Cheat-Proof Playout for Centralized and Distributed Online Games

IEEE InfoCom'01 Paper by
Nathaniel E. Baughman and Brian Neil Levine

CPSC 538A Presentation: Georg Wittenburg

Background of the Paper

- Authors:
  - Nathaniel E. Baughman – BS in 1998 @ Ohio Northern University
  - Brian Neil Levine – Professor at UMass since 1999

What is Cheating?

- What is fair?
  - *"[…] an online game is* fair *if state as perceived by every player is consistent with every other player's expectations, including the server, as defined by the game rules."*
- Cheats take advantage of a technical weakness to gain an unfair advantage over another player.
- Cheats are game (genre) dependant *and* implementation dependant.

Some Background on Security

- The three major goals of information security are:
  - Confidentiality – Data is protected against spying.
  - Integrity – Data is protected against manipulation.
  - Availability – Data (or services) can be accessed.

Some Background on Security

- Choices need to be made on how to reach these goals.
- The most crucial single aspect in this design process is what one knows about potential attacks.
- Hence we need to model the attacker.

Modelling the Attacker

Common characteristics of hackers are:

Modelling the Attacker

Common characteristics of hackers are:

- They are incredibly smart.

Modelling the Attacker

Common characteristics of hackers are:

- They are incredibly smart.
- They dress in black.

Modelling the Attacker

Common characteristics of hackers are:

- They are incredibly smart.
- They dress in black.
- They know Kung-Fu.

Modelling the Attacker

Common characteristics of hackers are:

- They are incredibly smart.
- They dress in black.
- They know Kung-Fu.

- So the typical hackers are:

Modelling the Attacker (2)

- Attackers are characterized by their capabilities:
  - read, write, and block messages
    - on parts of the network
    - on the entire network
  - modify the client
  - read and write data on the server
  - deny service (client or server side)

Centralized and Distributed Games

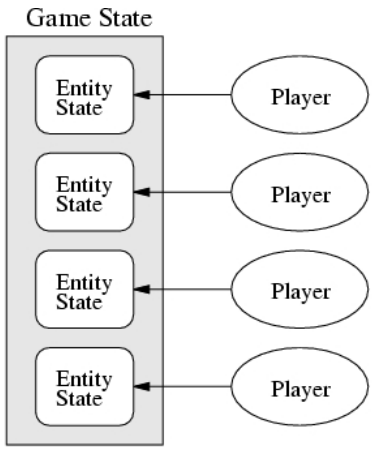


Fig. 1. Game state partitioning

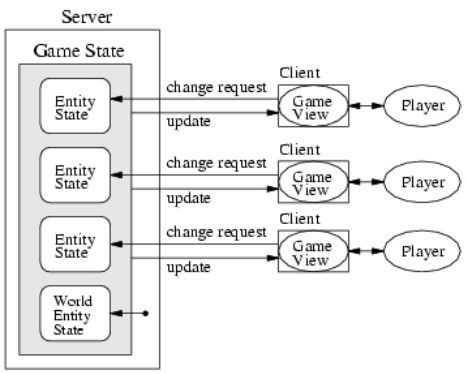


Fig. 2. Centralized-control client-server

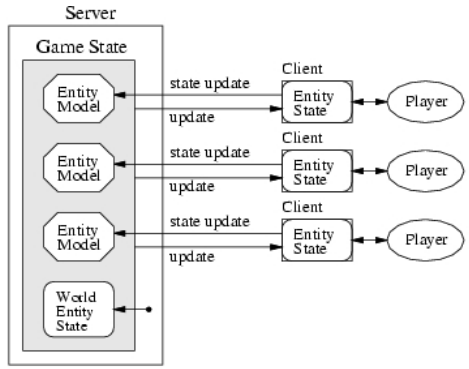


Fig. 3. Decentralized-control client-server

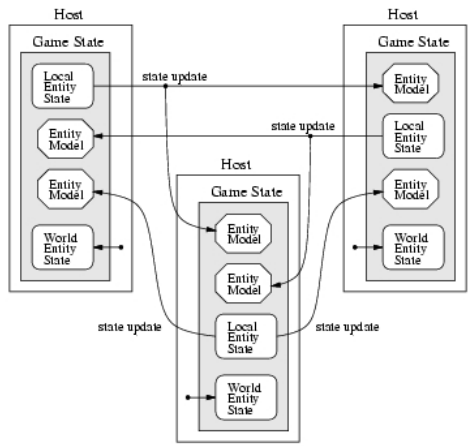


Fig. 4. Distributed

Attacks and Defenses

- Suppress-Correct Cheat
  - In a dead reckoning environment, gain advantage by delaying your actions.
- Lookahead Cheat
  - For simultaneous actions, gain advantage by being the last player to decide.
- Verifying Secret Possessions
  - Verify current claims (e.g. possession) based on previously secret actions.
- Verifying Hidden Positions
  - Verify information (e.g. a player's position) without giving that information away.

