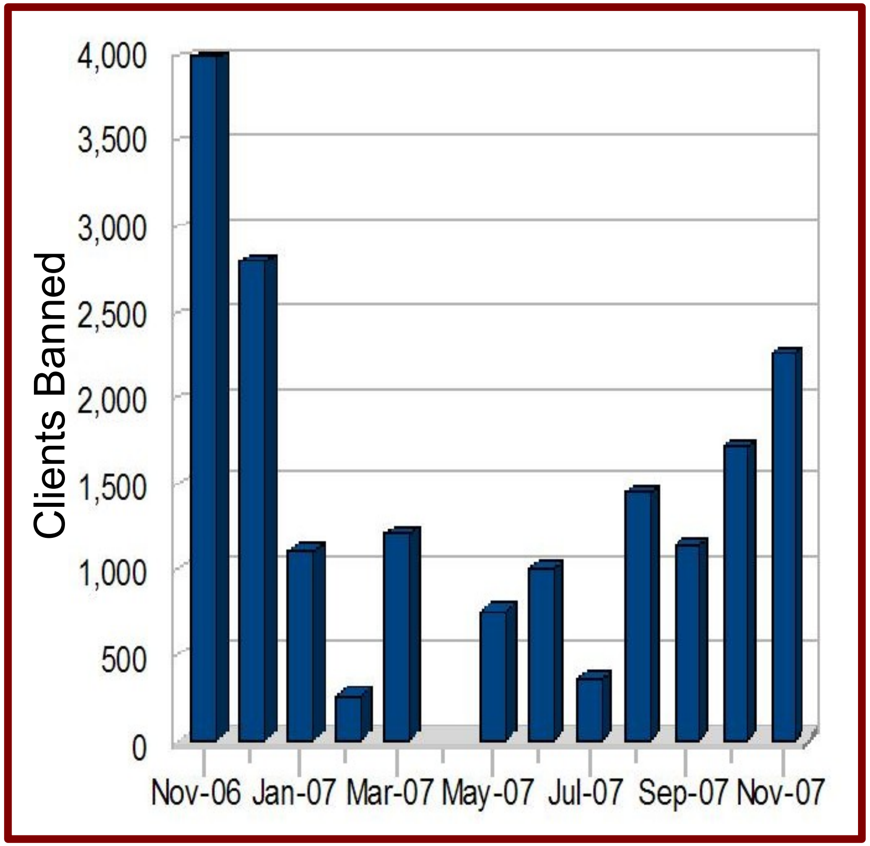
A Peer Auditing Scheme for Cheat Elimination in MMOGs

Josh Goodman — jgoodm7@cs.mcgill.ca
Clark Verbrugge — clump@cs.mcgill.ca
McGill University
School of Computer Science
Montréal, Canada

Table of Contents

Introduction: *Cheating Impact*

- Cheating in MMOGs can have an **important** impact
- Example: cheaters **banned** for using the "Movement Enhancing Hack" in Final Fantasy XI
- There is a FFXI cheating task force



Introduction: *Current Solutions*

- Current Cheat Elimination Solutions are:
  - Manual:
    - Log reviewing
    - Complaint based
  - Methods that focus on a specific cheat
  - Using to Client Server (C/S) models
- **But:** harder to implement and limit scalability (C/S over P2P)

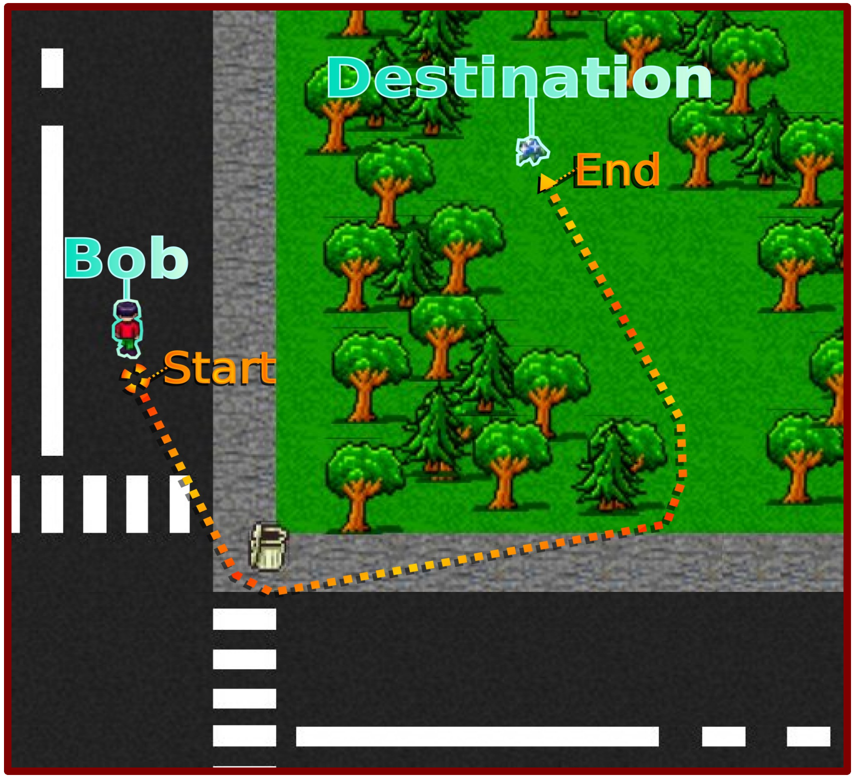
Introduction: *Other Solutions*

- Automatic, scalable cheat resistance is very desirable, however:
  - Cheating domain: it is hard to define exactly what *"cheating"* is
  - Performance: a solution must be scalable, having low overhead
  - Accuracy: a solution should punish **only** cheaters
    - Should avoid mistaking a trustworthy client as a cheater

      False positives

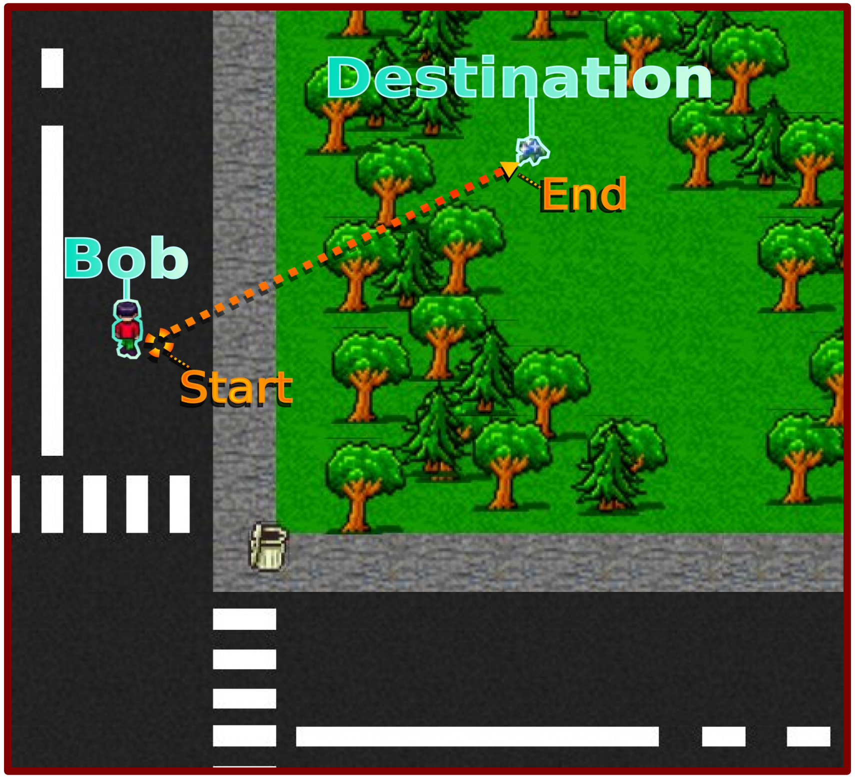
Introduction: *Motivating Example*

- McGill MMOG Testbed: ***Mammoth***
- Problem:
  - Path-finding done **client side**
  - Allows for **abuse** / **cheating**
- Example:
  - Normally, **Bob** finds the path leading to the **destination**



Introduction: *Motivating Example*

- McGill MMOG Testbed: ***Mammoth***
- Problem:
  - Path-finding done **client side**
  - Allows for **abuse** / **cheating**
- Example:
  - **Bob** can also cheat sending a path that ignores obstacles



