Scalability for Virtual Worlds

By Nitin Gupta, Alan Demers, Johannes Gehrke, Philipp Unterbrunner, Walker White
at ICDE 2009

Presented By,
Mayuri Khardikar

Net-VEs

- Networked Virtual Environments
- A virtual environment shared by many users connected over a network
- Users can interact with each other in real time
- e.g MMOs like WoW, virtual world like second life

Motivation

- So Net-VEs are very popular due to 3D immersive graphics,stereo sound, realistic, and highly multiplayer nature
- But current architecture is server centric, all game logic is executed on server
- Leads to severe scalability problem due to high computational intensive tasks.
- Clients generally have enough computing power, so need to leverage it to increase scalability

Contribution of the paper

- Proposes distributed action based protocol for Net-VEs
- Pushes most of the computation on player's machine(client's side)
- So no game logic on server side, thus, can achieve massive scalability
- Novel distributed consistency model: uses application semantics to reduce number of messages needed between clients and server
- Investigate the solution theoretically and experimentally

Virtual World – A Database Perspective

- The entire virtual world and all its components (World State) are stored in a high dimensional database where attributes can change in only predicted ways
  - Tuples - Each object/player information
  - Attributes - Characteristics like Health, position, speed, weapons of each object/player

- Any interaction in the world is a database transaction
  - Observations - Database Queries
  - Change in state - Database Updates

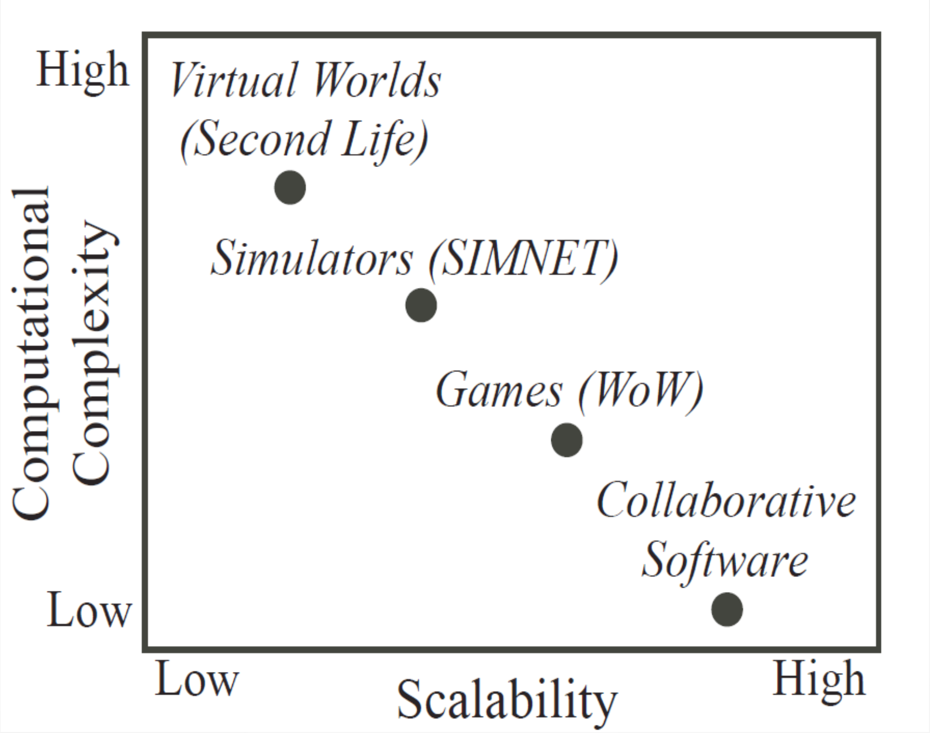
A Gaming Example

- A Shared Virtual Gotham City
- Avatars - Batman and Joker
- Event - Batman kicks Joker which reduces Joker’s health
- A look from Database perspective
  - Batman, Joker and their attributes including current health stored as tuples in the database in objects table
  - The game engine reads from the database, attacking power of Batman and health of Joker
  - The game engine determines the effect of the action on Joker's health and other parameters
  - The game engine updates the values of the new parameters in the database

What restricts Massive Scalability?

- Computational Complexity
  - Realistic graphics and physics based interaction
- Consistency
  - Consistent view of virtual world for all users called as world state. Required for realism.
- Response Time
  - Guaranteeing bounded response time to users thereby increasing action throughput. Required for real-time interaction.

Computational Complexity



- Similarly we expect scalability to decrease with increasing consistency requirement and decreasing response time requirement

Tackling Massive Scalability Problem

- Computational Complexity
  - Pushing complex computation to client machines
- Consistency
  - Using application semantics to reduce consistency requirements, such as visibility
- Response Time
  - Reducing messages communicated for an action
- Exploring the Trade-Offs in above requirements

Net-VE Architectures

- Centralized VEs
- Distributed VEs
  - P2P architecture
  - Client Server

    Consistency protocols:

