# The slime mold : A decentralized model for a decentralized entity

Arberet, Antonin
antonin.arberet@etu.sorbonne-universite.fr

## ABSTRACT

Mobile Ad-hoc Network is a powerful solution to quickly create a Network in an area where we can't get access to a classical one. They can be used to provide communications to rescuers after a natural disaster or to troops during military operations in enemy territory. In that kind of network, we wish to minimize the total length of the used connections, and to maximize the robustness of the network to disconnections. During this internship, we worked on a decentralized algorithm for communications in MANET inspired by the slime mold, a living being able to create a network between interest points and to optimize it very quickly. We are presenting here our model and the partial results that have been obtained.

## 1. INTRODUCTION

In one hand, the physarum polycephalum is myxomycete species. More known in France as "blob" or as "slime mold" in English-Speaking countries, this living being is able to extend its body to create and optimize a network between points it's interested in. Moreover, it can adapt this network if a point isn't interesting anymore or if the slime mold is cut in half and separated.
In the other hand, mobile ad-hoc networks (MANET) are networks in which every router is mobile, and connections between them can be interrupted or created during the exploitation of the network.
I have completed an internship at the LIP6 at Sorbonne Universite, within the Multi-Agents System team mentored by Dr. Cedric Herpson on a multi-agent model inspired by the slime mold to optimize a MANET.
The initial goals were to understand the behavior of the slime mold, to read research papers on similar work and to understand the models proposed, to design a new decentralized model, and to implement and test it.
We actually worked more on the model of the slim mold behavior (to get it as close as possible to the real one), than on the application of the slime mold behavior to the MANET due to lack of time.

## 2. CONTEXT

### 2.1 Slime mold

The slime mold is a very special species. Its classification is still debated today, but it is neither an animal, nor a plant nor a fungus. It is a monocellular organism but polynuclear so its cell has many cores.
Many scientists are very interested in this living being because of the intelligence it shows through its behavior considering the simplicity of its organism. Here, we will only describe the things that the slime mold is able to do which are useful for our problem, but that's just a little part of its impressive abilities and many others are studied by biologists or ethologists. [2]

**Figure 1: A slime mold in its natural environment - Wikipedia**

.

#### 2.1.1 Movement

This cell is composed of a membrane in which we can find the protoplasm, the liquid that can carry nutriments and allows the slime mold to move. In some locations of the slime mold cell, outgrowths can appear, we call them pseudopodium.
The slime mold will successively gather or disperse the protoplasm in the cell, creating a cycle of expansion and retraction of the pseudopodium. If the blob tries to go forward in the direction of a pseudopodium, the expansion will be stronger than the retraction. This process allows the slime mold to move at a speed between 1 centimeter per hour in a normal state and 4 centimeter per hour if it is starving.

#### 2.1.2 Feeding and network optimization

The main goal of the slime mold is to find and eat food. In the laboratory, it is fed with oat flakes. It will first use the movement as described before to explore its environment, leaving behind its mucus to keep a trace of the place it already explored.
If it does not find any food, it will retract its pseudopodium and just move and try somewhere else. Conversely, if it finds something it can eat the slime mold will expend on it and consume it. But meanwhile, the slime mold keeps exploring the environment around the food location in order to find more food. If it does, it will keep a connection between the different locations and retract useless pseudopodiums.
This behavior is very interesting for us because the network created is very optimized and robust: Toshiyuki Nakagaki has reproduced the map of the region of Tokyo, replacing each big city by an oat flake. [6] A slime mold has been placed at the location of Tokyo. After exploring the map, the slime mold has established a network which was evaluated better than the rail network of the region : the sum of the lengths of the connections was very low considering the robustness of the network (i.e. the ability to stay efficient despite the loss of a connection).

#### 2.1.3 Separation and fusion

The slime mold has another ability which is very interesting for MANET : it can be split and reassembled.
In one hand, if the slime mold is cut in two parts and separated, both parts will let protoplasm flow out a very short time. Then something very similar to the blood coagulation happens and the slime mold just recreates its membrane and keeps living in two different places.
In the other hand, if those two parts are placed in the same environment again and they enter in physical contact, they will open their membranes at the contact point to fuse and become one only cell again. Note that it's also possible for two different slime molds to fuse if they are genetically close enough.
Of course, the fusion and the splitting of the slime mold causes a new optimization of the network.
That behavior is exactly the kind that we would like a MANET to adopt in case of disconnection and reconnection of routers.

