Designing a Secure and Scalable Multiplayer Game Platform in C

Presented by Team 5, a skilled group of developers and designers.

We bring a diverse set of expertise, covering development and UI design.

OUR TEAM

Development Team:
Sneha Kumari
Ankit Roy
Nirvan Jain
Ashish Patel

Presenters:
Nikunj Buddhawar
Vedanga Bharadwaj

Design Team:
Naina Kotnak
Souvik Mandal

UI Designers:
Akash Sanapala
Sushanta Mahata

Introduction to Multiplayer Game Platform

Cross-Platform Support

Runs seamlessly on Linux and various operating systems.

Optimized Server-Client Model

Ensures smooth responsive gameplay with efficient communication.

User-Friendly Features

Easy game discovery and joining with strong security and privacy.

Initial Multiplayer Test: Tic Tac Toe

Core Features

- Real-time network play with two participants
- Turn-based logic, synchronized game states
- Server-client multiplayer communication validated

Challenges & Objectives: Simple design limits scalability and complexity

Expanding Game Library

Two-Player Number Guessing

Turn-based TCP socket game guessing secret numbers across network.

Chess

Turn-based strategy with move validation and game state saving.

Treasure Hunt

Real-time multiplayer treasure collection on a grid.

Technology Stack Overview

Programming

C language for high performance and cross-platform support

Networking

TCP/IP sockets for reliable client-server communication

fork() and UNIX signals for concurrency and event handling

Security & Platform

OpenSSL's SHA-256 for integrity verification

Linux CentOS 7 for robustness and security

Multiplayer Game Workflow

1. **Connection & Lobby**

   Clients connect, join lobby, configure options

2. **Matchmaking & Initialization**

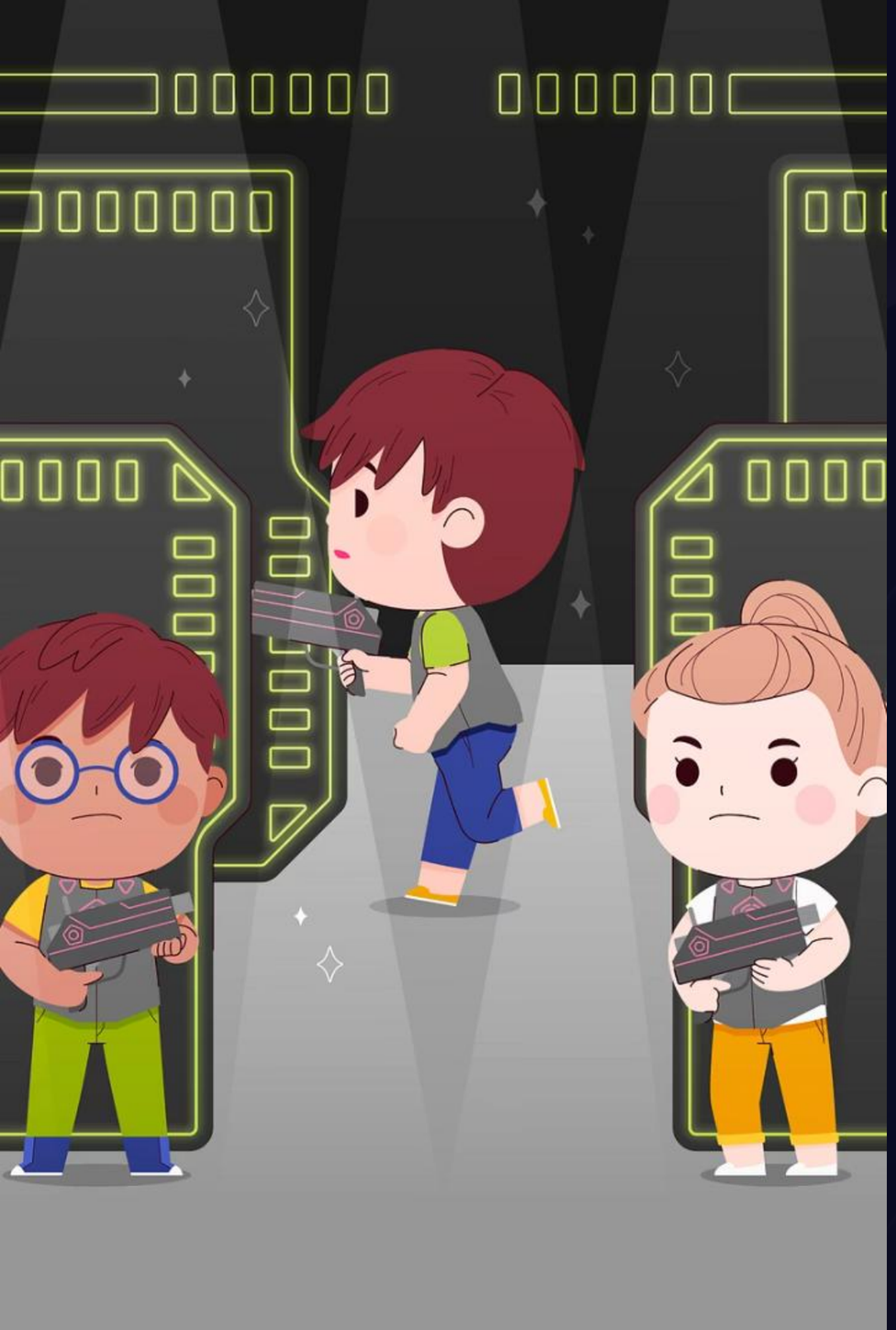
   Server confirms players, starts game state

