**TCR Project - Tower Defense Game Documentation**

**Table of Contents**

1. [System Architecture](https://claude.ai/chat/d8fb860a-01ca-47f0-99b0-fc4328e9a3b0#system-architecture)
2. [Application PDU Description](https://claude.ai/chat/d8fb860a-01ca-47f0-99b0-fc4328e9a3b0#application-pdu-description)
3. [Sequence Diagram](https://claude.ai/chat/d8fb860a-01ca-47f0-99b0-fc4328e9a3b0#sequence-diagram)
4. [Deployment & Execution Instructions](https://claude.ai/chat/d8fb860a-01ca-47f0-99b0-fc4328e9a3b0#deployment--execution-instructions)

**System Architecture**

**Overview**

TCR Project is a multiplayer tower defense game implemented in Go using TCP sockets. The system follows a client-server architecture where multiple clients connect to a central server for real-time battles.

**Architecture Components**

┌─────────────────┐ TCP Connection ┌─────────────────┐

│ Client │ ←──────────────────→ │ Server │

│ (clients.go) │ │ (server.go) │

└─────────────────┘ └─────────────────┘

│

▼

┌─────────────────┐

│ Data Layer │

│ JSON Files │

│ • players.json │

│ • troops.json │

│ • towers.json │

└─────────────────┘

**Module Structure**

**1. Server Module (server/server.go)**

* **Role**: Central game coordinator
* **Responsibilities**:
  + Accept client connections
  + Handle authentication
  + Manage lobby queue
  + Coordinate battles between players
  + Update player statistics

**2. Client Module (clients.go)**

* **Role**: Player interface
* **Responsibilities**:
  + Connect to server with retry mechanism
  + Handle user authentication
  + Process game interactions
  + Display game state

**3. Models Module (models/entities.go)**

* **Role**: Data structures
* **Components**:
  + Player: User account with stats
  + Tower: Defense structures
  + Troop: Attack units

**4. Utils Module (utils/)**

* **JSON Utils**: Data persistence operations
* **Level Utils**: Experience and leveling system

**Data Flow Architecture**

Client Input → Server Processing → Game Logic → State Update → Response to Clients

↑ ↓

└──────────────── Feedback Loop ←─────────────────────────────┘

**Application PDU Description**

**Protocol Data Units (PDUs)**

The communication between client and server uses text-based messages over TCP. All messages are terminated with newline characters (\n).

**1. Authentication Phase**

|  |  |  |  |
| --- | --- | --- | --- |
| Direction | Message Type | Format | Example |
| Server → Client | Username Request | "👤 Nhập username: " | "👤 Nhập username: " |
| Client → Server | Username Response | "{username}\n" | "player1\n" |
| Server → Client | Password Request | "🔑 Nhập password: " | "🔑 Nhập password: " |
| Client → Server | Password Response | "{password}\n" | "123456\n" |
| Server → Client | Auth Result | Success: "✅ Đăng nhập thành công! Đang chờ người chơi khác..." |  |
|  |  | Failed: "❌ Sai thông tin đăng nhập." |  |

**2. Game Setup Phase**

|  |  |  |  |
| --- | --- | --- | --- |
| Direction | Message Type | Format | Example |
| Server → Client | Troop Assignment | "🎲 Bạn đã nhận được 3 quân:\n • {name} (ATK: {atk}, DEF: {def}, Mana: {mana})" | "🎲 Bạn đã nhận được 3 quân:\n • Pawn (ATK: 150, DEF: 100, Mana: 3)" |
| Server → Client | Game Rules | "📋 Hướng dẫn:\n • Mỗi lượt bạn sẽ chọn 1 quân để tấn công..." |  |
| Server → Client | Game Start | "🎮 Trận đấu bắt đầu! Bạn là Người chơi {id}" | "🎮 Trận đấu bắt đầu! Bạn là Người chơi 1" |