### 2.2 MANET

The term "mobile ad hoc network" designates a network composed of mobile routers. Due to the mobility of the routers, each one of them can lose a connection with a neighbour at any time if it gets out of communication range, or create a new connection as well.
That kind of network is mostly used in military operations where we try to recreate a temporary network in a place there isn't any, and in case of natural disasters to replace the damaged network and to communicate with rescue teams.
The routers can be carried by cars or by drones, or even be smartphones in some cases.
The mobility aspect of the network can be the source of problems: the way a router acts when it loses or gains a connection, how this information is processed in the network of course, but also how to recreate the routing tables at each node quickly enough to recover after a disconnection and how to take into account the physical distance between two nodes, since MANET are composed of wireless routers and distance has a huge impact on bandwidth.
Typically, we want a MANET to be able to route a message between the routers using a minimal number of hop and distance of travel (since the routers are wireless and the bandwidth is very impacted by the distance between two routers), and we also want it to be robust to disconnections. [7]

### 2.3 The slime mold for MANET so far

The slime mold has already inspired researchers who were working on networks. Toshiyuki Nakagaki, with his Tokyo rail experimentation, proposed a mathematical model to simulate the slime mold behavior for networks, and to compute the values which need to be continuously updated like the flow between two nodes or the pressure in one of them. This model has been modified and used by another team of researchers [5] and used in a MANET, but a specific one. In their case, they worked on wireless sensor networks (WSN), where each node of the network can be a simple router, a sensor (able to route too), or a sink. The sensors are producing information that needs to be sent to a sink. Their model is based on an analogy between information and food for the slime mold: sensors create food and a sink consumes it. Nodes will create connections with their neighbors and update the diameter of these connections using the mathematical model. Diameters are used in the routing step when a message must be forwarded.
In this case, the network is not symmetric and the information can flow naturally from sensors to sinks and is centralized around the sinks.

### 2.4 The position of our work

We would like to present a new multi-agents model for MANET inspired by slime mold in a decentralized way. Each node will be controlled by an agent, communicating with the others, and have the same propriety at the beginning. Each node can be provided with food and will share it with its neighbours.
The network should be able to keep working if a node is disconnected, no matter which one. It should also be able to exist in two separated parts disconnected from each other, and to fuse when a connection is created.
Since we are working with agents, the model must handle asynchronous communications. The idea is that our network should be closer to the real behavior of the slime mold, and should be able to manage connections in any MANET.

## 3. REALIZATION

## 3.1 A decentralized multi-agent model

### *3.1.1 Overview of the model*

Our model is mainly inspired by the work realized with WSN, but adapted to the constraints we had. In our network, each node represents a spot of the slime mold able to create connections with other ones. Each node has an agent associated, which is in communication with other agents. Together, they will reproduce the slime mold behavior.
Two modes are available for our slime mold : in the first one, each node is placed and can be moved. The number of agents is static. The second one is similar but each agent will be able to explore the environment: it will create a new node and a new agent on this node, to expend the network as the slime mold expands itself.
Our agents have a main behavior composed by different states in which they pass through, on and on, in a cyclical way. We will call a cycle an execution of the full following process :
The agent will wait for updates about its neighbours state. Each agent can find food. Three modes exist to give food to agents:

- Random mode: each node has a probability to receive food at each cycle.
- Static food mode: some nodes are created as food sources and will always be provided in food.
- Food in the environment: food zones are created in the environment. Each zone contains a given amount of food. The agent on those zones will be able to pick food if there is still food in it.

If it does, the agent will share it with close agents, to mimic the propagation of the protoplasm inside the slime mold. The quantity of food picked depends on the agent and on its neighbours needs. The agent receiving the food will share it the same way.
An agent will process the diameter of the connections with its neighbour depending of the food quantity transiting between both, and of the distance.
Each agent consumes an amount of food at each cycle of the main behavior.
The agent is also sending updates about its state often enough to ensure its neighbours to receive a new one before the end of each update phase.
We propose two ways for the creation of the slime mold networks : the static mode in which every node are already placed and in which the user may move them in the space and they will recreate connections with close nodes; the other mode which is supposed to reproduce the exploration behavior of the slime mold: if a node is fed enough and its neighbours too, it has a probability to create new neighbors in its communication reach. This way, the network uses the food to explore the environment.