    -Lock based

    -Time stamp based

    -Object ownership based

    -Action based

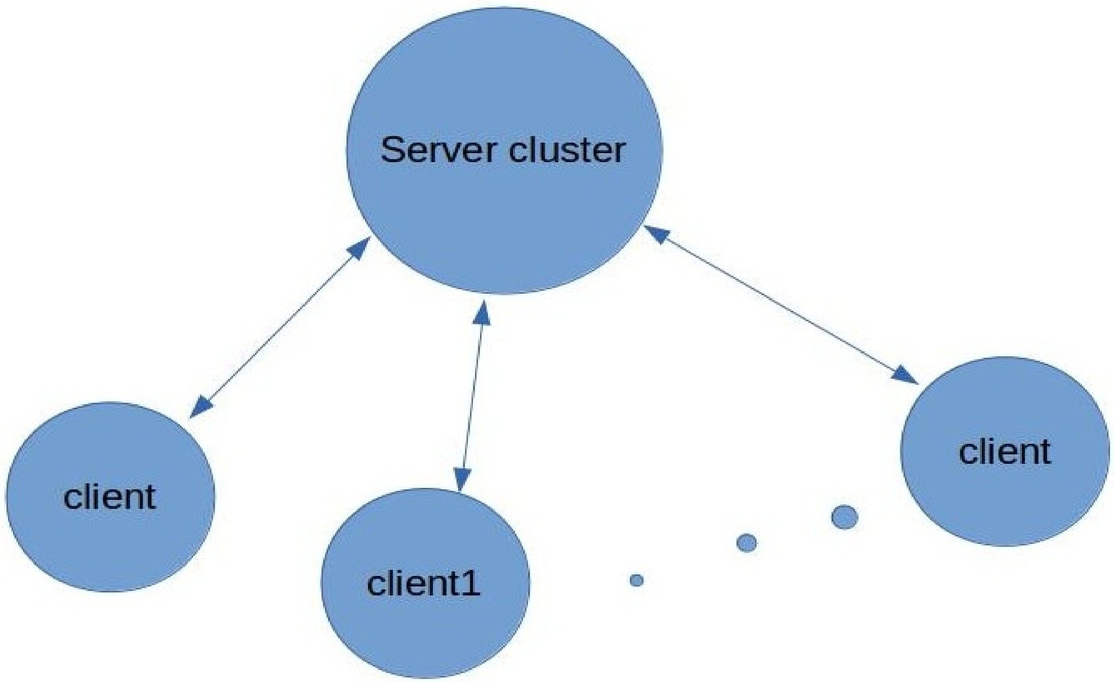
Net-VE Architectures

- Centralized VEs
  - All computations are done at a centralized server
  - World state updated only by server
  - The clients only read this world state and show it to the users
  - Scalability issues- as computational complexity increases, number of users handled by each server reduces.
  - e.g. In Second-life, max 25-30 users/server

Scaling Centralized VEs

- Zoning
  - Geographically partitioning virtual environments small enough for a server to handle
  - But user cannot move from one zone to other, if allowed, complexity is very high, will collapse if too many player gather in one zone
- Sharding
  - Different instances of virtual environments for geographically distant users, e.g. separate for Asian countries and separate for Europe
- Instancing
  - Private zones meant a personal experiences to some players, e.g in WoW
- Focus on partitioning user base
- Limits user interaction with each other
- Some virtual worlds require users to pay for playing with real friends

Net-VE Architectures



All Clients run Net-VE software(client programs)
containing game logic,
Clients initiates and processes actions,
server timestamps them & serializes,
Also server logs the actions.

Client Server Net-VEs

- Clients connected to server(s)
- Imposes central control by server
- Reduced load on server, so increases scalability
- Client
  - All clients contain virtual world logic(client program)
  - Clients initiate and process action
    - A sequence of atomic operations
    - At first, observation of world state
    - Followed by update of the state
- Server
  - Shoulders the responsibility of consistency of world state across clients
  - Can log actions for security and prevent cheating

Ensuring consistency in Client Server Net-VEs

- Distributed Lock Based Protocol
  - Global Locks on objects
  - Lock granted by server
  - Client Requests locks
  - Server multicasts request to other clients
  - Lock status reported to client
  - Client preforms transaction and sends result to server
  - Server again multicasts result to other clients
  - All clients update their local copy
  - Move to next conflicting transaction
  - Disadvantages
    - Min time required is 2 x RTT
    - All consistency issues should be mapped to object access

Ensuring consistency in Client Server Net-VEs

- Time-stamp based Protocols
  Optimistic concurrency control :
  - Servers associates versions with objects and timestamps with transactions
  - Clients execute actions optimistically on local copy
  - Server integrates the local copies into a global multi-version history ensuring consistency in the world
  - Disadvantages
    - Server should understand game logic
    - If server broadcasts global history then time required 2 x RTT

Ensuring consistency in Client Server Net-VEs

- Object Ownership based protocols:

  - Each object owned and managed by single client
  - Other clients use cached copy but cannot modify it, only owner can modify
  - Scalable but doesn't allow object contention
  - If allowed then need to compromise on consistency or use time-stamp-based serialization
  - Time-stamp-based serialization will increase response time

Back to the work in the paper…

- Action based Protocols

  - Consistency checked at action level
  - Actions are functions which update the world state
  - Virtual World is a progression of world states updated by client actions

- Assumptions
  - Standard model of simulation engine
  - World changes only at simulation ticks, so discrete
  - Inter tick interval ‘T’

Basic Algorithms

- Client sends actions to the server not objects.
- Whole application logic is executed at client.
- Server only timestamps and serializes actions for consistency and durability

- First, some notations and definitions
  - World State (WS): state of database of objects in virtual world
  - Client maintains two versions of world state
    - Optimistic version ZCO
    - Stable version ZCS
  - Actions performed by clients ai
  - Effect of applying ai to ZCO is vi

Basic Algorithms : A Bird's eye view

- Clients (when sending actions)
  - Preform action on optimistic copy and sent result to server
- Server
  - Gets actions from all clients, timestamps and orders them and relays these actions to the clients
- Clients (when receiving actions)
  - Applies received actions on ZCS and compares the result with those of ZCO
  - Reconciliation protocol is called in case of conflicts
    - Resolves conflict considering the ordering imposed by the server
    - Changes the action & its result and again sends it to the server

Basic Algorithms : Client

The client maintains a queue

$$Q = [\langle a_1, v_1 \rangle, \ldots, \langle a_k, v_k \rangle]$$

where each a_i is a locally generated action that has not yet been received back from the server, and v_i is the result of applying a_i to ζ_{CO} as described below. Whenever the client creates an action a, the action is first executed on ζ_{CO} producing a result v. We call this the optimistic evaluation of a. The pair $\langle a, v \rangle$ is then added to Q, and the action a is sent to the server.

Basic Algorithm : Client

Assume that the client receives an action b from the server. There are two possible cases:
(Action b originated at some other client): Action b is applied to ζ_{CS}. Each write $x \leftarrow v$ performed by b is also performed on ζ_{CO} if (and only if) $x \notin WS(Q)$. (This has the effect of updating items in the state that are not awaiting permanent values from the server).
(Action $b = a_1$): Action a_1 is applied to ζ_{CS} producing result u. If $u = v_1$, indicating the new evaluation of a_1 agrees with its optimistic evaluation, the entry $\langle a_1, v_1 \rangle$ is removed from the head of Q. Otherwise, ζ_{CO} is reconciled with ζ_{CS} using Algorithm 3.

