A Distributed Autonomous Swarm

or

how to tear down the Control Tower

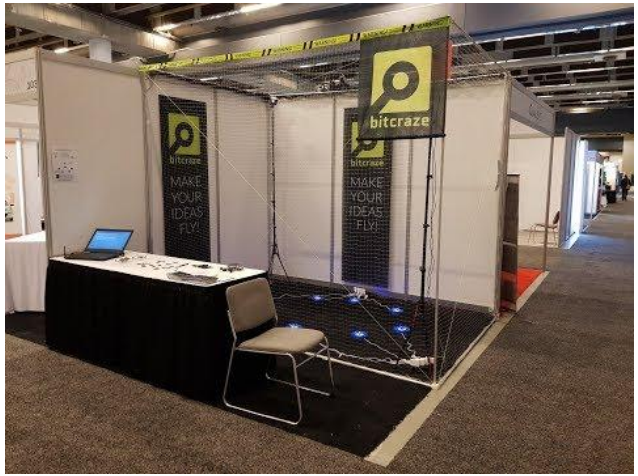
BAM days
21 October 2021
Kristoffer Richardsson

Contents

- The “old” swarm demo
- The new swarm demo
- The Pilot
  - State machine
- The Distributed Control Tower
  - Radio communication principles
  - The shared swarm state
  - Flight planning
  - Distributed consensus
  - State machine
- Live demo

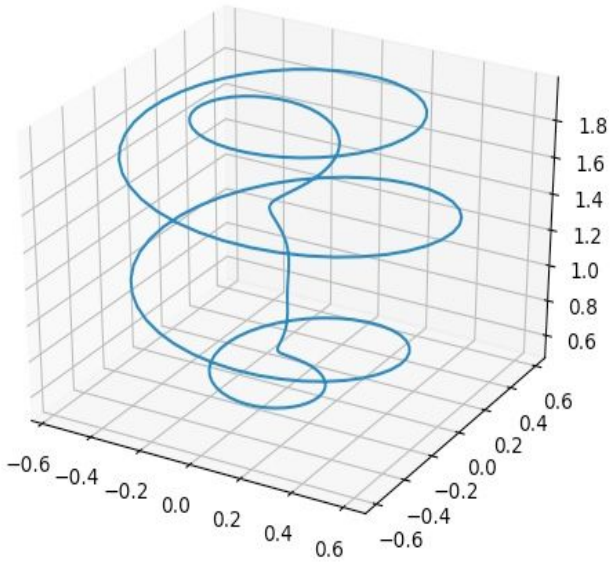
The “old” swarm demo

- Has been used since 2019 in various flavours
- Hybrid implementation
  - Crazyflies use an on-board application that manages the flight (called “the pilot”)
  - A python “control tower” runs on a PC and controls when a Crazyflie takes-off or lands



The “old” swarm demo - the Pilot

- An application on-board
- Fixed spiral trajectory
- When started it runs a deterministic trajectory
- Samples start point before take-off to be used as landing position
- Re-charge battery on charging pad
- Reposition if not charging
- Crash report to the tower



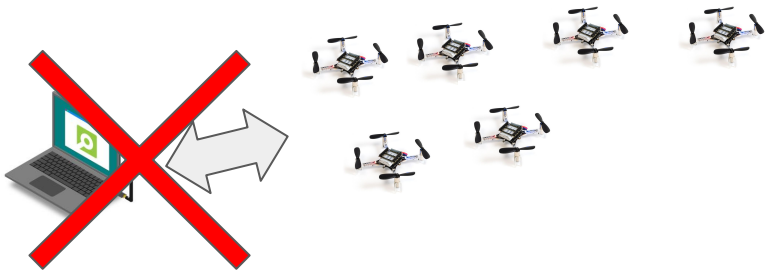
The “old” swarm demo - The Control tower

- Monitors battery levels
- Monitors who is flying (or has crashed)
- Simple commands to the pilot
  - Take-off
  - Land
- GUI
- Support for multiple Crazyflies flying at the same time
  - Slotted spiral
  - synchronized formation