### *3.1.2 Mathematical slim mold*

The mathematical model is based on the one proposed by the team which worked on WSN. But since our virutal slime mold is not synchronous, the pressure model that they used can not fit. Processing with pressure that should have been updated creates imbalance of the general pressure in the network and results in explosion of the node pressure.

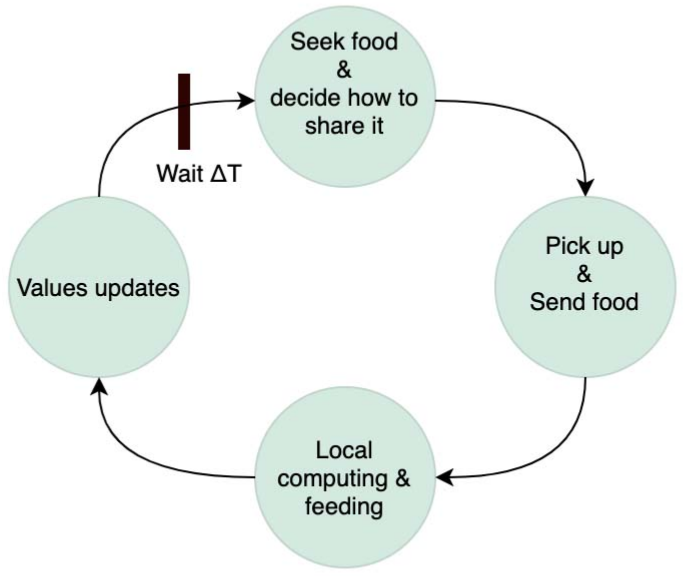


**Figure 2: Agent main behavior process**

Consequently, we decided to use the food to replace it, because we calculate the food amount depending on others and the way we use it guarantees us having at most a bit of exceeding quantity of food at some point on some nodes.

#### *3.1.2.1 Food.*

Each agent i owns a food quantity $F_i$. The goal for the agent is to reach the food bound $B$. $n_i$ is the need of the agent i:

$$n_i = B - F_i \tag{1}$$

At each cycle, the agent i picks a quantity of food $p$ defined by the available quantity of food $a$, the pick capacity $c$ which is the amount of food the agent is able to pick in one cycle, and the neighbours needs:

$$p = min(\{a, c, \sum_{j \in neighbours(i)} n_j + n_i\}) \tag{2}$$

$\overline{F}$ is defined as the mean of the needs of the agent and its neighbours:

$$\overline{F} = \frac{\sum_{j \in neighbours(i)} F_j + F_i}{|neighbours(i) + 1|} \tag{3}$$

The quantity kept by the agent $F_i^k$ is calculated to make $F_i$ tense toward the $\overline{F}$, but without giving more than a proportion $k$ of $F_i$:

$$F_i^k = min(\{min(\{\overline{F}, F_i + p\}), ((F_i + pickup)k)\}) \tag{4}$$

Obviously, the part given to the neighbourhood $F_i^g$ is defined as following. It will be split between neighbors proportionally to their needs:

$$F_i^g = F_i + p - F_i^k \tag{5}$$

$$\forall j \in neighbours(i) F_{ij} = F_i^g \frac{n_j}{\sum_{j \in neighbours(i)} nj} \tag{6}$$

#### 3.1.2.2 *Connections diameters.*

$q_{ij}$ is the quantity of food passing through the connection between $i$ and $j$. Note that the way to estimate this value is based on the WSN model based on pressure processing. Other models more adapted to the situation could be imagined.

$$q_{ij} = \frac{D_{ij}}{L_{ij}}(F_i - F_j) \tag{7}$$

We can calculate the new value of $D_{ij}$ based on the current one and $q_{ij}$.

$$g(q) = rD_{max}\frac{\alpha|q|^{\mu}}{1+\alpha|q|^{\mu}} \tag{8}$$

$$D_{ij}^{n+1} = \frac{D_{ij}^{n} + g(|q_{ij}^{n}|)\Delta_t}{1+r\Delta_t} \tag{9}$$

According to the work on the WSNs, the parameter $\alpha$ has a little impact on the routes selections, while $\mu$ impacts directly the trade off between robustness and efficiency: $0 < \mu < 1$ for robust path, $1 < \mu$ for efficient single path.

### 3.1.3 *Communications*

In this model, we assume that agents are able to broadcast messages to every agent closer than a given communication range, or to send a message to a given agent or group of agents if needed.

