

# **Playing Games with Consistent Mutable DHT Objects**

Athicha Muthitacharoen

Robert Morris

MIT Computer Science and Artificial Intelligence Laboratory (CSAIL)

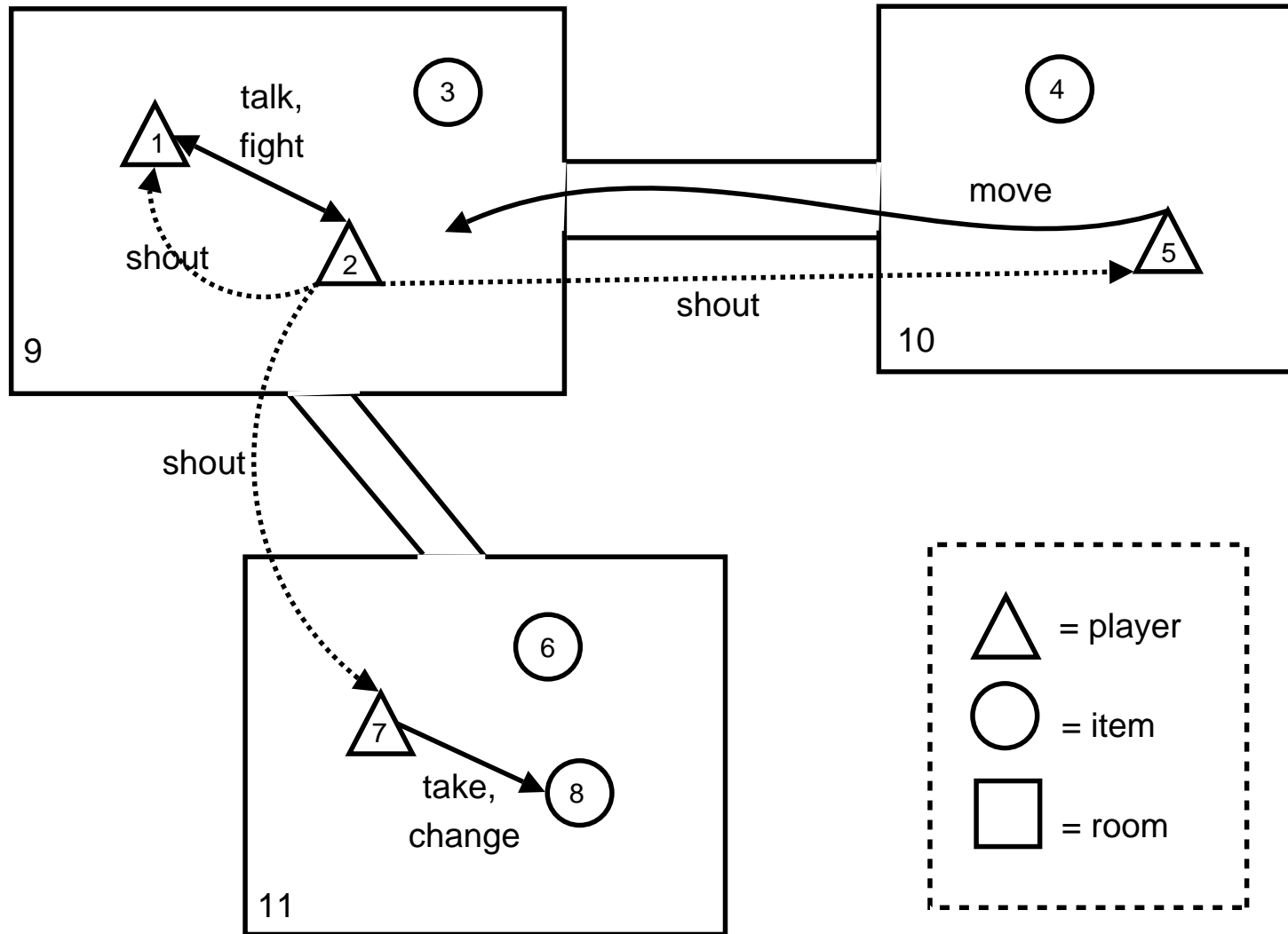
## The Case for Mutable DHT Data Support

- **DHTs have good support for immutable data.**
- **Many applications would benefit from mutable data.**
  - Read/write file systems, instant messaging, resolving web URLs, etc.
- **What do DHTs need to support mutable data?**
  - Transactions? Locking?
  - New interface? Data blocks or Object methods?

