Decentralized architecture for robust semi-autonomous logistics and construction robots

Motivation

- Logistics robots are increasingly adopted in warehouse settings to improve efficiency, decrease downtime of various industry supply lines
- Robotics for construction are presently being evaluated in many forms, especially to aid in lifting heavy equipment/supplies
- Safe and collaborative robot control architectures can be used to increase productivity in both of these settings
- Decentralizing the control architecture can increase safety and robustness to signal drops, robot equipment malfunction, and other unforeseen failures

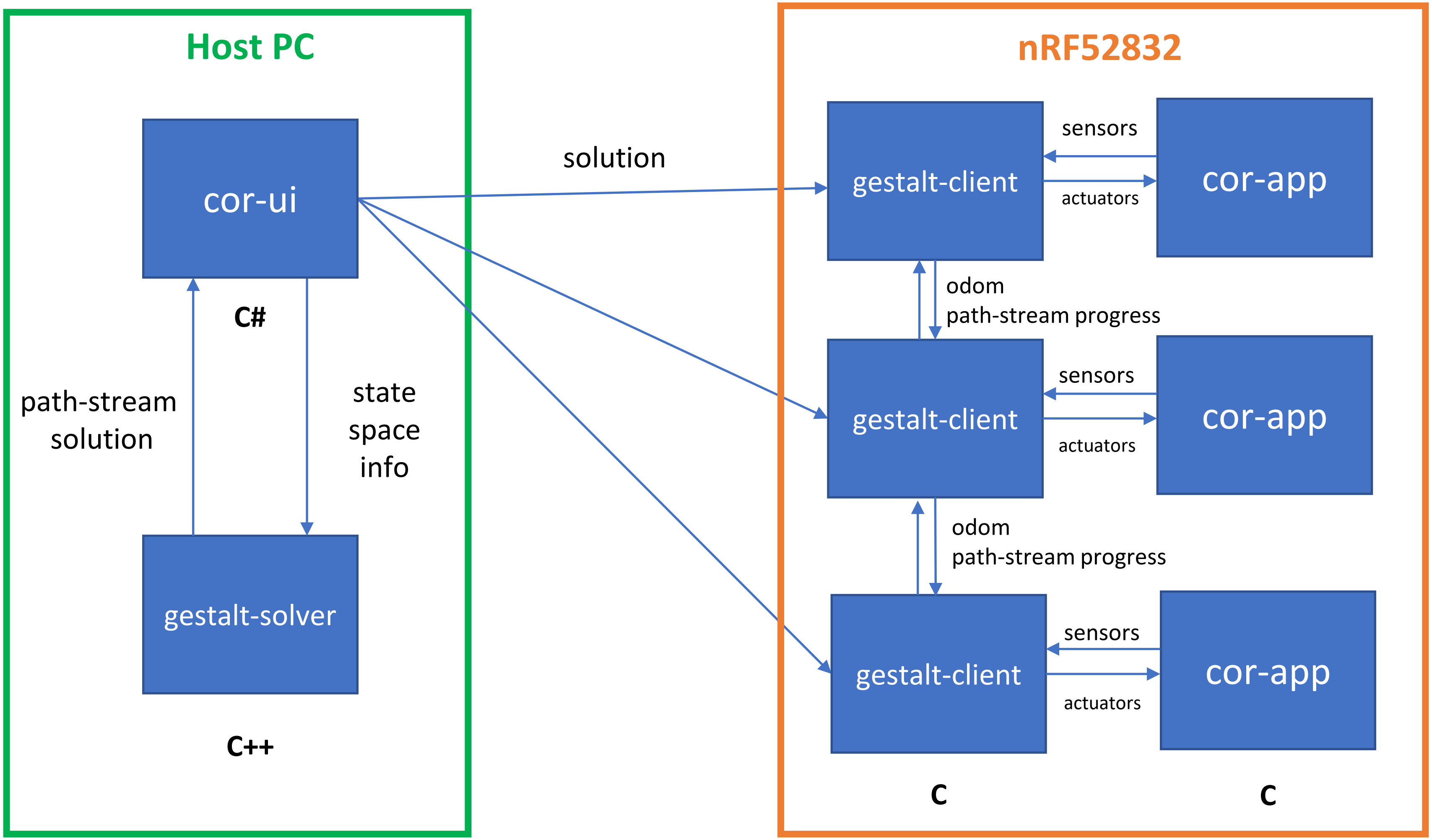
Goal

- This project will demonstrate a decentralized robot control architecture by which a user-controlled interface prescribes an objective to a semi-autonomous robotics network. This network is then able to arrange props collaboratively in arbitrary patterns safely without further guidance from the operator and is robust to losses within the network.