#### 3.1.3.1 *Initialization: ad broadcasting.*

At the begining, the agents broadcast an ad package. It contains the sender ID, two coordinates indicating the position of the sender, the quantity of food the sender currently own, the number of hops made by the package and the list of the agents who forwarded it. Those packages are rebroadcasted by receivers, and are supposed to be received at least once by every reachable agent in the network. Of course, if an agent receives a package it already processed, it will not be forwarded again. The purpose of those packages is to declare who in the network initializes a routing table.

#### 3.1.3.2 *Neighbors values update and food sharing.*

After the initialization step, each agent starts to seek for food and, if it finds it, it shares it with its neighbors. In order to do so, agents are broadcasting state packages. They contain the position of the sender and the food it owns and are used to allow neighbors to update their knowledge about the sender before the food sharing step. Those packages are sent at each $\delta_{Tsync}$ period.
State packages are not rebroadcast since non-neighbors agents do not need this information. Moreover, they are used as ping packages: if an agent does not receive state package of a neighbor during a given period (depending on $\delta_{Tsync}$), it will consider this agent as lost and reciprocally a state package of an unknown agent means that a new neighbor has arrived.
An agent that wants to share some food will wait a given time for neighbors updates, and decide how to share the food as seen in the previous part. It will then send a food package at each food receiver, that contains the amount of food.

#### 3.1.3.3 *Network update.*

As explained before, one of the main interest of the slime mold for MANET is its ability to recover from a cut connection in its network. The mathematical model is supposed to allow a fast recalculation of the connection diameter as the slime mold redirects its flow to smaller connections when a big one is lost.
In order to do so, any agent will send packages on a modification of its neighborhood:

- If a neighbor is found, the agent will send a new ad package as seen in the initialization part. This package will be forwarded in the whole network to allow other agents to detect a potential new path to reach the sender which can be better than the former ones.

- If a neighbor is lost, the agent will send a connection lost package containing the ID of the sender and the ID of the lost agent. The receiver of this package will forward it if the lost connection impacts their ability to reach the lost agent, (i.e. if the sender was the only way for the agent to reach the lost agent). The sender also sends an ad package to allow any agent which don't know how to reach it anymore to find a new path.

## 3.2 Implementation

The implementation made during this internship is based on the Cedric Herpson's Dedale Environment [3] which use JAVA Agent DEvelopment Framework (JADE) [4] for the agent programming.
The network formed by agents is handled as a graph, and as the Dedale environment allows it, we use GraphStream [1] to handle this graph.

### 3.2.1 *Agents implementation*

#### 3.2.1.1 *Agents and knowledge.*

We only use one type of agent in this project: the blob agent. It's implemented in the BlobAgent class which is only an instatiable version of AbstractBlobAgent. Each agent is representing a part of the slime mold, communicating with close agents to reproduce the way the slime mold is creating its network and expending.
In addition to handle the execution of the behaviors, this class is containing every parameter given to the agent in the ConfigurationFile such as the reference of the node the agent is associated to or the parameters of the mathematical model, but also tables the agent keeps up to date:

- nTab: Neighbors table, used to keep information sent by neighbors the agent will need during the processing step.

- lastContact: Used to keep the date of the last package received from an agent in order to detect a missing neighbor. Note that if a neighbor is missing, it is removed form nTab but not from lastContact

- routingTab: supposed to be used to forward packages, not really used in the current version of the project.

The tables entries are encapsulated in classes you can find in the knowledge package.

#### *3.2.1.2 behaviors.*

In JADE, each behavior of the agent is a class. We can split our agents behaviors in two categories: the behavior used in the main process (i.e. the process described in the model overview) and the behaviors instanced after the reception of a package to process its content.
The main process behaviors are described below. See the figure 3 for their organization.

- AdBroadcastingBehavior: Waits a random time, lower than a given bound then sends the first ad package at initialization.
- ParallelBlobBehavior: Composite behavior, allows to execute BlobingBehavior and SyncBlobBehavior in two separated thread.
- BlobingBehavior: Handles the local computing, seeks for food and decides how to share it using the mathematical model and decides if the agent will create a new node and a new agent to explore the environment.
- SyncBlobBehavior: Sends a state package every $\delta_{Tsync}$ period. Needs to be executed in a separated thread to avoid desynchronization.
- ReceiveMessageBehavior: First step in the processing of a new message, checks the type of the message and creates a behavior to process it. This behavior stays blocked until a message reception.

The behaviors that ReceiveMessageBehavior instances are AdProcessingBehavior, CoLostProcessingBehavior or FoodProcessingBehavior and StateProcessingBehavior depending on the message type.

