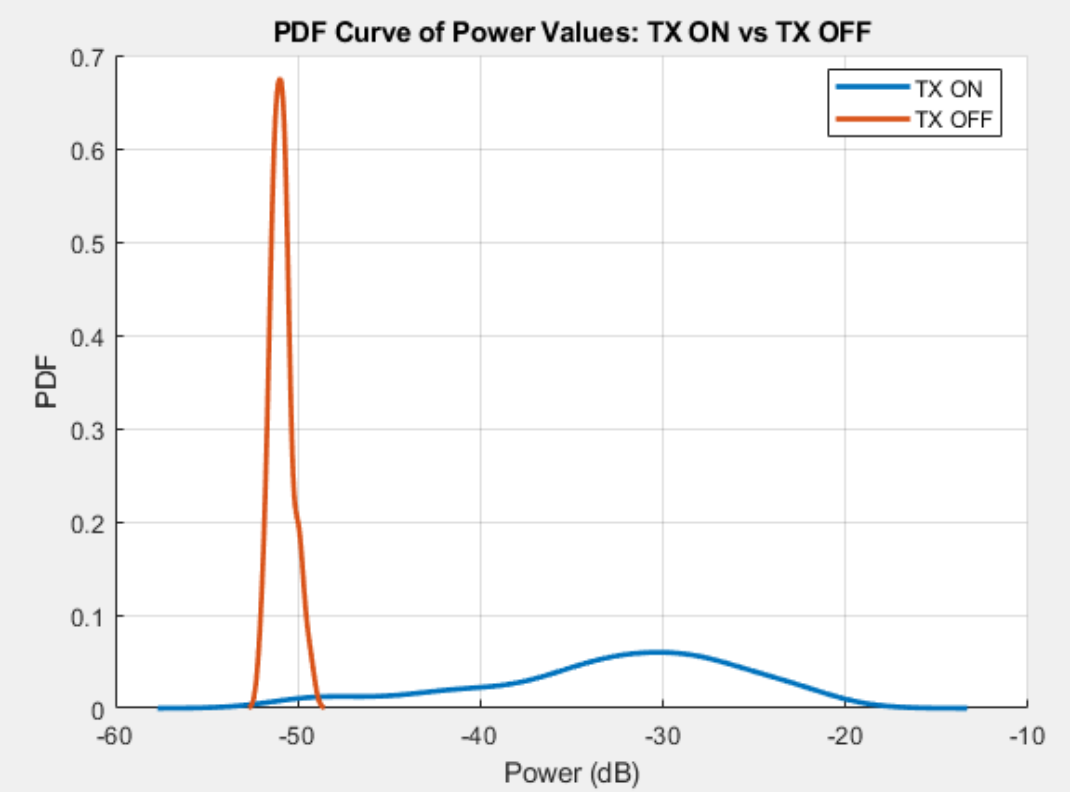
Number of tests: **30 Tests**
Cluster Radius: **2.5 Feet**
Test interval distance: **2 Feet**
Nodes per cluster: **3 Nodes**

