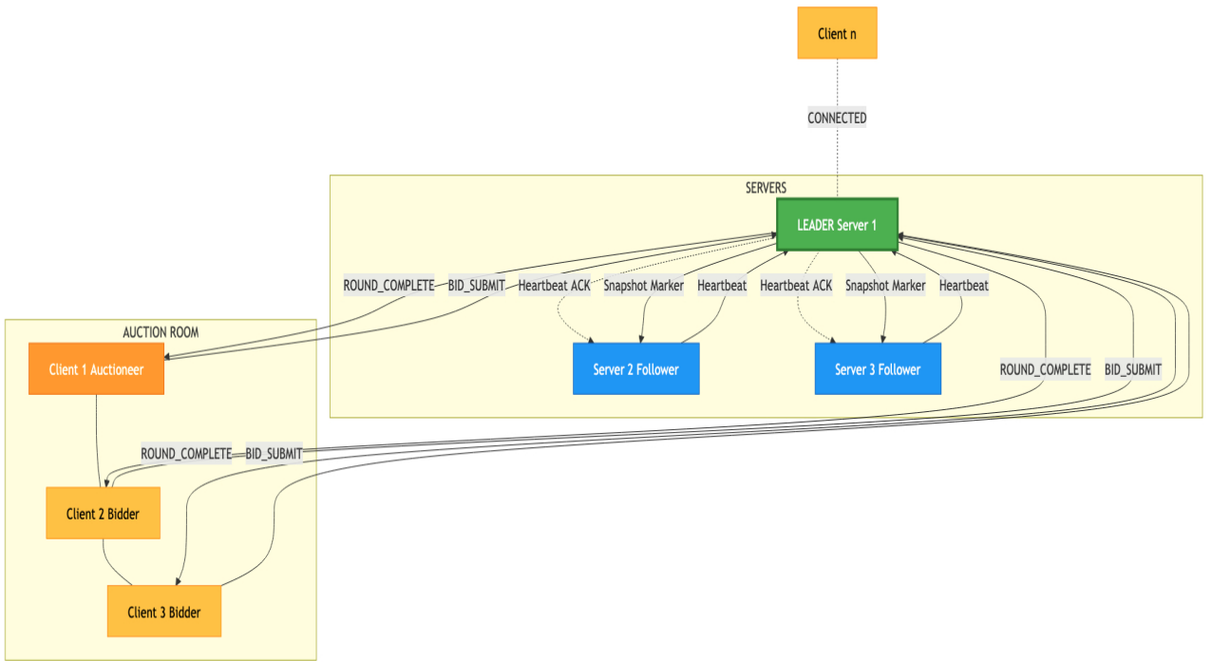
Project Information			
Group ID	29		**Semester (WS2?/2?)**
1. Student Name	Sohankumar Thaliyazcha Rajeeshkumar		4
2. Student Name	Srinidhi Krishna Kalpathy Sundaram		4
3. Student Name	Yash Chandrakant Amanaji		2
4. Student Name			

Project Title	Agora: A distributed auction system
Project Description	We the members of group 29 propose an auction system named Agora. In our system, clients can join Agora and either participate in an existing auction as a bidder or start a new auction.
	GitHub/GitLab/BitBucket etc. URL:
Code Repository Link	

Architectural Model	
Architecture Type	(●) Client-Server ( ) Peer-to-Peer ( ) Hybrid
Mechanism	
Communication	Explain briefly what it is used for:
TCP	
	Explain briefly what it is used for:
UDP	UDP is used for all communication between server and server as well as client and server. The messages are in a JSON format.
	Explain briefly what it is used for:
Other	
Concurrency	Explain briefly what it is used for:
Multithreading	Python's daemon threads are used to run concurrent tasks for network I/O and message processing. Timer threads are used for periodic tasks like sending heartbeat and snapshots. The async architecture decouples the network I/O and processing.
	Explain briefly what it is used for:
Multiprocessing	

Text should stay within text boxes. Hidden text and text running outside text boxes will not be considered.

System Architecture Desin
Include a clear diagram of your system architecture (key components and connections between them).



Dynamic Discovery of Hosts	
Discovery Mechanism	☑ Client discovers server ☑ Server discovers servers ☐ Other. Explain briefly:
Discovery Implemented	☑ Broadcast ☐ Multicast ☐ Other. Explain briefly:
Discovery Occurs	☑ When system starts ☑ Whenever new component comes in ☐ Other. Explain briefly:

Voting	
Voting Implemented Using	◉ Bully Algorithm ○ LeLann-Chang-Roberts Algorithm ○ Hirschberg-Sinclainr Algorithm
Group View Used	○ No ◉ Yes. Explain briefly what the group view is used for:
Nodes Identified Using	○ IP addresses / IP addresses + ports ◉ UUIDs ○ Other (describe ID format and reasons for using it):
Election Starts	☑ When system starts ☑ When a new server joins ☑ When the leader fails

Text should stay within text boxes. Hidden text and text running outside text boxes will not be considered.

Fault Tolerance	
Faults Tolerated	☑ Crash Faults (Leader Server, Regular Server, Client) ☐ Omission Faults ☐ Byzantine Faults
Fault Detection	Explain who sends heartbeats to whom, their frequency, and retries: Servers in follower state and client send heartbeats to leader. The leader responds to the heartbeat with an acknowledgement. Heartbeats are sent periodically every 20 seconds via timer threads. When the leader misses ACK for the heart beat above a threshold (2 misses), the leader is detected as failed. Leader tracks the timestamp of received heartbeats, remove clients or followers when they timeout (60 seconds).
Recovery Strategy	Chandy-Lamport snapshot algorithm is used to capture consistent distributed snapshots for fault recovery. The leader initiates a snapshot every 15 seconds by first recording its local state (discovered servers, clients, running auctions, bids, and queue contents). Each server persists its snapshot locally. Upon leader failure, followers detect the crash via missed heartbeats and triggers election. The newly elected leader restores the

Reliable Ordered Multicast	
Type of Ordering	☐ FIFO Ordering ☐ Causal Ordering ☑ Total Ordering
Reason for Chosen Ordering	Explain why the selected ordering the right type for the application: FIFO could not be used because two bids could interleave differently at different receivers. In causal ordering, the bids made concurrently can be delivered differently at different receivers. Hence total ordering is required to ensure that all bidders see the message in the same order.
Reliability Mechanism	☐ Acknowledgements ☐ Negative Acknowledgments ☑ Sequencing ☐ Other. Explain briefly:
Implementation Details	The auction proceeds in synchronized rounds coordinated by the leader: 1. Each participant submits their bid via BID_SUBMIT 2. Leader sequences bids and multicast each via BID_BROADCAST to all participants 3. Leader waits for all participants to submit 4. Leader broadcasts ROUND_COMPLETE with finalized bids for that round

Text should stay within text boxes. Hidden text and text running outside text boxes will not be considered.

Make sure the final file is a non-editable PDF (save or export as a PDF).