Basic Algorithms : Server

Algorithm 2: Server-Side Protocol

1 The server maintains a global queue of actions. For each client C, the server maintains the index pos_C of the action in the queue that was last sent to C. At the start of the protocol, $pos_C = 0$ for all clients C.

2 When the server receives an action a from client C (Step 2 in the client-side protocol), it performs two steps:

3 (a) It timestamps a and puts it into the queue, assigning a a unique order number $pos(a)$ that is a's position in the queue.

4 (b) The server returns to C all actions between positions pos_C and $pos(a)$, and it sets $pos_C = pos(a)$.

Basic Algorithms : Reconciliation

Algorithm 3: Reconciliation Protocol

Require: $Q = [\langle a_1, v_1 \rangle, \ldots, \langle a_k, v_k \rangle]$ is the results of optimistic evaluation of locally generated actions.

$\zeta_{CO}(WS(Q)) \leftarrow \zeta_{CS}(WS(Q))$

$Q \leftarrow []$

for $(j = 1; j <= k; j++)$ **do**

  apply a_i to ζ_{CO} producing result v

  insert $\langle a_i, v \rangle$ into Q

Is the proposed solution enough?

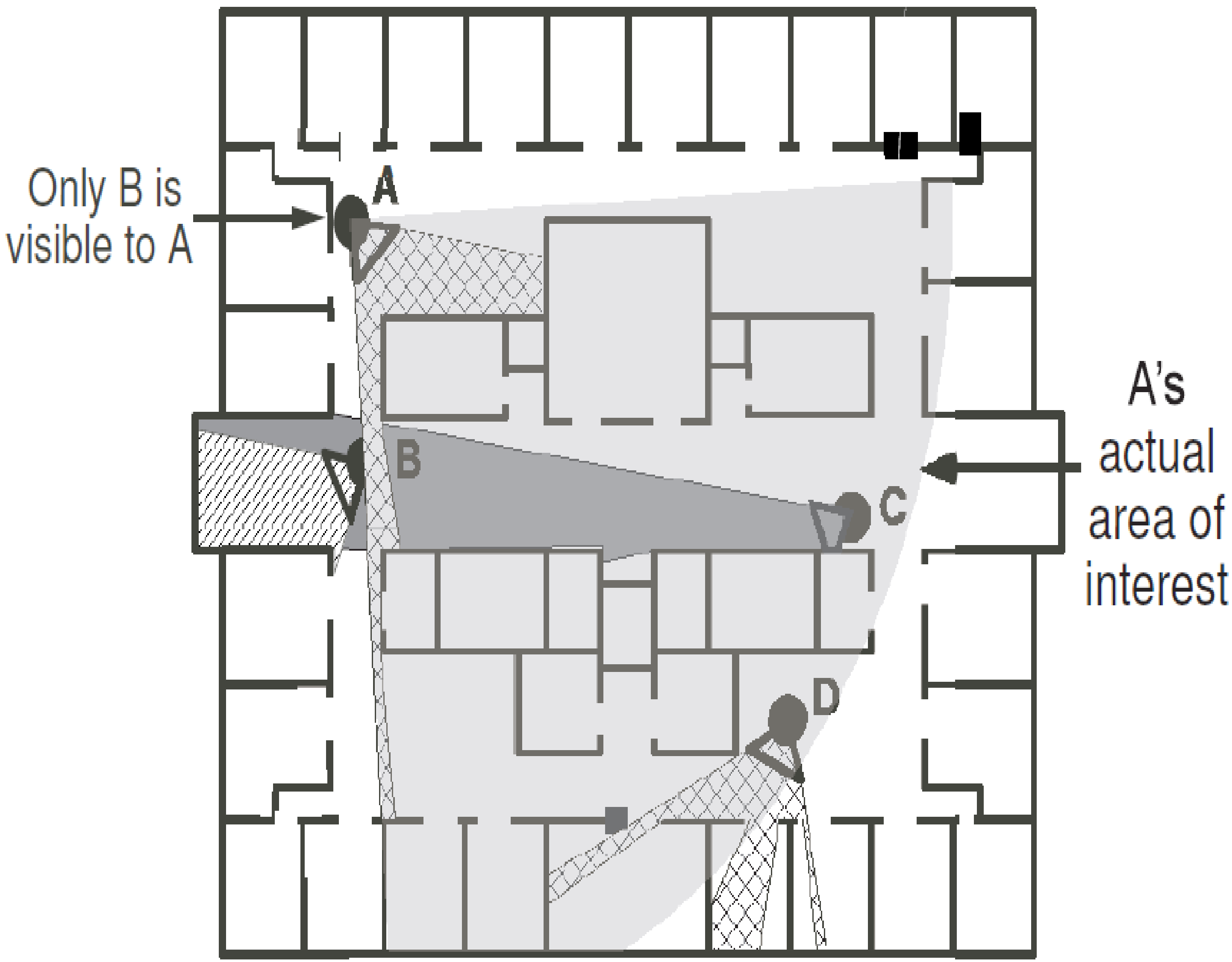
- Response Time= RTT for most actions, so good enough.
- Allows any interaction including object contention
- Server can handle large number of clients
  - server is free from game logic
  - only timestamps actions. Queues them, manage n/w traffic
- Consistency
  - The server ensures consistency using time-stamp ordering
  - Each client execute all actions on its stable copy in same order imposed by server
  - So it is broadcast based protocol e.g. used by SIMNET

    BUT……
- Computational Load on clients
  - Clients need to process actions of all the clients in the world
  - Incurs high computation load on clients
  - Server sends each message to all clients so high BW requirement

