UNIVERSITÉ DU LUXEMBOURG

TP1 Networks 2 – War Card Game

Student surnames, first names and student ID number	ZUIDBERG Chris – 0230289906	
Title of the course	Networks 2	
Name of the study program	Bachelor in Applied Information Technology (BINFO)	
Names of the instructors to whom the task is submitted	STEENIS Bernard	
Date of submission		
Number of pages:	Number of words:	Number of characters:

1. Code Architecture and Design Choices

Overall Architecture

The War Card Game implements a client-server architecture using Python socket programming. The system is designed around two main components that communicate over a network to provide a real-time multiplayer gaming experience.

Core Components:

- Game Server (war_game_server.py): Central game coordinator and state manager
- Game Client (war_game_client.py): Player interface and network communication handler

Key Architectural Decisions

1. Dual Protocol Approach

TCP for Game Communication:

- Choice Rationale: The War card game requires reliable, ordered message delivery to maintain consistent game state between players
- Implementation: All game commands, state updates, and player coordination use TCP connections on port 5555
- Benefits: Guarantees message delivery and prevents game state desynchronization

UDP for Service Discovery:

- Choice Rationale: Automatic server discovery across local networks requires broadcast capability
- Implementation: Server continuously broadcasts its location on UDP port 54545; clients listen for these broadcasts
- Benefits: Eliminates need for manual IP configuration and enables plug-and-play networking

2. Multi-threaded Server Design

Thread Architecture:

- Main Thread: Handles client connections and game orchestration
- Client Handler Threads: One per connected client for message processing
- UDP Broadcast Thread: Continuous service advertisement
- Heartbeat Monitor Thread: Connection health monitoring
- Connection Rejection Thread: Manages excess connection attempts

Design Benefits:

- Concurrent client handling without blocking
- Separation of concerns for different network functions
- Fault isolation between different system components

3. Robust Connection Management

Heartbeat System:

- Client Side: Sends heartbeat every 15 seconds
- Server Side: Monitors with 20-second timeout tolerance
- Purpose: Early detection of network failures and silent disconnections

Reconnection Strategy:

- Grace Period: 120-second reconnection window
- State Preservation: Complete game state maintained during disconnections
- Seamless Resume: Players rejoin exactly where they left off

4. Message Protocol Design

Frame Structure:

[4-byte length][pickled message data]

Protocol Benefits:

- Variable-length message support
- Binary efficiency with Python pickle serialization
- Simple parsing with length-prefixed framing
- Support for complex data structures (dictionaries, lists)

5. Error Handling Strategy

Network Resilience:

- Comprehensive socket exception handling
- Graceful degradation on communication failures
- Automatic fallback mechanisms (localhost discovery)
- Timeout management for all network operations

Game State Protection:

- Atomic game state updates
- Consistent state across all clients
- Recovery from partial failures

2. Game Description

Game Overview

War is a classic card game implemented as a networked multiplayer experience. Two players compete using a standard 52-card deck, with the goal of capturing all cards through comparative card battles.

Game Rules

Setup

- Standard 52-card deck is shuffled and divided equally (26 cards each)
- Each player receives their cards in a face-down stack
- Players cannot see their cards before playing them

Basic Gameplay

1. Simultaneous Play: Both players simultaneously reveal the top card from their stack
2. Comparison: Cards are compared by rank (2 lowest, Ace highest)
3. Winner Takes All: Player with higher card wins both cards
4. Card Collection: Won cards go to the winner's separate winning pile

Card Ranking (Low to High)

2,3,4,5,6,7,8,9,10, Jack, Queen, King, Ace

War Conditions (Ties)

When both players reveal cards of equal rank:

1. War Declaration: Each player places one card face-down, then one face-up
2. New Comparison: The face-up cards are compared
3. Winner Takes Pot: Winner collects all cards in play (minimum 6 cards)
4. Chained Wars: If second comparison also ties, process repeats

Stack Management

- Empty Stack: When a player's main stack is empty, their winning pile is shuffled and becomes their new stack
- No Cards Available: If a player cannot participate in a war due to insufficient cards, they lose immediately

Victory Conditions

- Complete Victory: A player wins when their opponent has no cards remaining in either stack or winning pile
- War Elimination: A player wins if their opponent cannot complete a war due to insufficient cards

Network Game Features

Enhanced Multiplayer Experience

- Real-time Synchronization: Both players see results simultaneously
- Automatic Opponent Matching: Server pairs two players automatically
- Reconnection Support: Players can rejoin games after network interruptions
- Graceful Disconnect Handling: Remaining player wins if opponent disconnects permanently

