Radio Direction Finding for Everyone

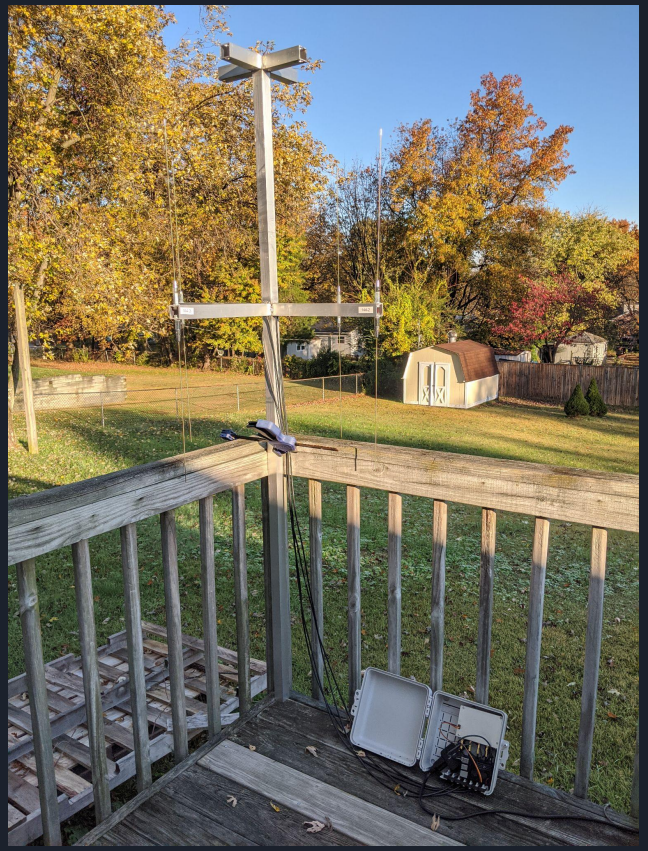
Corey Koval, Maryland, USA

Who am I?

- RF and SDR Enthusiast. I’ve been playing with SDR since 2013
- SDR Challenge Developer for the RF Hackers Sanctuary (rfhackers.com)
- Radio Direction Finding Nerd
- Contributor to KerberosSDR and KrakenSDR Software
- Developer of DF Aggregator

What is Radio Direction Finding (RDF)

- RDF is the process of determining the direction or angle of arrival of a received signal
- The direction of arrival information is often referred to as a Line of Bearing (LOB)
- Aggregating LOBs can be used to geolocate a transmitter.



Example RDF Sensors

What is DF Aggregator?

- DF Aggregator is an open source, network capable, direction of arrival bearing aggregation software.
- DFA collects Lines of Bearing (LOBs) from one or more RDF sensors and computes the best possible location of a transmitter.
- DFA displays a summarized version of the computed data on a map
- https://github.com/ckoval7/df-aggregator

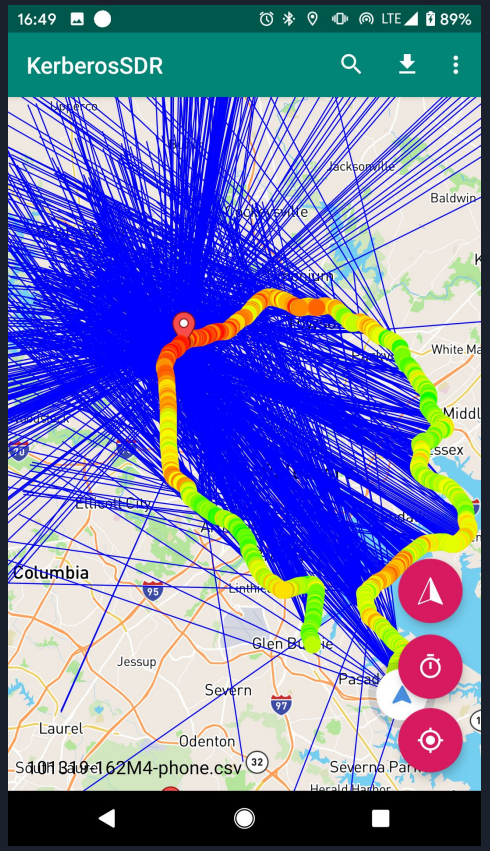
What DFA is NOT

- DFA is not command and control software
- DFA is not a general purpose mapping utility

