# Program Design

**Lakshmi Narasimha Guptha Munuhur Rajagopal**

I have used a custom leader election mechanism and Lamport's clocks for this lab.

## Leader Election Algorithm

My program uses a modified version of the Designated Forwarder election process used in RFC 5015 (Bidirectional PIM). The pig which is closest to the bird wins the election. As the positions of the pigs change with every bird launch, there is a new leader each time.

When the pigs are spawned for the first time, they wait for 1 s for the other pigs to get spawned and for the topology to settle down. Then, they choose a random position on the grid and advertise their position to all other pigs in the form of Offer messages. This is possible because the pigs know the process IDs of all the other pigs. They wait for these advertisements from all the other pigs. At this stage, they have the positions of all the pigs. They then locally compute the winner based on which pig is the closest to the bird on the grid. In case of ties (which is possible because there is no limit on how many pigs can be present at one position on the grid), the pig with the highest process ID is the winner.

The process is completely multithreaded. When a message is received, a new thread is spawned to handle the message, and the pigs are message driven primarily.

## The Leader Pig

Once the leader has been elected, it sends a Winner message to the bird and expects a bird position message in return, which indicates where the bird is expected to land as well as when in the future in terms of the number of timestamps.

The leader calculates the pigs which will be impacted by the bird. This has been significantly simplified compared to the first assignment because I believe this is not the focus of this assignment. Pigs can roll over onto other pigs and this is chained, and so, a group of pigs which are bunched together will be impacted. We assume pigs get chopped into two when the bird lands, and thus, they affect both their neighbors. Walls topple only if the bird falls directly on them, and walls crumble in both directions. If there is a wall adjacent to the hit wall, the second wall crumbles partially, such that the affected locations are exactly what would have been affected even without the presence of the adjacent wall.

Once the affected pigs are calculated, the leader sends the timing information to them as a status request message and expect a status reply message indicating whether the pigs were hit or not.

Once the pigs which were affected reply back, the leader computes the score. The score basically keeps track of how many pigs were hit. The number of rounds is tracked too. This combination enables us to calculate the hit rate after a few iterations, and helps us identify how performance changes with adjustment of the various parameters. These parameters are displayed to the user at the end of every round.

Once each round is done, the leader sends out a Pass message to all the other pigs. This Pass message passes along the score to each other pig, so that if any other pig is elected as leader in the next round, it is able to keep track of the score seamlessly. After sending a Pass message, the leader starts a new election.

## The Loser Pigs

The losers of the leader election don't have a lot of things to do. They of course update their timestamps upon receipt of messages from other pigs. They also respond to status request messages. An important difference is that they don't have any interactions with the bird process.

When these pigs receive the status request message from the leader, they move their logical clocks forward in order to simulate network delays. They then check to see if their local clock has moved beyond the time during which the bird was expected to hit. If the bird has not yet been "logically" hit, they can move out of the way, and they send back a status reply message to the winner indicating that they were not hit. On the other hand, if the local clock has advanced beyond the timestamp embedded in the status request message, they send the status reply message indicating that they were hit.

Upon receipt of a Pass message, each of the loser pigs starts off their parts of the new election, and the cycle continues.

## The Bird

Contrary to what was done in the previous lab, the bird process has a very limited role. It gives out a bird position message in response to a winner message from the leader. It also maintains time and sends out a measure of its time in its messages.

## Logical Clocks

Every node, be it a bird or a pig, maintains a logical clock. Any message which is sent is piggybacked with the timestamp of the sender. The receiver uses the standard Lamport's clock mechanism to update/not update its own clock.

The time that the bird takes from its launch to its landing is sent by the bird process to the leader. The leader's clock is closely synchronized to that of the other pigs, and so, the leader just sends the expected logical timestamp at which the bird is expected to land to the other pigs.

By the time the other pigs receive the status message that contains the bird landing time; their clocks would have ticked a few ticks. We just simulate the network delay by introducing a random change in the timestamps of the other pigs. Thus, when the timestamp is received from the leader, we can just compare timestamps and find out whether it is a hit or a miss.

One quirk of the design is that the leader sends a message to itself, and that too involves a network delay. I could have just added a check upon receipt of the status message and if the recipient is the leader, I could have just not added a random "network delay" but I decided to let it be, reasoning that the leader will just take a longer time to move than the other pigs.

## The Messages

***ELECT_OFFER_MSG*** This message is used by the pigs to advertise their location to the other pigs as part of the leader election process.

***ELECT_WINNER_MSG*** This message is used by the leader to inform the bird that it has won the election.

***ELECT_PASS_MSG*** This is used by the leader to indicate that it wants to pass on the leadership responsibilities to another node. This usually happens at the end of each round.

***BIRD_POSN_MSG*** This is sent by the bird to the leader upon the receipt of a winner message.

***STATUS_REQ_MSG*** This is sent by the leader to the pigs which will be affected by the bird in this round. It contains information about the bird's location and expected time of arrival.

***STATUS_REPLY_MSG*** This is used by the loser pigs to indicate to the leader whether they have won or lost.

## Important Functions

I am not listing the send and handle functions for the messages and the message handlers as they are pretty straightforward.

**listenerFlow** In both the bird and the pigs, this function is used to listen on the socket for incoming messages. This then spawns threads for handling the messages

**processAffectedPigs** This is used by the leader. Once the bird position message has been received from the bird, the leader calculates the affected pigs and gets their status via the status request message

**pigFinishElection** This takes place in every pig. Once the location of each of the other pigs is known via incoming offer messages, this function finds out the winner locally in every pig

## Other Characteristics

1. The language used is C++ 11, but using procedural programming for the most part.
2. Both the bird and the pigs use multithreading. Any message that has been received is handled by a newly spawned thread. As most actions occur on the receipt of a message, this works out pretty well.
3. The grid is one dimensional and the size of the grid is flexible
4. When a pig receives a move indication, if it has the time, it will move to a location that is not affected by the bird. There is no possibility of it trying to move but failing to do so. This is because more than one pig can occupy the same position on the grid, and so, pigs can phase right through other pigs and walls.
5. If the pig on which the bird is supposed to land on moves out of the way, it does not mean that the neighboring pigs are automatically safe. In the toy world, we can assume that the bird sees that there are neighboring pigs, and it just expands the scope of its impact.
6. UDP messages are used for communication. A UDP socket wrapper class called PracticalSockets has been used, which is under the GNU General Public License

## Improvements

1. A more elegant clock algorithm could probably be used.
2. In the leader election, it is better for pigs to keep quiet once they hear lower positions on the channel. Later, once the leader gets elected, the loser pigs can transmit their positions to the leader. In the program, I have combined the two steps, leading to extra messages on the channel.

## Use Instructions

Go to the birdypigs2 directory and give make. This creates two executables: bin/bird and bin/pig. Run bin/bird. This automatically spawns the pigs.

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 birdypigs2 directory