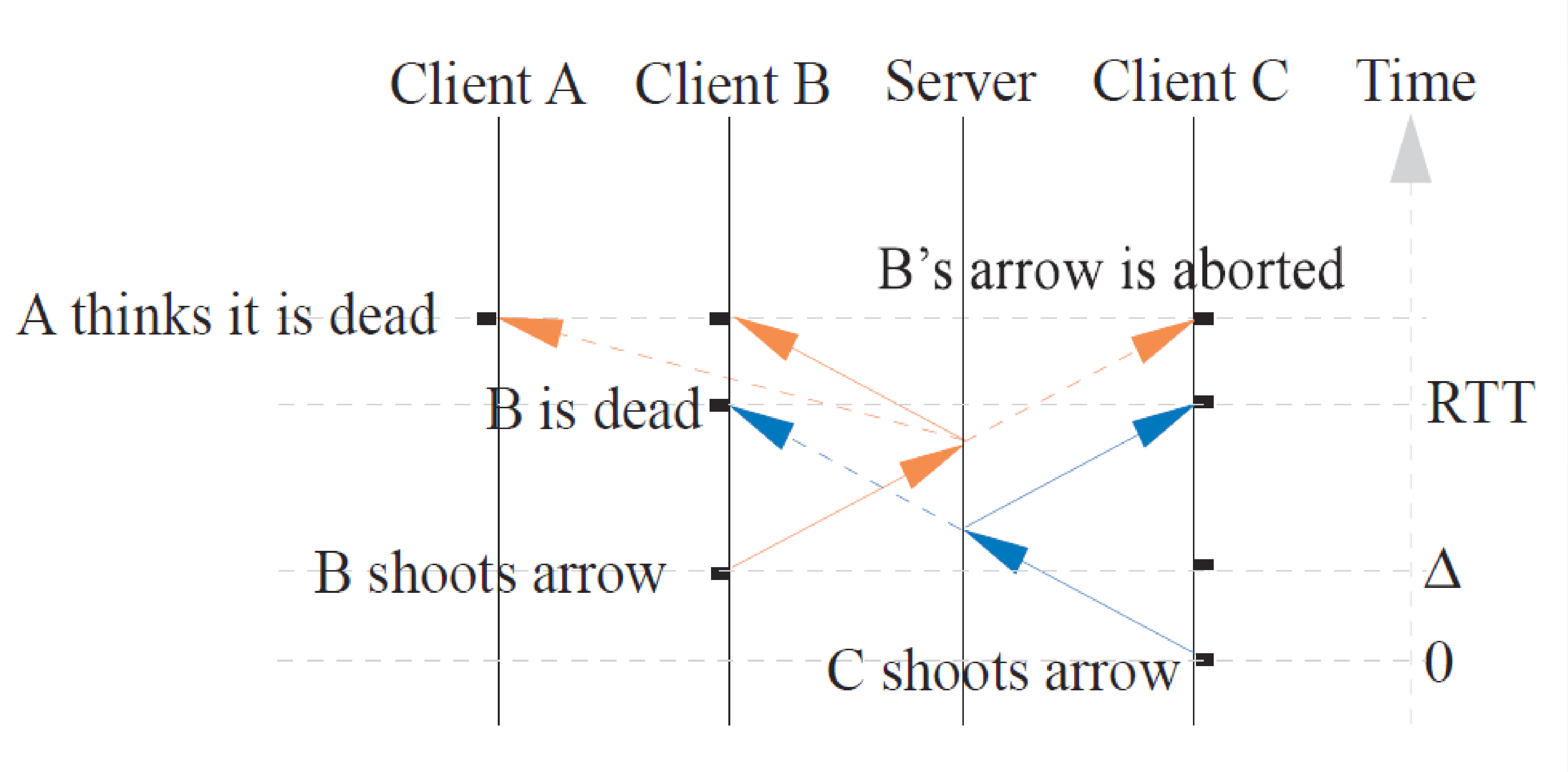
Leveraging Application Specific Information

- Current optimizations focus on area-of-interest paradigm in
  - Restrict set of update messages by syntactic constraints like visibility
    (fig on next slide)

- Problems with the approach
  - Does not generalize to arbitrary actions like scrying spell
  - Different obstruction layers for actions based on different senses
  - Transitive propagation of effects of actions need to be taken into account

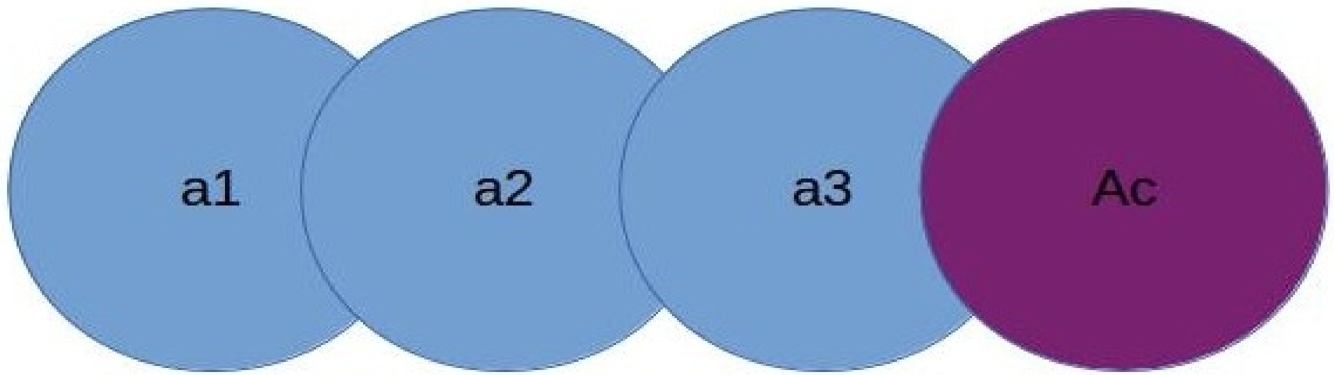

Only B is visible to A
A
B
C
D
A's actual area of interest

Transitive propagation of actions by users



Thus, actual area of influence of an Avatar is much larger than its visibility area. This is mainly because of transitive effect of actions. These are based on application semantics

Transitively affecting actions



So, Ac is transitively affected by a1, a2, a3. Client C must execute a1,a2,a3 and then Ac.
So server need to send all these in sequence to C

An action ai affects action aj if,
Read Set (aj) $\cap$Write Set (ai) $\neq \emptyset$

Incomplete World Model

- Semantic-based, action based protocol
- Resolve previous inconsistency in earlier model

- Clients maintain incomplete world state in their databases
  - World State variables which concern them are only updated
- Now server has the responsibility to maintain a complete world state
  - Also since we don’t want the server to evaluate game logic, the actions would still be evaluated by the clients
  - Their result and a completion message is sent to the server
  - The server then updates the authoritative state

- Client sees an incomplete world while server sees a complete world

Incomplete World Model : Client

- Every client does not need to execute every action, executes only relevant ones

- Now after application of each of its own action successfully, it sends a completion message to the server in both cases
  - If Zco and Zcs match
  - If not, then reconciled and new action added
- Completion message indicates the successful application of an action

Incomplete World Model : Server

- The server maintains
  - Authoritative state Zs
  - Global queue of ordered actions
  - And for each action in the queue, the clients it was sent
- Time stamping of actions is similar as in previous protocol
- For every action, it computes the set previous actions that must be sent to each client (See Next Slide)
- Upon receipt of an completion message of an action from a client, the action is removed from the global queue
- Only completed actions are applied to the authoritative state

Which updates should be sent?

