Showdown

Distributed Systems (A.Y. 2025/2026)

Concept

- **Idea:** *real-time* game based on Brawl Stars "Showdown" mode, last standing player wins
- **Primary Users:** casual gamers
- **Distribution:** Local Area Network (no NAT traversal)
- **Roles:** peer (client/host) + lobby server
- **Architecture:**
  - **P2P:** players directly exchange game data with *authoritative host*
  - **Client-Server:** lobby discovery service

Requirements

Functional requirements:

- **Scalability:** Support multiple users in the same lobby
- Real-time player movement & shooting
- Lobby discovery without manual IP entry

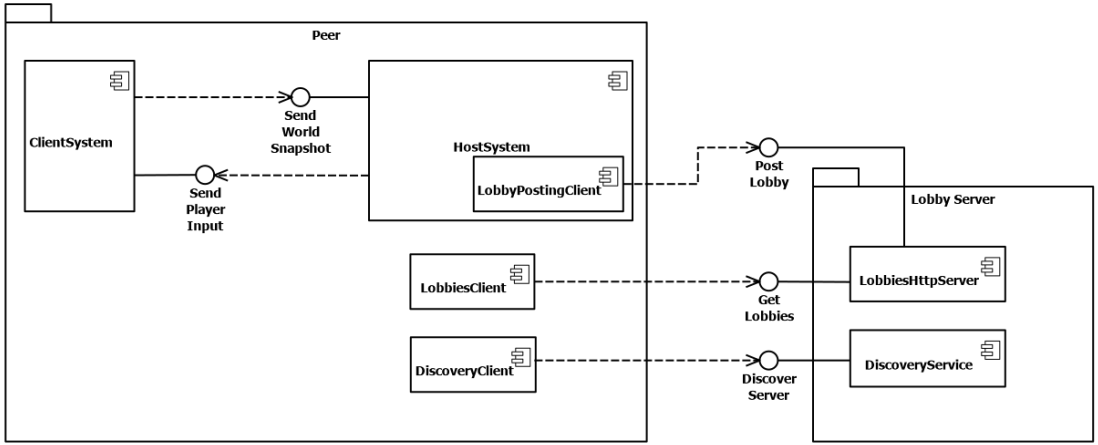
Non-functional requirements:

- **Availability:** always responsive to inputs
- **Performance**: Gameplay should always be smooth
- **Fault Tolerance:** Continue despite packet loss

System Architecture

Hybrid Architecture Design:

- **Peer-to-Peer:** Direct communication between players during gameplay
- **Peer-Server:** Lobby server for discovery and matchmaking



Peer-to-Peer Communication

Peers communicate by using a double TCP+UDP connection.

TCP:

- Initial handshake
- Connection keep-alive

UDP:

- **Client → Host:** Player input packets
- **Host → Client:** World snapshot packets (authoritative game state)
- Low latency

Lobby Server Communication

The Lobby Server enables 2 services:

- Discovery Service (UDP)
  - Responds to discovery requests
  - Must work in broadcast
- Lobby Posting Service (HTTP)
  - Maintains a list of active lobbies
  - Peers can query or post lobbies via GET/POST requests

Data & Consistency

- **Data Representation:** world snapshots and input packets are represented as JSON strings over the network
- **No Persistent Storage:** game data is keep is memory and does not need to persist across sessions
- **Eventual Consistency:** when clients receive updates from the host, they reconcile their local state, correcting any differences

Availability

Availability over Consistency (AP):

- **Client-side prediction**: clients continue to be responsive during a Network Partition

- **Eventual Consistency:** reached once a snapshot arrives from host

Fault Tolerance

- **Graceful Degradation:**
  - **Client failure:** removed from game
  - **Host failure:** all clients disconnected
  - **Lobby Server failure:** peers continue to operate without it

- **Timeouts:** abort operations if no response from Lobby Server

- **Heartbeats:** host need to periodically update their active lobby

Security

For simplicity reasons, these features were not included (needed in real world):

- Authentication and Authorization
- HTTPS

Implementation

Libraries:

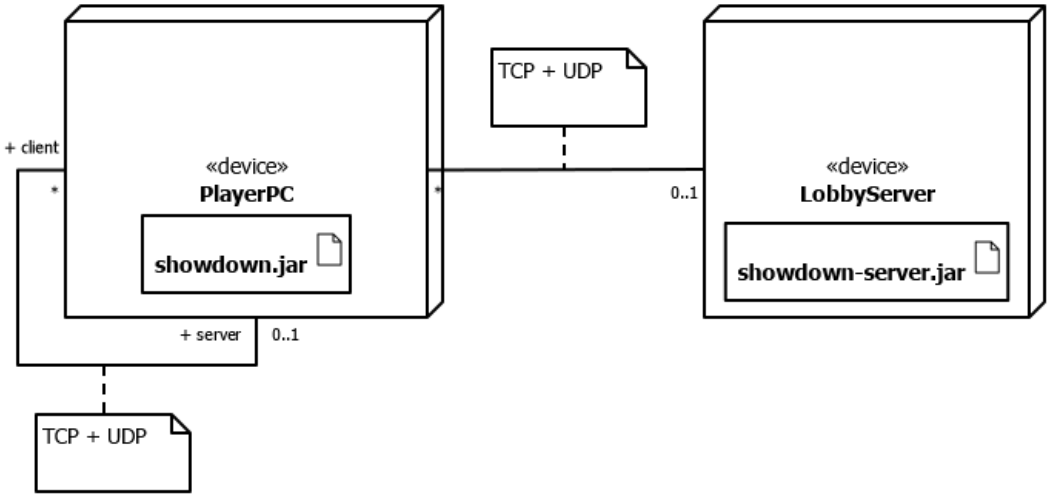
- Kotlin stdlib
- Kotlin Serialization (JSON)
- KTX (game features)
- Ktor (network)
- JUnit (testing)

Network Protocols:

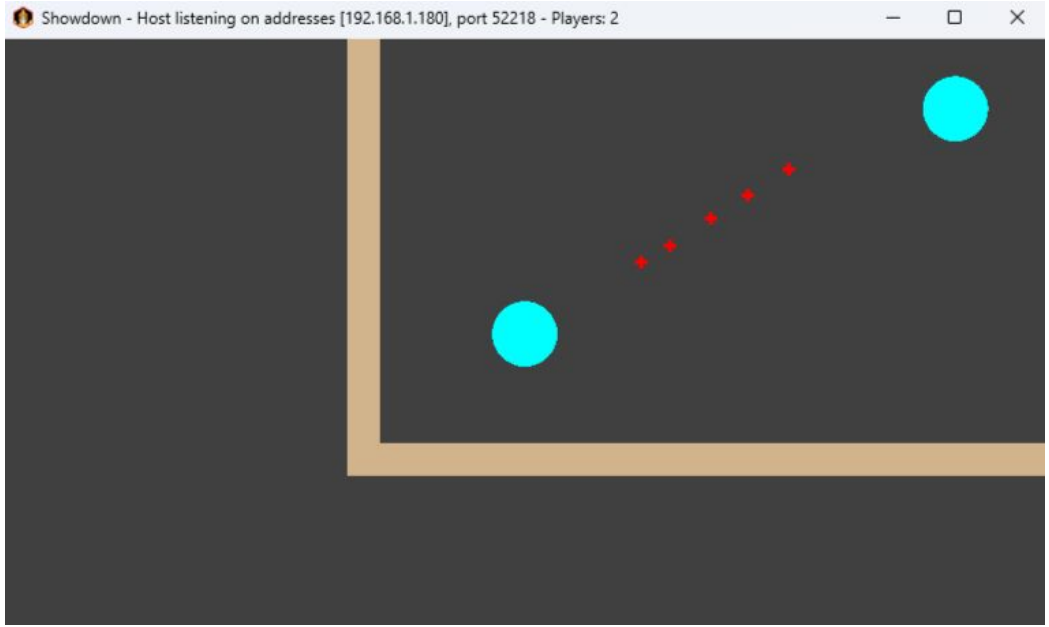
- **Host-Client:** TCP+UDP
- **Lobby Server Discovery:** UDP
- **Lobbies HTTP Server:** HTTP (over TCP)

Deployment

The system is a native Java application. Releases are packed in .jar files (easily deployable). **Java 25** needed.



Demo


Showdown - Host listening on addresses [192.168.1.180], port 52218 - Players: 2