Introduction: *Motivating Example*

- Alternate approach
  - Path-finding done **server** side
  - Lowers chances for **abuse** / **cheating**
  - Path-finding is expensive
  - Can cause a bottleneck
- Idea: Marry both approaches
  - Use **P2P** for <u>load management</u>
    - Use **Peers** to resolve path requests
  - Use **C/S** for <u>cheat resistance</u>
    - Use **server** as an arbiter

Hybrid Solution: *The IRS Model*

- Approach MMOGs with a Hybrid Model
  - Try and create a network model that is the best of both worlds
- The *IRS* hybrid model:
  - Uses a centralized server for verification / persistence
  - Uses P2P communication for message handling
- Goal:
  - Reduce the occurrence / accessibility of **Cheating**
  - Reduce the computational requirements of the Server

Hybrid Solution: *Cheat Detection*

- Detection of suspicious behaviour
  - Use *peer auditing*
    - Send copies of requests to an extra client
    - Compare both answers
  - If both answers are the same
    - Assume they are both correct
  - If both answers differ
    - Assume either is cheating
    - Compute the **true** result and compare both answers

Hybrid Solution: *Cheater Identification*

- There are many causes for suspicious behaviour
  - Hardware differences
  - Communication failure
  - Cheating
- Differentiating between errors and cheating:
  - Use a ***Trust Metric:***
    - Group the failures by severity
    - Count the number of failures against successes
    - Since random hardware or communication errors are rare
    - Use this to determine if a client is likely cheating

Hybrid Solution: *Summary*

- We propose the IRS model as a cheat reduction solution that is:
  - *Scalable with Low overhead:* allows P2P communication and reduces server **CPU load**
  - *Automatic*: peer auditing allows for the identification of suspicious behaviour
  - *Accurate:* Trust based scoring differentiates between random *errors* and *cheating behaviour*

Design: *Overview*

- The IRS Model incorporates the following:
  - Communication Model
  - Message Verification Scheme
    - Auditing
    - Monitoring
    - Quick Testing
  - Trust method for identifying cheaters
  - Disciplinary system that removes malicious clients

Design: *Components*

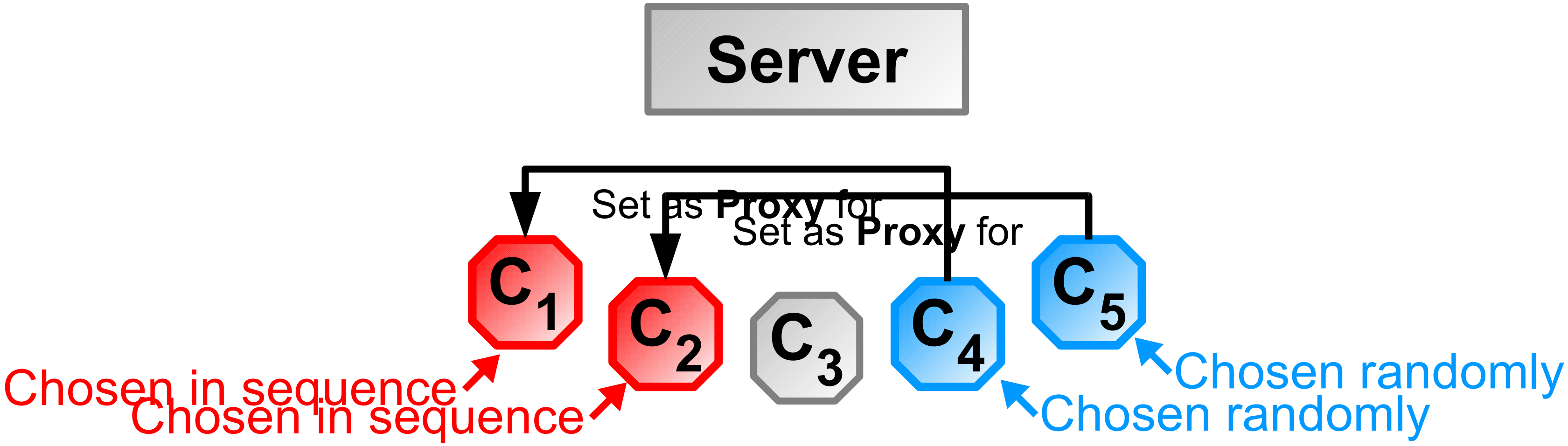
- Components of the IRS Model:
  - **Server**
    - Acts as arbiter for clients
    - Manages gamestate
    - Handles Login
  - Monitors **M_1**
    - Owned by the game providers
    - Monitor and verify audits
  - Clients **C_1**
    - The game players
    - Acts as a **proxies**
    - Has a client **proxy**

Design: *Load Distribution Protocol*

- The IRS model's *load distribution protocol* is:
  - P2P oriented:
    - *<u>Proxies</u>* are clients that compute message results for others
    - Each client has a proxy and acts as a proxy for others
  - C/S oriented:
    - Server handles login
    - Result monitoring
    - Gamestate maintenance
    - Message relaying
    - Matching clients and proxies

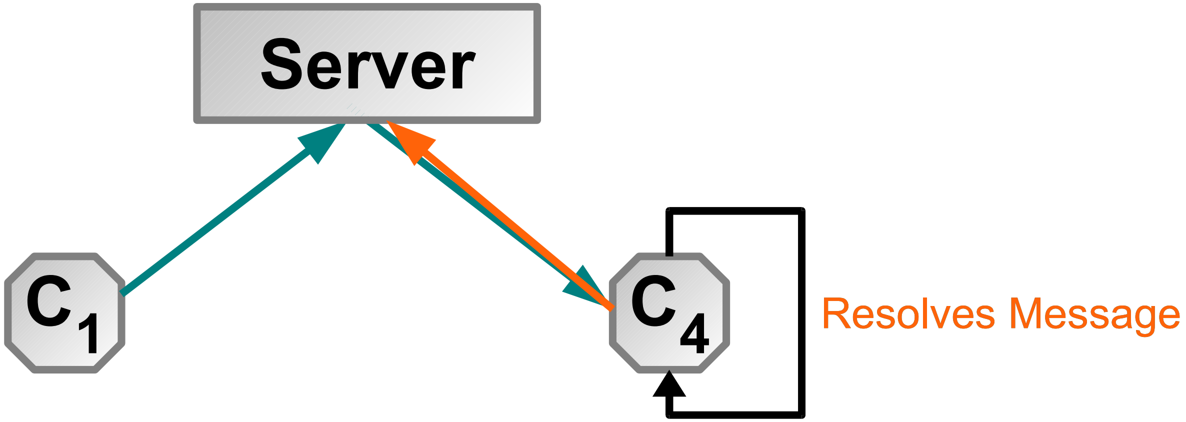
Design: *Load Distribution Protocol*

- 4 Protocol Phases (Server)
  - 1. Proxy Assignment
    - Randomly matches clients to proxies
    - Proxies are assigned by server at regular intervals



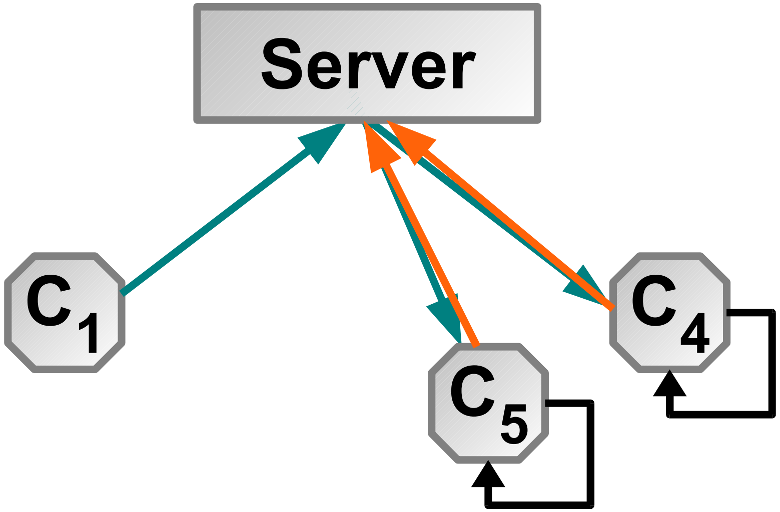
Design: *Load Distribution Protocol*

- 4 Protocol Phases (Server)
  - 2. Message Relaying
    - Server relays path finding requests from a client to its proxy
    - The proxy is responsible for resolving said request