**3. Battle Phase**

|  |  |  |  |
| --- | --- | --- | --- |
| Direction | Message Type | Format | Example |
| Server → Client | Turn Notification | "🕐 Lượt của bạn (Player {id})" | "🕐 Lượt của bạn (Player 1)" |
| Server → Client | Action Request | "💂 Chọn quân (1, 2, 3):\n {id}. {name} (ATK: {atk}, DEF: {def})" | "💂 Chọn quân (1, 2, 3):\n 1. Pawn (ATK: 150, DEF: 100)" |
| Client → Server | Action Response | "{choice}\n" | "1\n" |
| Server → Client | Battle Result | "🎯 {troop} tấn công {tower}:\n • Gây ra: {damage} sát thương\n • HP còn lại: {hp}" | "🎯 Pawn tấn công Guard Tower 1:\n • Gây ra: 50 sát thương\n • HP còn lại: 950" |

**4. Game End Phase**

|  |  |  |  |
| --- | --- | --- | --- |
| Direction | Message Type | Format | Example |
| Server → Client | Victory/Defeat | "🏆 Người chơi {id} đã chiến thắng!" | "🏆 Người chơi 1 đã chiến thắng!" |
| Server → Client | Experience Gain | "🎉 Bạn nhận được +30 EXP!" |  |
| Server → Client | Level Up | "🎉 Bạn đã lên {levels} level mới!" | "🎉 Bạn đã lên 1 level mới!" |
| Server → Client | Game End | "🎮 Trận đấu kết thúc!" |  |

**5. Special PDUs**

|  |  |  |
| --- | --- | --- |
| Type | Purpose | Format |
| Heartbeat | Connection check | Empty byte array []byte{} |
| Error | Error notification | "❌ {error\_message}" |
| Timeout | Timeout warning | "⚠️ Hết thời gian chờ, vui lòng thử lại." |
| Disconnect | Connection lost | "🔌 Mất kết nối." |

**Sequence Diagram**

sequenceDiagram

participant C1 as Client 1

participant S as Server

participant C2 as Client 2

participant DB as JSON Files

Note over C1,DB: Connection & Authentication Phase

C1->>S: TCP Connect

S->>C1: "👤 Nhập username: "

C1->>S: username

S->>C1: "🔑 Nhập password: "

C1->>S: password

S->>DB: Validate credentials

DB-->>S: Auth result

alt Authentication Success

S->>C1: "✅ Đăng nhập thành công!"

S->>S: Add to lobby queue

else Authentication Failed

S->>C1: "❌ Sai thông tin đăng nhập"

S-->>C1: Close connection

end

Note over C1,DB: Player 2 Connection (Same process)

C2->>S: TCP Connect

S->>C2: Authentication flow

S->>S: Add to lobby queue

Note over C1,DB: Game Setup Phase

S->>S: Match players (C1 + C2)

S->>DB: Load troops data

DB-->>S: Troops list

S->>S: Generate random troops for each player

par Send to Player 1

S->>C1: "🎲 Bạn đã nhận được 3 quân:"

S->>C1: Troop details

S->>C1: Game rules

S->>C1: "🎮 Trận đấu bắt đầu! Bạn là Người chơi 1"

and Send to Player 2

S->>C2: "🎲 Bạn đã nhận được 3 quân:"

S->>C2: Troop details

S->>C2: Game rules

S->>C2: "🎮 Trận đấu bắt đầu! Bạn là Người chơi 2"

end

Note over C1,DB: Battle Loop

loop Game turns until King Tower destroyed

alt Player 1's Turn

S->>C1: "🕐 Lượt của bạn (Player 1)"

S->>C1: "💂 Chọn quân (1, 2, 3):"

C1->>S: Choice (1-3)

S->>S: Process attack

S->>C1: Battle result

S->>C2: Tower status update

else Player 2's Turn

S->>C2: "🕐 Lượt của bạn (Player 2)"

S->>C2: "💂 Chọn quân (1, 2, 3):"

C2->>S: Choice (1-3)

S->>S: Process attack

S->>C2: Battle result

S->>C1: Tower status update

end

S->>S: Check win condition

alt King Tower destroyed

break Game Over

par Victory/Defeat messages

S->>C1: Victory/Defeat message

and

S->>C2: Victory/Defeat message

end

end

end

end

Note over C1,DB: Game End Phase

S->>DB: Load player data

DB-->>S: Player stats

S->>S: Update winner EXP

S->>S: Check level up

par Send results

S->>C1: EXP gain message

S->>C1: Level up (if applicable)