The new swarm demo

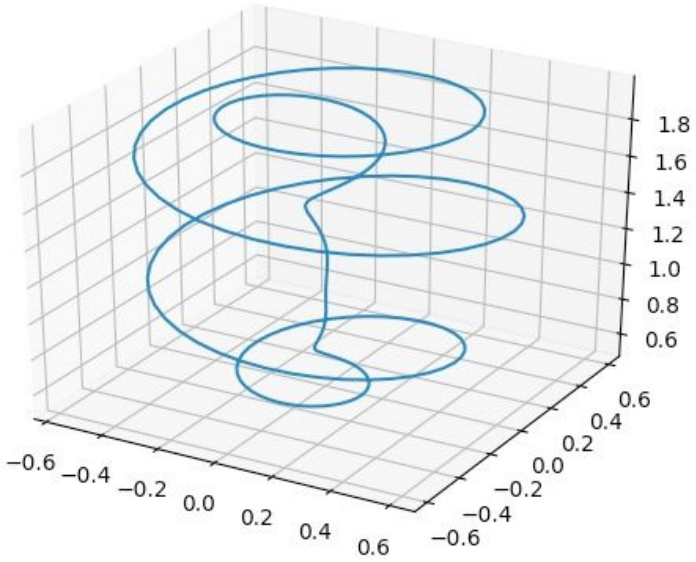
Goals:

- No central Control Tower - distributed decision of when to take off
- P2P communication between Crazyflies
- No collisions
- Slot into the spiral at the right time

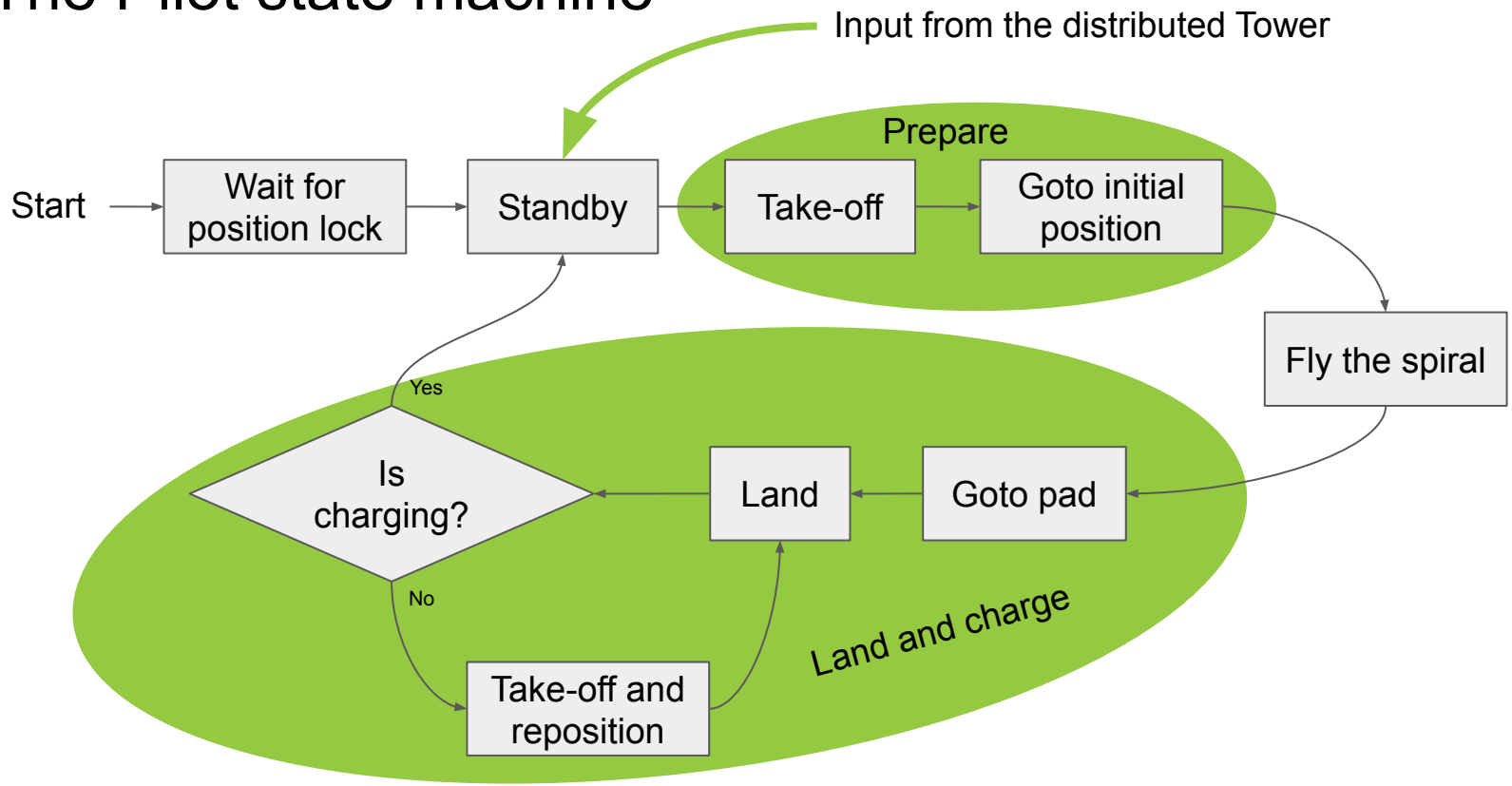


The Pilot

- Very similar to the old demo
- An application on-board
- Fixed spiral trajectory
- When started it runs a deterministic trajectory
- Samples start point before take-off to used as landing position
- Re-charge battery on charging pad
- Reposition if not charging