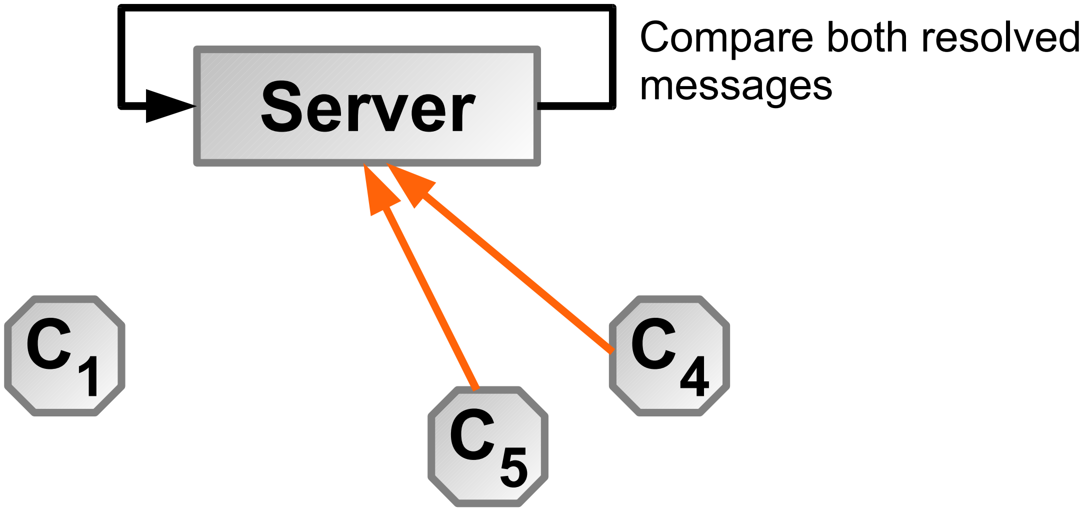
Design: *Load Distribution Protocol*

- 4 Protocol Phases (Server)
  - 3. Peer Auditing
    - The server randomly audits clients by simultaneously sending the request message to an extra client (co-auditor)



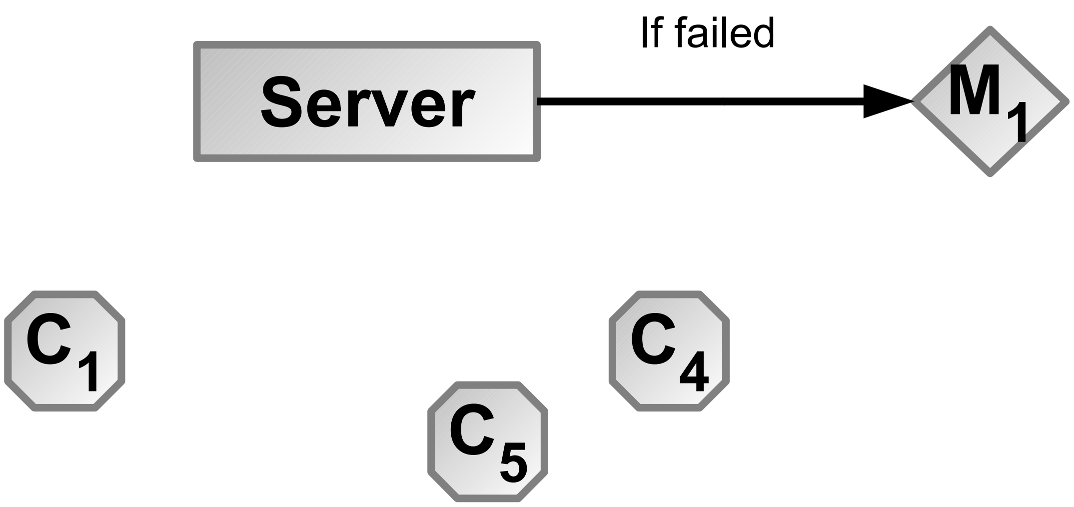
Design: *Load Distribution Protocol*

- 4 Protocol Phases (Server)
  - 3. Peer Auditing
    - The proxy's message is quick-tested and forwarded
    - The server then compares both resolved messages



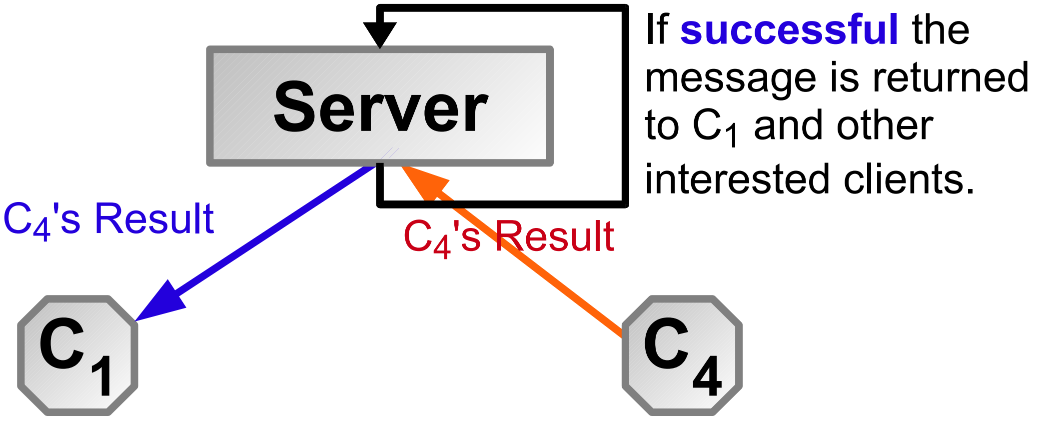
Design: *Load Distribution Protocol*

- 4 Protocol Phases (Server)
  - 3. Peer Auditing
    - If the comparison fails, the audit is sent to the monitor



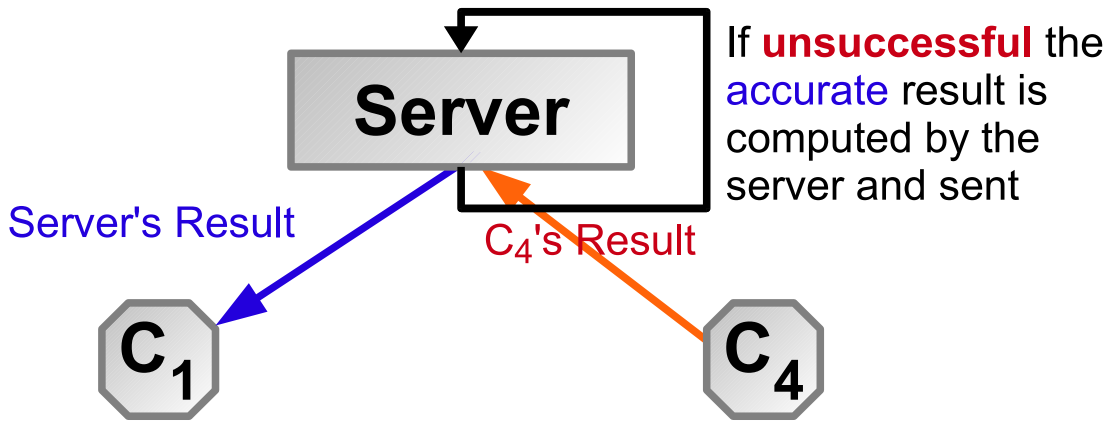
Design: *Load Distribution Protocol*

- 4 Protocol Phases (Server)
  - 4. Message Handling
    - Quick Testing of resolved messages
    - Relaying the resolved message to appropriate clients



Design: *Load Distribution Protocol*

- 4 Protocol Phases (Server)
  - 4. Message Handling
    - Quick Testing of resolved messages
    - Relaying the resolved message to appropriate clients

