Networks Class Project

For my class project, I would like to implement a directory service broker using ZeroMQ.

I am enrolled in an independent research course with Dr. Dubey in which I am writing a peer to peer networking library for use in multiplayer games. My goal is to develop strategies which deal with the more stringent performance requirements of games in virtual reality—higher frame rates, more fine-grained movement from more input sources, etc. One of the constraints of the project is that all the players are on the same local network.

Since the project is limited in scope and focused on networking between peers, I am hard-coding each peer's local IP address into my code for testing. While that is fine for the work I am doing this semester, if I ever want to release or use my code in a real application or use it outside of my local network, I would need a directory service to connect the nodes. Dr. Dubey recommended that I consider writing a broker for my networks project, and I thought that was a nice idea.

I would like to write the directory service in Python using ZeroMQ. The clients of this directory service would be instances of the C# project that I am developing for my independent research. Most of the code for this project will reside in the directory service (Python). Both Python and C# have stable ZeroMQ implementations.

The directory service would need to support two use cases: First, a peer can tell the directory service that it would like to start it's own "game" along with the number of peers it would like to join the party. The directory service would then register a new game internally and add that peer's IP address and port to the registry as the "owner" of the new game. Second, a peer can ask the directory service to connect it to an existing game. The directory service would inspect its list of open games and add the user to the whichever game is closest to the number of peers it would need for a game to start. That peer would receive back a list of IP addresses that it should connect to.

There are two important edge cases to handle: First, when a new user connects, the directory service must notify all existing peers in that game that the peer has connected. Second, if a peer disconnects, the registry must notify the peers that it has disconnected. To handle both cases, I plan to keep an open TCP connection between the directory and peers. When a new peer connects, all connected peers will be notified. The directory will periodically send a heartbeat request to all of the peers; if a peer does not respond after two or three retries, it will be disconnected.

My primary goal is for connecting to a game to be as fast as possible. I am not as particular about how long it takes for peers to be notified of a disconnecting peer.