S->>C1: "🎮 Trận đấu kết thúc!"

and

S->>C2: EXP gain message

S->>C2: Level up (if applicable)

S->>C2: "🎮 Trận đấu kết thúc!"

end

S->>DB: Save updated player data

Note over C1,DB: Connection Cleanup

S-->>C1: Close connection

S-->>C2: Close connection

**Deployment & Execution Instructions**

**Prerequisites**

* **Go**: Version 1.24.1 or higher
* **Operating System**: Windows, macOS, or Linux
* **Network**: TCP port 8080 available

**Project Structure**

tcr\_project/

├── main.go # Application entry point

├── clients.go # Client implementation

├── go.mod # Go module file

├── data/ # Data directory

│ ├── players.json # Player accounts

│ ├── troops.json # Troop definitions

│ └── towers.json # Tower definitions

├── models/

│ └── entities.go # Data structures

├── server/

│ └── server.go # Server implementation

└── utils/

├── json\_utils.go # JSON operations

└── level\_utils.go # Level system

**Setup Instructions**

**1. Create Data Directory**

mkdir -p data

**2. Setup Data Files**

Create the required JSON files in the data/ directory using the provided content:

* data/players.json - Player accounts
* data/troops.json - Troop definitions
* data/towers.json - Tower definitions

**3. Initialize Go Module**

go mod init tcr\_project

go mod tidy

**Execution Steps**

**Start Server**

# Run from project root directory

go run main.go

Expected output:

🔌 Server đang chạy trên cổng 8080...

**Start Client(s)**

Open **separate terminal windows** for each client:

# Terminal 1 - Player 1

go run clients.go

# Terminal 2 - Player 2

go run clients.go

**Usage Flow**

**Client Connection Process**

1. Client connects to server
2. Enter username (use player1 or player2)
3. Enter password (use 123456 or 654321)
4. Wait for second player to join
5. Game automatically starts when 2 players connected

**Gameplay**

1. Each player receives 3 random troops
2. Players take turns selecting troops (enter 1, 2, or 3)
3. Troops automatically attack opponent's towers
4. Game ends when a King Tower is destroyed
5. Winner receives +30 EXP and potential level up

**Configuration Options**

**Server Configuration (in server.go)**

const serverPort = ":8080" // Change port if needed

const lobbyQueueSize = 10 // Max players in queue

const connectionTimeout = 30 // Seconds for client timeout

**Client Configuration (in clients.go)**

const maxRetries = 3 // Connection retry attempts

const retryDelay = 2 \* time.Second // Delay between retries

const serverAddress = "localhost:8080" // Server connection string

**Troubleshooting**

**Common Issues**

1. **Port Already in Use**
2. Error: listen tcp :8080: bind: address already in use

**Solution**: Change port in server.go or kill process using port 8080

1. **Connection Refused**
2. Error: dial tcp [::1]:8080: connect: connection refused

**Solution**: Ensure server is running before starting clients

1. **JSON File Errors**
2. Error: Không thể mở file players.json

**Solution**: Ensure data/ directory exists with proper JSON files

1. **Authentication Failed**
2. ❌ Sai thông tin đăng nhập.

**Solution**: Use correct credentials from players.json:

* + Username: player1, Password: 123456
  + Username: player2, Password: 654321

**Network Configuration**

* **Firewall**: Ensure port 8080 is not blocked
* **Local Testing**: Use localhost or 127.0.0.1
* **Network Testing**: Replace localhost with server IP address in clients.go

**Performance Tuning**

* **Concurrent Players**: Modify lobbyQueueSize for more simultaneous games
* **Timeout Settings**: Adjust connection timeouts based on network conditions
* **Resource Management**: Monitor memory usage with multiple concurrent games

**Development Notes**

**Adding New Features**

1. **New Troops**: Add entries to troops.json
2. **New Towers**: Add entries to towers.json
3. **New Players**: Add entries to players.json

**Code Modifications**

* Server logic: server/server.go
* Client interface: client/clients.go
* Data models: models/entities.go
* Utility functions: utils/ directory

**Testing**

* Test with multiple client connections
* Verify data persistence after games
* Check error handling for network issues
* Validate game balance and mechanics