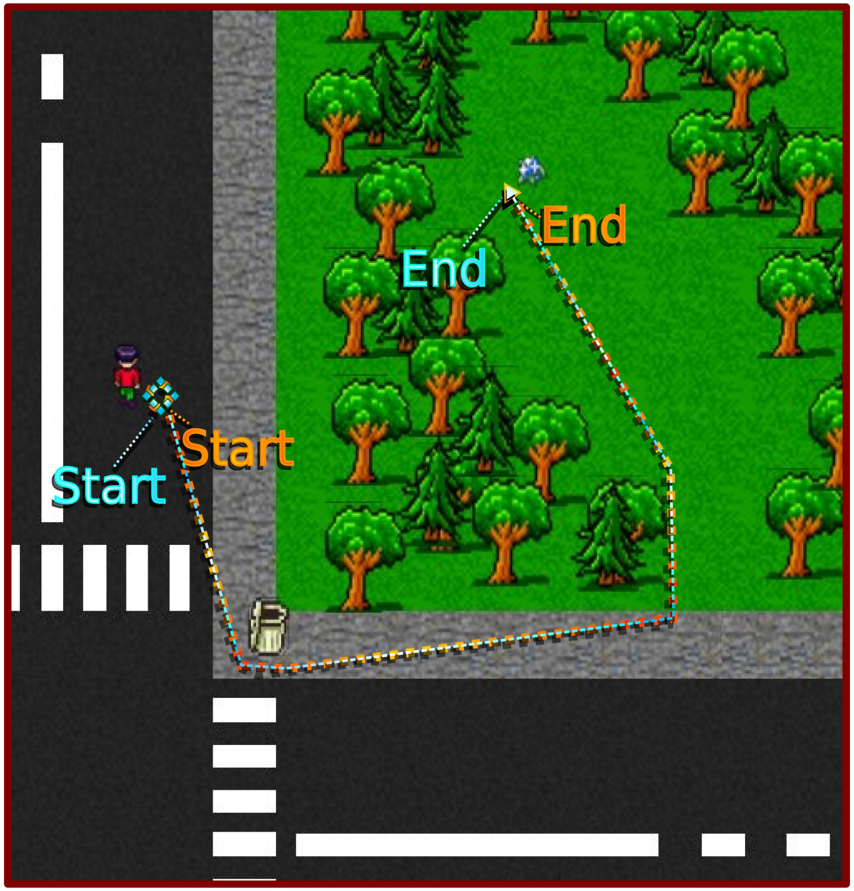


Design: *Auditing Scheme*

- Peer Audits
  - Examine resolved messages returned by proxies
  - Started randomly
  - Opened during the message relaying phase
  - Compared at a **later** time
- Audits yield the following:
  - Identical
  - Equivalent
  - Inequivalent
  - Infeasible

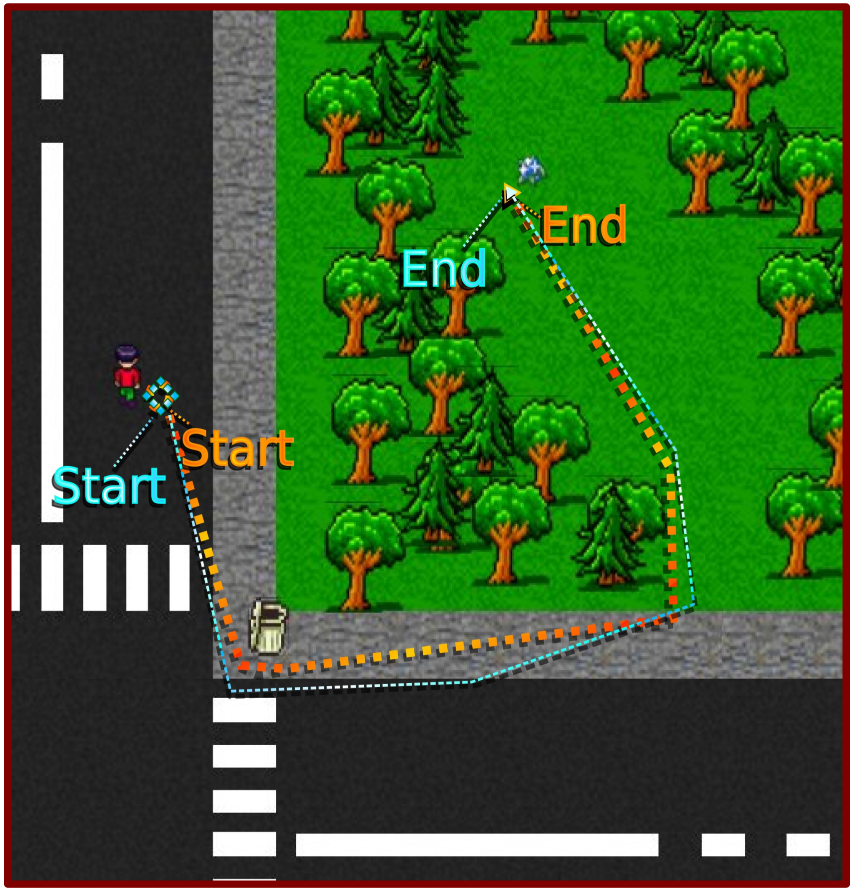
Design: *Comparison Types*

- **Identical:**
  - All points are coincidental
  - This is the **best** possible comparison result.



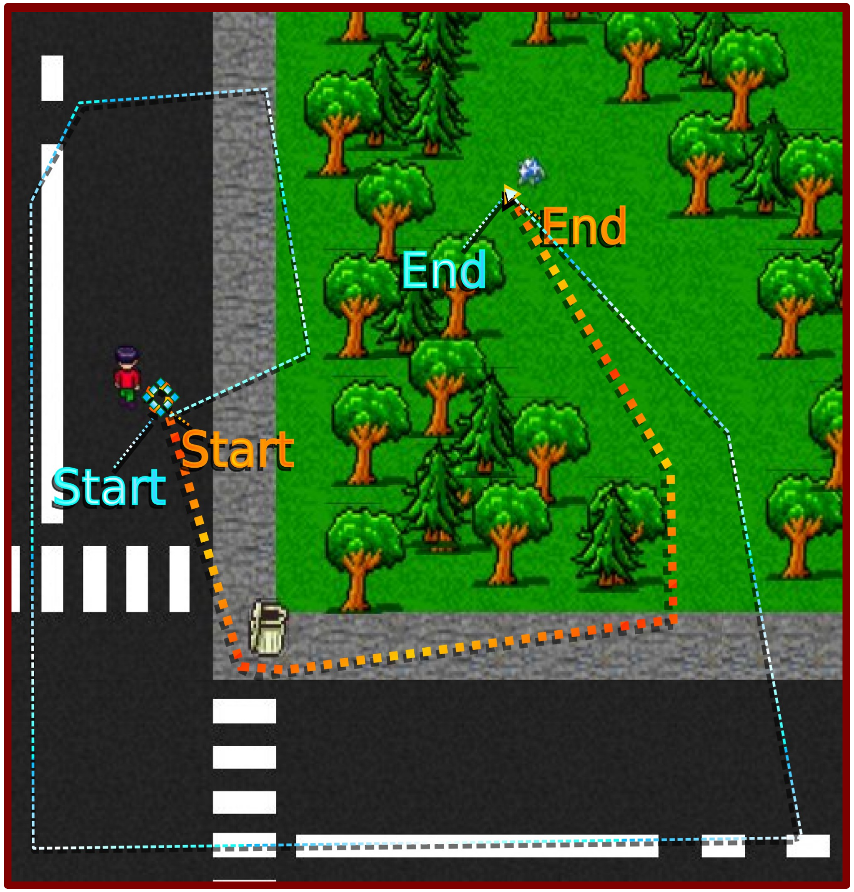
Design: *Comparison Types*

- **Equivalent:**
  - Same **starts** point
  - Same **ends** point
  - Similar **lengths**
  - Regarded as a **positive** result



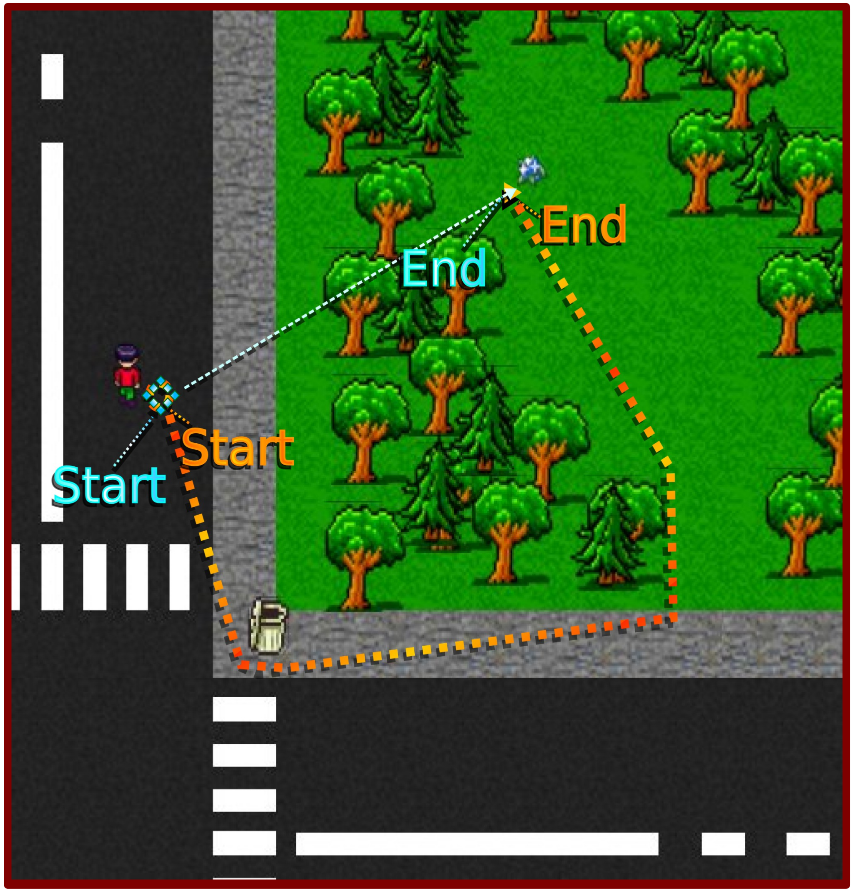
Design: *Comparison Types*

- **Inequivalent:**
  - Different **start** points <u>or</u>
  - Different **end** points <u>or</u>
  - Dissimilar **lengths**
  - Regarded as a **negative** result