- Which part of the world is client concerned with? Application semantic information can be used to determine if an action affects another action
- A bomb explosion in a area affects the health of an avatar if the avatar is within the maximum radius of explosion
- So calculate transitive closure of action using RS and WS of the actions

Determining update set

- An action has
  - Read Set – The world state variables it reads
  - Write Set – The world state variables it updates
- An action ai affects action aj if,
  - Read Set (aj) ∩Write Set (ai) ≠ ø
  - Now compute which actions ak affect ai
  - Continue transitively for all actions in the ready queue of actions

- The determination of actions would go on but terminates when
  - The action queue is finished since these actions have completed message sent
  - The values for the remaining read set are read from the authoritative database. As all completed actions have been applied to authoritative database
  - Thus, transitivity has bound

A Theorem

- If clients follow algorithm 4, and server follow 5 and 6 then in a distributed snapshot of the system, Zcs at all clients are consistent with Zs at the server
  - Observe that all clients and server apply the updates relevant to them in the same order

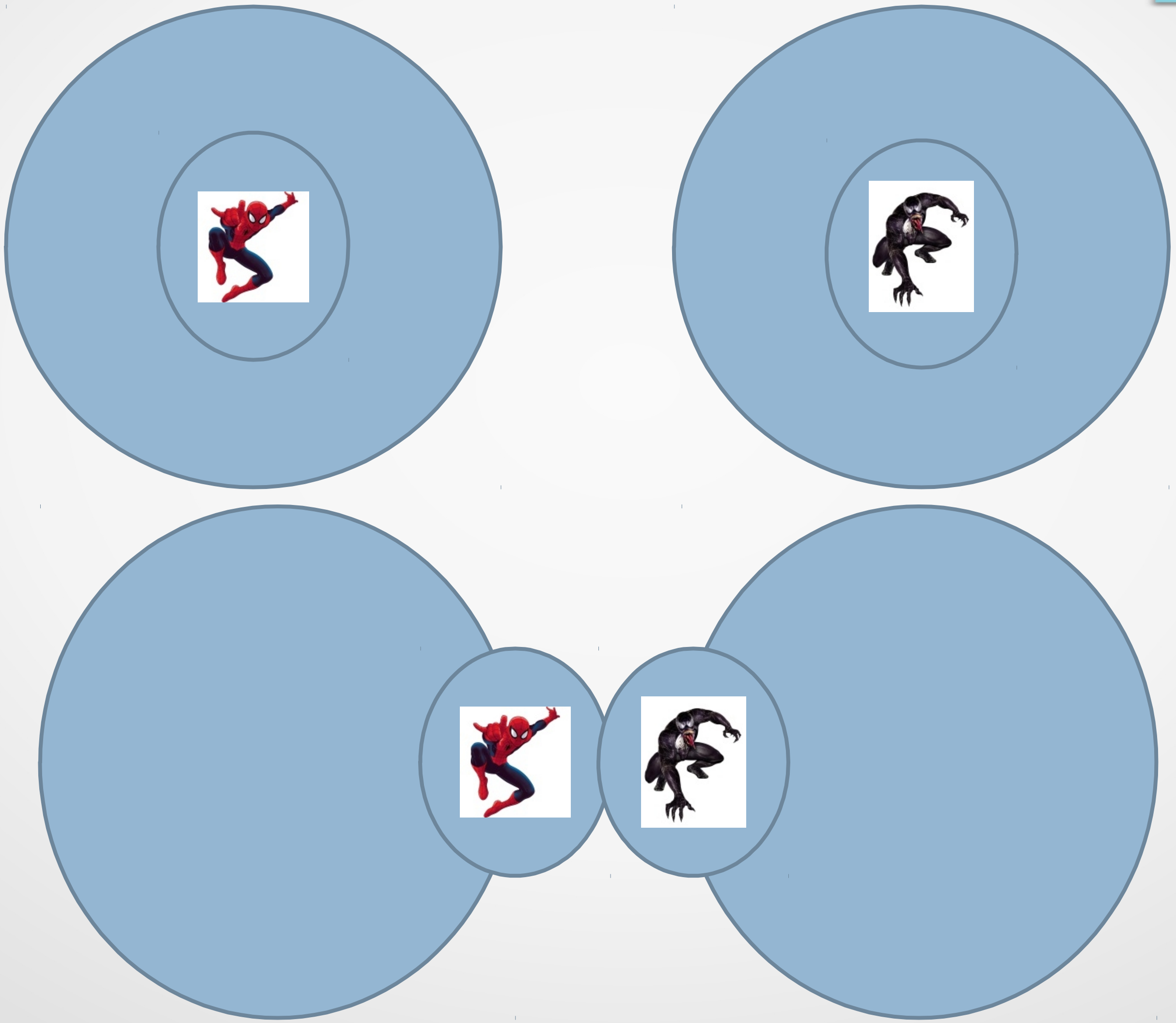
Analysis of the Protocol

- Depends on the bound on the number of actions to be included in the update set which affects the computational complexity at the clients
- Transitive closure
  - Determines which previously unsent actions can affect the evaluation of current action
- First Bound Model
  - Maximum number of actions that need to be sent to a client due to direct conflicts with client's current action
- Information Bound Model
  - Maximum number of actions that can be a part of any action's transitive closure. It is represented as a function of distance

Which actions to consider?