The Pilot state machine



The distributed control tower

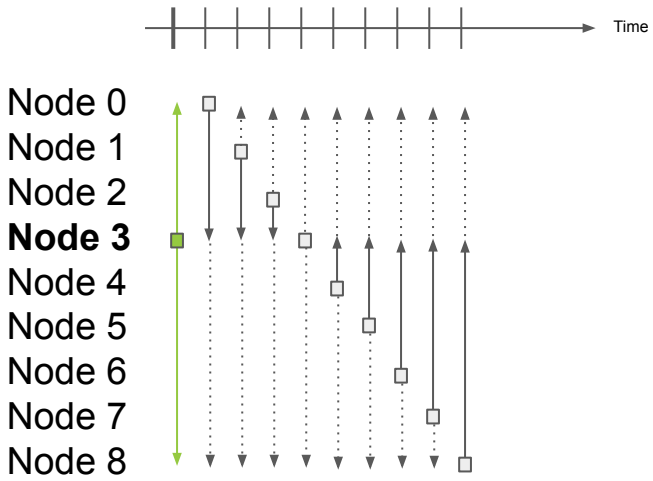
- Implemented in the Crazyflie application
- Uses P2P communication
- Keeps track of the shared swarm state
- Monitors the flight of other Crazyflies
- Monitors the battery level
- Triggers the take-off in the pilot state machine

P2P radio communication

- P2P support in the Crazyflie firmware (experimental)
- Broadcast functionality
- An application can implement unicast on the protocol level, in this application we use broadcast

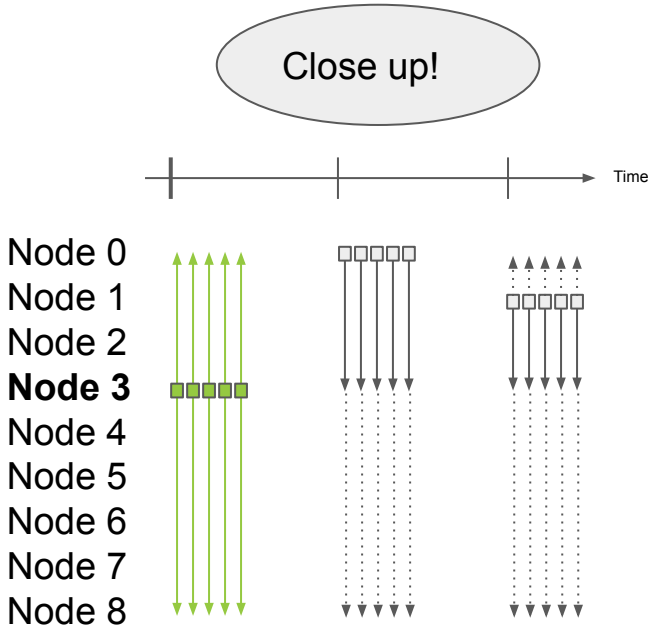
Time-slotted TX scheme

- The radio can not RX and TX at the same time
- A transaction is one request with responses from all peers
- Use time slots (20 ms) based on RX time and node id