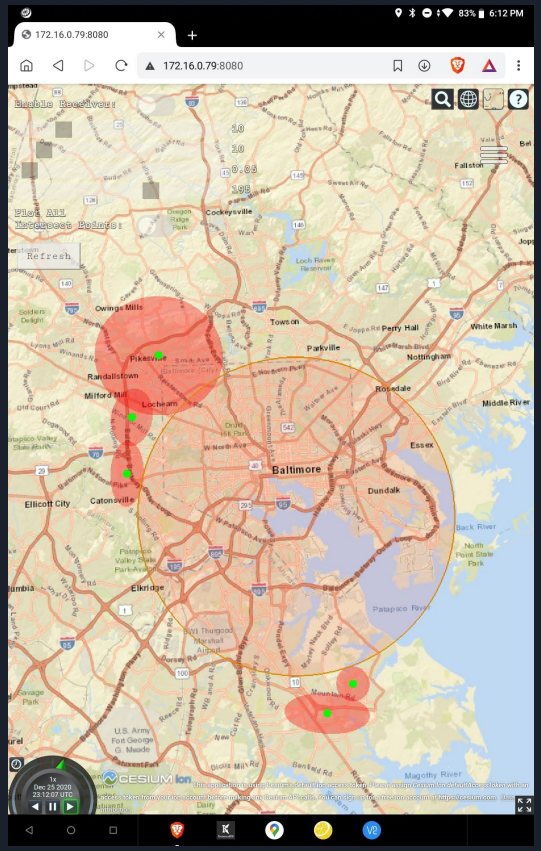
How I got here

The goals of DF Aggregator

- Build a clean interface without bearing lines going everywhere
- Have simple controls for the user
- Make the software accessible from any web browser
- Compatible with consumer RDF units, primarily KerberosSDR
- Make it Free/Open Source



Let’s get from this…



…to this

Overcoming Obstacles

- The KerberosSDR Software doesn't provide location information.
  - Solution: Integrate GPSd into a custom fork of the software

- No other code to cheat off of
  - Solution: Actually learn Python

- I can’t be DFing all the time to build test data sets
  - Solution: Build a data simulator

Step Zero:
Data Collection, Storage, and Display

- KerberosSDR outputs XML data through an HTTP request
- SQLite is an efficient way to store and retrieve the data
- CesiumJS Provides an easy to integrate, 3D interactive map