3. **Gameplay & Updates**

   Players send moves; server broadcasts updates

4. **Game End & Cleanup**

   Results announced; server frees resources



Two-Player Number Guessing Game

Game Description

Server picks number 1–100; players alternate guessing via TCP.

Key Features

- Turn-based with live feedback ("Too high", "Too low").
- Clear message protocol for synchronization.

Modular Flow Overview

Client

setup_client_socket()
- Create and connect sock

interact_with_server()
- Create server messages
- Send user input

cleanup()
- Close socket

Socket
(Two-way connection)

Server

setup_server_socket()
- Create, bind, listen
- Accept 2 players

handle_player() [Thread]
- Prompt guess
- Calculate score

send_scorecard()
- Display final results

cleanup()

Multiplayer Chess Game

Game Flow

Menu options; 8x8 board with numeric codes.

Gameplay

Turn-based with move validation, check, and checkmate.

Networking

Server-client communication via send() and recv() for real-time sync.

Features

Live board updates and seamless handling of disconnections.

Modular Flow Overview

Client

setup_client_socket()
- Create an connect soket
- Input IP & port

connect_to_server()
- Send player info
- Request board state

interact_with_game()
- Display board
- Validate move format

cleanup()

Socket
(TuO way corection)

Server

setup_server_socket()
- Create, bind, listen
- Accept 2 players

initialize_board() [Thread]
- Receive moves
- Update board state
- Checkmate detection