Suppress-Correct Cheat

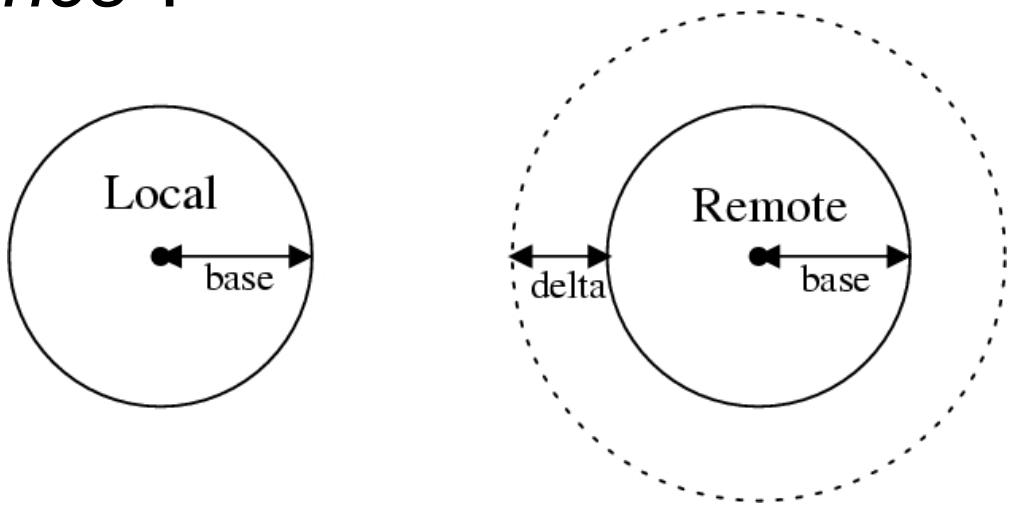
- Bucket implementation, that assumes disconnect after *n* lost packets; compensates with dead reckoning.
- Delay your replies in a way so that you miss only *n-1* packets.
- In your reply you can take other players' actions into account, thus "seeing into the future."

Lookahead Cheat

- In turn-based games, delay your action until you have received the actions of all other players.
- *Proposed Solution:* Lockstep Protocol
  - Instead of sending actions, players send a cryptographic hash of their intended action.
  - Only after the hashes of all other players have been received, the plain text actions are sent.
  - This has performance issues as it effectively synchronizes all players.

Lookahead Cheat - Optimization

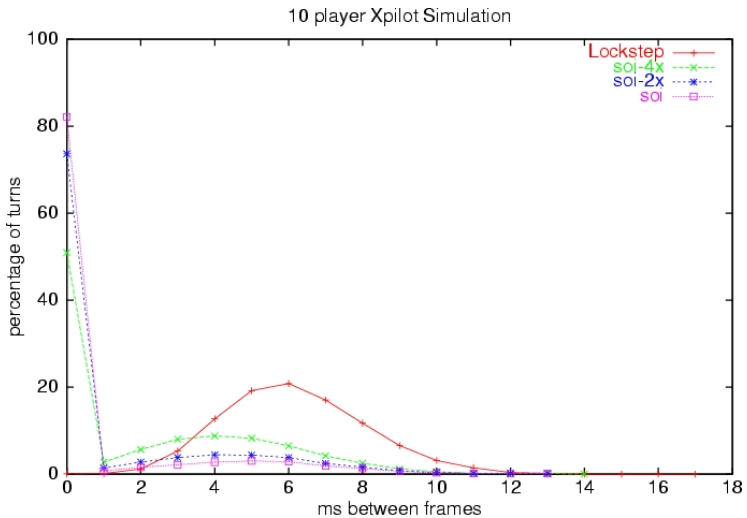
- Optimization: Only synchronize with players whose actions can affect you.
- Model possibility of interaction with *"Spheres of Influence"*:

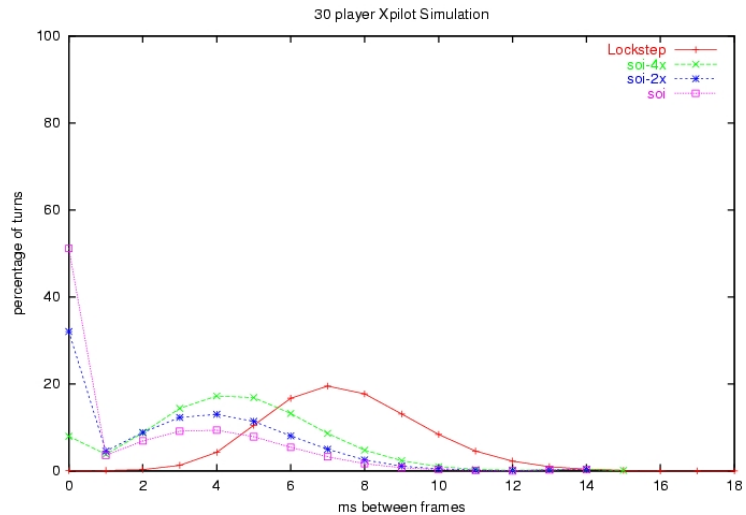


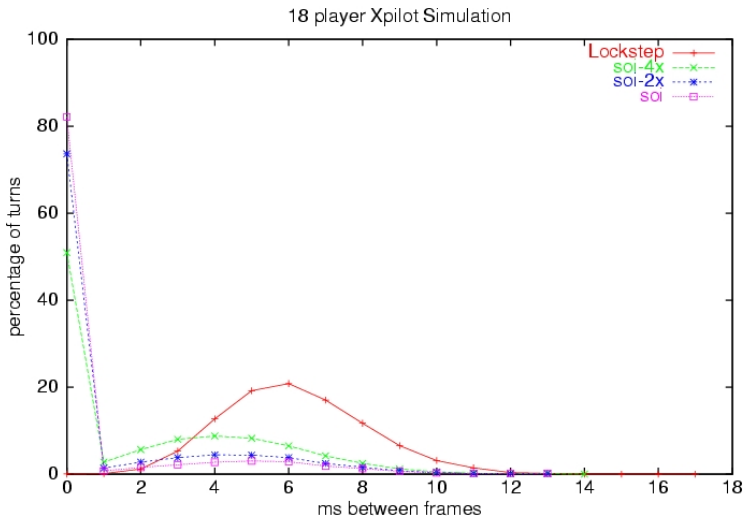
Lookahead Cheat – Optimization (2)

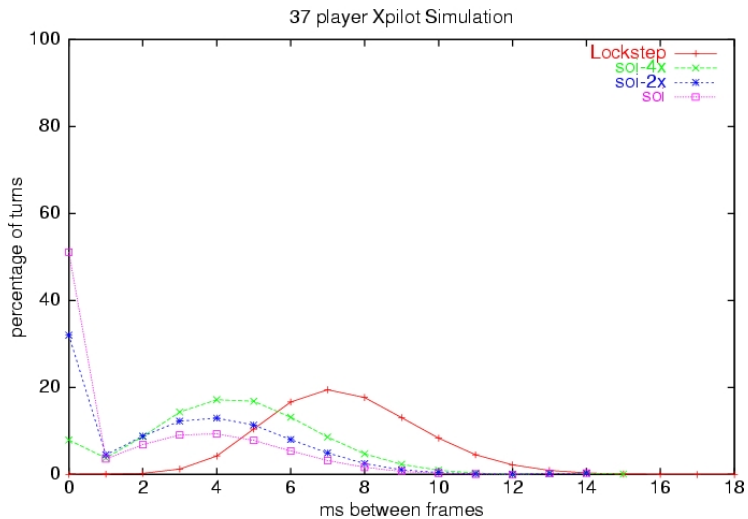
- Based on current position / state, the number of steps is calculated that it takes to reach another player's sphere of influence.
- Gameplay proceeds asynchronously until spheres intersect or could intersect during the next turn.
- Players with intersecting spheres synchronize as before.
- Additional benefit: Packets may be lost as long as spheres have a safe distance.