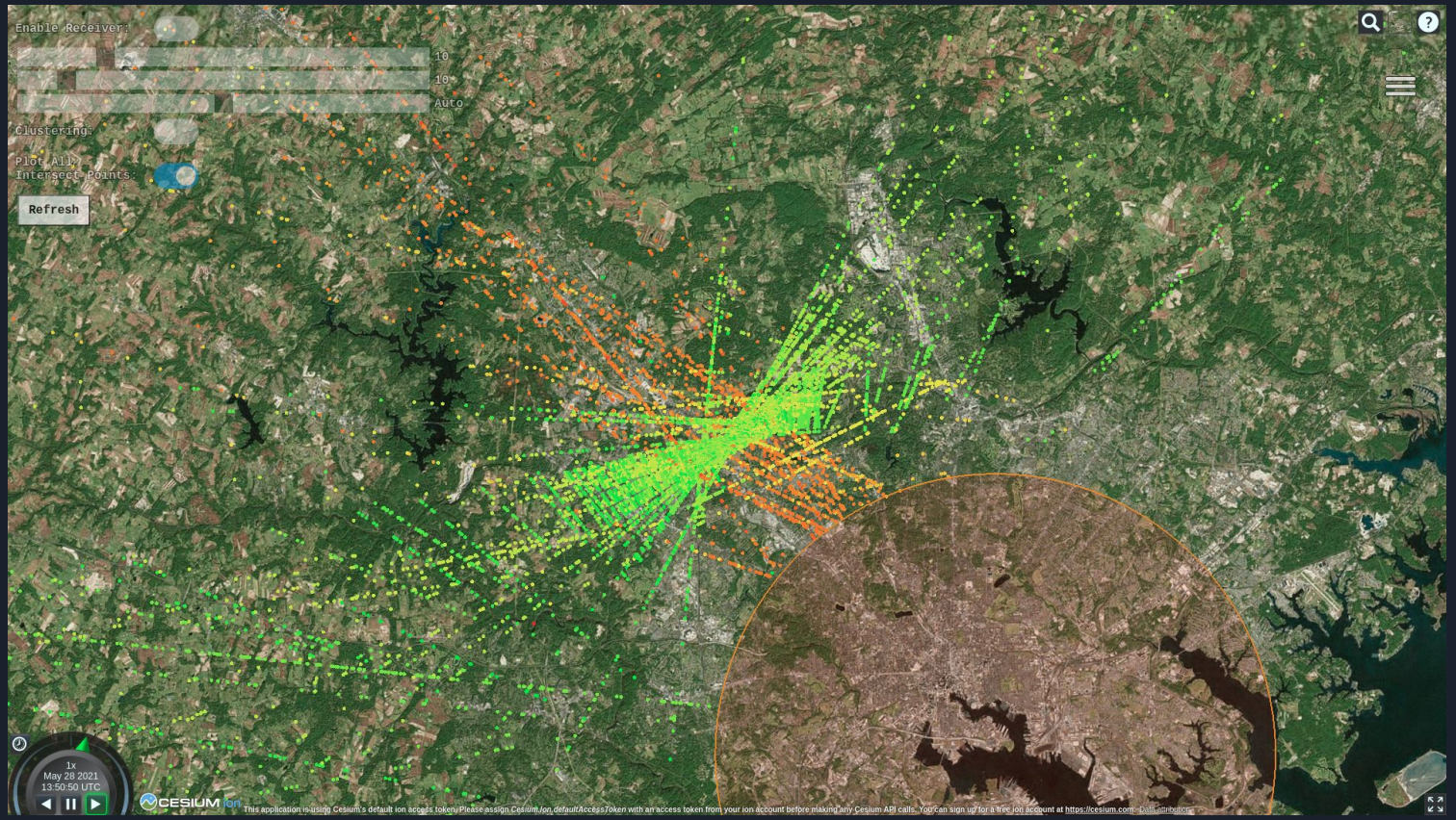
Step One: Computing Intersections

- Let’s assume the earth is round
- Great circle intersections improve accuracy over longer distances
- When computing intersections for two receivers, a single point is generated
- Three or more receivers will generate multiple points.
- An average of all the intersections computed in a single cycle creates a single point
- Weighted averages did not seem to improve accuracy

