Contents

1. Main Scenarios & Subscenarios (Keywords)

Core Objective:

Implement a client-server system capable of:

- **Pinging** (latency measurement).

- **Broadcasting messages** to all clients.

- **Identifying clients** (server logs which client sent a ping/message).

Message	Ping	Broadcast	Connection	Identification
Create	Timestamp register	Multicast	TCP connection	Assign Client ID
Send	Latency calculate	Broadcast Receive	Client validation	Log actions
Receive	Echo ping	Exclude Sender	Disconnect Handle	Map endpoints
Parse	Ping sender identification	Thread-safe distribution	Async IO	
Queue				

Message	Description
Create	Build packets
Send	Transamit data over network
Receive	Accept incoming data
Parse	Decode message content
Queue	Message buffer

Ping	Description
Timestamp register	Tag with send time
Latency calculate	Measure round trip
Echo ping	Server bounces ping back
Ping sender identification	Track sender ID

Broadcast	Description
Multicast	Sends to all clients
Broadcast Receive	Clients get broadcasts
Exclude Sender	Skip original sender
Thread-safe distribution	Avoid data races

Connection	Description
TCP connection	Link client-server
Client validation	Authenticate clients
Disconnect Handle	Correct termination
Async IO	Non-blocking operation

Identification	Description
Assign Client ID	Assign Client ID
Log actions	Log actions
Map endpoints	Map endpoints

2. SWOT Analysis

Strengths	
Templating	Lightweight, modular design
TSQueue	Thread-safe message queue prevents data races
ASYNC	Cross-platform async networking efficiently

Weaknesses	
No encryption	Raw messages are vulnerable to interception
Minimal error recovery	Recovery plan
Scalability	Hardware limitation for testing

Opportunities	
SSL Security	Enhance security
Message compression	Memory management and efficiency
Database integration	Persistent event logging

Threats	
Network latency	Latency spikes distort ping measirements
Message validation	Wrong message structure could crash clients/servers
Competitive protocols	Websockets, gRPC

3. Operational and Technical considerations and measurement:

Function	Operational	Tecnical	Metric
Ping Larency	Slow ping responses	TSQueues contentions	Round-trip time
Client identification	Duplicate Ids cause message routing error	ID Assignment logic	Duplicate Ids occurrance
Message Reliability	Messages could fail to send or receive	async_write error handling	Message drop rate %
Server Performance	Server becomes unresponsive underload	ASIO thread workload	CPU usage spikes % over time

4. User Roles & Activities

User	Activities	
Client	Connect to server	Send ping
	Broadcast message	Receive messages
Server	Accept clients	Relay broadcast
	Log client pings/ID	Monitor connection

---

5. Functionalities: Input → Expected Output

Functionality	Input	Expected Output
Ping server	Client sends ping	Server echoes ping and calculate RRT
Broadcast Message	Client sends a Message to all clients	All clients receive it (except the sender)
Client connect	IP/Port connect requests	Server assigns IP, adds to connected pool, sends Accept confirmation
Error handle	Client disconnects	Async context doesn't crash, pushed to wait for new connection

6. Backlog for Architecture

Phase 1 Core Networking	ASIO Context setup (Client-Server)
	Message templating (message<T>)
	Thread-safe queue design (tsqueue)
Phase 2 Client-Server Logic	Ping with timestamp (Custom Message -> Server Ping)
	Broadcast (Custom message -> Message all clients)
	Client ID management design (ID counter in server)
Phase 3 Enhancements	Message validation
	Feedback logic
	Measurement logging
Phase 4 Security	TLS Encryption
	Authentication (Client Tokens)