Game Statistics Tracking

- Round Counter: Display of current round number
- War Tracking: Count and notification of war occurrences
- Stack Status: Ongoing display of remaining card counts

3. User Manual

System Requirements

Minimum Requirements:

- Python 3.7 or higher
- Network connectivity (LAN or localhost)
- Terminal/Command prompt access

Network Requirements:

- TCP port 5555 available for game communication
- UDP port 54545 available for server discovery
- Firewall configured to allow these ports (if applicable)

Installation and Setup

Quick Start (Same Computer)

1. Download Files: Ensure both war_game_server.py and war_game_client.py are in the same directory
2. No Installation Required: Uses only Python standard library

Network Play Setup

1. Server Computer: Must be accessible to client computers on the network
2. Firewall Configuration:
    - Allow incoming connections on TCP port 5555
    - Allow UDP broadcast traffic on port 54545
3. Network Requirements: All players must be on the same local network for automatic discovery

Running the Game

Step 1: Start the Server

python war_game_server.py

Server Status Indicators:

- Server automatically begins broadcasting its location
- Waits for exactly 2 players to connect
- Displays connection messages as players join

Step 2: Start Client 1

Python war_game_client.py

Connection Process:

1. Automatic Discovery: Client searches for server on local network
2. Name Entry: Prompt for player name (or auto-generates if empty)
3. Connection Confirmation: Displays successful connection and player number
4. Wait State: Shows "Waiting for other player..." message

Step 3: Start Client 2

Python war_game_client.py

Game Initialization:

1. Same Connection Process: Automatic discovery and name entry
2. Game Start: Once both players connect, game begins automatically
3. Card Distribution: Each player receives 26 cards

Playing the Game

Initial Display:

Game started! Opponent: [Player name]
You have 26 cards in your stack
Press Enter to play your next card, or type ‘q’ to quit.

Player Actions

Playing a Card:

- Action: Press Enter key
- Result: Your top card is revealed and compared with opponent's card
- Display: Shows both cards and round winner

Example Round Output:

You play 7 of Hearts, opponent plays 4 of Clubs. You WIN this round! (+2 cards)

War Scenario Output:

You play King of Spades, opponent plays King of Hearts. WAR!
You play 9 of Diamonds, opponent plays 6 of Clubs. You WIN this round (after 1 war)! (+6 cards)

Quit Game:

- Action: Type 'q' and press Enter
- Effect: Gracefully disconnects and shuts down server
- Result: Opponent is notified of your departure

Game Progression

Round Flow:

1. Both players press Enter when ready
2. Server reveals cards simultaneously
3. Winner determination and card distribution
4. Updated game status display
5. Continue until victory condition met

Status Information:

- Current round number
- Cards played each round
- Round winner announcement
- War occurrences and outcomes

Victory Conditions

Winning the Game:

Congratulations! You won the game!

Losing the Game:

Game over. [Opponent Name] won.

Troubleshooting

Connection Issues

Problem: "Server discovery timed out"

- Solution: Server may not be running or not on same network
- Fallback: Client automatically tries localhost (127.0.0.1:5555)
- Manual Fix: Ensure server is running and firewall allows connections

Problem: "Could not connect to server"

- Check: Server is running and listening on correct port
- Verify: No other application is using port 5555
- Network: Ensure client and server can communicate

During Gameplay

Problem: "Disconnected from server"

- Cause: Network interruption or server shutdown
- Reconnection: Restart client within 120 seconds to resume game
- Resume: Game state is preserved and will continue from last round

Problem: "Timeout: One or both players did not respond"

- Cause: Player took too long to press Enter (90-second limit)
- Solution: Respond more quickly in future games
- Result: Game ends automatically

Advanced Features

Reconnection System

- Automatic Detection: Server detects when players disconnect
- Grace Period: 120 seconds to reconnect
- Seamless Resume: Game continues exactly where it left off
- State Preservation: All cards and game progress maintained

Network Discovery

- Automatic: Clients find server without configuration
- Broadcast: Server announces itself every 2 seconds
- Fallback: Automatic localhost connection if network discovery fails

Limitations and Considerations

Current Limitations

- Two Players Only: Exactly 2 players required, no more, no less
- Single Game: One game at a time per server instance
- No “play again” functionality (if implemented one could also add a score keeping feature)
- Local Network: Designed for LAN play (not internet-scale)

Best Practices

- Stable Network: Use reliable network connection for best experience
- Timely Response: Respond to prompts within reasonable time
- Graceful Exit: Use 'q' command rather than forcibly closing terminal

Security Considerations

- Local Network Use: Designed for trusted local networks
- No Authentication: Players identified by name only