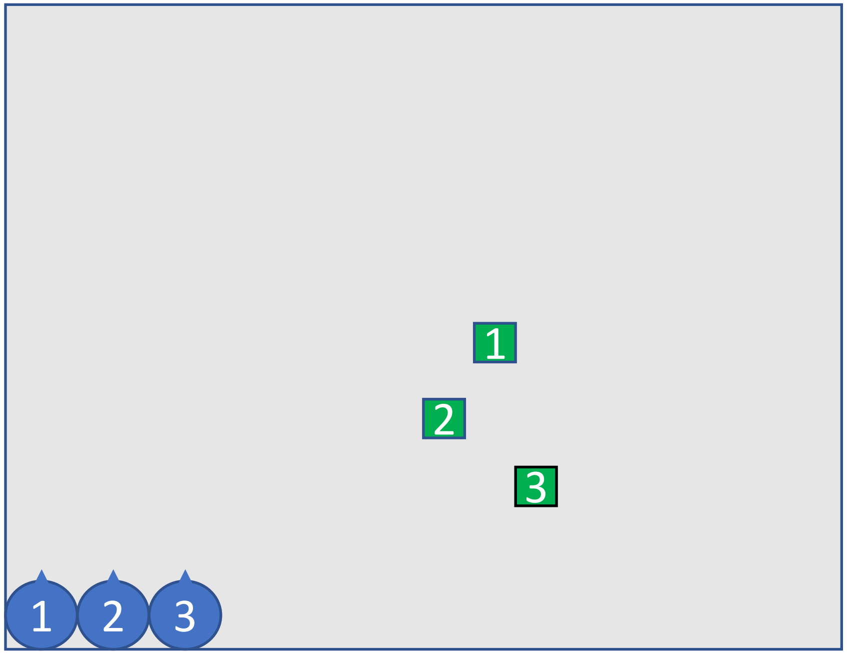
Project Approach

- The project involves a demonstration of 2+ robots operating in collaboration within an isolated environment to arrange props in arbitrary patterns as assigned by an operator.
- At tasking-time, a solver will calculate a stream of paths for the robots to follow, and this path is then downloaded to all robots in the network.
- During execution-time, the operator is disconnected from the network, and the path-stream will be correlated with live sensor data on each robot to move toward the solution state. Furthermore, during this phase, state space measurements and path-stream progress will be communicated between all operating robots, enabling higher spatial certainty and decreasing chances of collision.
- To demonstrate loss-tolerance, all robots will receive redundant path-streams such that network members may be removed at any time and the same outcome will be achieved.