Performance Analysis









Verifying Secret Possessions

- Players need to verify that their current state was reached by legal means, e.g. to have item X, you need to find it in the past.
- *Proposed Solution:*
  - Have a designated entity (“*Logger*”) store cryptographic hashes of critical parts of a player’s current state.
  - Make this information available when required in the future.

Verifying Hidden Positions

- A piece of information (e.g. player's position) needs to be compared without revealing it.
- *Proposed Solution:* Basic Cryptography
  - Use a commutative cryptosystem.
  - Exchange random numbers, XOR them, add secret, encrypt, and trade results.
  - Due to commutative nature of the cryptosystem, repeated encryption with own key will yield a comparable value.

Conclusion

- Four attacks / problems were discussed.
- Three solutions were proposed.
- One solution was evaluated extensively.

Evaluation

- The process of cheating was not modelled, the definition of fairness is weak.
- The description of possible attacks is helpful.
- The proposed solutions have merit, pending evaluation on a wider variety of games.
- As a side note, I had problems with scope and structure of the paper.

References

- SySL Reading Group Presentation by Chris Chambers @ OGI
  www.cse.ogi.edu/sysl/readings/slides/CheatProof.ppt

- “Cheatproof Playout Summary” by Chris GauthierDickey @ UOregon
  www.cs.uoregon.edu/~chrisg/summaries/baughman01.pdf

The End

Discussion

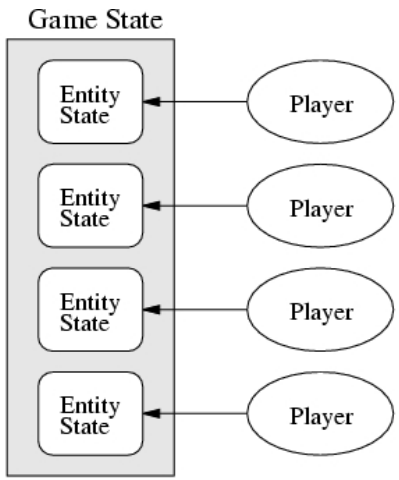


Fig. 1. Game state partitioning

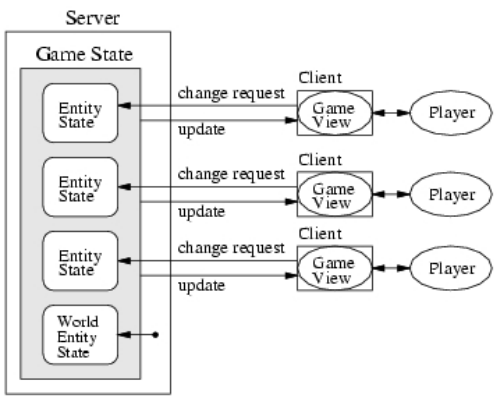


Fig. 2. Centralized-control client-server

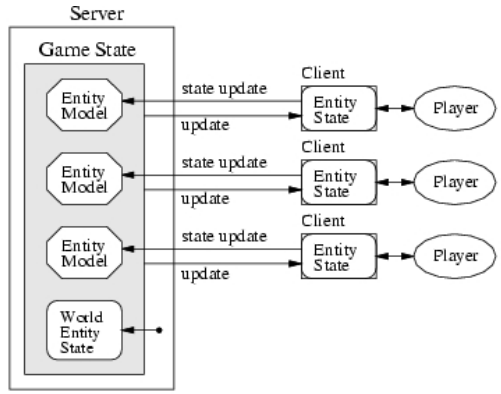


Fig. 3. Decentralized-control client-server

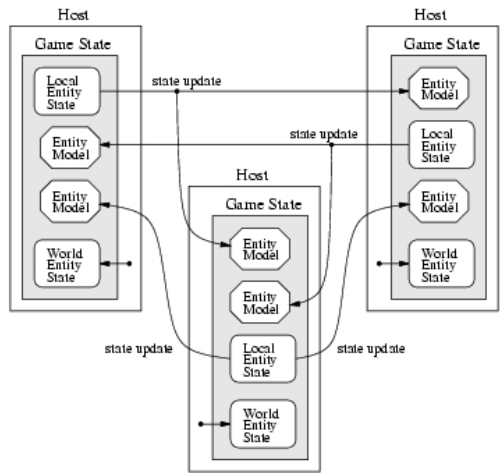


Fig. 4. Distributed