Distributed Machine Learning Test Bench with Raspberry Pis

Nikhil Shenoy and Revan Sopher
nrs76@scarletmail.rutgers.edu, rsopher@gmail.com
Advisor: Prof. Anand Sarwate

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Project Repository
https://github.com/revan/RPi-distributed-ML

Overview

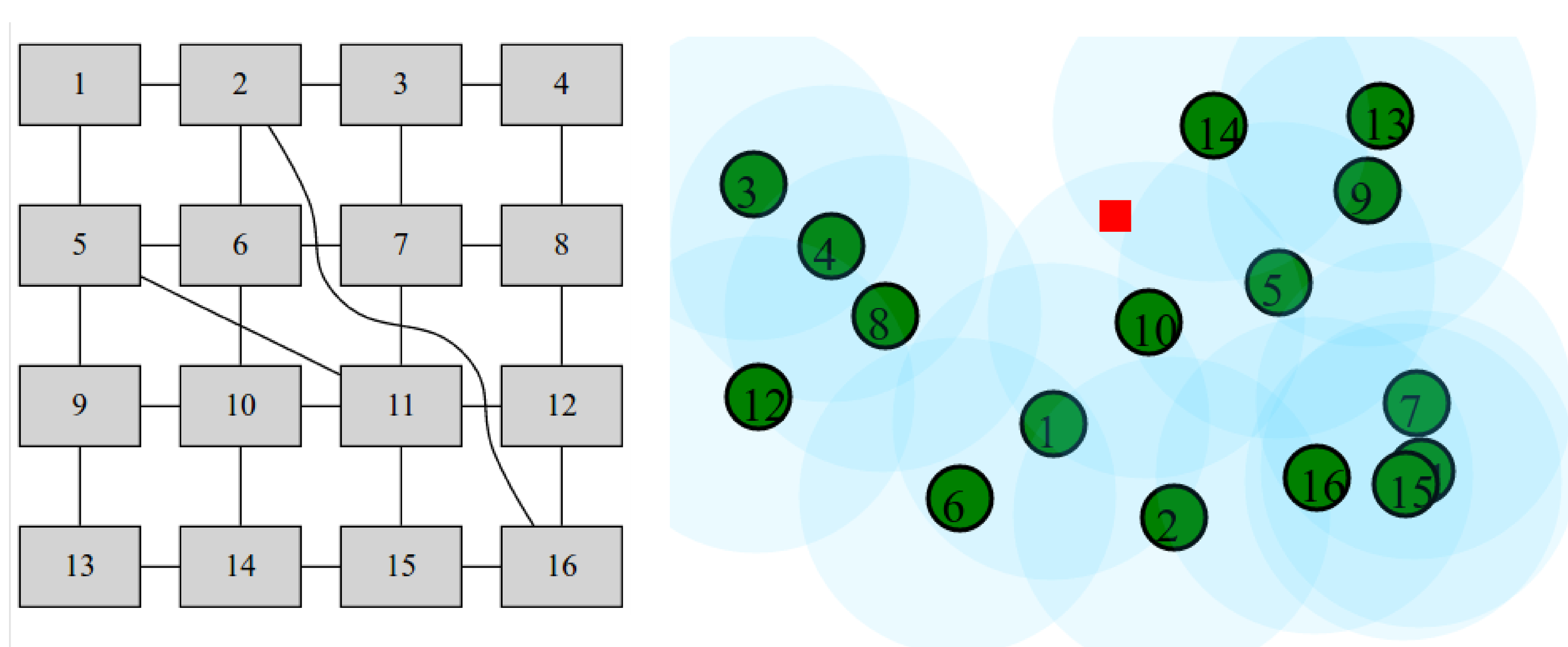
We develop a Python library facilitating the development of distributed machine learning algorithms using embedded devices. As a demonstration of the library's functionality, we present implementations of geo-routing and classifier training.

