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

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

- 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
- 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: Table of notation

(r, θ)	Polar coordinates, (meters, radians)
(x, y)	Cartesian coordinates, meters
$n \in N$	Node index
$k \in K$	Channel index
t	Time in seconds
Δt	Simulation timestep
d_{ij}	Distance between nodes i and j at a gigven timestep.
m_{nk}	Indicator function if node n communicates on channel k.
I_{ijk}	Intensity between two nodes i and j on channel k. $= m_{jk}\sqrt{d}_{ij}$

Timesteps t are often omitted when notation is only used in calculations that have no dependencies between timesteps (such as distance d_{ij}).

Appendix B: Particle movement patterns

> **TLDR:** The particles move in a spiral with several local optimum per radius. The particles are then perturbed by adding Gaussian noise, allowing them to "jump" to other local optimum.

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 another and obey basic Newtonian physics.

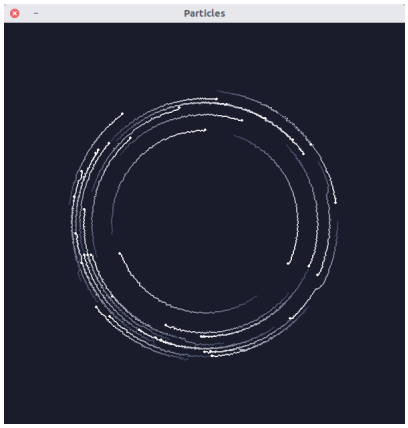

Figure 1: Particles in normal movement conditions.

First, 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 $\mathbb{R}^2$):

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

$$\frac{dr}{dt} = \frac{R - r}{R} + a_2 \cos(a_3 r)$$

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 $a_2 \cdot \cos(a_3 \cdot r)$ 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 $\mathcal{N}\left(0, \frac{1}{2}\right)$. This added noise allows particles to 'jump' lanes.

The motion of recently-initialized particles is shown in Fig 2, where distant particles quickly move toward the $r = R$ center, moving as shown 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'.

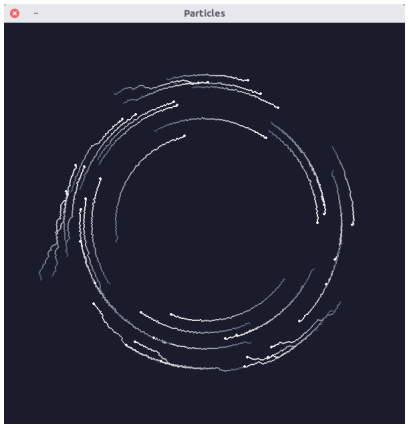

Figure 2: Early stages of particle movement.

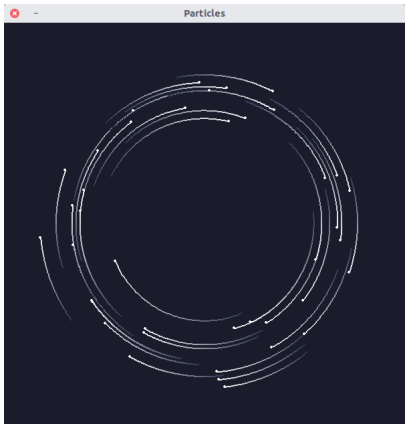

Figure 3: Particles in normal movement conditions without noise added

Appendix C: Physical layer contention modeling

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

We make significant assumptions here, primarily that:

1. there is no cross-talk between channels,
2. there is no background interference and no scattering / absorption from objects the environment,
3. and messages start, finish, and are received in only one timestep,
4. Messages all contain different content and are of the same length,
5. Destruction is identified in the same timestep as a message is received.

So, the only contention that can happen is when two channels talk at the same time. That is, for a receiver receiving signals simueltaneously from J nodes $n_1, n_2, ..., n_j$, with distances $d_1, d_2, ..., d_J$, which will be the dominant message?

We assume no significant signal attenuation the medium and no beamforming, so signals are only degraded per the square root law. 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. So, a given node i will see the intensity I_{ijk} from node j as $\sqrt{d}_{ij}m_{jk}$. So, the relative intensity of a message from the perspective i of node j broadcasting on channel k is given as

$$\frac{I_{ijk}}{\sum_{n=1}^{N} I_{njk}} = \frac{\sqrt{d_{ij}}m_{jk}}{\sum_{n=1}^{N} \sqrt{d_{in}}m_{nk}}$$