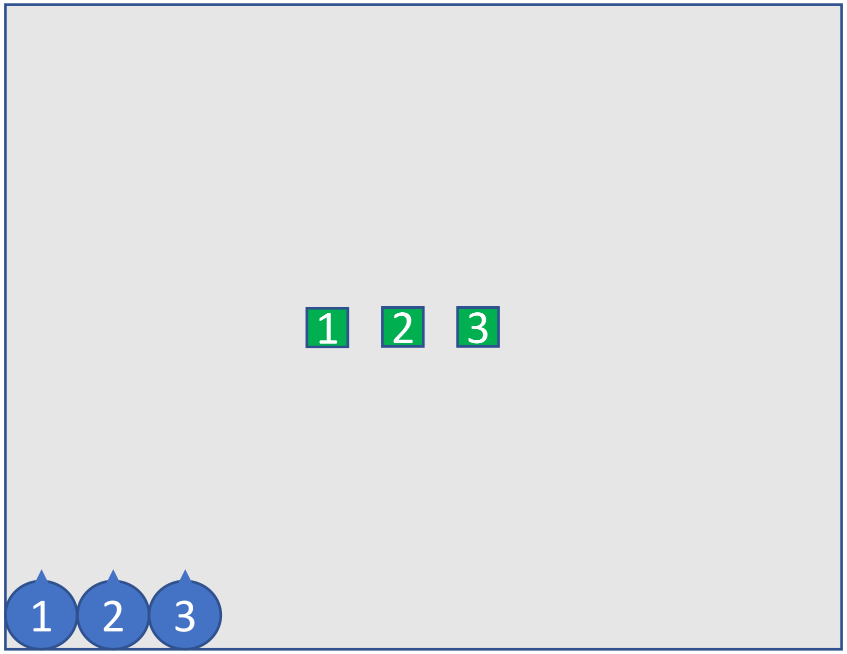
Top Level Architecture



Proof of Concept

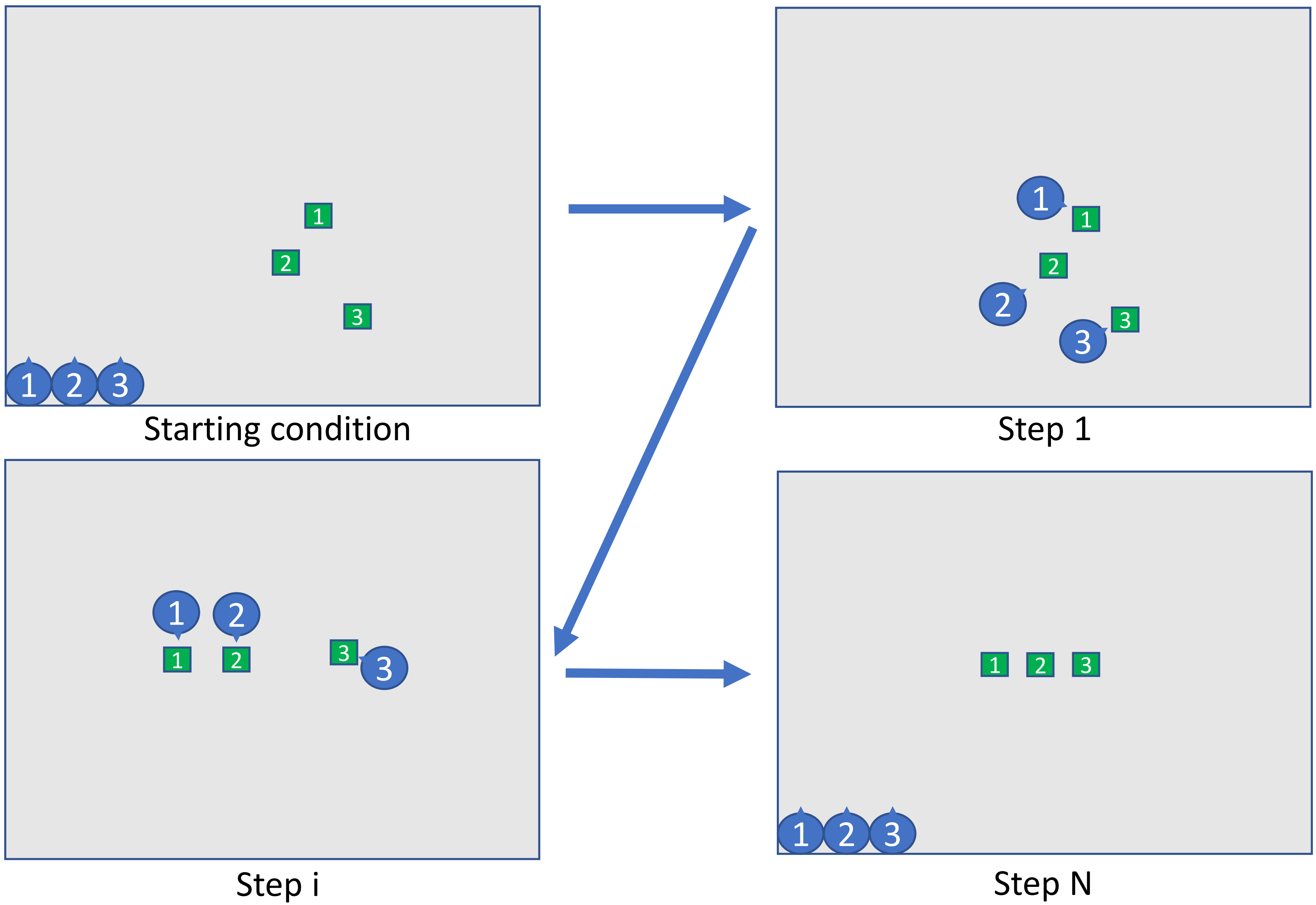


Starting condition

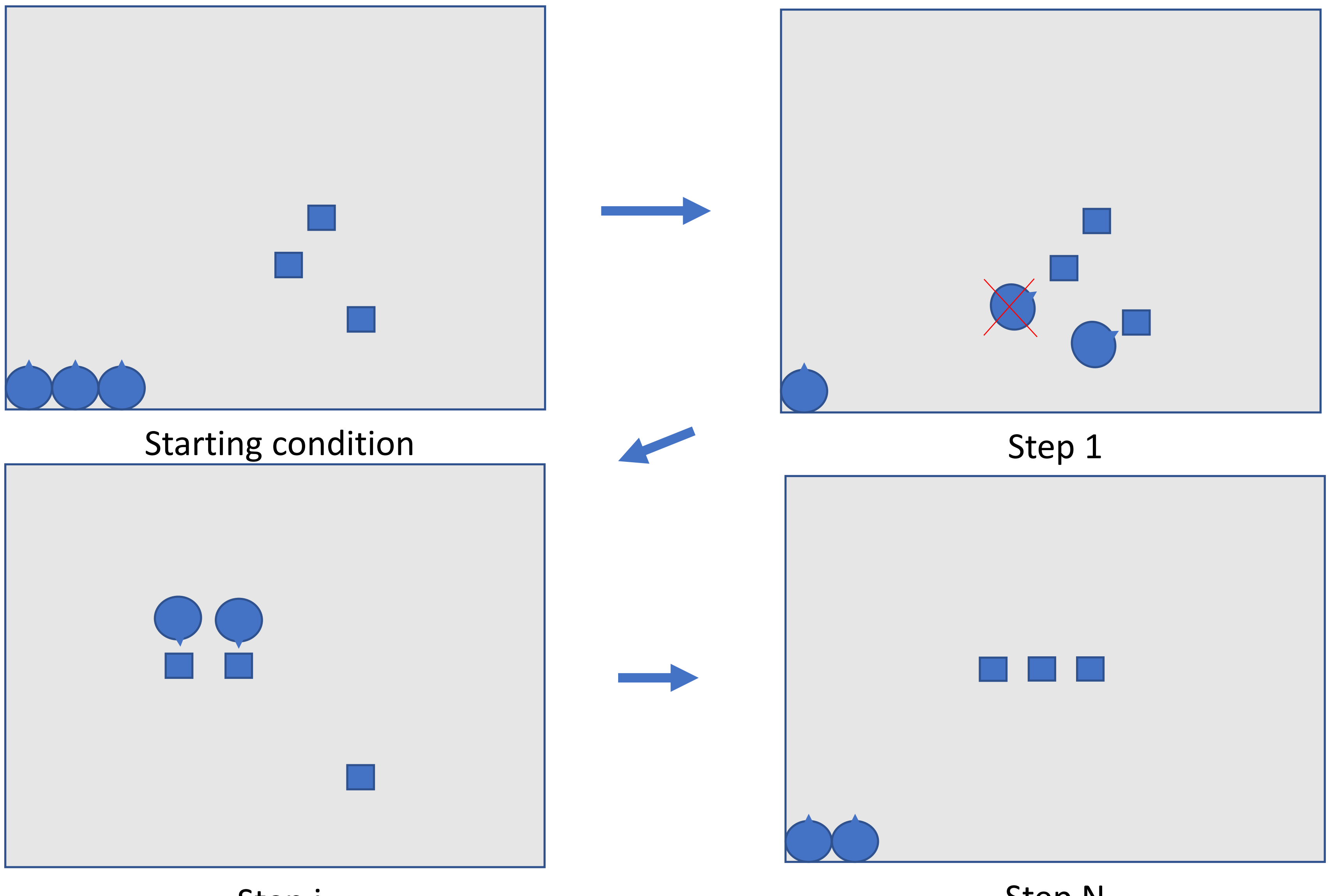


User-prescribed target

Proof of Concept - Solver

