Final report: Particle swarm network simulation

Course	Modeling Abstractions for Embedded/Networked Systems (CSE5309)
Instructor	Fei Miao
Date	Spring 2022
Student	Lynn Pepin
NetID	tmp13009, 2079724
Due:	May 7th

Basic functionality

The goal of this project is to create the foundations for an advanced network simulator. This work makes many (intended!) ommissions and simplifications, using a modularity approach combining entity/component/scene composition and functional programming.

This project simulates a discrete-time network wireless physical-level mesh network. The goal is to measure network throughput in the presence of congestion for different collision-avoidance mechanisms. The network consists of N particles in a swarm communicating over k wireless channels.

This simulation is implemented primarily using Python3, numpy, and PyGame[1], the latter of which is used for rendering the simulation.

As stated, the entity/component/scene system is applied with functional programming to increase modularity:

- The **Entity Component Scene** pattern:
- Each `Node` is an entity representing a particle with components such as the game screen or messages.
- A `Swarm` manages the movement of these Nodes, passing the same components as Node, functional patterns.
- The `Spectrum` simulates the passing of messages in a wireless spectrum.
- Components include the virtual screen used for rendering, channels, and messages.
- One whole simulation is one scene.
- **Functional patterns** are used to make the code re-usable, allowing the logic of one entity to be modified without requiring large structural changes. For example,
- Rendering transformations are passed to entities as
- The Scene is the PyGame main loop, and while these can be modularly composed
- The PyGame loop instantiates one Swarm, which has one Spectrum class and many Nodes.
- The Node and the Spectrum interact

[1] https://www.pygame.org/

- The Swarm defines the physics controlling the nodes and handles the logic

Model description

- Main challenges in system design
- How solved
- Other existing models

Appendix A: Tables of notation

General simulation notation:

(r, θ)	Polar coordinates, (meters, radians)
(x, y)	Cartesian coordinates (meters, meters)
$n \in N$	Node index
$k \in K$	Channel index
t	Time (seconds)
Δt	Simulation timestep (seconds)

We omit timestep t in calculations which have no dependency between timesteps (which is most of them).

Notation used in networksimulation:

d_{ij}	Distance between nodes i and j.
m_{jk}	Indicator if node j communicates on channel k.
$a_{i,j,k}$	Portion node i perceives from j on channel k. $= d_{ij}^{-2} m_{jk}$, with unit $\left(\frac{W}{m^2}\right)$
$a'_{i,k}$	Total intensity node i perceives on channel k. $= \sum_{j=1}^{n} d_{ij}^{-2} m_{jk}$ with unit $\left(\frac{W}{m^2}\right)$

Notation used in network simulation (matrices):

These matrices are not used in this paper, but are used in the implementation and are worth describing.

$D \in \mathbb{R}^{i,j}$	Inverse-square distance matrix. Let $D_{i,j} = d_{i,j}^{-2}$
$M \in \{0,1\}^{i \times j}$	Message indicator matrix.
$A \in \mathbb{R}^{i,j,k}$	Node-node-channel intensity tensor. $A_{ijk} = D_{ij} M_{jk}$ as Einstein-Summation.[2] Equivalently `np.einsum('ij,jk->ijk', D, M)`
$A' \in \mathbb{R}^{i,k}$	Node-channel intensity matrix. $= D_{ij} \times M_{jk}$.

[2] Einstein-Summation notation is described succinctly here: https://rockt.github.io/2018/04/30/einsum

Appendix B: Particle movement patterns

TLDR: The particles move in a circle with several "lanes". The particles are then perturbed by adding Gaussian noise, allowing them to "jump" to other lanes.

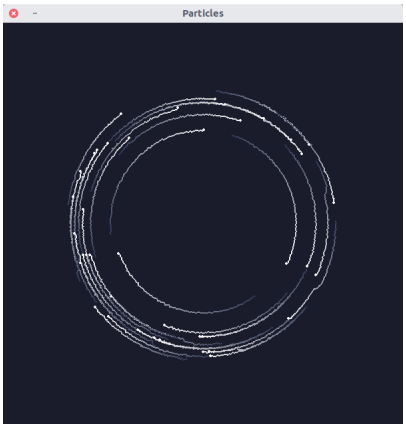

Figure 1: Particles in normal movement conditions.

The N particles are initialized at random at radius $\mathcal{N}(\mu = 200, \sigma = 30)$ and angle uniform $\mathcal{U}(2\pi)$ from center. The particles do not collide with one.