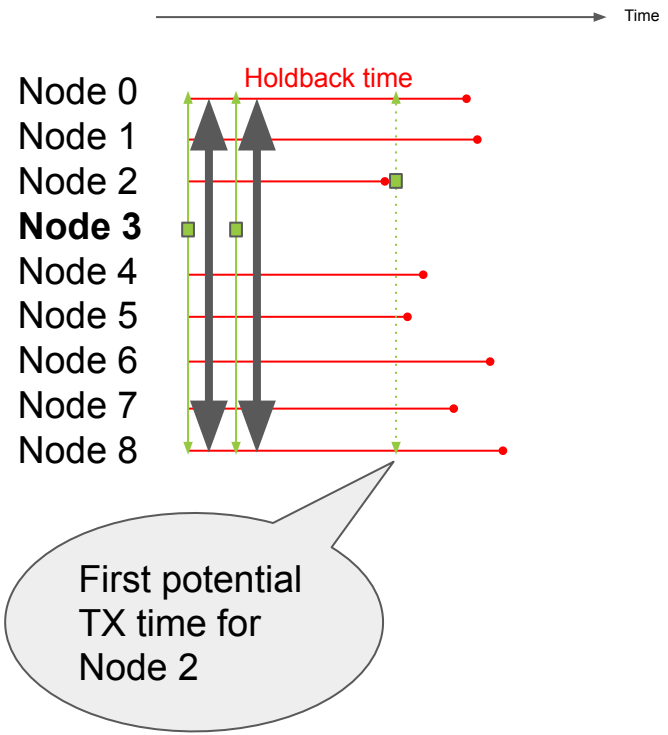
Broadcast retransmission

- Packets are lost
- Too many lost packets causes delays on higher levels
- Handled by transmitting all packets 5 times in bursts
- A sequence number in the packets is used to determine if the packet has already been received

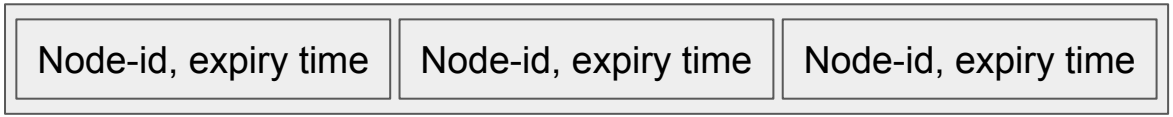


Transaction holdback

- Transactions often come in pairs
- Randomized holdback time to avoid transaction collisions



The shared swarm state

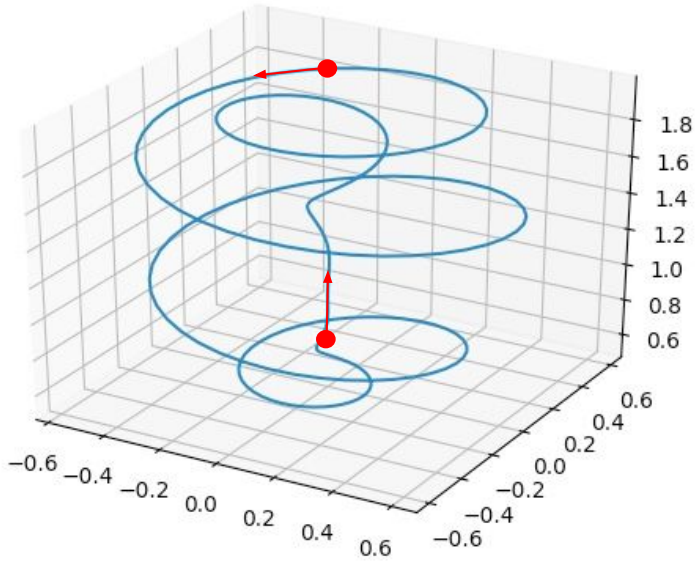


- A shared state with three time-out locks
- Each lock represents a flying Crazyflie
  - Two locks for the two flying Crazyflies
  - One lock for the Crazyflie that will take off next
- When a lock expires another Crazyflie is free to take the lock and set a new end time
- Time-out locks will expire automatically and do not require communication to be released.
- A complication is that the clocks in the Crazyflies are not synchronized, use relative time when communicating the state