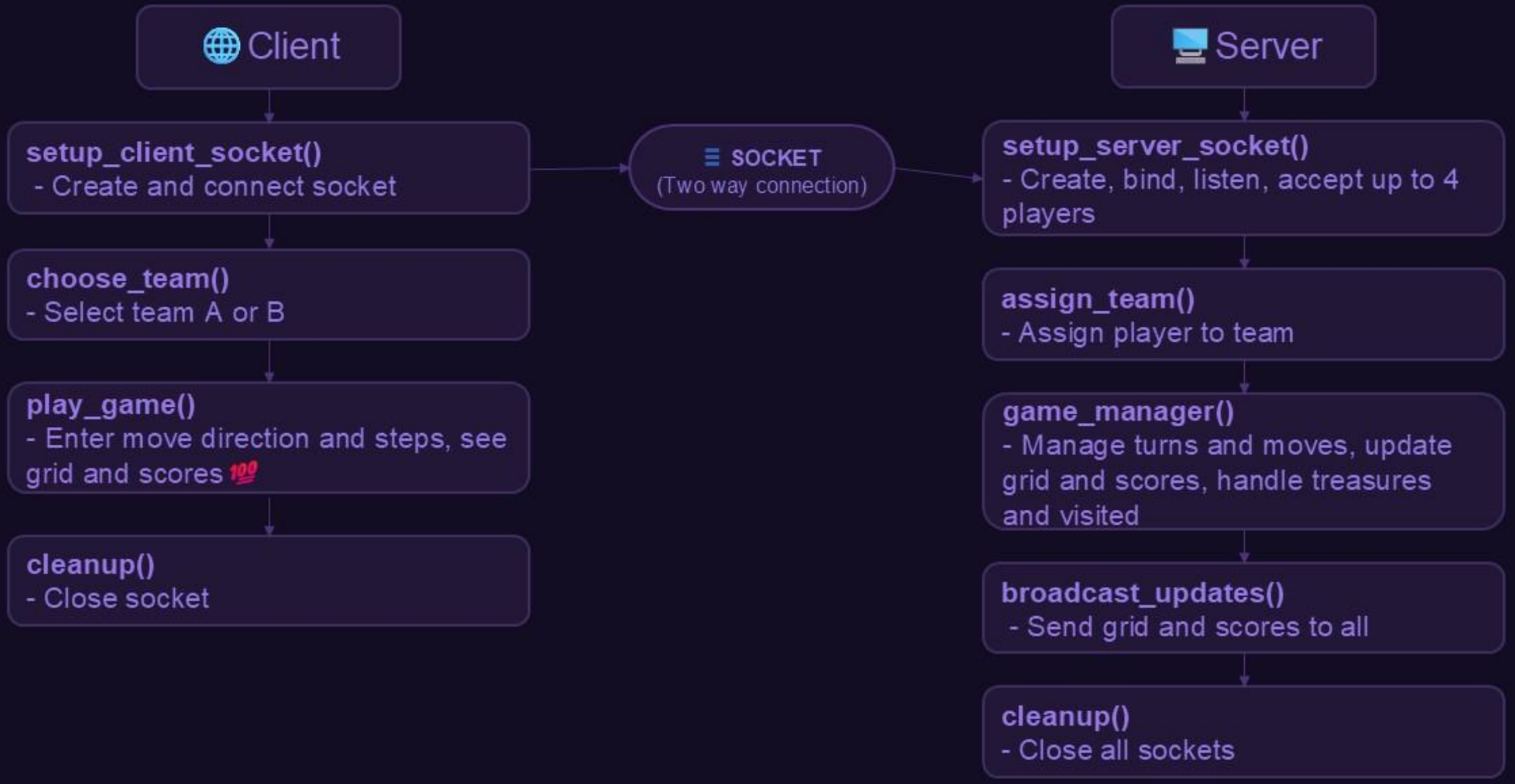
send_game_state()
- Sync board to both clients
- Display winner

Treasure Hunt Game Overview

Gameplay Highlights

- Team-based treasure collection on 10x10 grid
- Real-time multiplayer engagement
- Text-based movement commands
- Live score updates drive competition
- Random treasure placement adds challenge
- Encourages strategic teamwork

Modular Flow Overview-Treasure Hunt



🎮 Project Highlights – Multiplayer Gaming Platform

A unified web platform for real-time multiplayer gaming. Play with friends or anonymous users seamlessly.

User Experience

- Clean, responsive UI for smooth gameplay
- Dynamic game rooms with unique IDs
- Real-time chat, scoreboards, and timers

Key Features

- Matchmaking and private rooms support
- Game state sync via Flask routes
- Modular design for easy game addition

Server-Client Architecture

1. **Server**: Creates and binds socket, listens and accepts connections.
2. Uses fork() to handle multiple players concurrently.
3. Controls gameplay logic and scoring mechanisms.
4. **Client**: Connects to server, sends moves, and receives updated game board.

Process Synchronization & Signal Handling

Parallel Player Handling

fork() creates separate processes for each player interaction.