Design Goals

- ❑ Create a testbed for easy, cost-effective testing of distributed and machine learning applications
- ❑ Abstraction of system-related tasks (e.g. synchronization); allows user to focus on his own application
- ❑ Implemented in Python; includes functionality from libraries such as Numpy, Scikit-learn
- ❑ Use of cheap, commonly available hardware allows for easy setup and prototyping
- ❑ Minimal Infrastructure
  - ❑ Code is deployed to cluster via "git push"
  - ❑ Nodes are addressable by name, not IP
  - ❑ Device Agnostic

Configurable Topology

- ❑ Library loads JSON file to restrict node connectivity
- ❑ Web UI allows generating configurations simulating Ethernet or WiFi



The Averaging Problem

- ❑ Model: Segment data, take local action, exchange results with neighbors
- ❑ Decentralized computation (e.g. sensor networks)
  - ❑ Asynchronous and synchronous
- ❑ Naturally extended to algorithms such as gradient descent and support vector machines

$$\overrightarrow{x_{t+1}} = W^{\alpha}\overrightarrow{x_t}$$

The synchronous update rule, with the *x* vector as each node's value at time *t* and *α* as the number of iterations.

$$W = \begin{bmatrix} 1-\frac{d_i}{d_{max}+1} & \frac{1}{d_{max}+1} & \frac{1}{d_{max}+1} \\ \frac{1}{d_{max}+1} & 1-\frac{d_i}{d_{max}+1} & \frac{1}{d_{max}+1} \\ \frac{1}{d_{max}+1} & \frac{1}{d_{max}+1} & 1-\frac{d_i}{d_{max}+1} \end{bmatrix}$$

A sample stochastic matrix, with d_i as node *i*'s degree, and d_{max} as the maximum degree in the topology.

