SCALABILITY

FOR

VIRTUAL WORLDS

Sistemas Distribuídos e Tolerância a Falhas

2010/2011

Adolfo Peixinho n.º m4067

Baseado no artigo

"Scalability for Virtual Worlds"

de

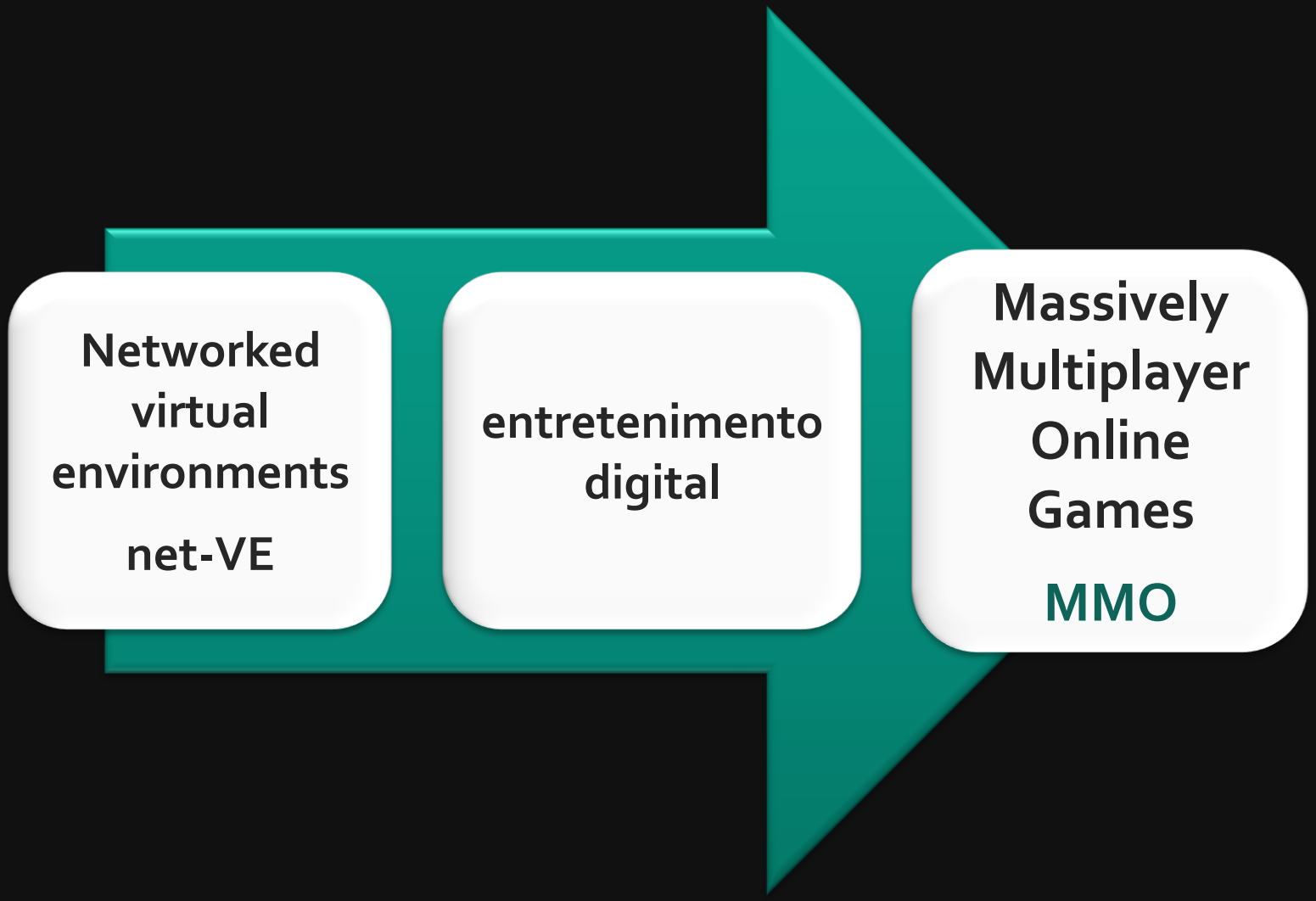
Nitin Gupta, Alan Demers, Johannes Gehrke