- Use application semantics to bound actions
- Spatial attributes can change at most by maximum velocity
- A player can damage other player at most by the maximum attacking power

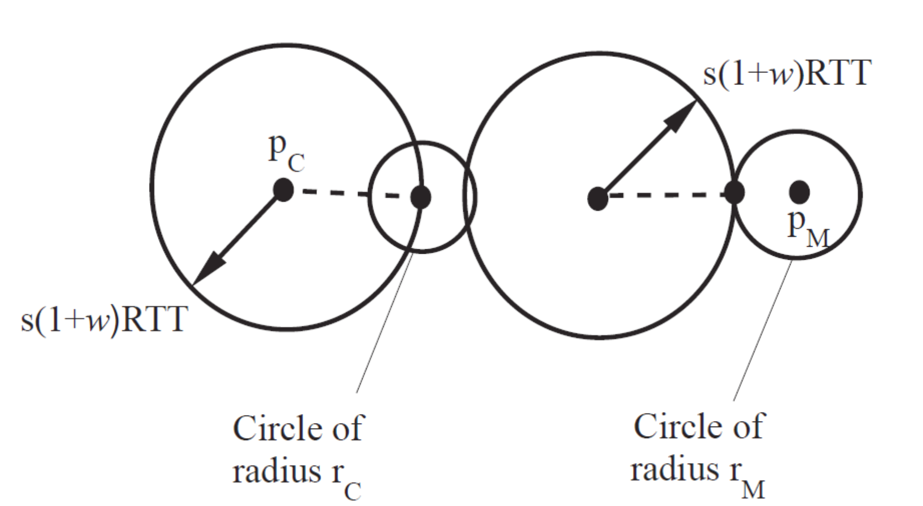
First Bound Model : Intuition



First Bound Model

- Computing Complexity
  - Time for server to receive response for an action from client is RTT + Y (initial processing)
  - Server needs to send all actions that it has seen in the previous (RTT + Y) / T ticks
  - Later as actions increase Y increases proportionally increasing the bound geometrically
- A little change in the protocol
  - The server now proactively pushes action sets to clients at regular intervals of w RTT ($0<w<1$)
  - The server receives a response for any action from the client in time (1+w) RTT after sending the action to the client

First Bound Model : The condition

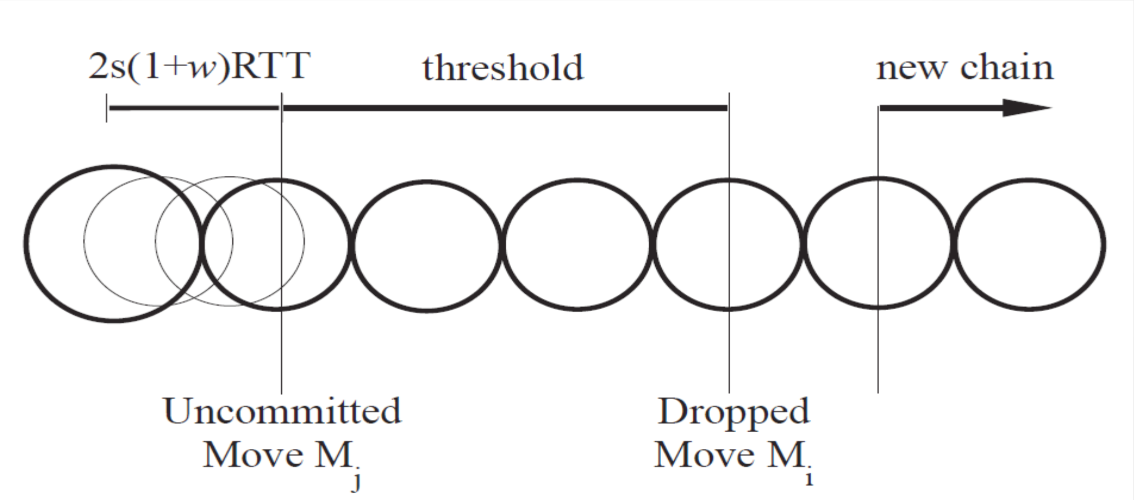


$$\| \bar{p}_A - \bar{p}_C \| \leq (2s \times (1+\omega)\ RTT) + r_C + r_A$$

- Pa and Pc are positions of the users
- S is the maximum velocity of the object
- rc and ra are radii of areas of influence
  (in above fig, consider Pa and ra instead of Pm and rm)

Information Bound Model

- Transitive effects of actions can sometimes affect other actions through very long sequence of actions
- Bound on the number of action to be considered for transitive effect
- The bound is decided arbitrarily and actions are dropped and not considered
- Raises some other issues like fairness but performance is good enough



Considering relevant actions

function onActionSubmission($action$)
begin
  $A_{actionCount} \leftarrow action$
  let $i = actionCount$
  for $(j = 0; j < clientCount; j+ = 1)$ **do**
    if $|p_{A_i} - p_{C_j}| \leq (2s \times (1 + \omega)\ RTT) + r_C + r_A$
    then
      $clientConflicts_{i,clientConflictCount_i} \leftarrow j$
      $clientConflictCount_i + = 1$
    end
  end
  $actionCount + = 1$
end

