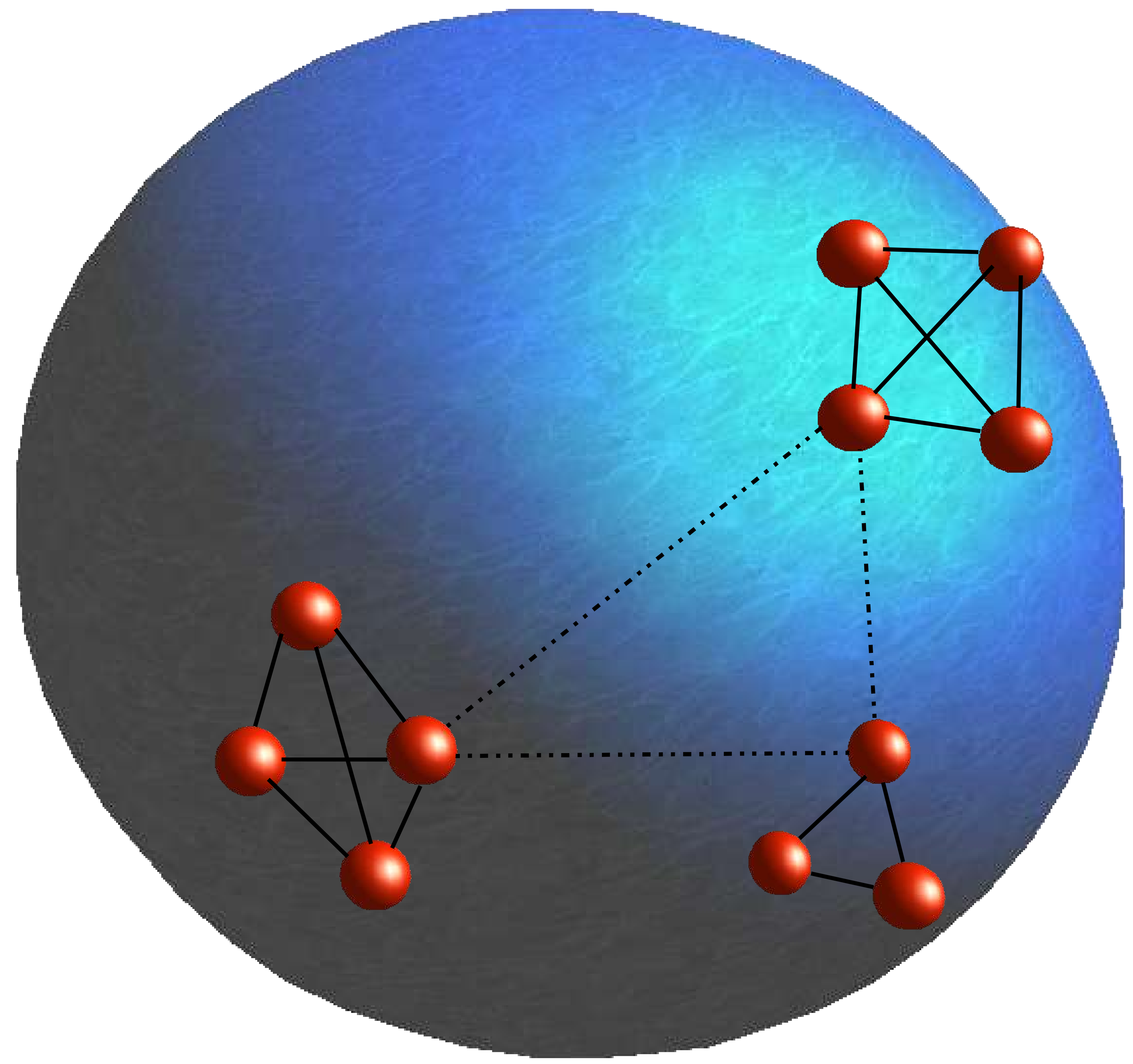
A Distributed Architecture for Massively Multiplayer Online Games

Chris GauthierDickey, Daniel Zappala and Virginia Lo

Benefits of a Distributed Architecture

- Any player can start an MMG without the incredible investment in hardware and bandwidth required by the client/server architecture.
- Players can send messages directly to each other, thereby reducing delay.
- Localized congestion centered at a server is eliminated.
- Storage capacity of the game is increased with the number of players.
- Processing power from individual machines can be harnessed for game computations.
- A distributed architecture has the potential for better scalability.



Players join NEO groups (solid lines) and communicate through leaders between groups (dashed lines). NEO groups are joined in a hierarchy to reduce bandwidth and propogate events.

Communication

The communication component is the building block of the game. We use NEO[1], a cheat-proof, low-latency, peer-to-peer protocol designed for games.

- Players form NEO groups based on locality in the virtual world. NEO orders events locally in each group and prevents cheating at the protocol level.
- To scale to a large number of players, each NEO group elects one or more leaders which then communicate with other groups to form a hierarchy of groups.
- The virtual world is subdivided by zones, and further by an efficient organizational structure such as an octree. This structure allows events, which have a larger scope than a NEO group, to be propagated to other players.

Authentication

The autentication component ensures that only authorized players can participate and bans cheating players or expired accounts from the game.

- A trusted authentication server digitally signs the key triple: (player_id, public_key, expiration_date). These keys are published on a DHT.
- Players can query the DHT for the public key of other players and to determine if another player is still a valid player.
- The trusted server can also publish a special key (player_id, *banned*), that informs other players if someone has been banned from the game.
- If a player cheats, proof can be presented to the trusted server to ban the player from the game.
- The trusted server uses a DHT for scalability purposes: once a triple is published, players no longer have to contact the authentication server.

Storage

The storage component is designed to provide persistent, long-term storage. All objects in the game have digital signatures that verify their validity and prevent tampering. We categorize data into two basic types: permanent and ephemeral.

- Permanent data: data which must *always* be available to a player. Permanent data can further be broken down into *existential* and *participatory* data.
  - Existential data always exists even when the player is offline, such as a player's house.
  - Participatory data is data which exists only when the player participates in the game, such as their player inventory.
- Ephemeral data: data which is stored long term for game effects, but does not alter game play if it is lost. Objects discarded by a player are considered ephemeral data.

Permanent data is stored and backed up by interested parties, while ephemeral data is stored over a DHT or distributed file system.

Computation

The computation component schedules remote process execution on players' machines.

- Processes, such as AI execution, are scheduled on randomly chosen machines outside of the NEO group interacting with the AI.
- More than one machine is chosen per process to prevent cheating. To prevent cheating, we assume that c of n total players will collude and we replicate r times such that our tolerance T is low enough by the given formula:

$$\frac{\binom{c}{r}}{\binom{n}{r}} < T$$

- NEO provides consistency and event ordering locally so that all clients are synchronized in a global point of view.

References:
1. C. GauthierDickey, D. Zappala, V. Lo, J. Marr, "Low-Latency and Cheat-Proof Event Ordering for Peer-to-Peer Games", NOSSDAV, June 2004.

chrisg@cs.uoregon.edu / http:www.cs.uoregon.edu/~chrisg

Network Research Group @ University of Oregon