CS 351: Design of Large Programs
Project 4: Mobile Agents Monitoring Forest Fire

Brooke Chenoweth

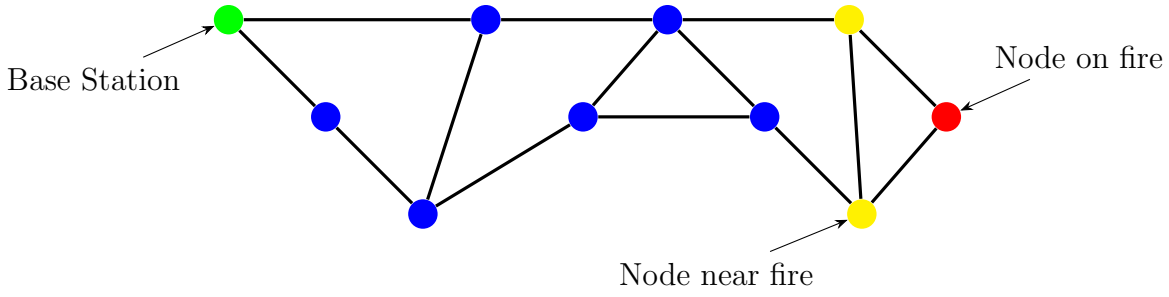
Spring 2025

Imagine a situation where we have placed sensors throughout a forest to monitor it remotely during fire season. Each sensor can communicate wirelessly with other sensors within a certain range, but won't necessarily have the broadcast power necessary to talk directly to the base station back at the fire tower.

We are going to model a system where mobile software agents travel among these sensor nodes to discover and monitor forest fires. In a real system, once the fire is detected, some sort of steps might be taken to put it out (or possibly not, as forest fires are a natural occurance needed for the health of the forest), but in our simulation, we'll just watch it spread.

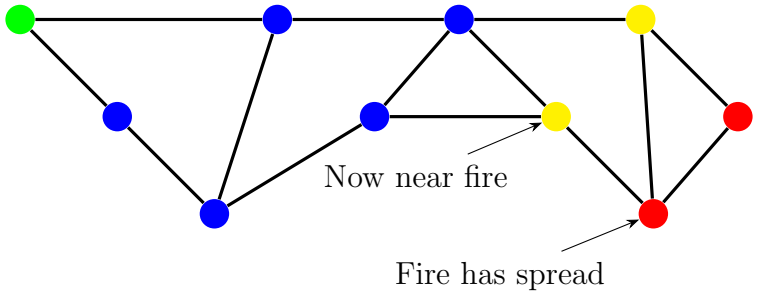
Problem Statement

- Design, implement, and test a simulation system for a wireless sensor network application in which mobile agents discover and monitor a forest fire
- The wireless sensor network is modeled as a planar graph
  - each vertex represents a sensor, modeled as an independent concurrent thread
  - each edge in the graph denotes communication link among two sensors, modeled as object references held by the respective sensors
- Mobile agents are modeled as independent concurrent threads that can traverse the sensor network from one sensor to one of its neighbors
- Mobile agents have the ability to copy/clone themselves
- Fire conditions are modeled as states of the sensor nodes
- Fire will spread along the same connections that are used for communication.
- The initial condition of the simulation will be given in a configuration file.



Sensor Network and Fire Behavior

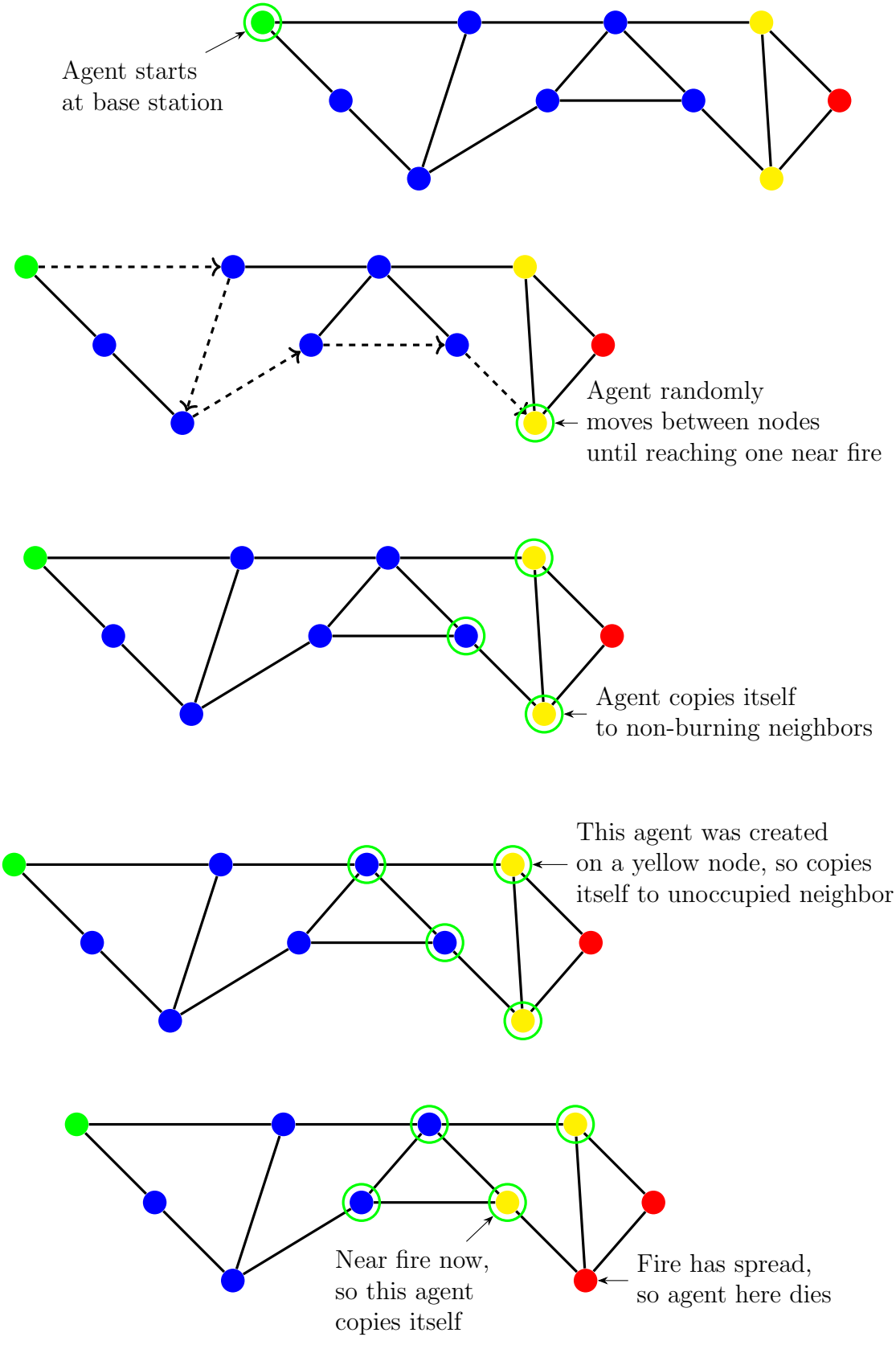
- A sensor/node can communicate only with its neighbors (and the agent currently on the node, if there is one present)
- Red nodes are on fire and can no longer support agents or communication.
- Yellow nodes surround the destroyed red nodes and indicate proximity to the fire.
- The fire starts at one location and spreads slowly.
- Yellow nodes can turn red and their blue neighbors should turn yellow.
- When a node turns red, it is allowed to communicate one last time with its live neighbors