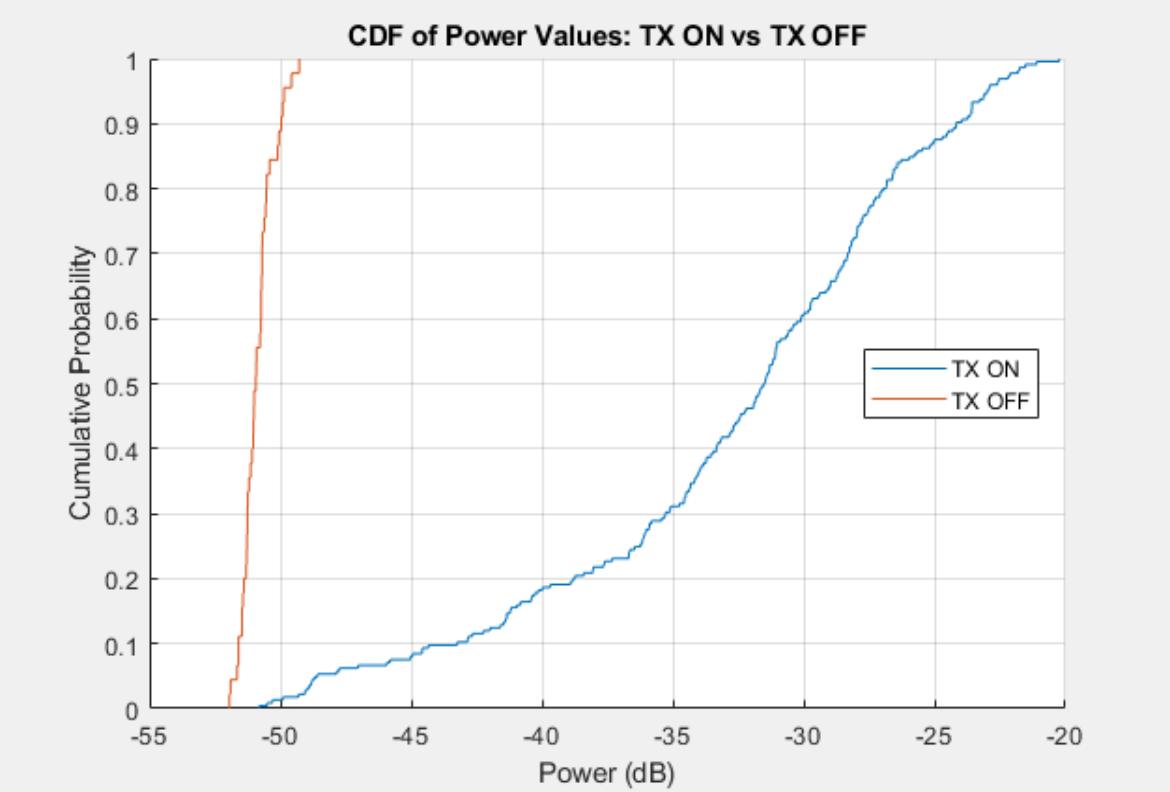
Test2, Transmitter **Off**:

Number of tests: **10 Tests**
Cluster Radius: **2.5 Feet**
Test interval distance: **2 Feet**
Nodes per cluster: **3 Nodes**



TX On / Off Data Visualization





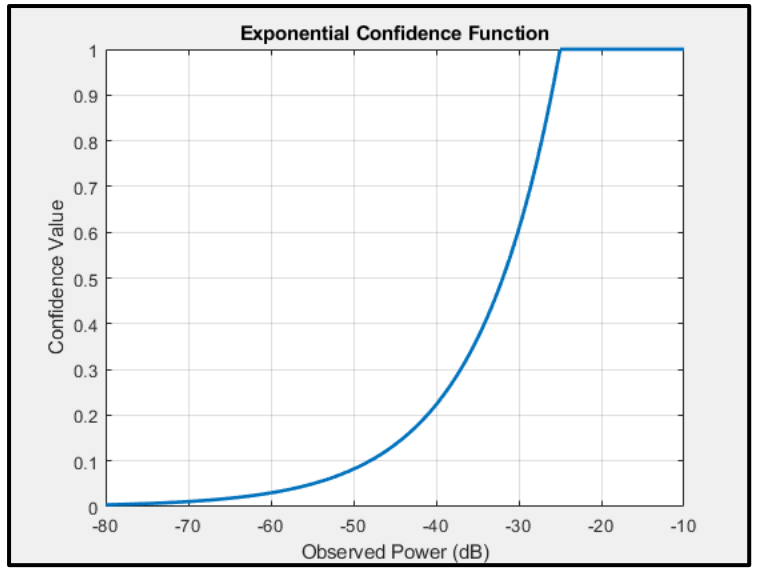
What does the PDF show?

