CS5412 Project Guidelines

Theodoros Gkountouvas, Stavros Nikolaou

Project Organization

- ➢ Project Proposal (February 10)
- ➢ Intermediate Report I (March 13)
- ➢ Peer Reviews (March 26)
- ➢ Intermediate Report II (April 14)
- ➢ Final Report (May 6)
- ➢ Poster/Demo (May 7-9)

Deliverable 1: Project Proposal

- Why? Goals, Expectations
  - Target applications
- What? Analytical description (paragraph, bullets) of your project.
  - Dates of completion (tentative)
- How? How will you achieve each step:
  - Tools, cloud
- Evaluation (visualization)

Example: Marauder’s Map

- ➢ Proposal: We want to build a tool for online games between smartphone users.
- ➢ Why: Provide a distributed gaming platform for smartphone games such as Human vs Zombies
  - ○ Framework for real-time interactive multiplayer games.
  - ○ Service guarantees for developers (availability, fault-tolerance, consistent)

Example: Marauder’s Map

- Analytical steps:
  - Define a game model:
    - players:
      - state: location, human or zombie bit, ID
      - actions: move, use item, use environment object
      - interactions: zombies eat humans, humans use objects
    - environment:
      - levels or real map
      - items placed in the world humans can use
      - obstacles: landmines, electric fences etc.
    - events:
      - capture the flag,
      - get the grenade to the nest/safe-house etc.
    - rules

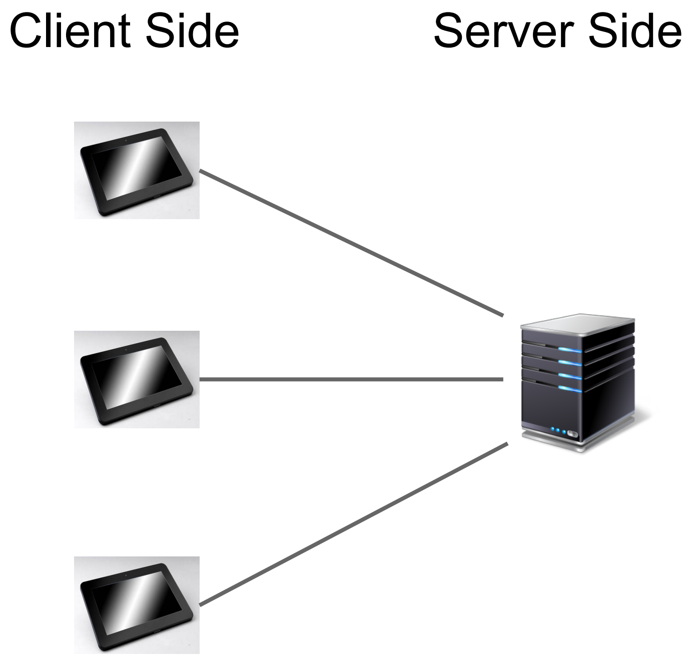
Example: Marauder’s Map

- Analytical steps:
  - Build server: given model create an instance of a game
    - Download PlayN game abstraction library
    - clients connect to/select/play games
    - Superimpose live data (collect[, decode], merge), apply updates on the fly -> new state
    - Compact and serve requesting clients
  - Build client: find server, select/connect to game, play
    - download Android development tool, install and learn how write apps which access sensor data
    - Implement connections, IDs, and participation
    - find how to stream info (locations, actions, items) with low bandwidth costs

Milestone 1: Application

1. Create maps for Human vs Zombies (map generator, 3-4 areas hand made).
2. Create interface and GUI for the game (PlayN).
3. Create server-client communication framework.
4. Clients should not witness delays >100ms when the systems operates without unexpected events (network overload, failures).