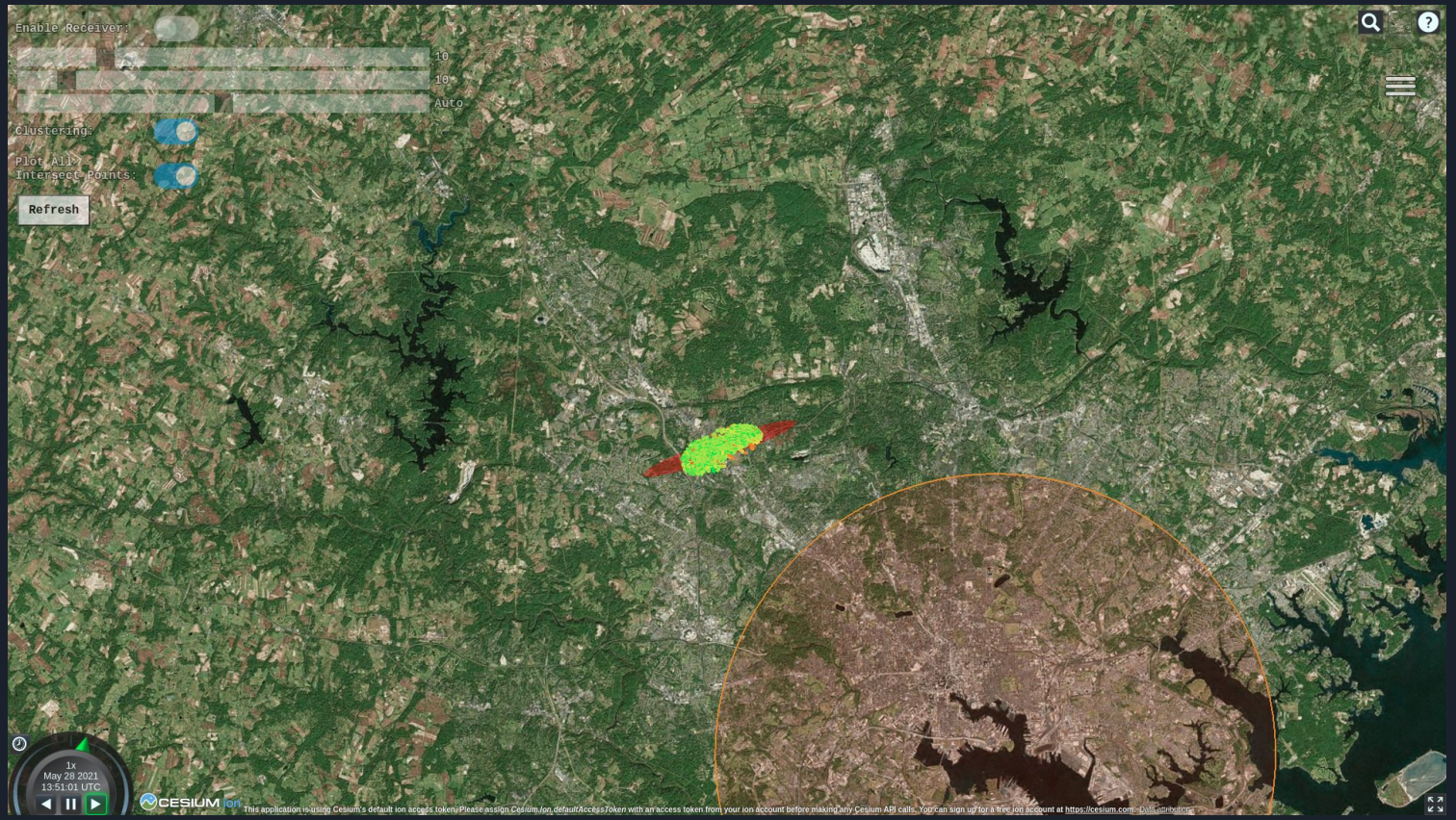
Accuracy over 21km

Cluster Estimation

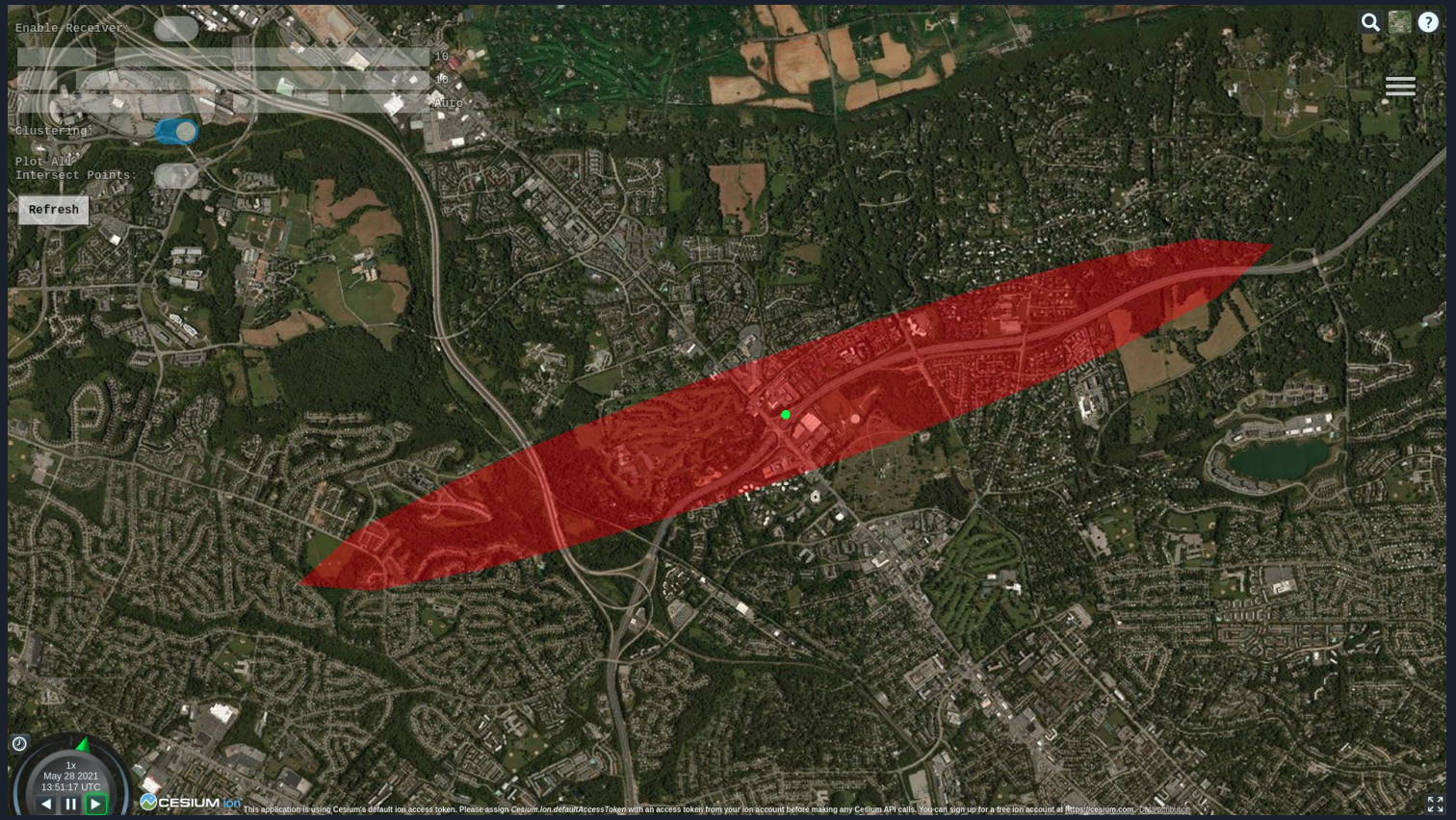
- Take out the human element when identifying intersection-dense regions
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
- minPts: The minimum number of points (a threshold) clustered together for a region to be considered dense.
- eps (ε): A distance measure that will be used to locate the points in the neighborhood of any point.
- Auto epsilon estimation: https://iopscience.iop.org/article/10.1088/1755-1315/31/1/012012/pdf
- Memory Intensive, it had to be limited to around 20k intersections