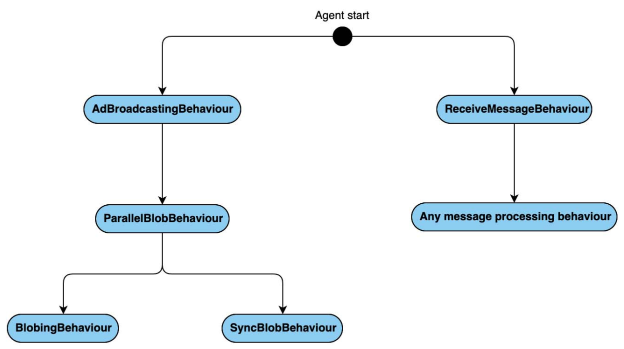


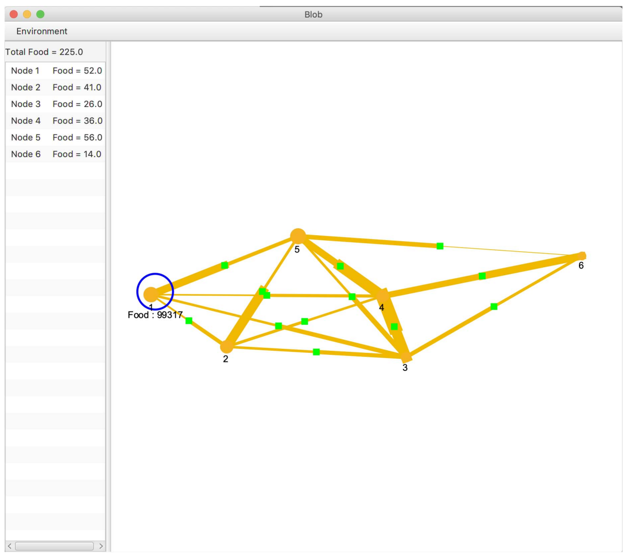
**Figure 3: behaviors creation hierarchy**

#### *3.2.1.3 Packages.*

The packages sent by agents are represented by simple serializable classes containing values and getters for them. You may find them in the msgContent package.

### *3.2.2 GUI*

A graphical user interface was necessary to allow the user to visualize the slime mold on screen, and to compare the way it expands and optimizes its connections. We made it from the GUI of the Dedale project, which is based on the JavaFX implementation of the GraphStream GUI.
This GUI displays the slime mold (i.e. the network) as a graph in which the nodes are the agents nodes and the connections between agent are edges.
The size of a node is proportional to the food quantity the agent owns. Note that you can find the exact quantity of food of each node in the column on the left of the window.
Each communication edge displays the connection state using the form on the middle of it: a green square is a connection established in both directions between the agents, a red one is a unidirectional connection and a black cross is a connection between two agents not in communication range anymore.
The thickness of each half of an edge is proportional to the diameter of the connection as processed locally by the agent on this side of the edge.
Note that every pair of node in communication reach of each other will create a connection which is displayed even if the diameter is null.
If the mode of execution allows the slime mold to find food in its environment, the food spot will be displayed as a blue circle, with the quantity of food on a label under it.
The user is also able to move the nodes in the space with a simple drag and drop in order to make modifications in the network and to see the impact of those on the slime mold.



Total Food = 225.0

| | |
|---|---|
| Node 1 | Food = 52.0 |
| Node 2 | Food = 41.0 |
| Node 3 | Food = 26.0 |
| Node 4 | Food = 36.0 |
| Node 5 | Food = 56.0 |
| Node 6 | Food = 14.0 |

**Figure 4: GUI**

## 4. EXPERIMENTAL EVALUATION

For the experimental evaluation, we were supposed to compare the behavior of the real slime mold to the virtual one. Sadly, our slime mold culture has not been fruitful probably due to the summer conditions and to the quality of the scletortials (which are a state of the slime where it is dehydrated and inactive which is used to ship them).
Here we can only present the experimental setup and give the results of observations on the algorithm.