Design: *Comparison Types*

- **Infeasible:**
  - *Violates game rules*
  - Passes through obstacles or
  - Leads to inaccessible areas
  - This is the **worst** possible comparison result



Design: *Monitoring*

- Failed audits are subjected to *Monitoring*
  - *Monitors* are controlled by game company
  - Monitors resolve the original request message
  - Compares its result to the two results contained in the audit
  - Determines which clients are responsible for the audit failure

Design: *Trust*

- *Trust*
  - Designed to distinguish cheats and error
  - History based
  - Identical and equivalent messages cause an increase
  - Inequivalent and infeasible messages causes a drop
  - Can require a discount factor in order to forget older infractions

Design: *Quick Testing*

- *Quick Testing* eliminates worst-case inaccuracies
  - Computed cheaply
  - Can only determine if a message is infeasible or not
  - Is used **before** relaying messages back to clients
  - If failed, the server will compute its own resolved message

Design: *Disciplinary Action*

- Disciplinary Action
  - Booting: when an inaccuracy is caught
    - Temporary
    - Early warning
    - Breaks up consecutive cheating
  - Banning: when trust falls below the *ban threshold*
    - Permanent
    - Ultimate deterrent
    - Lowers the number of cheaters in the system

Results

- Cheat reduction tests in 2 environments
  - Static client base
  - Dynamic client base
- Load analysis
  - Determine CPU load reduction
  - Bandwidth increases
  - Costs of cheat reduction

Results: *Parametrization*

- Client Simulation:
  - Legit Clients:
    - Trustworthy clients
    - Never attempt to cheat
    - Have a small chance to fail
  - Are 99% accurate

Results: *Parametrization*

- Client Simulation:
  - Griefers:
    - Want to disturb others
    - Cheat in order to ruin other's game play
    - Example: sending clients in the wrong direction
  - Will “grief” 50% of the time, returning inequivalent results

Results: *Parametrization*

- Client Simulation:
  - Hackers:
    - Malicious clients
    - Attempt to destabilize the game
    - Example: returning a result with a different start point in order to “teleport”
  - Will cheat 50% of the time
    - 50% of said cheats will be infeasible
    - The other 50% will be inequivalent

Results: *Parametrization*

- Client Simulation:
  - Monitors:
    - Owned by the game providers
    - Used to monitor audits after the fact
  - Assumed to resolve messages 100% accurately
  - Compares its result to audit
  - Determines whch client is responsible

Results: *Parametrization*

- Client Overview:
  - Legit Clients: 99% accurate, 1% error
  - Hackers: 50% accurate, 25% inequivalent, 25% infeasible
  - Griefers: 50% accurate, 50% inequivalent
  - Monitors: 100% accurate
  - Clients make requests every ~{0,3] seconds
    - Based on practical game data

Results: *Parametrization*

- Cheat Reduction:
  - Audit: 10% of requests, Monitor: 5% of positive audits.
  - Boot time: 30 secs
  - Ban threshold: -15
    - Determined as best candidate experimentally
  - Trust metric:
    - Also Determined as best candidate

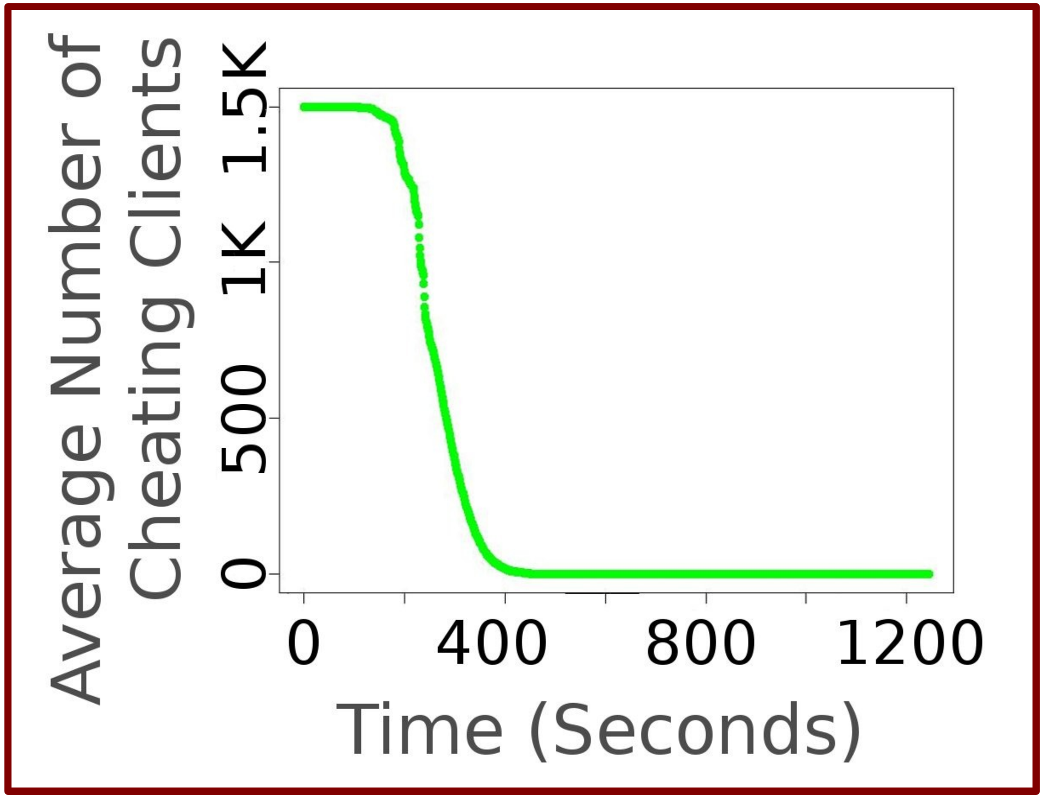
$$\boxed{T = [\text{identical}] + [\text{equivalent}] - [\text{inequivalent}]^{1.5} - [\text{infeasible}]^{2}}$$

An exponent of 1.5 causes less serious cheats to ramp up quickly, but not too quickly as to effect legit cleints

An exponent of 2 causes more serious cheats to ramp up exceedingly quickly removing malicious clients effectively