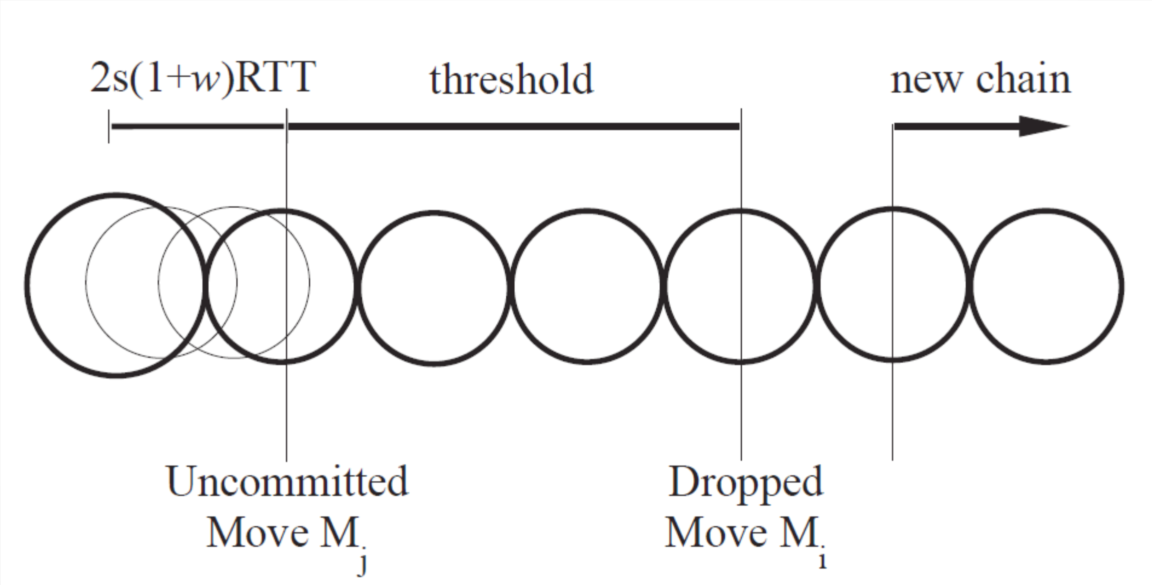
Computing Update set

function onNextTick()
begin
  for $(i = previousCount; i < actionCount; i += 1)$
  do
    let $S = RS(A_i)$
    let $invalid =$ **false**
    for $(j = i - 1; j > lastCommitted; j -= 1)$ **do**
      if $isValid_j$ **and** $S \cap WS(A_j) \neq \emptyset$ **then**
        if $|p_{A_i} - p_{A_j}| > threshold$ **then**
          $invalid \leftarrow$ **true**
          break
        end
        $S \leftarrow (S - WS(A_j)) \cup RS(A_j)$
        $conflicts_{i,conflictCount_i} \leftarrow j$
        $conflictCount_i += 1$
      end
    end
    $isValid_i \leftarrow$ **not** $invalid$
  end
  previousCount $\leftarrow$ actionCount
end

The complete bound

49

- Using both the first bound and information bound



$$\| \bar{p}_A - \bar{p}_C \| \leq (2s \times (1+\omega)\ RTT) + r_C + r_A + threshold$$

Experimental Evaluation

- Paper’s algorithm – SEVE (Scalable Engine for Virtual Environment)
- The game - Manhattan People