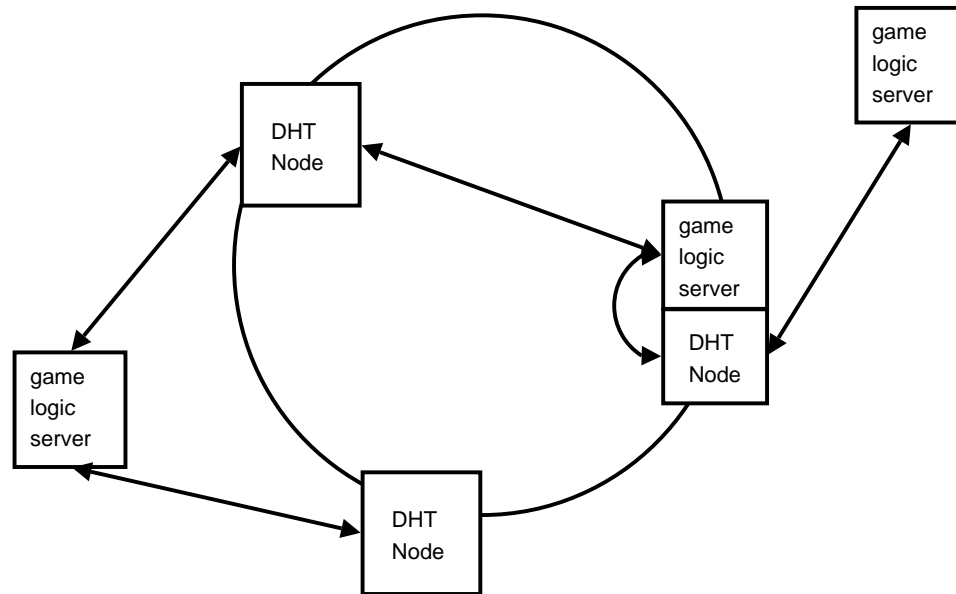
## How to find out application requirements?

- **Few applications store mutable data in the DHT.**
- **Use Multi-User Dungeon Games as a case study:**
  - Distributed game structure.
  - Make heavy use of shared mutable data.

# Game Characteristics



## DHT-based Game Design



- Each room, item, and player is implemented as a mutable DHT object.
- Replicate objects for availability and fault-tolerance.

## Game Semantic Requirements

- **Atomic multi-step operations:**
  - Atomic Read/Write to individual DHT mutable objects.
  - Atomic Read-Modify-Write to individual mutable objects.
  - Nothing else.
- **Atomic Set Insertion and Deletion.**
  - To efficiently implement players entering/leaving a room.

## Achieving Read/Write Atomicity

- **Object replication, read and write follow the Etna protocol.**
  - Replicates an object at its  $k$  immediate successors.
  - If the current  $k$  immediate succs do not match the replica configuration:
    - change the replicas to the current succs.
    - do so consistently by using Paxos to get the current configuration to agree on the next configuration.
  - Read and write to an object start at the primary.

## Achieving the Rest

- **Read-Modify-Write Atomicity:**
  - Extend writes in Etna to only accept writes that contain the latest version number.
- **Add a set insert and a set delete operation to the DHT API.**

## Summary

- **Investigated application requirements for mutable DHT data.**
- **Case study application: DHT-based MUD game.**
- **Defined new DHT APIs for mutable data:**
  - `write(k, v)`: Atomic write.
  - `read(k)`: Atomic read.
  - `write(k, v, i)`: To implement read-modify-write.
  - `set-insert(k, e)`, `set-delete(k, e)`.
- **Implemented initial prototype of the game and APIs.**