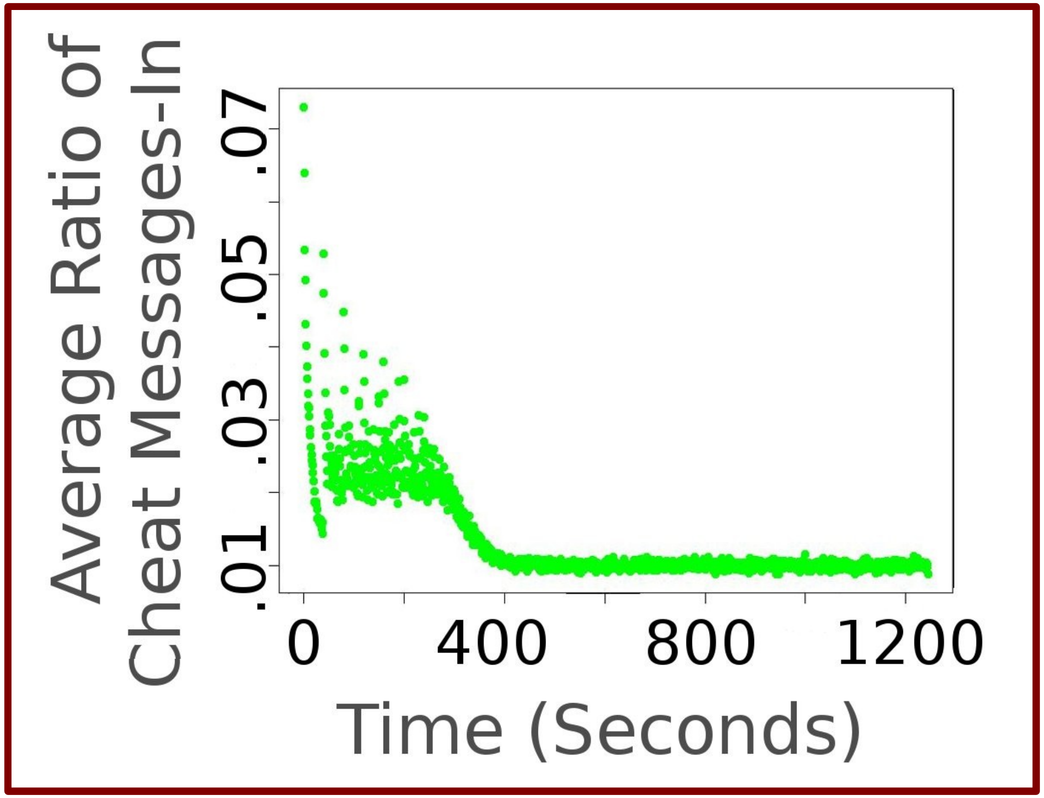
Results: *Experiment 1*

- Experiment in a **static** setting
- Initial clients:
  - 8,500 legit
  - 750 hackers
  - 750 griefers
- 20 minute experiments
- Very few false positives ~ 0.4 clients per experiment



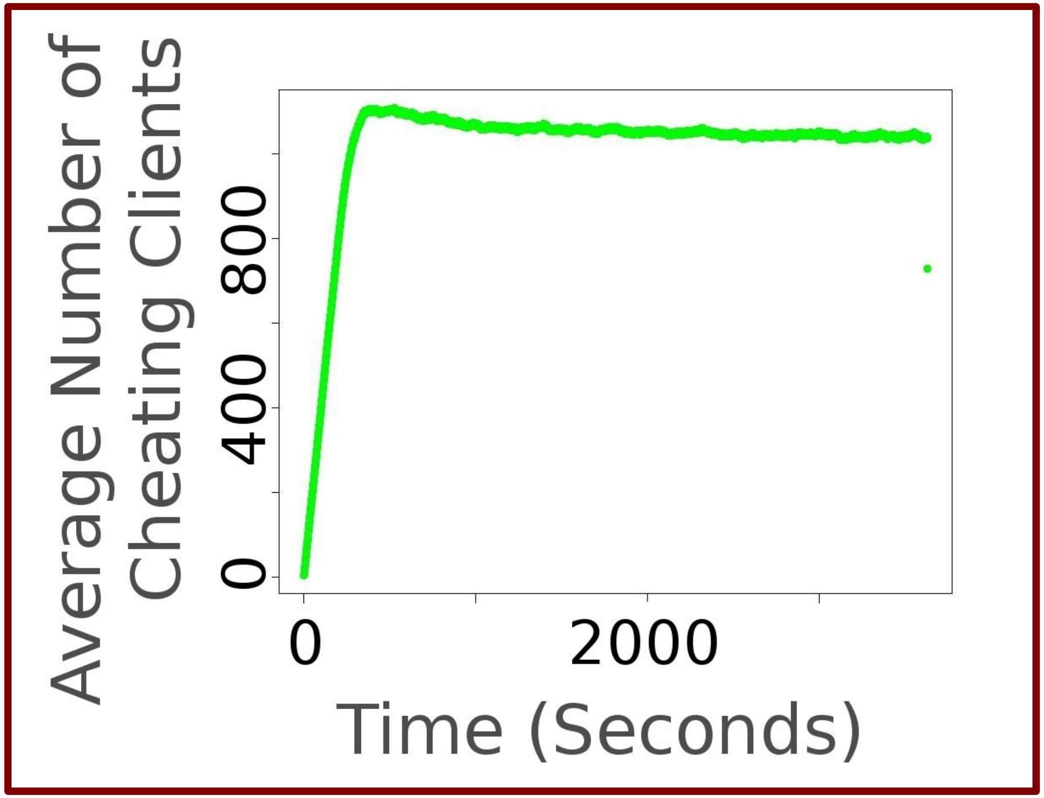
Results: *Experiment 1*

- Experiment in a **static** setting
- Initial clients:
  - 8,500 legit
  - 750 hackers
  - 750 griefers
- 20 minute experiments
- Very few false positives ~ 0.4 clients per experiment



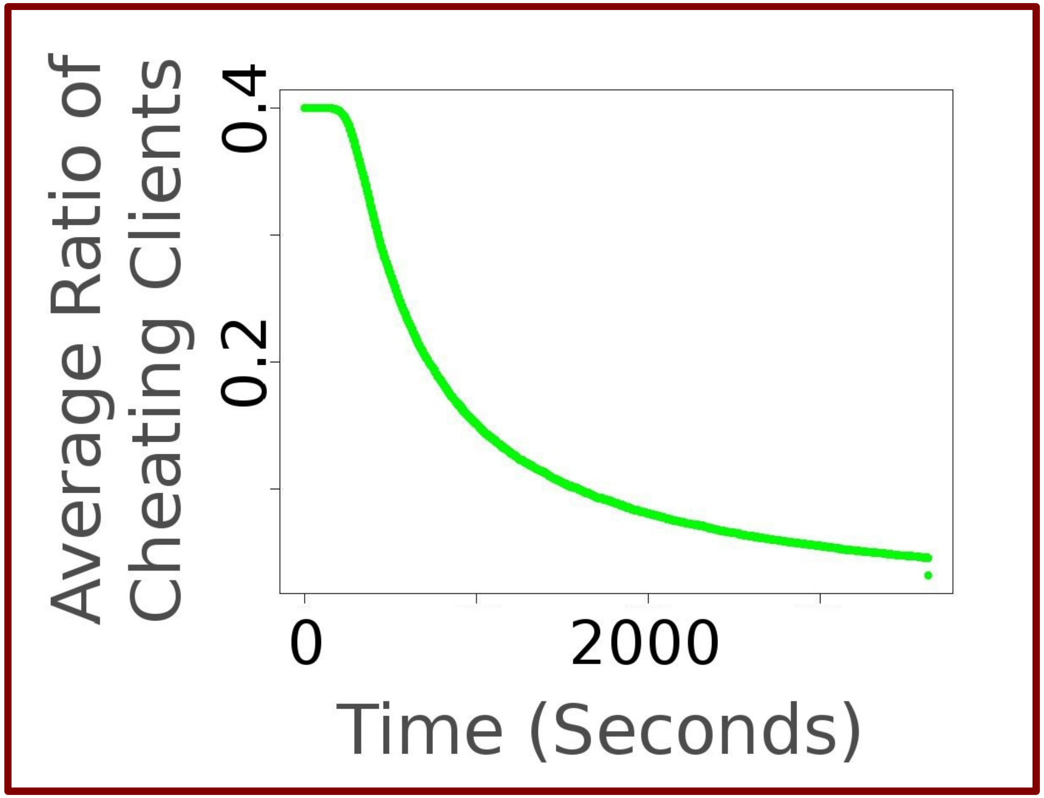
Results: *Experiment 2*

- Experiment in a **Dynamic** setting
- Initial clients: 0
- Per second:
  - 6 legit
  - 2 hackers
  - 2 griefers
- 60 minute experiments
- More false positives ~ 8 per experiment



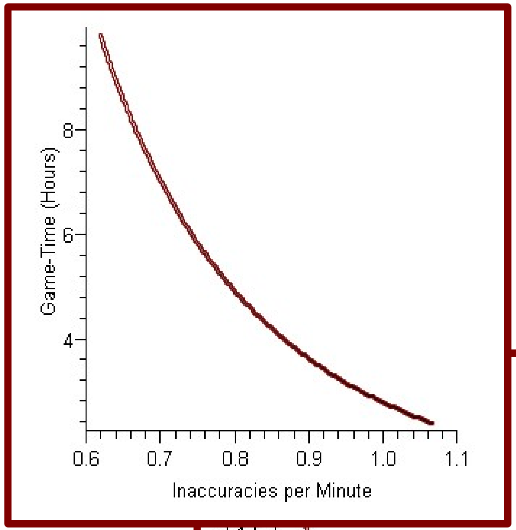
Results: *Experiment 2*

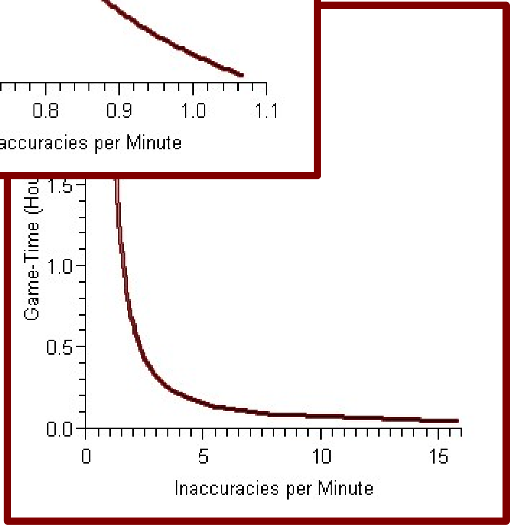
- Experiment in a **Dynamic** setting
- Initial clients: 0
- Per second:
  - 6 legit
  - 2 hackers
  - 2 griefers
- 60 minute experiments
- More false positives ~ 8 per experiment