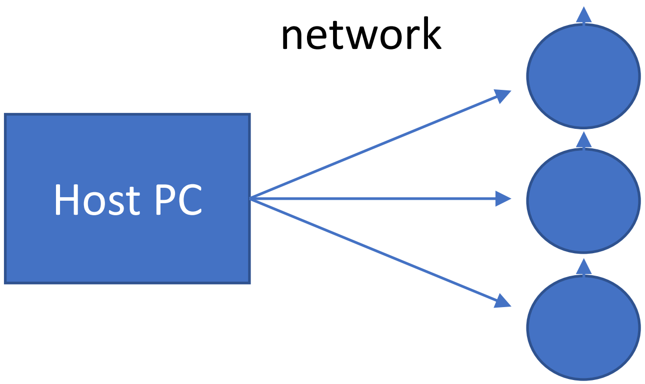


Proof of Concept – Loss-Tolerant Solver



Architecture

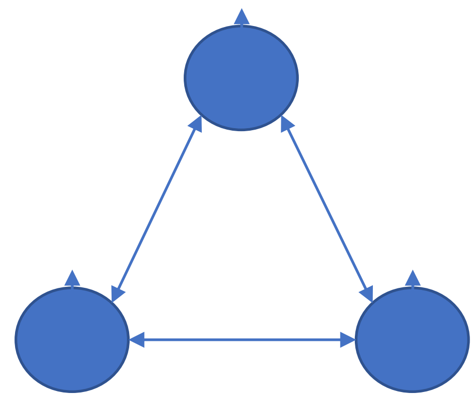
Solution path-streams are uploaded to robot network



Stage 1

Architecture at tasking