Flight planning

When the currently flying CF is going back to the pad the next one takes off

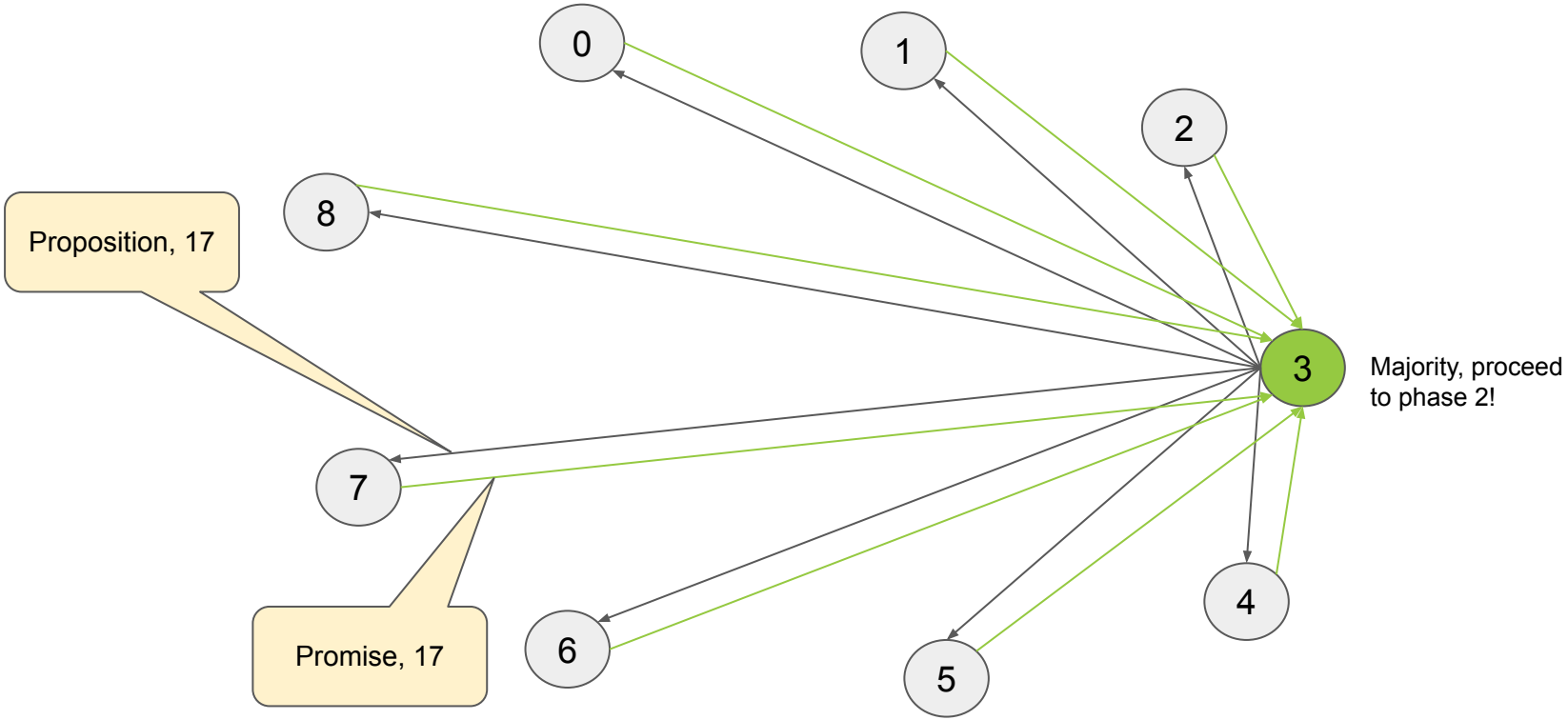
(We also tried to let two Crazyflies fly at the same time with a spacing of ~½ spiral cycle time. Sometimes we got problems with downwash when entering/leaving the spiral and abandoned the idea.)