Results: *Rate of Cheating Analysis*

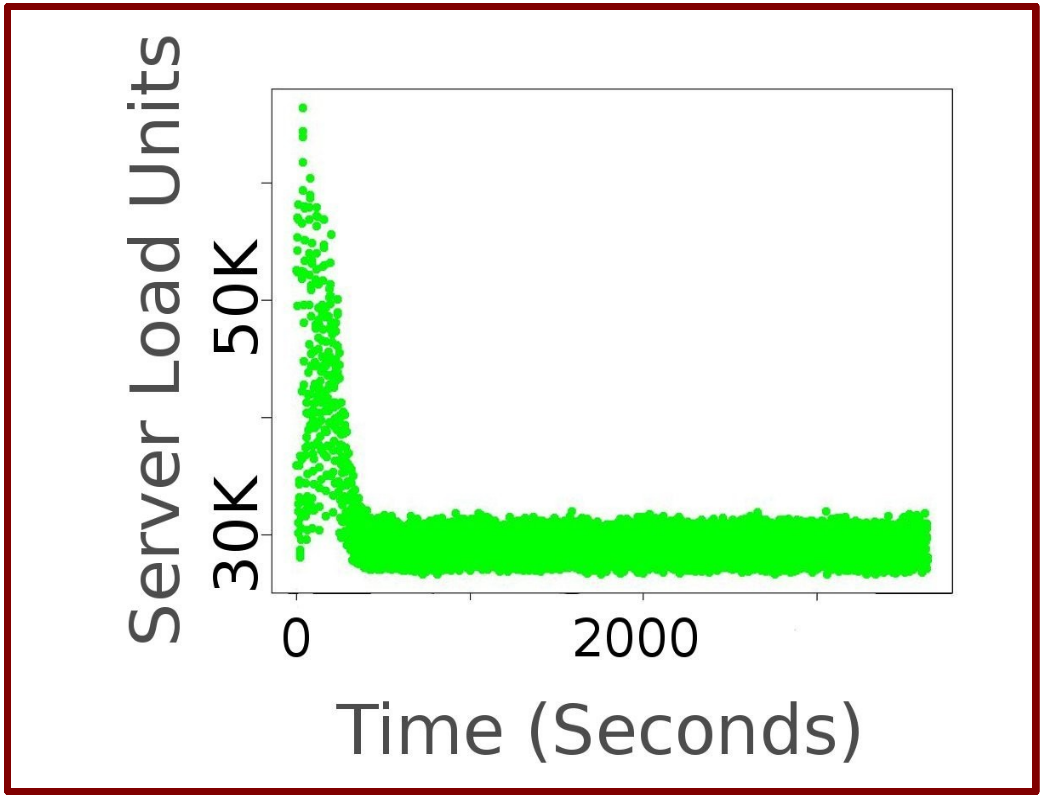
- Formal analysis
  - Relates rate of cheating to expected ban time
  - Shows:
    - A cheater must reduce its rate of cheating to last
    - A lower rate of error extends game time drastically
    - A client with a 0.1% error rate is expected to last will last ~7.5 months of continual gameplay





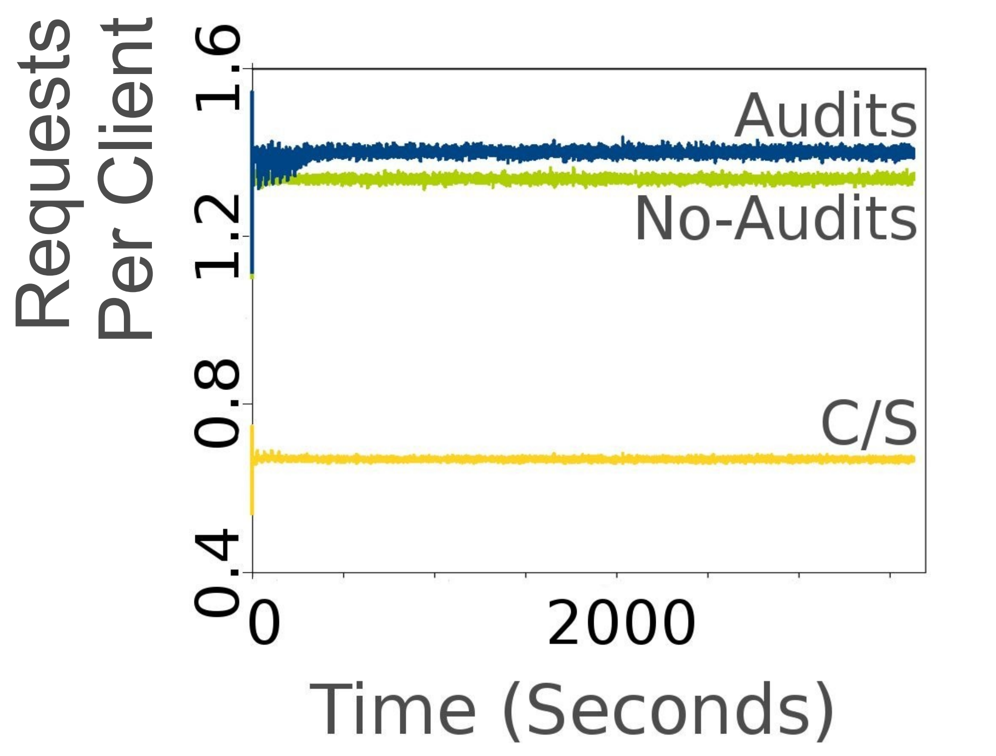
Results: *Experiment 3a*

- Experiment on Load/Overhead
- From static experiment data
- 60 minute experiments
- C/S results depict a load of around 250,000-275,000 units.



Results: *Experiment 3b*

- Experiment on Load/Overhead
- From static experiment data
- 60 minute experiments
- Compares
  - C/S
  - IRS w/ audits
  - IRS w/o audits



Conclusion: *Summary*

- Trade-off between **scalability** and **Cheat Resistance**
- IRS model shows
  - Good CPU load reduction ~ 10%
  - Ability to eliminate cheaters quickly
    - In approximately 400 seconds (due to booting)
  - Higher bandwidth > 200%
  - Higher Number of Hops > 200%

Conclusion: *Future Work*

- The examination of models which:
  - Ensure the IRS cheat reduction guarantees
  - Lower bandwidth cost
  - Lower latency
- The examination of auditing systems which:
  - Use adaptive auditing based on trust
- Integration of the IRS model into Mammoth
  - Alleviate cost of server side path-finding
  - Investigate IRS properties in a concrete setting

References

[1] Blizzard Entertainment, World of Warcraft. http://www.worldofwarcraft.com/index.xml.

[2] Mcgill University, Mammoth. http://mammoth.cs.mcgill.ca/.

[3] SQUARE ENIX, Final Fantasy XI. http://www.playonline.com/ff11us/index.shtml.

[4] B. Ali, W. Villegas, and M. Maheswaran. A trust based approach for protecting user data in social networks. In IBM CASCON 2007, pages 288–293, Richmond Hill, Ontario, Canada, Jan. 2007.

[5] N. E. Baughman and B. N. Levine. Cheat-proof playout for centralized and distributed online games. In IEEE InfoCom, pages 104–113, 2001.

References

[6] X. bin Shi, L. Fang, D. Ling, C. Xiao-hong, and X. Yuan-sheng. A cheating detection mechanism based on fuzzy reputation management of P2P MMOGs. In SNPD 2007, pages 75–80, Washington, DC, USA, 2007.

[7] F. R. Cecin, C. F. R. Geyer, S. Rabello, and J. L. V. Barbosa. A peer-to-peer simulation technique for instanced massively multiplayer games. In DS-RT 2006, pages 43–50, Washington, DC, USA, 2006.