1. TX OFF data is concentrated around -53 dB to -48 dB
2. TX ON data is concentrated around -50 dB to -20 dB
3. TX ON and OFF data overlap, meaning there is a probability of false positives and false negatives.

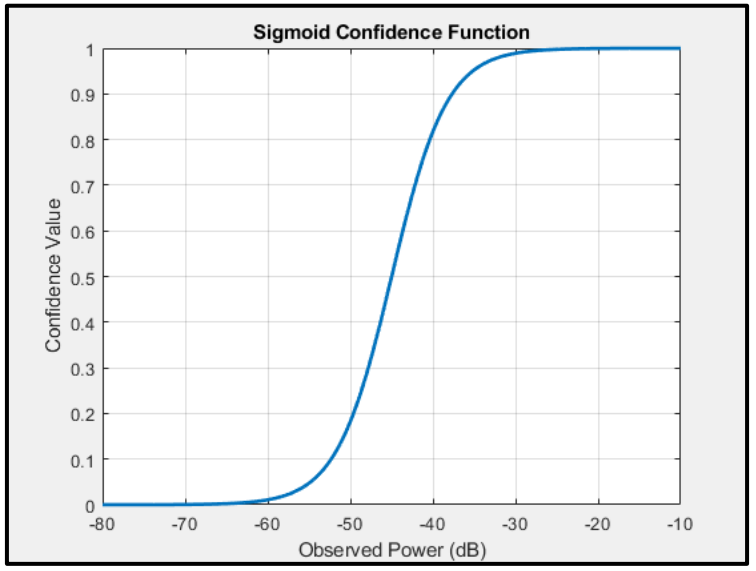
What does the CDF show?

1. 50% of our data lies above -32 dB
2. TX OFF data is more concentrated while TX ON data is more spread out

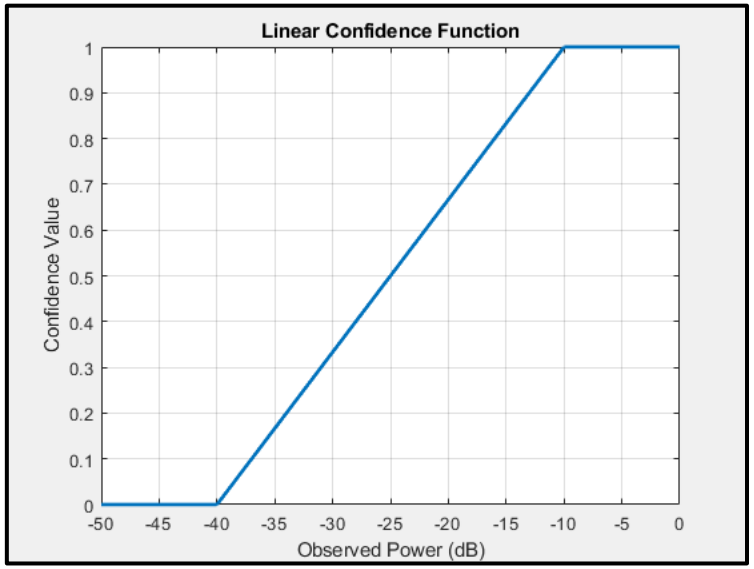
INITIAL DECISION FUNCTIONS (F_D)



Works best when requiring a high threshold of confidence before certainty. **Best used when precision is a goal**, such as security sensitive applications or similar critical environments.