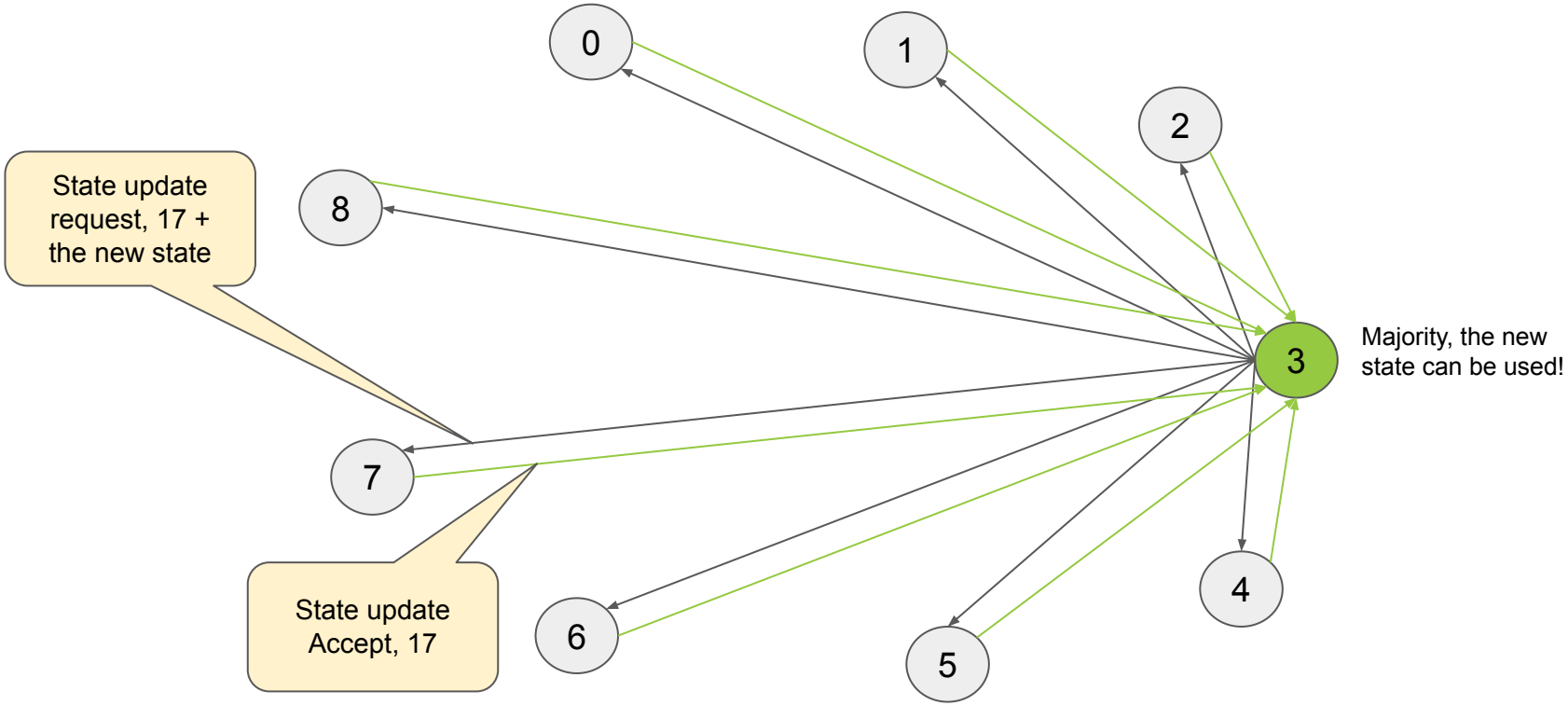
Distributed consensus

- Loosely based on Paxos
- Majority (5 nodes) required for consensus
- Two phase commit
- We use broadcast instead of unicast
- Packets are lost - must be supported
- All nodes are “nice” and behave as expected