Signal Management

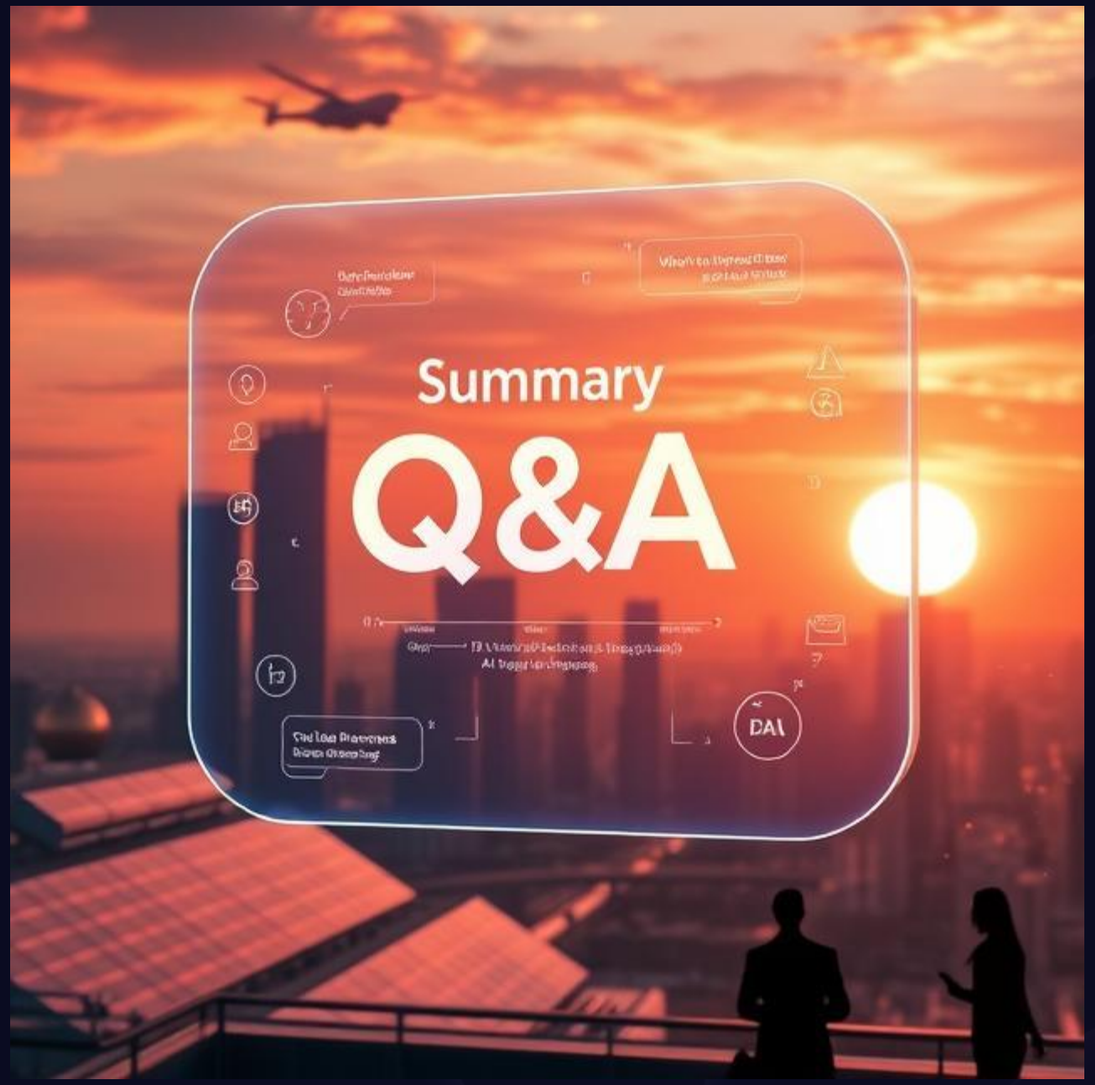
Signals update victory status and manage disconnections effectively.

Synchronization

Avoids conflicting moves ensuring game state consistency.

Conclusion & Future Scope

- Deployed a networked Tic-Tac-Toe with real-time process control.
- Plans include GUI development and 3+ player support.
- Future remote play with matchmaking functionality.



Summary and Q&A

Secure Architecture

Framework implemented primarily in C for robustness and performance.

Integrated Technologies

Combines networking, concurrency, and cryptography seamlessly.

Ready for Growth

Designed for scaling and addition of advanced features.