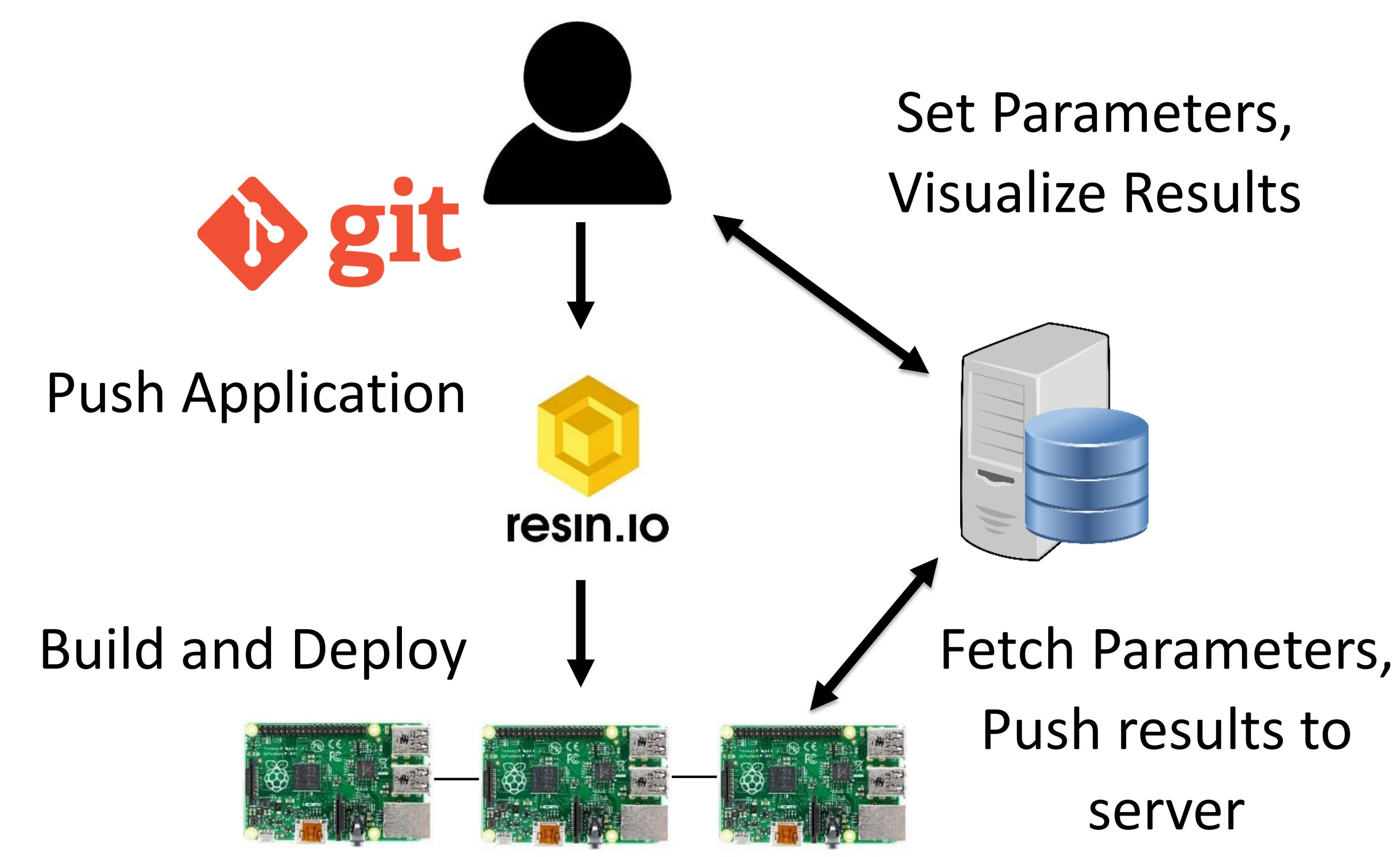
Geo-Routing

- ❑ Sensor network task
- ❑ Route message to given GPS coordinates over unknown topology
- ❑ Every node only knows own and neighbors' locations
- ❑ Greedily pass to closest neighbor

Future Work

- ❑ Testing with physical sensors
- ❑ Simulation of packet dropping, dying nodes
- ❑ Bandwidth/Power consumption tests for field deployments

System Architecture



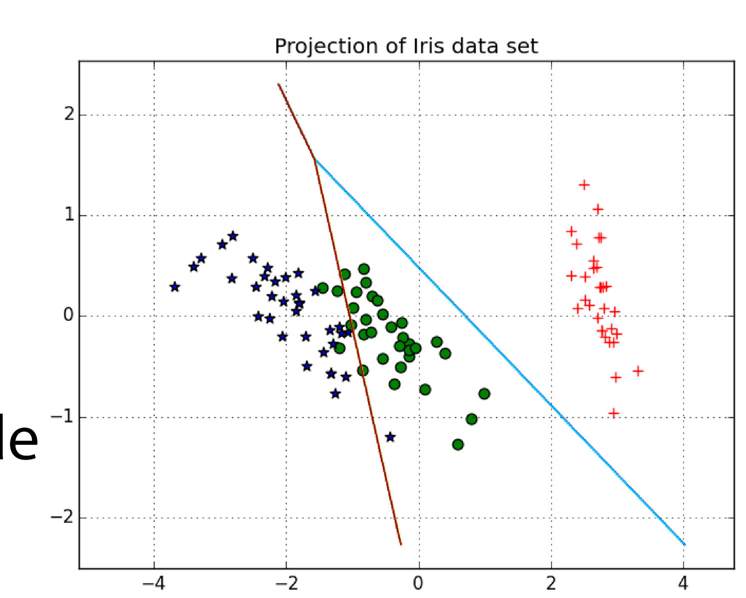
Message Sending and Callbacks

```
m = Messager()
neighbor = '5'
message = { 'num': 42 }
m.sendMessage(neighbor, message)

def callback(message, name):
    print('Got message from %s: %s' % (name,
    message))
m.registerCallback(callback)
```

Classifier Training

- ❑ Gradient Descent, SVM
- ❑ Use synchronous averaging to get feature vector
- ❑ Compute individual node errors and final classification



Acknowledgements

We would like to thank resin.io for generously donating the Raspberry Pis, and Professor Sarwate for his guidance on this project!

resin.io