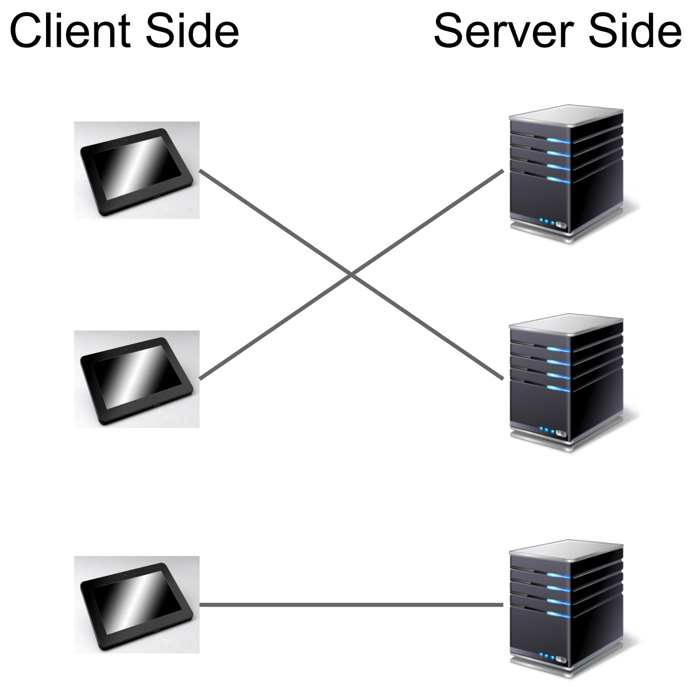
Implemented until 20 February.



Milestone 2: Availability

1. Use multiple servers and assign to them different game instances (load balancing).
2. Each server should operate on similar number of game instances as much as possible.

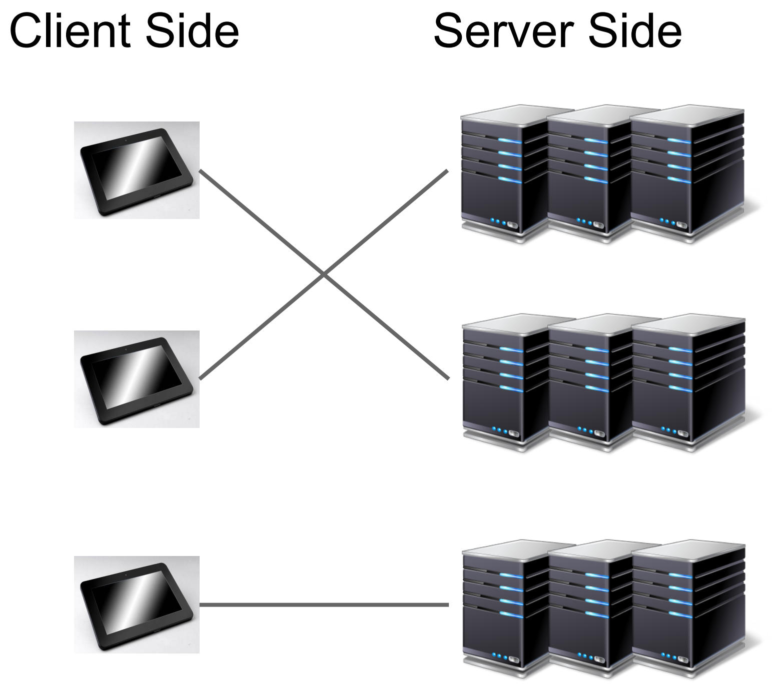
Implemented until 15 March.



Milestone 3: Fault T. - Consistency

1. Replicate servers to have fault tolerance (handle 2 failures).
2. Clients can communicate with every replica that handles their game's instance.
3. Clients should witness changes in a consistent way. (Isis tool)
4. The delay limit (100ms) should be maintained.