Primarily, the particles move according to a system of differential equations, where (r, θ) are polar coordinates (with the usual transform to Cartesian (x, y) coordiates in $\mathbb{R}^2$):

$$\frac{d\theta}{\Delta t} = \frac{a_1}{r^2}$$

$$\frac{dr}{\Delta t} = \frac{R - r}{R} + a_2 \cos\left(a_3 r\right)$$

Here, R, a_1, a_2, a_3 are constants. This series of equations defines a system in which particles rotate at radius R around center $(0, 0)$, with rotational speed proportional to $\frac{\pi}{r^2}$. The term $\frac{R-r}{R}$ yields local optimum around $r = R$, but the term $a_2 \cdot \cos\left(a_3 \cdot r\right)$ adds local optimum "lanes" around R. In this manner, we get an interesting spread of particles.

We set $R = 100$ is the radius around which the particles rotate, $a_1 = 20000$ to control the rate of rotation. The combination of $a_2 = \frac{3}{2}$ and $a_3 = \frac{pi}{3}$ yield a nice spread of locally-minimum "lanes" around which particles can fall into.

The radius is then perturbed by $\mathcal{N}\left(0, 30\right) \Delta t$ at each timestep. This added noise causes particles to occasionally 'jump' between the local-optimum 'lanes'.

The motion of recently-initialized particles is visualized in Fig 2, where distant particles quickly move toward the $r = R$ center, moving as visualized in Fig 1. Note the trails showing how particles 'jump' to different optimum due to added noise. Particle trails without noise are seen in Fig 3. Note the smoother motion and the lack of 'jumps'.

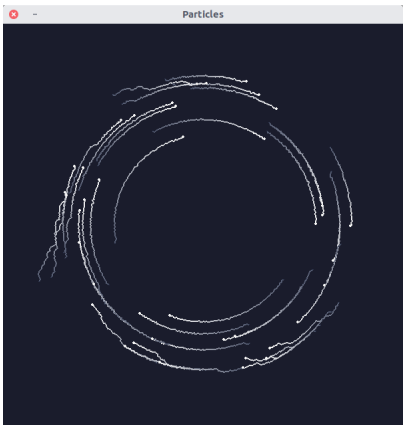

Figure 2: Early stages of particle movement.

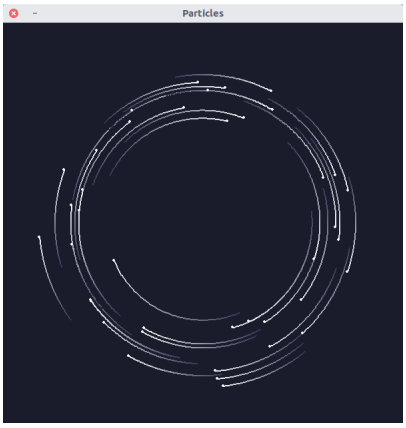

Figure 3: Particles in normal movement conditions without noise added

Appendix C: Physical layer contention modeling

> **TLDR:** We model signal intensity and contention on a channel according to the inverse-square law. Node i succesfully receives a message from node j on channel k if and only if that message accounts for at least 50% of the amplitude. (Put simply, only if that message is the "loudest".)

We want to simulate a raw wireless physical layer and measure the throughput in bits. This means no error-correction or other protocol-level improvements. The biggest difficulty here is contention (destructive interference when multiple nodes are talking over the same channel.)

We make significant assumptions here, primarily that:

1. The messages have no signal attenuation, background interference, scattering, absorption, or cross-talk.
2. Messages are fixed-length, and are sent and received within one timestep.
3. Destruction is identified instantly.
4. All messages are sent with a fixed power.
5. Nodes account for interference caused by their own messages.
    - This means two nodes i and j can talk over channel k without interference to one another.

This allows us to model contention only according to signal strength governed by the inverse-square law, that is, we assume Intensity $= \frac{\text{constant}}{\text{distance}^2}$. The constant does not matter, since it disappears in all calculations below.

Let d_{ij} denote the distance between nodes i and j, and m_{jk} indicate if a node j is broadcasting a message on node k. The intensity node i perceives of a signal from node j on a given channel k is given as

$$a_{i,j,k} = d_{ij}^{-2} m_{jk},$$

while the total intensity node i perceives over channel k is given as

$$a'_{ik} = \sum_{j=1}^{n} d_{ij}^{-2} m_{jk}.$$

We assume node i succesfully receives message j over channel k if $\frac{a_{ijk}}{a'_{ik}} \geq \frac{1}{2}$.

We assume a message is constructively received by node i from j over k iff $I'_{ijk} \geq \frac{1}{2}$.