Response Time vs Scalability

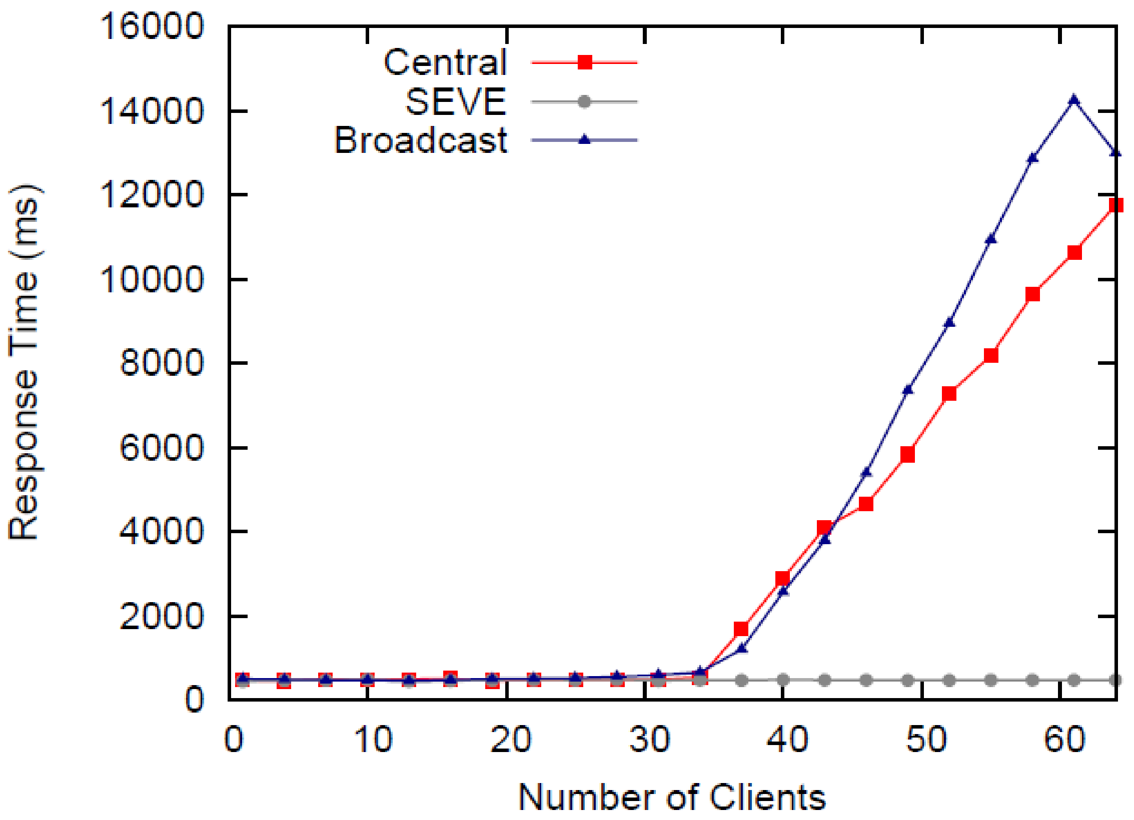


Fig. 6. Scalability of SEVE vs. Central architecture

Response Time vs Complexity

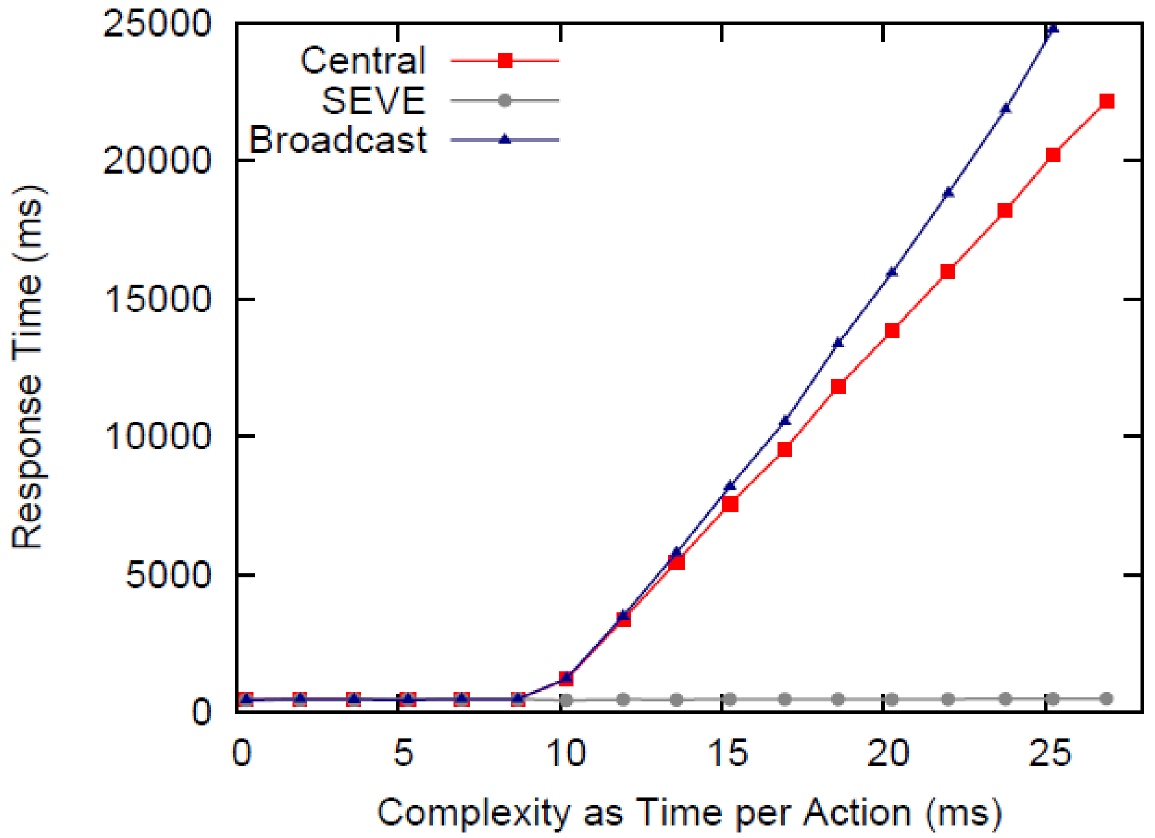


Fig. 7. Response Time vs. Action Complexity

Data Transfer vs Number of Clients

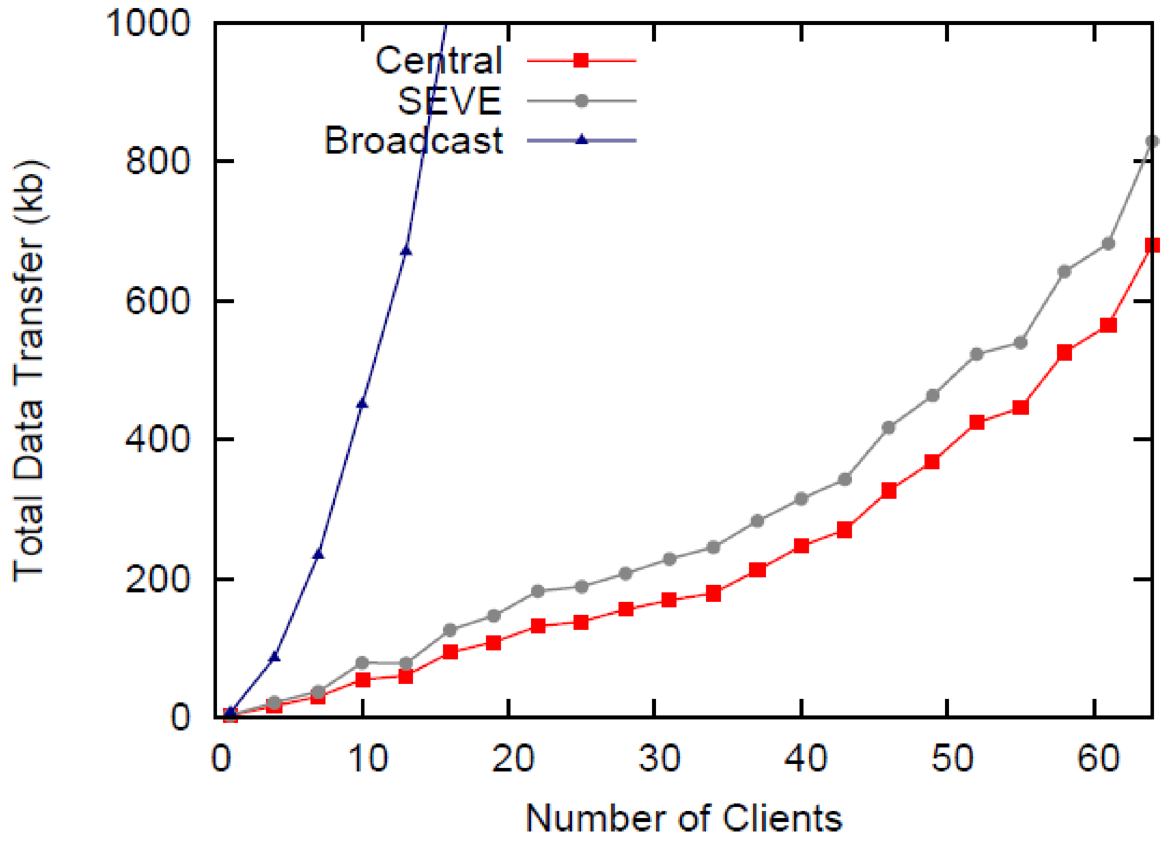


Fig. 9. Total data transfer

Conclusion

At the core of networked Virtual Environments, lie **data-management problems.**

- Identified a novel solution to an interesting concurrency problem, **using DBMS paradigms**.
- Using the proposed solutions, VEs can be made massively scalable with achieving high consistency

- Applications ranging from collaborative problem solving to online games can benefit from the database community

References

[1] Scalability for Virtual Worlds
Nitin Gupta, Alan J. Demers, Johannes Gehrke, Philipp Unterbrunner, Walker M. White

- ICDE 2009
- [2] SEMMO: A Scalable Engine for Massively Multiplayer Online Games (Demonstration Paper)
  Nitin Gupta, Alan Demers, and Johannes Gehrke, SIGMOD 2008
- [3] Database Research Opportunities in Computer Games
  Walker White, Christoph Koch, Nitin Gupta, Johannes Gehrke, and Alan Demers, In SIGMOD Record, September 2007

Thank you :)