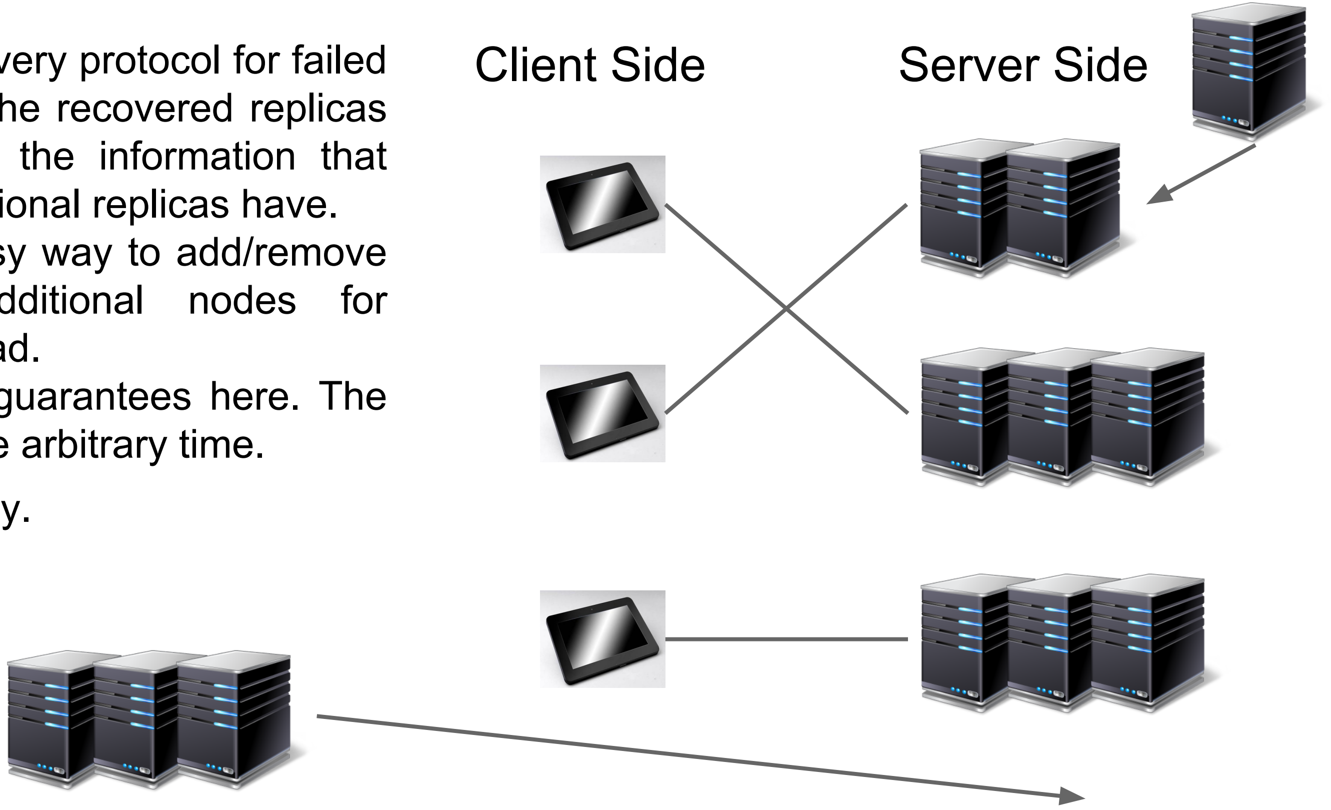
Implemented until 10 April.



Milestone 4: Recovery - Expansion

1. Implement a recovery protocol for failed server replicas. The recovered replicas should obtain all the information that the current operational replicas have.
2. Implement an easy way to add/remove replicas and additional nodes for partitioning the load.
3. No performance guarantees here. The operation can take arbitrary time.

Implemented until 1 May.



