AMIGO DOCKER CONTAINER

Below we provide instructions for running the main Amigo codebase.

We provide samples of code which is easy/quick to run.

Setting up our docker container

We provide a docker container which contains the primary Amigo code base.

To **build** our docker container, run:

```
cd amigo-docker
chmod +x runners/setup-docker.sh
./runners/setup-docker.sh
```

Setting up ns-3 for model code

Once you are in our docker container, to run our simulations, you will need to configure and build ns-3, and implement our custom mods to ns-3 to simulate Wi-Fi Direct.

We provide an executable to implement this process, which you may run inside the docker container, using:

```
chmod +x /runners/setup-ns3.sh
./runners/setup-ns3.sh
```

Running model code

Our paper evaluates the impact of different routing mechanisms and protest features on message delivery rates (Section 7). We use this script to execute a simulation run in ns-3, and collect information about packet and message delivery.

We provide an executable to implement this process, which you may run inside the docker container, using:

`chmod +x /runners/run-test-sim.sh`

`./runners/run-test-sim.sh`

Our code entails the following parameters:

```
# <nodeCount> : # of nodes in the sim
# <buffSize> : buffer size (bytes)
# <trafficModel> : traffic model type (standard or event)
# <mobilityModel> : mobility model (static, random, chain, march, gather, bloc
kade1, blockade2)
# <groupSize> : group size (number of nodes in each group)
# <runIter> : iteration number (for multiple runs)
```

We include the mobility and traffic models used to produce our paper's results for use.

Our shell script as is runs for a short run which should complete in <60 minutes; other runs will take much more time (on the scale of 1-2 days).

On completion, the output will be written to:

`/protest/results/...`

Producing mobility models

Our paper provides a mechanism for producing realistic protest mobility models (Section 6). We use this script to produce sample mobility models.

To run this code, we provide a script to set up our python configuration:

`chmod +x /runners/setup-pyenv.sh`

`./runners/setup-pyenv.sh`

To produce mobility models:

`chmod +x /runners/run-test-model-gen.sh`

`./runners/run-test-model-gen.sh`

Our mechanism includes the following parameters:

```
# <mobType> : mobility model (static, random, chain, march, gather, blockade
1, blockade2)
# <nodeCount> : # of nodes in the mobility model
# <runIter> : iteration number (for multiple runs)
# <seconds> : duration of the simulation in seconds
```

These mobility models are

On completion, the output will be written to:

`/protest/mobility_models/...`

Producing traffic models

Our paper provides a mechanism for producing realistic protest traffic models to accompany our mobility models (Section 6). We use this script to produce sample traffic models.

To produce traffic models:

`chmod +x /runners/run-test-traffic-gen.sh`

`./runners/run-test-traffic-gen.sh`

Our mechanism includes the following parameters:

```
# <mobType> : mobility model (static, random, chain, march, gather, blockade
1, blockade2)
# <nodeCount> : # of nodes in the mobility model
# <groupSize> : group size (number of nodes in each group)
# <trafficModel> : traffic model type (standard or event)
# <runIter> : iteration number (for multiple runs)
```

On completion, the output will be written to:

`/protest/traffic_models/...`

and

`/protest/group/...`

Parameters from our testing

We ran our test code with the following:

- Docker version: 4.41.2
- Ubuntu version in docker container: 22.04
- ns-3 version: 3.43
- python version: Python 3.10.12

We performed runs on 64 CPU cores and 250 GB RAM.

We tested our provided sample code on a machine with 4 cores and 16 GB RAM.