### 4.1 Experimental setup

The comparison between the real slime mold and the virtual one is based on a particular instance of the network that you can find on figure 5.
On one hand, the virtual slime mold will have every node

already placed, and will have to optimize connections while it shares food placed on different nodes.
On the other hand, the real slime mold will evolve in a 3D printed labyrinth which is reproducing the constraints of the virtual instance: the slime mold can explore corridors which are matching to the edges of communication of the virtual instance, and to nodes are corridors ends. Note that two nodes in communication reach must be linked by a direct corridor in the labyrinth. The slime mold will reach the food on the node, create a network and optimize it.
Since the slime mold can not move on plastic, the labyrinth bottom must be covered with a couple millimeters of agar substrate.
We can compare both networks and see if the same connections are the most used.
Positioning food on each node should give interesting results for a comparison since the algorithm will try to feed each node whereas the real slime mold will probably not explore isolated node if there is no food on it.

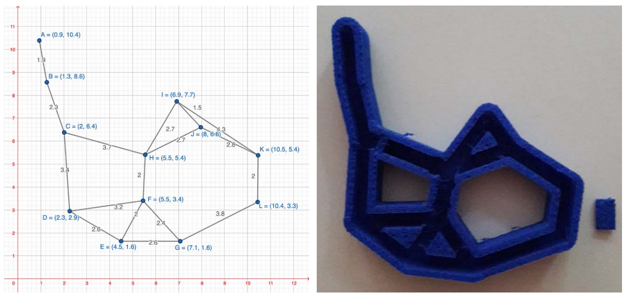


**Figure 5: Map of the instance for the comparison and 3D printed labyrinth (Communication reach = 4)**

We also want to test the ability of the algorithm to adapt its network after a disconnection. The GUI allows the user to move nodes and the connections will be lost if they get out of communication reach. To simulate a disconnection with the virtual slime mold is trivial, but we need to recreate the same disconnection on the real one.
In order to do it, we added doors to the labyrinth (you can see one on the figure 5). Once the doors are positioned, the access to some of the connections is forbidden to the slime mold. In our instance we can put the doors to block the C-H and F-H connections.
The slime mold can be grown in the restricted labyrinth as well as it can be in the complete one and cut at the time to put the doors. If needed, the corridor can be filed with salt, which is a slime mold repulsive.
This experiment is supposed to give a good comparison support for the disconnection case.

### 4.2 Observations

This version of the algorithm seems to be the closest to the reality beneath the versions we tested during this internship. The way of sending the food insures a conservation of the total quantity of food in the network (except for consumed food) unlike other models experimented. Moreover, the agents sharing their food with their neighbors causes a flow of food in the network even in if two nodes (one with a starving agent and one with food) are far from one another.
On the contrary, the edges sizes are not very meaningful since the way to calculate them is based on the difference of food, as it was based on pressure difference in the WSN model: in the case of two nodes fully fed and in communication reach of each other, the connection will be small with the virtual slime mold, but will not be with the real slime mold.

## 5. PERSPECTIVES AND CONCLUSION

Since we could not perform it, the first thing to do is obviously the experimentation proposed. If the model happens to be interesting, a new way to calculate communication diameters should be a good improvement for the model.
There is still a lot of work to do on the network part too: how to forward the message taking count of the communications diameters and how to manage the routing table locally in each agent (maybe a trade-off between the number of hops and the diameters) and probably other problems which are going to appear at this moment.

## 6. REFERENCES

[1] Graphstream. http://graphstream-project.org/. Online; consulted on the 2019/08/29.
[2] Audrey Dussoutour. *Tout ce que vous avez toujours voulu savoir sur le blob sans jamais oser le demander*. Des Equateurs Eds.
[3] Cedric Herpson. Dedale. https://dedale.gitlab.io/. Online; consulted on the 2019/08/29.
[4] Telecom Italia. Jade. https://jade.tilab.com/. Online; consulted on the 2019/08/29.
[5] Ke Li, Kyle Thomas, Louis Rossi, and Chien-Chung Shen. Slime mold inspired protocol for wireless sensor networks. pages 319–328, 10 2008.
[6] Atsushi Tero, Seiji Takagi, Tetsu Saigusa, Kentaro Ito, Daniel Bebber, Mark Fricker, Kenji Yumiki, Ryo Kobayashi, and Toshiyuki Nakagaki. Rules for biologically inspired adaptive network design. *Science (New York, N.Y.)*, 327:439–42, 01 2010.
[7] Wikipedia. Manet. https://en.wikipedia.org/wiki/Mobile$_{a}d_{h}oc_{n}etwork.Online; consulted$

## Thanks

I would like to thank Cedric Herpson, my mentor during this internship, for this very original topic and for the help he gave me during the whole process and the multi-agent team for its warm welcome.