Calculated intersections without clustering



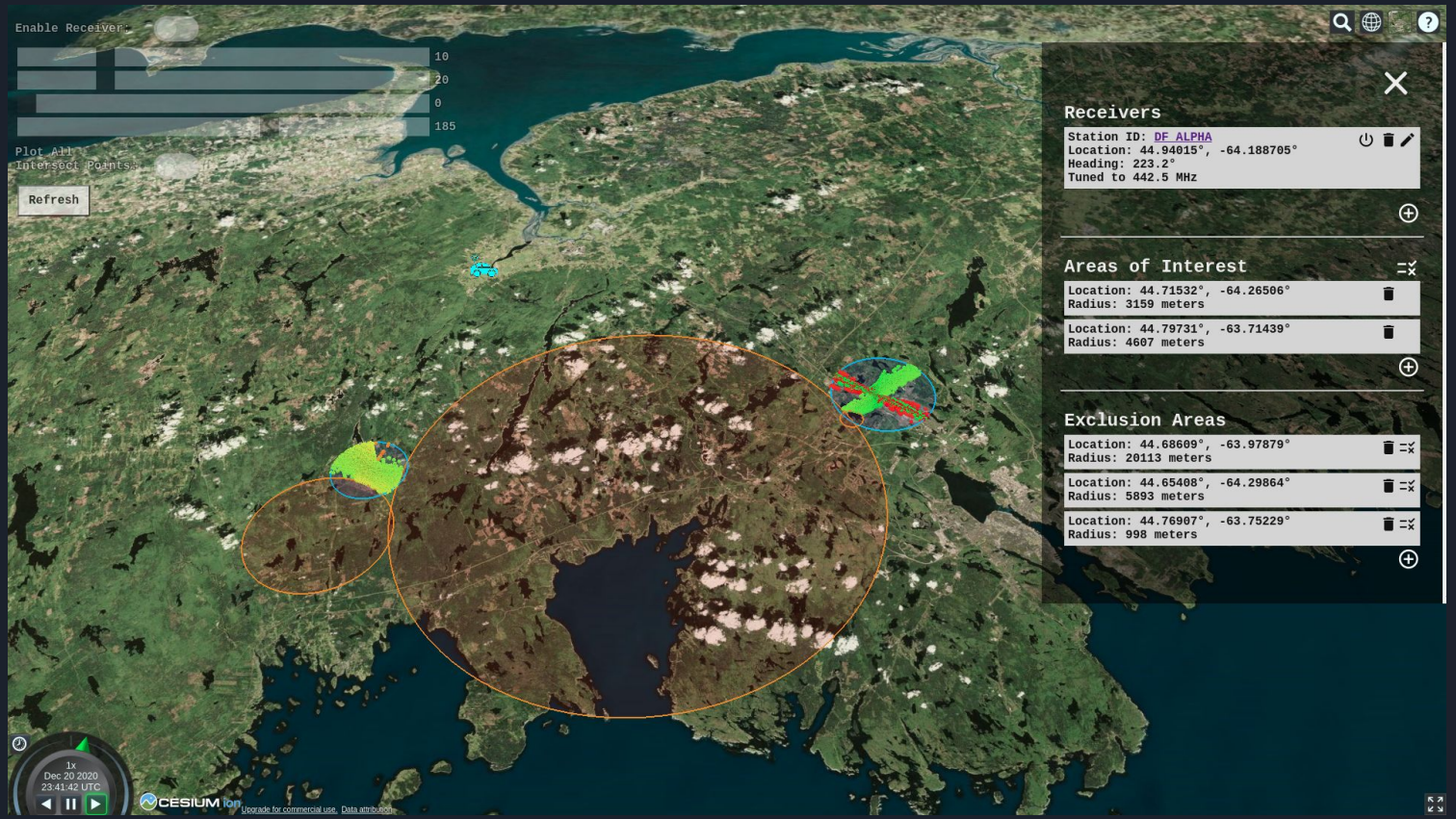
Clustering enabled, intersections outside the cluster are hidden



Intersection points turned off, showing an ellipse as a data summary

Filtering data: Areas of interest

- Identify positive areas (I know it's around here)
  - Areas of Interest (AOI)
- Identify negative areas (I know it's not here)
  - Exclusion Areas



Using AOIs and Exclusion Areas to isolate two transmitters when using a single receiver

Plans for the Future

- Add KrakenSDR support
  - Multiple LOBs per cycle
  - Updated Data Format
- Add a timeline to review historical data?
- Submit your ideas and pull requests on GitHub

Questions?