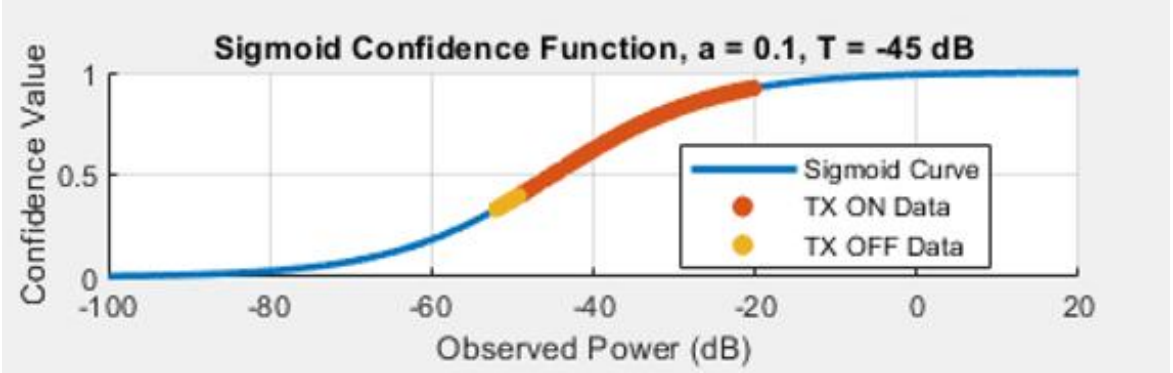
Works best when wanting to **minimize the effects of extremities**. Smooth transition from each extremity; allowing data to maintain an impactful distribution throughout.

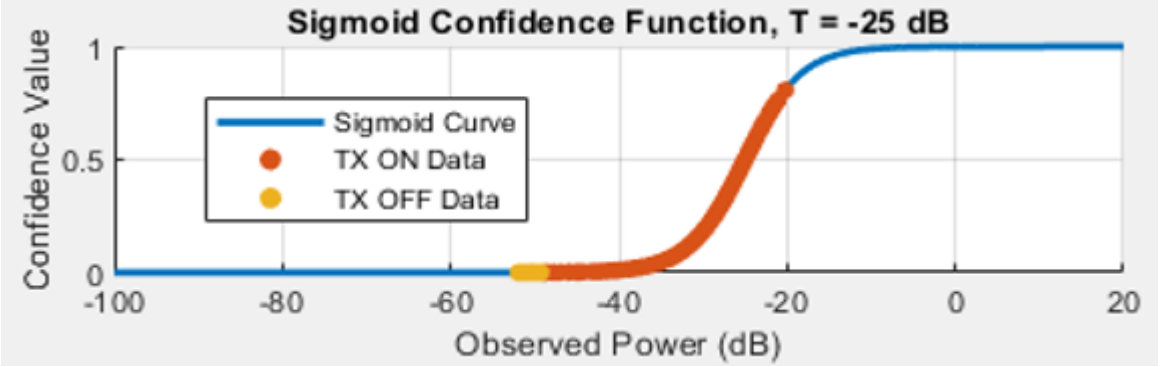


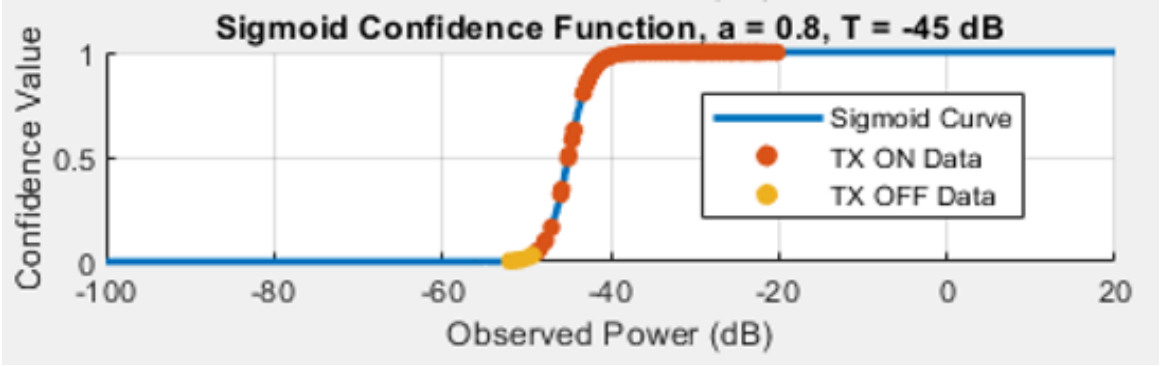
Works best when **wanting a normalized mapping excluding extremes**. Likely to be used when network is somewhat pre-defined and bounded.