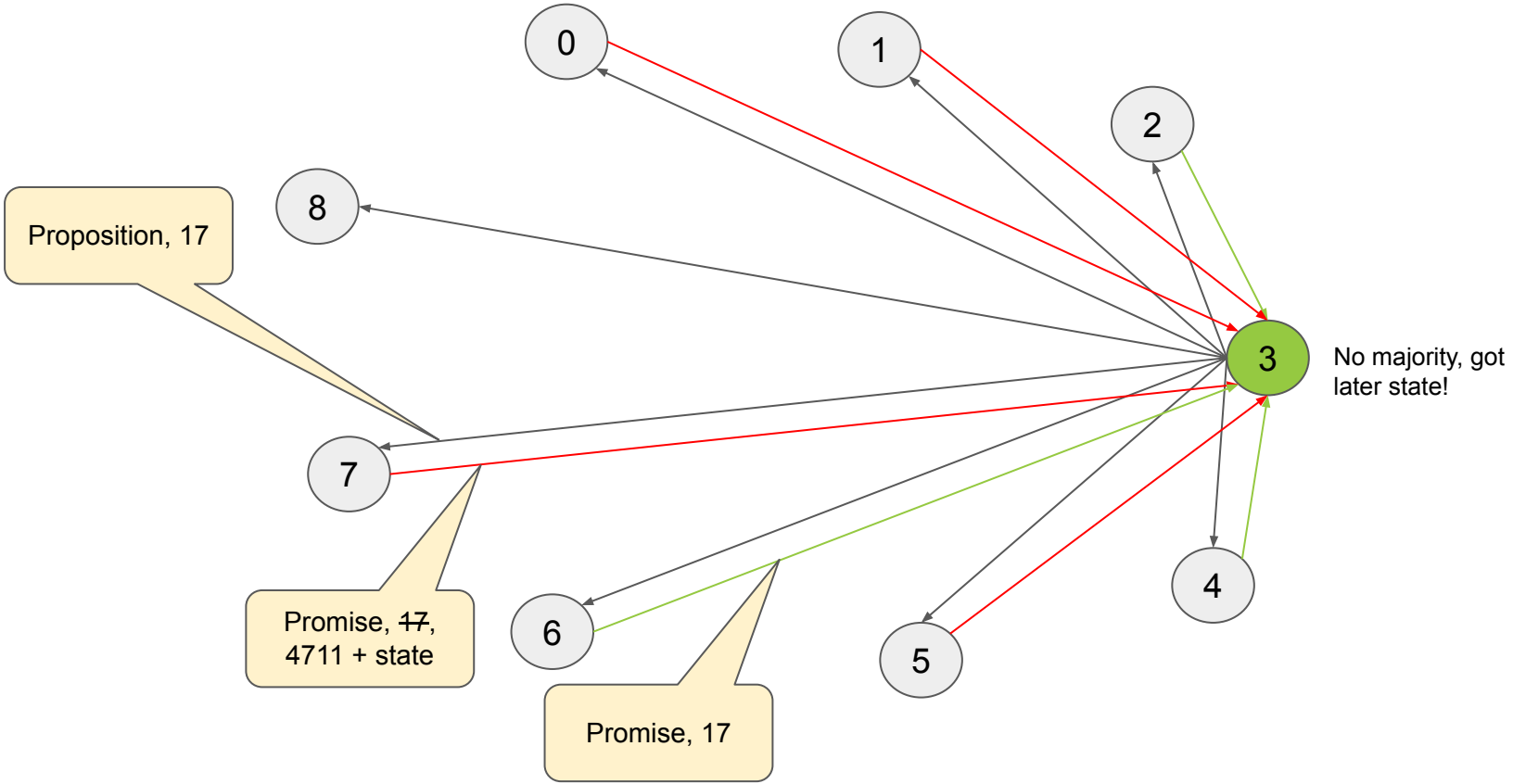
Phase 1 - prepare for state update



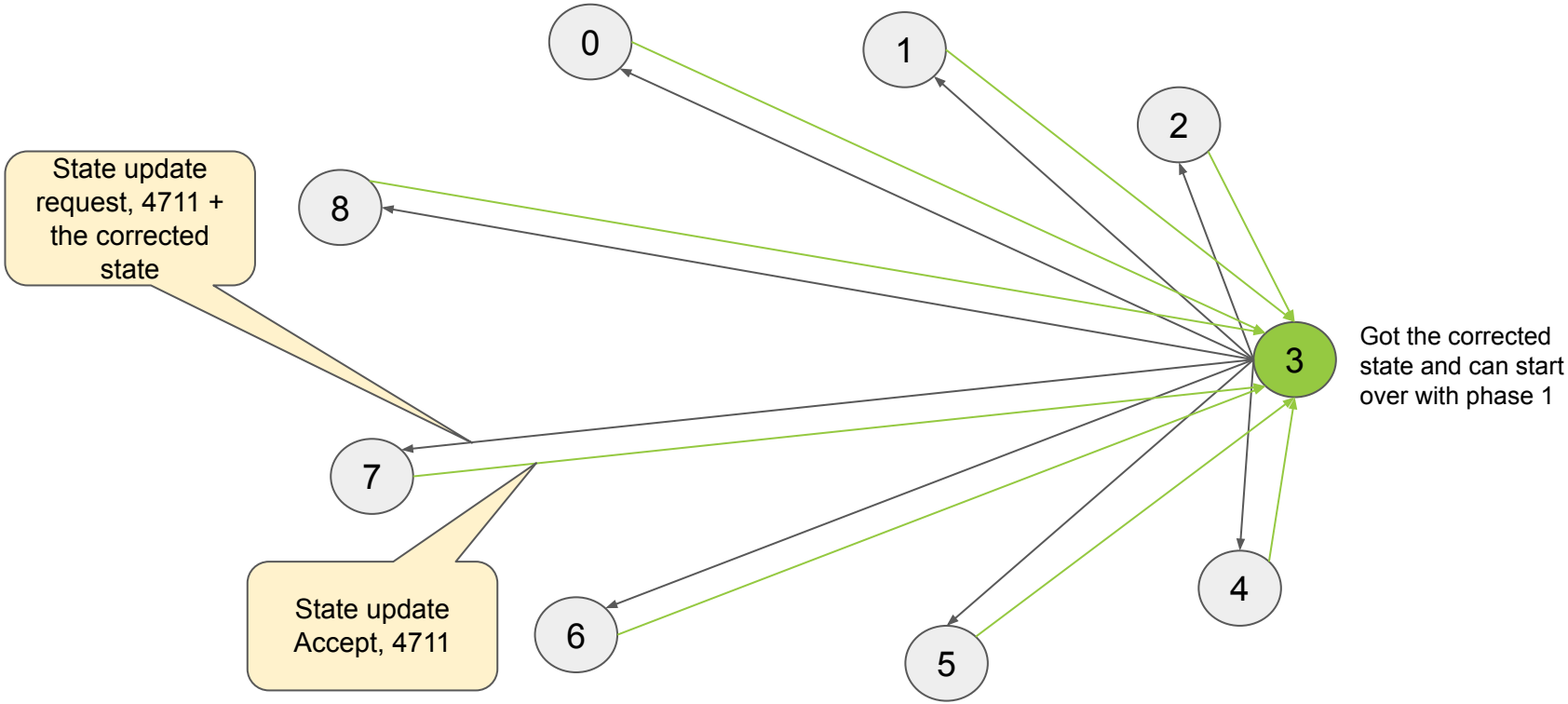
Phase 2 - update state



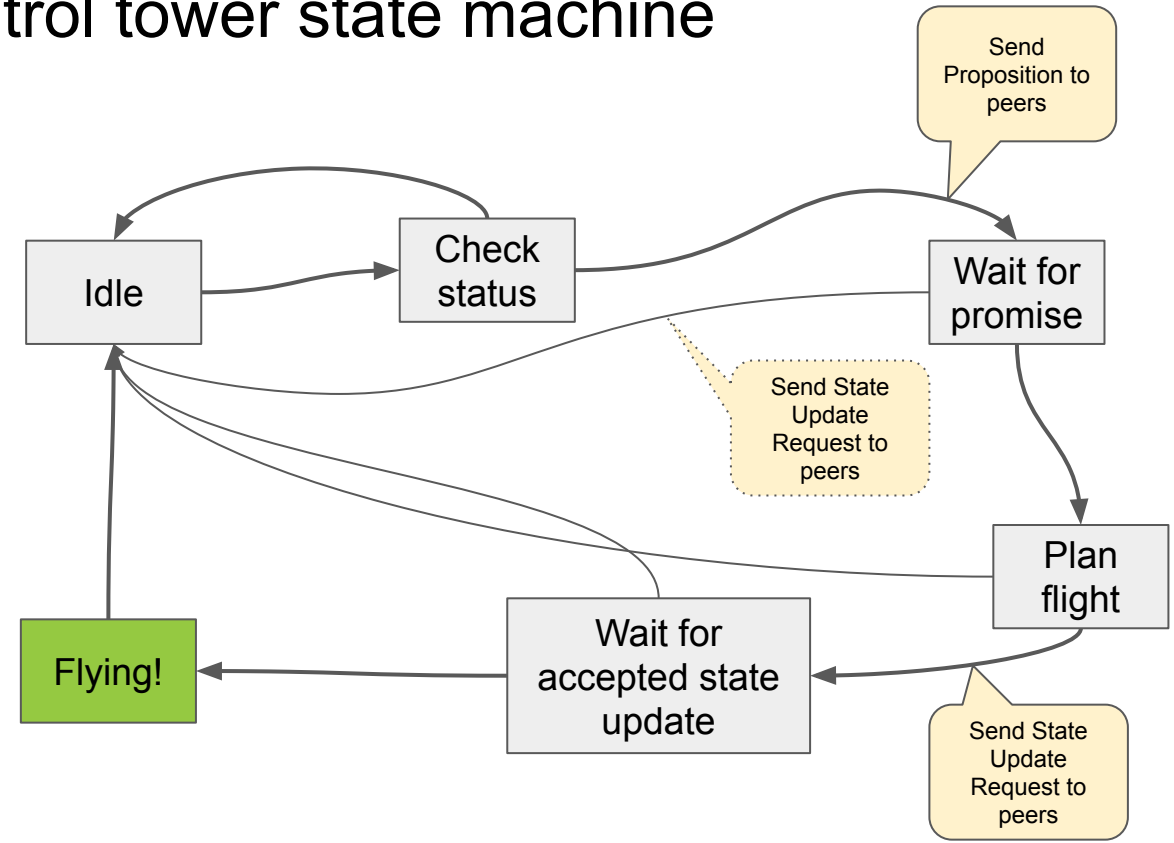
Phase 1 - swarm state out of synch



Phase 2 - swarm state out of synch



Control tower state machine



Hardware and code

We used:

- 9 Crazyflies with Lighthouse decks and Qi charger decks
- 2 Lighthouse V2 base stations
- Qi chargers from IKEA with 3D-printed pads

- A Computer with a Crazyradio for starting/stopping the demo
- A Crazyflie (connected via USB) for sniffing

The code is available in
https://github.com/bitcraze/crazyflie-firmware-experimental/tree/bam-2021

What did we learn?

- Losing more packets than expected
- Debugging when using P2P is not easy
- Distributed consensus is tricky
- Found some issues related to P2P and the High Level Commander

Live demo!