Mobile Agent Behavior

- Initially, a single mobile agent is present in the system at some distance from the fire on a base station.
- The mobile agent performs a random walk searching for yellow nodes.
- Once a yellow node is discovered, the agent copies itself on all its yellow and blue neighbors trying to create a ring surrounding the fire. The agents are merely monitoring the fire, not actually stopping it.
- No two agents can coexist on the same node, thus agents cannot be created on nodes that already have a local agent.
- Agents communicate only with the sensor which they are visiting.

- Agents on a red fire node will be destroyed.



Base Station Log

- Keep a log of all the created agents at the base station, with each entry containing:
  - unique id of the agent
  - location where it was created
- Since new agents created near the fire, this log would allow firefighters to be dispatched from the base station to the appropriate area of the forest. In your simulation, there are no firefighters, so the fire will just keep spreading.
- It is allowed to develop a message routing structure before the first agent is created. That is, you might choose to use send messages between nodes as part of setup to let them discover their connections and build a map. This *does not* mean that you can just initial nodes with initial knowledge of the the whole map.[1]
- *Do not* use any global resources (e.g., static fields or methods) for communication, unique agent naming, etc.
- You must somehow include the log in your program display.

Configuration File Format

- Text file with each line giving information about a node, edge, base station, or fire location. These lines can occur in any order, so you can interleave node and edge info, put the station location at the beginning or end of the file, etc.
- Node line starts with "node" and then is followed by two numbers representing the x and y coordinates of the node
- Edge line starts with "edge" and then is followed by four numbers representing the coordinates of the endpoints (x and y of first point, then x and y of second). Endpoints should be at locations of nodes specified somewhere in the file, though the nodes may be specified after the edge.
- Base station info starts with "station" and is followed two numbers representing the x and y coordinates of the node the station is on. This should be the location of a node specified somewhere in the file. Our simulation assumes there will be only one base station, so you may decide how to handle extra station lines.
- Fire location info starts with "fire" and is followed two numbers representing the x and y coordinates of the node that is initially on fire. This should be the location of a node specified somewhere in the file. Our simulation assumes there will be only one initial fire location, so you may decide how to handle extra fire lines. (Maybe you'd like to start a simulation with more fire for testing purposes?)

[1] Yes, you do have a global configuration read from a file, but in the simulation the nodes only know of their immediate neighbors until they are told about locations over the next hill.

- Choices you make when parsing the configuration file should be documented in your readme file.

Example Configuration

The following configuration would produce the graph used in this document.

```
node 0 0
node 3 0
node 5 0
node 7 0
node 8 1
node 1 1
node 2 2
node 4 1
node 6 1
node 7 2
edge 0 0 3 0
edge 1 1 0 0
edge 1 1 2 2
edge 2 2 3 0
edge 2 2 4 1
edge 3 0 5 0
edge 5 0 4 1
edge 5 0 6 1
edge 5 0 7 0
edge 4 1 6 1
edge 6 1 7 2
edge 7 2 7 0
edge 7 0 8 1
edge 8 1 7 2
station 0 0
fire 8 1
```

Program GUI

- Program initially reads configuration file to set up simulation. This file may be specified on the command line, or by prompting the user somehow when the program starts. (Document whatever you choose in your readme file.

- Display of simulation graph should have edge connections, status of nodes, location of base station and agents clearly displayed and updated as the simulation progresses.

- Simulation should be displayed at human speed, not just jump from initial configuration to world on fire. That said, your simulation logic should not depend upon timing induced by making threads sleep. (So, okay to sleep to slow down enough for humans

to see, but you should have more or less the same simulation result if you turn off the the sleeping.)

- Don't forget to include the base station log!