To develop the initial Decision Model, different networks can use an experiential system to collect and process power measurement data. Using this data, an accurate function can be determined. That includes, but not limited to, the functions above.

LOOKING AT SIGMOID RESULTS(F_D)

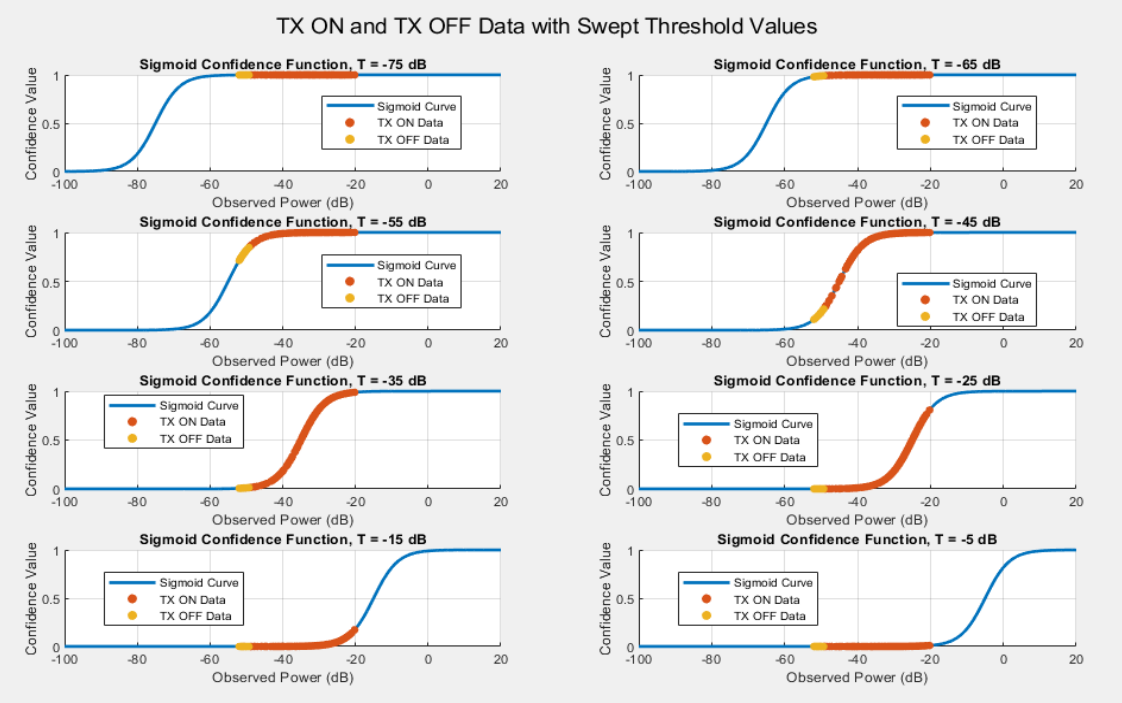


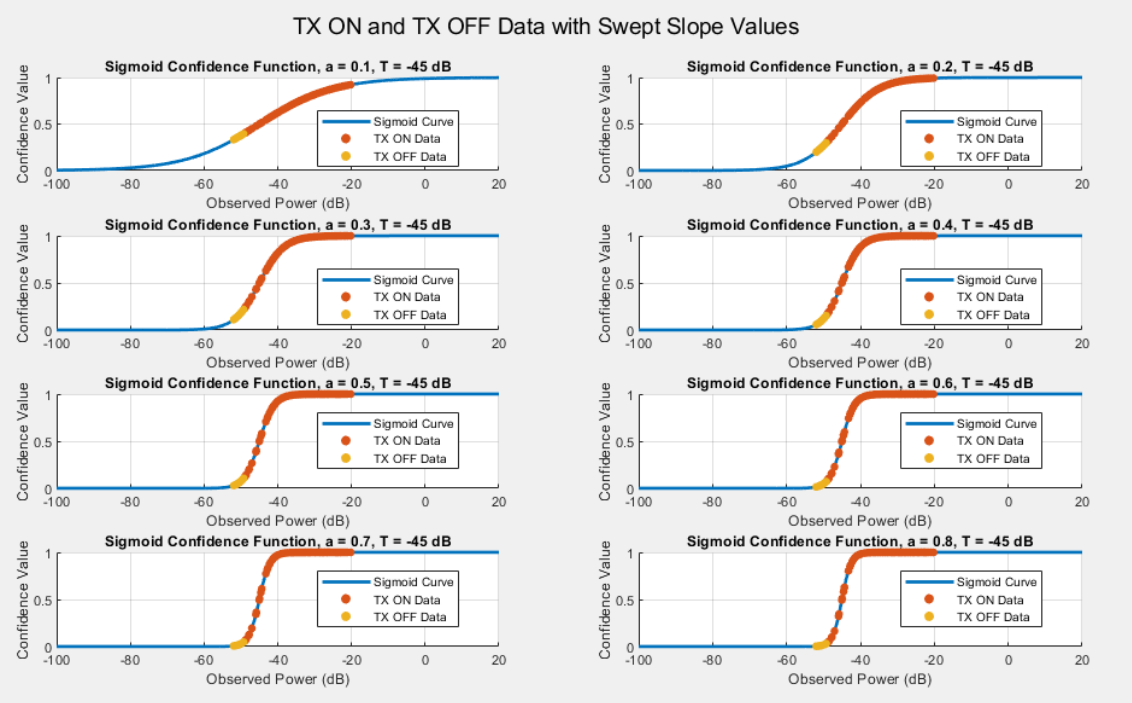




TX ON / OFF SIGMOID SWEEP

$$S = \frac{1}{1 + e^{-a(P-T)}} \rightarrow S = \frac{1}{1 + e^{-0.3(P-(-45))}}$$





How does changing threshold affect our confidence value?

1. Lowering Threshold can misclassify TX OFF data as TX ON data (False Positives)
2. Higher Threshold values can misclassify TX ON data as TX OFF data (False Negatives)
3. False positives in our initial decision model yield longer consensus times, and less accurate consensus results.

How does changing slope affect our confidence value?

1. Lower slope values yield weak initial decision confidence values, possibly causing a longer consensus time
2. Higher slope values classify some TX on Data closer to TX off, possibly causing a longer consensus time or even an incorrect consensus result

Final Conclusions

- **The system provides a repeatable testing framework**
  - The same setup can be reused by changing one control variable at a time
- **Power measurements can be used beyond TX ON/OFF**
  - The same data collection method can test different transmitter powers, receiver locations, interference levels, or channel conditions
- **CDF/PDF helped choose a better decision model**
  - The PDF showed where the power values were concentrated
  - The CDF showed how much data falls below or above a power threshold

Together, they help decide where to place the TX ON/OFF threshold

- **The sigmoid function gave a smoother confidence decision**
  - Instead of making a hard ON/OFF choice, the data can be mapped into a confidence value
  - This helps the system make better decisions when the data is uncertain or overlapping
- **The framework can support future spectrum sensing tests**
  - New nodes, channels, transmitters, or environments can be added without changing the main idea of the system
- **Threshold and distance are key characteristics for establishing link connectivity (SU to SU)**
  - These are parameters that will always affect the link connectivity in some way
- **Modular framework design allows for further research and analysis of the DSS algorithm**
  - We can change models in the framework to observe different things within the DSS algorithm by subsystems rather than a big system

THANK YOU

- **GITHUB REPOSITORY -** Utin0313/Independent_Study2026: Documentation and scripting used in Professor Rahaim's Independent Study 2026 aim to create a highly congested cluster environment in maritime settings to simulate the Dynamic Spectrum Sensing Algorithm.