Path-stream progress and odometry are exchanged

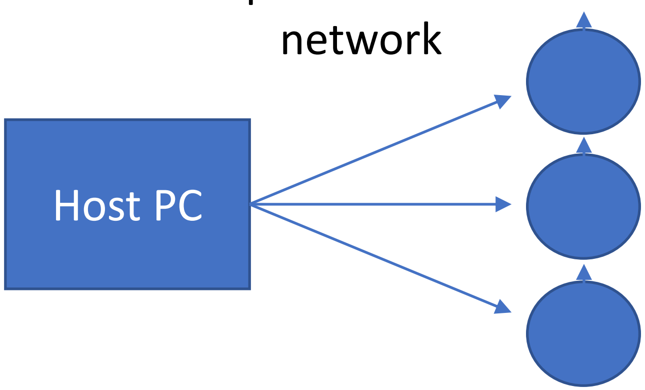


Stage 2

Architecture at execution

Architecture + Stretch Goal

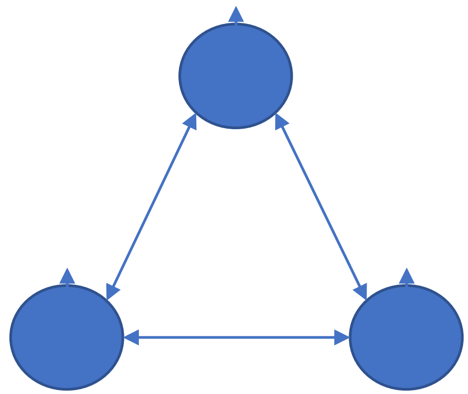
Solution path-streams are uploaded to robot network



