ID	Channel	Message
1	heartbeat	timestamp, platoon_id, robot_id, robot_type (e.g. detector or fixer), lane_id, position_in_platoon, position (x, y, z, theta: representing the estimated position in world coordinates and orientation), speed
2	action	timestamp, action_id, source_robot_id, target_platoon_id, target_robot_id, msg (string representing the message payload)
3	feedback	timestamp, action_id, my_robot_id, other_robot_id, msg
ID	Actions	Description
a	setPlatoon	set platoon_id for robot with robot_id equal to target_robot_id to msg
b	setID	Lane id could be 4 (outermost) to 1 (innermost); -1 if robot is not following a lane.
c	setType	set robot_type of robot with robot_id equal to target_robot_id to msg
d	setLane	set lane_id of robot with robot_id equal to target_robot_id to msg
e	setRole	set position_in_platoon of robot with robot_id equal to target_robot_id to msg
f	setPosition	set position of robot with robot_id equal to target_robot_id to msg
g	setSpeed	set speed of robot with robot_id equal to target_robot_id to msg
h-l	-	(free to add)
m	changeLane	request lane change for robot with robot_id equal to target_robot_id to new lane msg
n	changeRole	request change of position in platoon for robot with robot_id equal to target_robot_id to new position msg
o	moveToPosition	request position change for robot with robot_id equal to target_robot_id to go to position msg
p	changeSpeed	request speed change for robot with robot_id equal to target_robot_id to move at speed msg
q	specialRequest	request special action of robot with robot_id equal to target_robot_id (based on msg and robot_type)
r	mergeRequest	Request to merge to the one common lane
s-z	-	(free to add)
	NOTES:	
	Timestamps can be like ROS uses (rospy.get_time()).	
	Default platoon ids can be used; -1 can mean no platoon, then there can be platoons 1, 2, etc so more than one platoon could be moving at once.	
	Each robot can have a unique robot_id (like a mac address of a computer): e.g. 5 and 6 for the robots of group 1 (following Josef IDs).	
	Robot type could be like 1 for a detector, 2 for fixer, -1 for no type or other.	
	Lane id could be 4 (outermost) to 1 (innermost); -1 if robot is not following a lane.	
	Default position_in_platoon can also be used; 1 could be (leader), 2-10 followers, 0 (leader, robots fanned out), -1 (not the leader, robots are fanned out)	
	Position can be 0,0,0 at bottom left corner of table from ceiling camera; z is added just in case some group wants to try using slopes/tunnels; theta can be from 0 (pointing right on the overhead camera image).	
	Speed can be -1 or 0 for halted, 255 for full speed.	
	Special actions could include anything that is specific to the groups, like detecting or fixing something, or starting/stopping line following, moving to a certain distance from the leader, parking, avoiding obstacles, etc.	
	Setting will just change variables on a robot (e.g. which could be useful for debugging/recovering from some error), whereas requesting will get a robot to perform some action to change its variables.	
	The source_robot_id identifies, for debugging, which robot is sending commands.	
	Channel 3 feedback can be used by robots to acknowledge receiving a command (msg 1), say the command has been carried out successfully (msg 2), or say a command has failed (msg -1).	