Evaluation Plan

- Demo:
  - Setup the service
  - Simulate the clients: connect, select various games, play
    - move around, use environment objects if found, avoid zombies, avoid obstacles
  - Visualization: Draw map on clients' machine with locations, show interactions (human becoming a zombie)
  - Run test cases:
    - Multiple clients single server (stress test for capacity checking)
    - Multiple servers; no replication (Load Balancing)
    - Replicated servers
      - introduce faults: availability, latency, consistency, load
    - Recovery:
      - load restoration

Evaluation Plan

- Poster:
  - architecture of our system
  - what technologies did you use and how (e.g. PlayN for building the game, Isis for consistency etc.)
  - Experiment results:
    - Scalability
    - Low latency
    - Availability under heavy load
    - Graceful failure handling
    - Recovery

Team Coordination

- ➢ Team mates: Theodoros, Stavros
- ➢ Theodoros: Single server implementation, game maps, and communication framework
- ➢ Stavros: multi-server implementation, GUI, consistency
- ➢ Both: game model, replication, recovery
- ➢ Meeting schedule (2 meetings per week, 3hours/meeting)

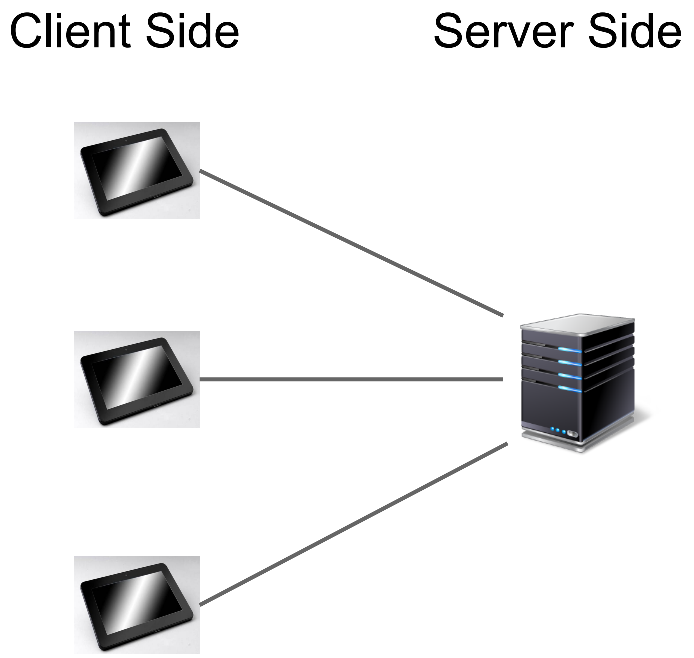
Intermediate - Final Reports

- ➢ State the progress of the project.
- ➢ Identify any changes made to the previous milestones.
- ➢ Everything specified as green or yellow status in the milestones for the final report should be demonstrated or explained in the Poster/Demo session.

Intermediate Report I: Milestone 1 - Application

1. Create maps for Human vs Zombies (map generator, 3-4 areas hand made).
2. Create interface and GUI for the game (PlayN).
3. Create server-client communication framework.
4. Clients should not witness delays >100ms when the systems operates without unexpected events (network overload, failures).

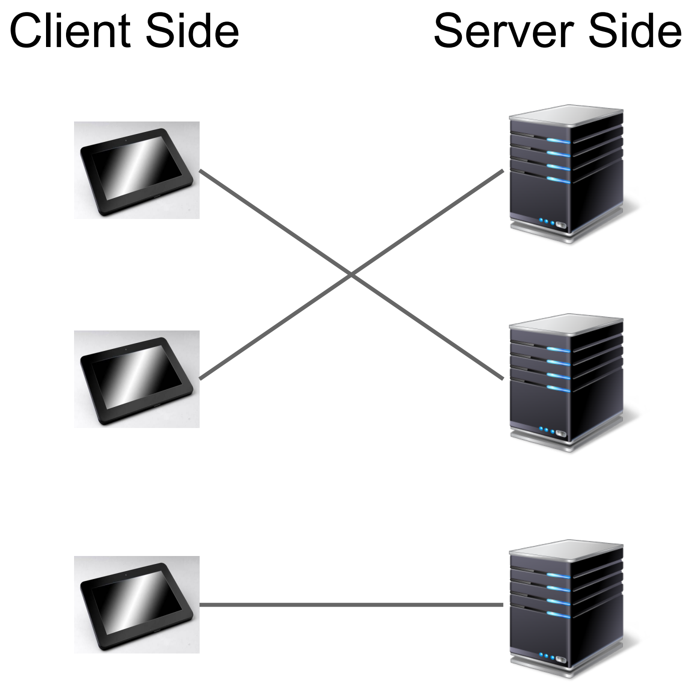
Implemented until 20 February.