Stage 1

Architecture at tasking

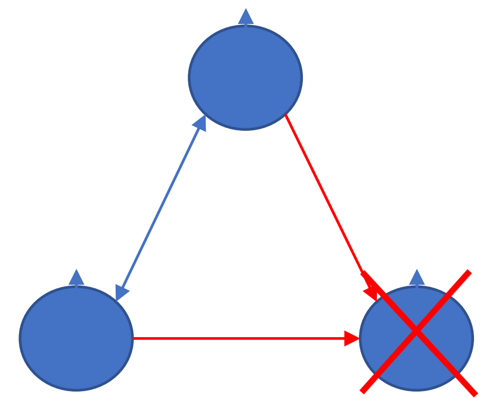
Path-stream progress and odometry are exchanged



Stage 2

Architecture at execution

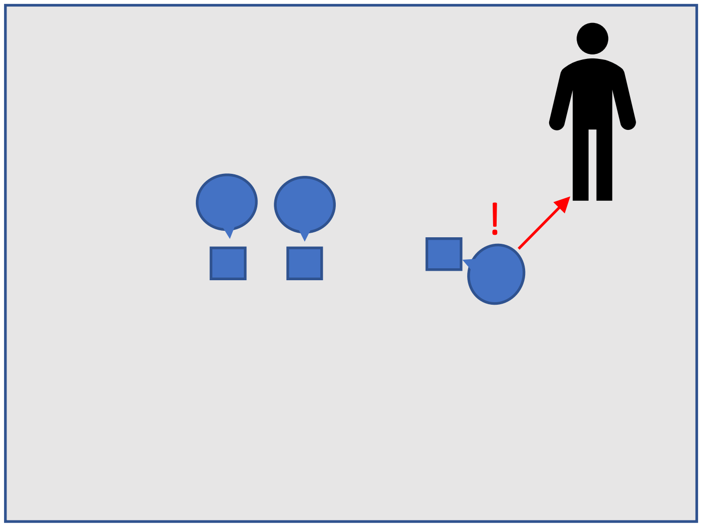
Remaining path-streams are consolidated and distributed between robots



Stage 2

Architecure at loss-time

Safety



STOP

If any single member of the network senses an unrecognized object within the operational space, all members will be stopped

