Program Design

Lakshmi Narasimha Guptha Munuhur Rajagopal

Salient Features

1. The grid is one dimensional and the size of the grid is flexible
2. The pigs are arranged in a ring, with each pig able to talk to only 2 of its neighbors directly
3. The time taken for the bird to land is randomized at each run. The maximum value for the uniform distribution can be configured
4. 2 pigs cannot occupy the same position. Pigs cannot climb walls
5. Each movement of the pigs is not transmitted to neighbors all the time
6. Walls can knock over pigs, pigs can knock over other pigs, but pigs can't knock over walls in a cascading manner. I found that having pigs knock over walls completely overshadowed any other observations in a tightly packed world. It was like all the pigs die all the time when the total number of grid positions is very small, unless the bird just happens to land on an empty position
7. UDP messages are used for communication. The system is somewhat resilient to the loss of packets –if a pig does not know its position or is not able to communicate with neighbors, it is still able to respond to enquiries about whether it has been hit or not based on whether the bird was actually targeting it
8. The initial position message is transmitted from the closest pig in 2 directions, with hop counts ensuring that there are no infinite loops.
9. The status messages proceed in a single direction, with the reply coming back along the same path. The above two approaches demonstrate 2 different ways of communication – one faster, but the other less wasteful
10. Pigs warn their physical neighbors in case they are affected and only if they have the ability to move. This eliminates futile messages
11. A uniform hop delay of 1 s is assumed for the initial position messages. The status messages were not delayed because the bird land message, which is unicast to each pig from the coordinator, finalizes the state of the pigs and the results are not going to change anymore. So, any delay for the status messages is not going to make any difference
12. A UDP socket wrapper class called PracticalSockets has been used, which was released under the GNU General Public License
13. Finally, the programming language used is C++ 11. Specifically, vectors, mutexes and threads are used from C++. The other programming follows C's procedural style. I found this style more suited than the object oriented paradigm. Things are pretty much event based – the pigs do stuff only in response to messages

Discussion

A coordinator process takes responsibility of spawning the pigs and performing the tasks of the bird. First, the pigs are spawned. Then, the coordinator process computes the locations of the pigs, walls and the landing site of the bird. This information is encapsulated in a message and transmitted to the closest pig. The closest pig transmits this information via the p2p network.

Pigs that receive this message check to see if they are affected by the bird. If they are, they try to move to a safe place. A design decision adopted is that a block cannot be occupied by 2 beings – 2 pigs cannot be on the same spot and a pig cannot be on a wall. Also, each wall has a standard height of 3, which dictates how far it affects its surroundings when it falls. Walls and pigs, when hit, affect their surroundings in both directions. Pigs also signal their neighbors in case they are affected. This is very selective – this happens only when a pig is going to affect another pig and the second pig can do something about it.

Each movement of a pig is not transmitted all the time. In other words, pigs know their surroundings only through the standard p2p messages. So, a pig thinks that a neighbor is still in its original spot even though it has moved. The reason for this is that there is limited time before the bird hits and it is infeasible to expect the pigs to notify other pigs of their every move and act on that information.

At a time of the coordinator's choosing, it informs all the pigs that the bird has landed. This message contains information that tells the pigs whether they have been hit, even if they have not yet received the initial messages. The coordinator then asks the closest pig for the status. The closest pig queries the other pigs about whether they have been hit, and upon receiving the information, updates the results back to the coordinator.

The coordinator keeps score, and starts off the next iteration of the game.

Flaws

1. The biggest problem with the assumption that pigs cannot occupy the same location is that it is difficult to keep the pigs updated about each other's locations all the time. So, pigs just know their neighbors' initial positions and decide whether they can move or not based on that. For example, let us assume 3 pigs A, B and C at locations 4, 5 and 6 on the grid. Now, pig B will not move from its location even if A has moved left and C has moved right. The other consequence of this issue is as follows – if A and B are adjacent to each other, the bird is estimated to land on A and A can move out of the way, but B cannot, B will not be able to know that A has moved out of the way. A way of looking at this is that the bird has winged B even though it didn't fall on B directly. This problem could be surmounted by the pigs updating the others about their locations all the time, and this takes time to implement. As I am doing this project alone, I couldn't find the time to implement this.

2. In some cases, if the request for status comes through before the pigs know about their neighbors, the run has to be aborted. This problem should be solved with a future version of the program with proper happens-before. So, if we are able to establish a proper happens-before relation such that we know when the pigs learn about their neighbors, this problem will go away. One more solution is to have 2 different set of messages – one informing the pigs about their neighbors and another which tells the pigs about the position of the bird. So, the request for status can be generated once we know that the pigs know about their neighbors. Right now, the number of runs is not incremented in the aborted case. This becomes a problem only because the status messages have to propagate through the p2p network, and they can't do this if they don't know their neighbors

Mappings from Interfaces to Messages

1. Bird_Approaching - INIT_POSN_MSG
2. Take_Shelter - INFORM_PHY_NBR_MSG
3. Status(pig_ID) - STATUS_REQ_UNI_MSG
4. Status_all - STATUS_REQ_MUL_MSG
5. Was_Hit - WAS_HIT_*_MSG
6. The bird land message is unicast from the coordinator to all the pigs – BIRD_LAND_MSG

Usage Instructions

For a default run, just run "bin/coord" after running "make". This spawns the pigs automatically.

If you want to adjust any of the macros, change them and run "make". Make timestamps suck, so it is always better to "make clean" and then "make." The process is persistent, so to kill, just "chmod 777" the file called kill and run "./kill" All commands have to be given in the birdypigs directory