[8] F. R. Cecin, R. Real, R. de Oliveira Jannone, C. F. R. Geyer, M. G. Martins, and J. L. V. Barbosa. FreeMMG: a scalable and cheat-resistant distribution model for internet games. In DS-RT 2004, pages 83–90, Washington, DC, USA, 2004.

References

[9] C. Chambers, W. chang Feng, W. chi Feng, and D. Saha. Mitigating information exposure to cheaters in real-time strategy games. In NOSSDAV 2005, pages 7–12, Washington, USA, June 2005.

[10] L. Chan, J. Yong, J. Bai, B. Leong, and R. Tan. Hydra: A massively-multiplayer peer-to-peer architecture for the game developer. In Netgames 2007, pages 37–42, Melbourne, Australia, Sept. 2007.

[11] W. chang Feng, D. Brandt, and D. Saha. A long-term study of a popular MMORPG. In Netgames 2007, pages 19–24, Melbourne, Australia, Sept. 2007.

[12] B. D. Chen and M. Maheswaran. A cheat controlled protocol for centralized online multiplayer games. In NetGames 2004, pages 139–143, Portland, OR, USA, Aug. 2004.

References

[13] E. Cronin, B. Filstrup, and S. Jamin. Cheat-proofing dead reckoning multiplayer games (extended abstract). In Conf. on Appl. and Dev. of Comp. Games, Jan. 2003.

[14] E. Cronin, B. Filstrup, A. R. Kurc, and S. Jamin. An e□cient synchronization mechanism for mirrored game architectures. In NetGames 2002, pages 67–73, Bruanschweig, Germany, 2002. IEEE.

[15] L. Fan, H. Taylor, and P. Trinder. Mediator: a design framework for P2P MMOGs. In Netgames 2007, pages 43–48, Melbourne, Australia, Sept. 2007.

[16] P. Golle and N. Ducheneaut. Preventing bots from playing online games. Computers in Entertainment, 3(3):3–3, 2005.

References

[17] R. Greenhill. Diablo and multiplayer game's future. http://www.gamesdomain.com/gdreview/zones/shareware/may97.html, May 1997.

[18] X. Jiang, F. Safaei, and P. Boustead. An approach to achieve scalability through a structured peer-to-peer network for massively multiplayer online role playing games. Computer Communications, 30(16):3075–3084, 2007.

[19] P. Kabus, W. W. Terpstra, M. Cilia, and A. P. Buchmann. Addressing cheating in distributed MMOGs. In Netgames 2005, pages 1–6, 2005.

[20] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In WWW 2003, pages 640–651, 2003.

References

[21] B. Knutsson, H. Lu, W. Xu, and B. Hopkins. Peer-to-peer support for massively multiplayer games. In IEEE InfoCom, Mar. 2004.

[22] J. Kuecklich. Other playings: cheating in computer games. In Other Players Conf., IT University of Copenhagen, Dec. 2004.

[23] P. Laurens, R. F. Paige, P. J. Brooke, and H. Chivers. A novel approach to the detection of cheating in multiplayer online games. In ICECCS 2007, pages 97–106, Washington, DC, USA, 2007.

[24] S. Mogaki, M. Kamada, T. Yonekura, S. Okamoto, Y. Ohtaki, and M. B. I. Reaz. Time-stamp service makes real-time gaming cheat-free. In Netgames 2007, pages 135–138, Melbourne, Australia, Sept. 2007.

References

[25] T. Schluessler, S. Goglin, and E. Johnson. Is a bot at the controls? detecting input data attacks. In Netgames 2007, pages 1–6, Melbourne, Australia, Sept. 2007.

[26] J. Smed, T. Kaukoranta, and H. Hakonen. A review on networking and multiplayer computer games. Technical Report Tech Report No. 454, University of Turku Centre for Computer Science, 2002.

[27] S. D. Webb and S. Soh. Cheating in networked computer games: a review. In DIMEA 2007, pages 105–112, 2007.

[28] J. Yan and B. Randell. A systematic classification of cheating in online games. In Netgames 2005, pages 1–9, Hawthorne, New York, USA, Oct. 2005.

Example Bullet Point slide

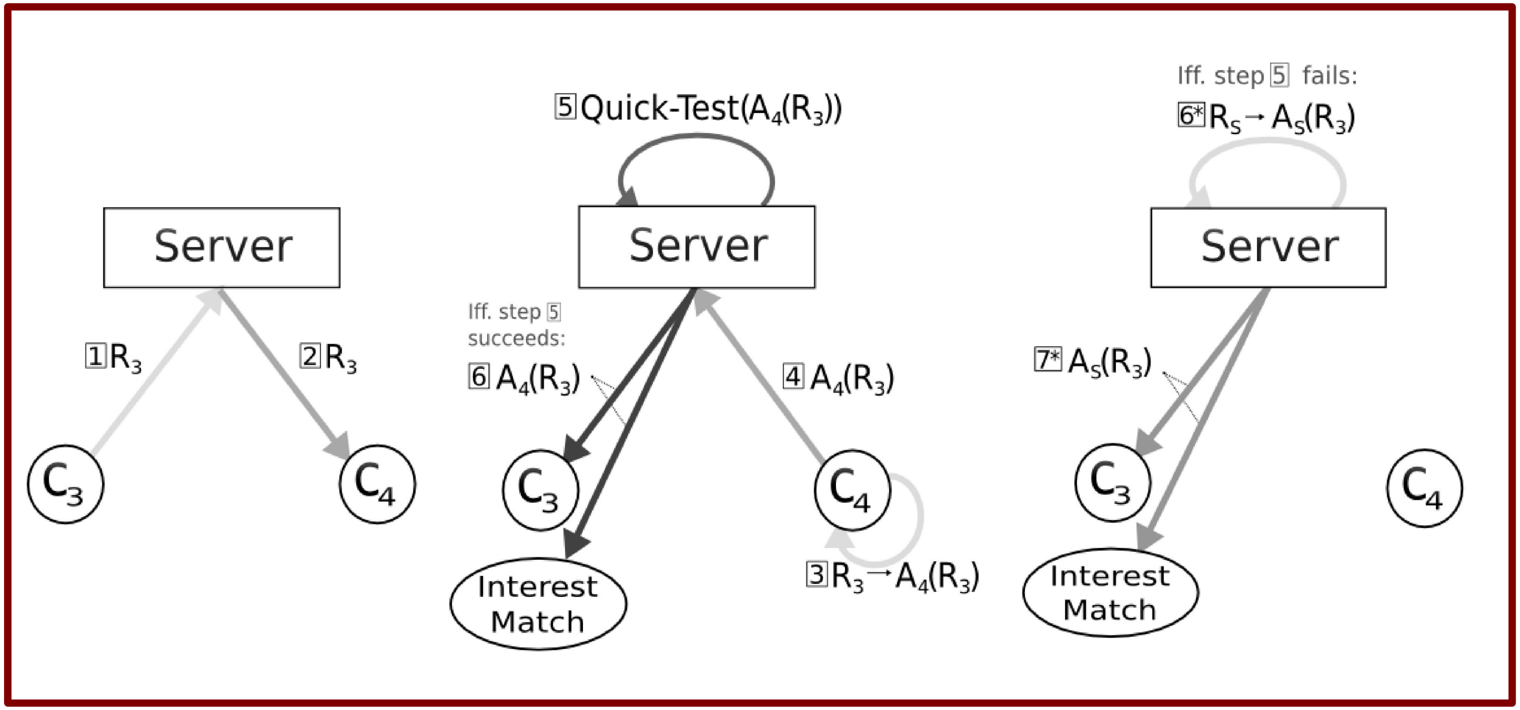
- These templates are for personal use only and must not be distributed, sold or displayed on the web by anyone other than Presentation Helper.
- Bullet point
  - Sub Bullet

Design: *Communication Model*

- 4 Communication Phases (Server)
  - 1. Proxy Assignment
    - Done by server at certain intervals
  - 2. Message Relaying
    - Server relays messages from a client to its proxy
    - The proxy is responsible for resolving said message
  - 3. Peer Auditing
    - Resolved messages computed by different clients on identical requests are compared
  - 4. Message Handling
    - Quick Testing of resolved messages
    - Relaying the resolved message to appropriate clients

Design: *Communication Model*

- Diagram of phases 2-4:



Introduction: *Current Solutions*

Introduction: *Cheating Focus*

- In MMOG's there are a vast variety of cheating behaviours
- It is also difficult to formulate a precise definition of cheating
- Many ad hoc cheat elimination systems exist
- However, with P2P communication avoiding **abuse of authority** is imperative
- Therefore: we *focus* on reducing/eliminating **abuse of authority** cheats