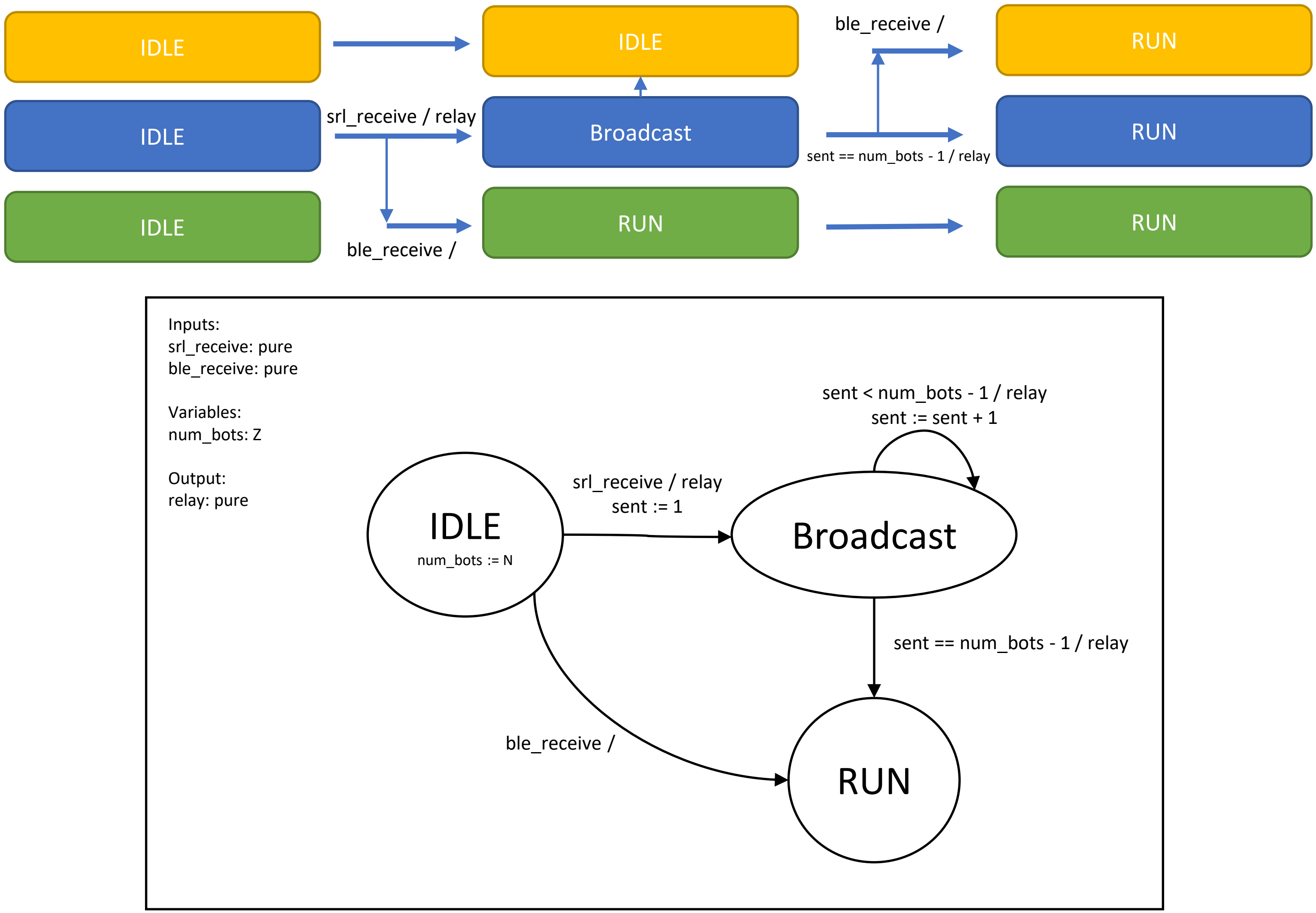
RESUME

All members of the network must confirm that the unrecognized object has exited the space before the entire network may resume

Equipment

- 3x Robot platform – Kobuki
- SoC
  - nRF52832
- Sensors
  - Accelerometer from Buckler
  - Gyro from Buckler
  - 2D top-mounted LiDAR $100
  - Bluetooth from nRF
- Actuators
  - One-dimensional linear actuator to grip cubes
- Host PC
  - Any OS running Unity

Initialization Flow



UART protocol

- 2 path stream example
- Little Endian

S = Path stream size = (16 * length) + 1
END = num_bytes – 1

Byte	0	1	2	3	4	5	6 to (5+S)	(5+S)+1	(5+S)+2 to ((5+S)+1) + S	END
Byte contents	‘g’	‘s’	num_bytes[0]	num_bytes[1]	num path streams	path length[0]	path stream[0]	path length[1]	path stream[1]	\n
Path stream	n/a	n/a	n/a	n/a	n/a	0	0	1	1	n/a

Path Stream Solution

Header: num_path_streams = 1 (uint8_t)

Path Stream

Path Stream

Path Stream

Header: path_length = 1 (uint8_t)
Path Stream:

- bot_id = 1 (uint8_t)
- x_pos_stream = 4 * length (single)
- y_pos_stream = 4 * length (single)
- action_stream = 4 * length (int32_t)
- exclusion_stream = 4 * length (int32_t)

BLE Broadcast Packet Definition

- 26 byte manufacturer's data
- Byte 0 (1 byte)
  - Bot ID (uint8_t)
- Bytes 1-8 (8 bytes)
  - Current Position (X, Y) (2x single)
- Bytes 9-12 (4 bytes)
  - Current Theta (θ) (single)
- Bytes 13-20 (8 bytes)
  - Localization Pole Position (X, Y) (2x single)
- Byte 21 (1 byte)
  - Path stream progress (uint8_t)