Currently an RTS style game with epic combat involving up to thousands of units is most often done through a synchronization technique called lockstepping. This works really well for managing the large number of units between all of the players but it limits the player count to around a maximum of 8. Adding more players can quickly overflow the network. RTS games have been a rather overlooked genre especially in recent years. The genre has a core fanbase that truly love RTS games however the lack of mainstream attention has led to there being less experimentation and growth for the genre than many others.

There has been lots of research into distributed architectures and systems for massively distributed games. One such example is *Colyseus: A Distributed Architecture for online Multiplayer Games.* However in general much of the research is not directly applicable to RTS games due to the vast difference in content that needs to be pushed through the network. There has also been quite a bit of research on super peer networks, however once again these are based on peer to peer networks.

My approach is to focus on RTS network growth through the lens of cloud computing. Specifically the idea is to connect multiple nodes within the network to one another. Each node is similar to the concept of a super peer. When players connect to the game they would be assigned underneath one of these nodes. Any given node would have a number of users assigned, for example 3-5 users. This node would then use lockstepping strategies to ensure synchronization between its clients. Further all nodes would maintain a state of the game and synchronize with each other.

My expected deliverables are as follows

- A defined strategy of implementation, documenting the network architecture.(1-2 weeks)
- Converting badlogics “Quantum” from standard client server lockstepping to my defined architecture. (Just a POC, not perfect.)
    - Setting up Quantum (1 week)
    - Reading through, getting familiar with the netcode (1 week)
    - The process of transferring the code over will depend on the design from step one. Although I imagine this will be my biggest time sink.
- If time permitting, (hopefully it will be) analyze the network latency and rollback cost and frequency

Website: https://sirduck145.github.io/CCS-research-project/
[3] - https://github.com/badlogic/quantum
Interesting articles
- Network infrastructure for massively distributed games | Proceedings of the 1st workshop on Network and system support for games
- **Proximity-Aware Superpeer Overlay Topologies**
- How do Superpeer Networks Emerge? https://ieeexplore.ieee.org/abstract/document/5461968