Intermediate Report I: Milestone 2 - Availability

1. Use multiple servers and assign to them different game instances (load balancing).
2. Each server should operate on similar number of game instances as much as possible.

Implemented until 15 March.



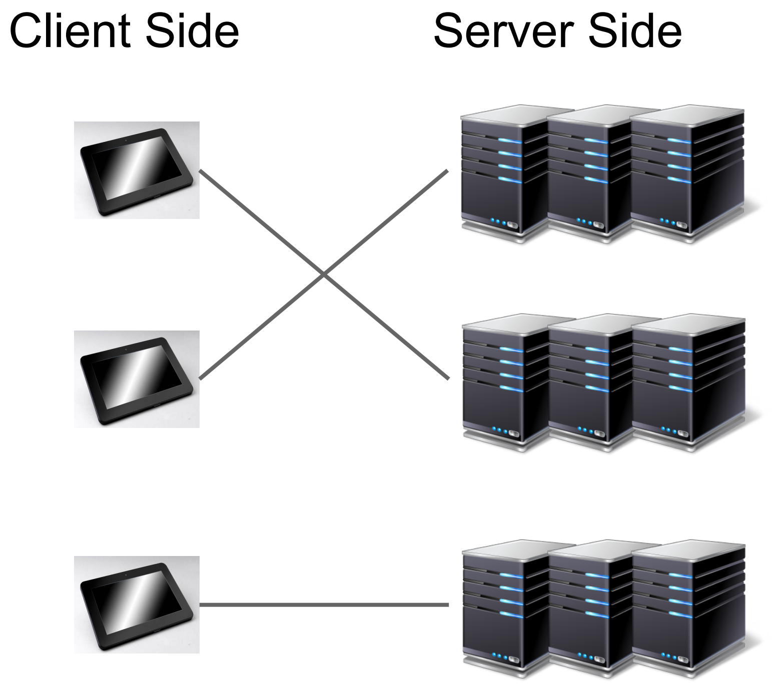
Intermediate Report I: Milestone 3 - Fault T. - Consistency

1. Replicate servers to have fault tolerance (handle 2 failures).
2. ~~Clients can communicate with every replica that handles their game's instance.~~ *

*Clients can communicate only with one client. Because of CAP theorem, we cannot tolerate network partition if we allow multiple connections.

3. Clients should witness changes in a consistent way. (Isis tool)
4. The delay limit (100ms) should be maintained.

Implemented until 10 April.



Peer review

- ➢ Each student:
  - 3 project proposals (as of intermediate report I) to review.
- ➢ Each review:
  - Summary
  - Good
  - Bad (nicely and politely expressed)
  - Recommendations (additions, removals, alterations)

Intermediate report II

- More green bullets
- What are you doing about the yellow bullets [a summary will suffice]
- What is your plans for your red tasks.

Any changes from the original proposal have to be justified.

Final Report

Only green bullets.

Justification for any dropped bullets.

Demo day

- Poster:
    - Architecture
        - Components, Interactions
    - Technologies:
        - how do they fit your project?
    - Features:
        - Capabilities of your system: Fault-tolerance, consistency guarantees, load-balancing, scalability
- Demo:
    - Showcase all the features described on your proposal and poster.

Questions?