Computer Science - Cornell University

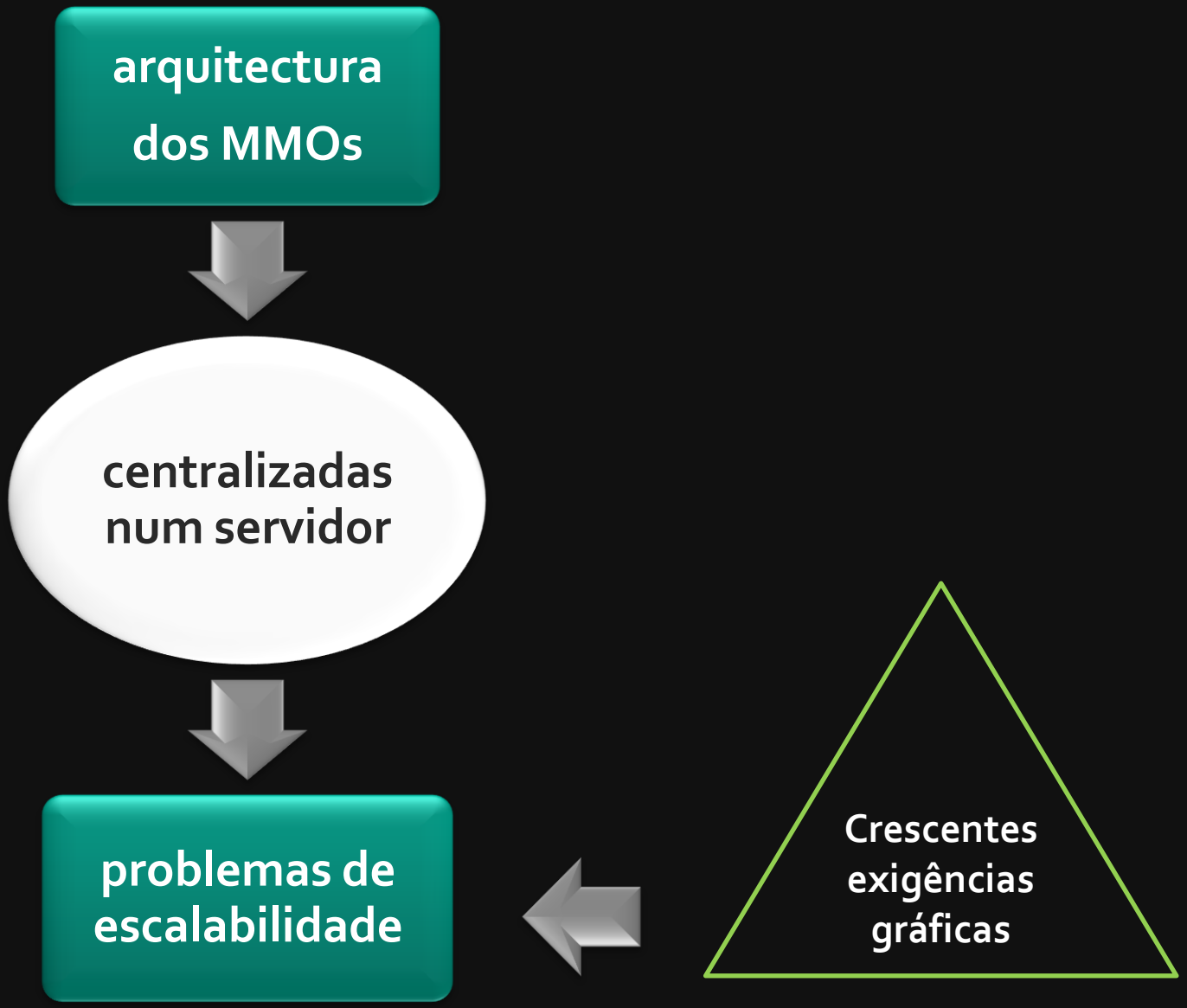
Philipp Unterbrunner, Walker White

Computer Science - ETH Zurich, Switzerland

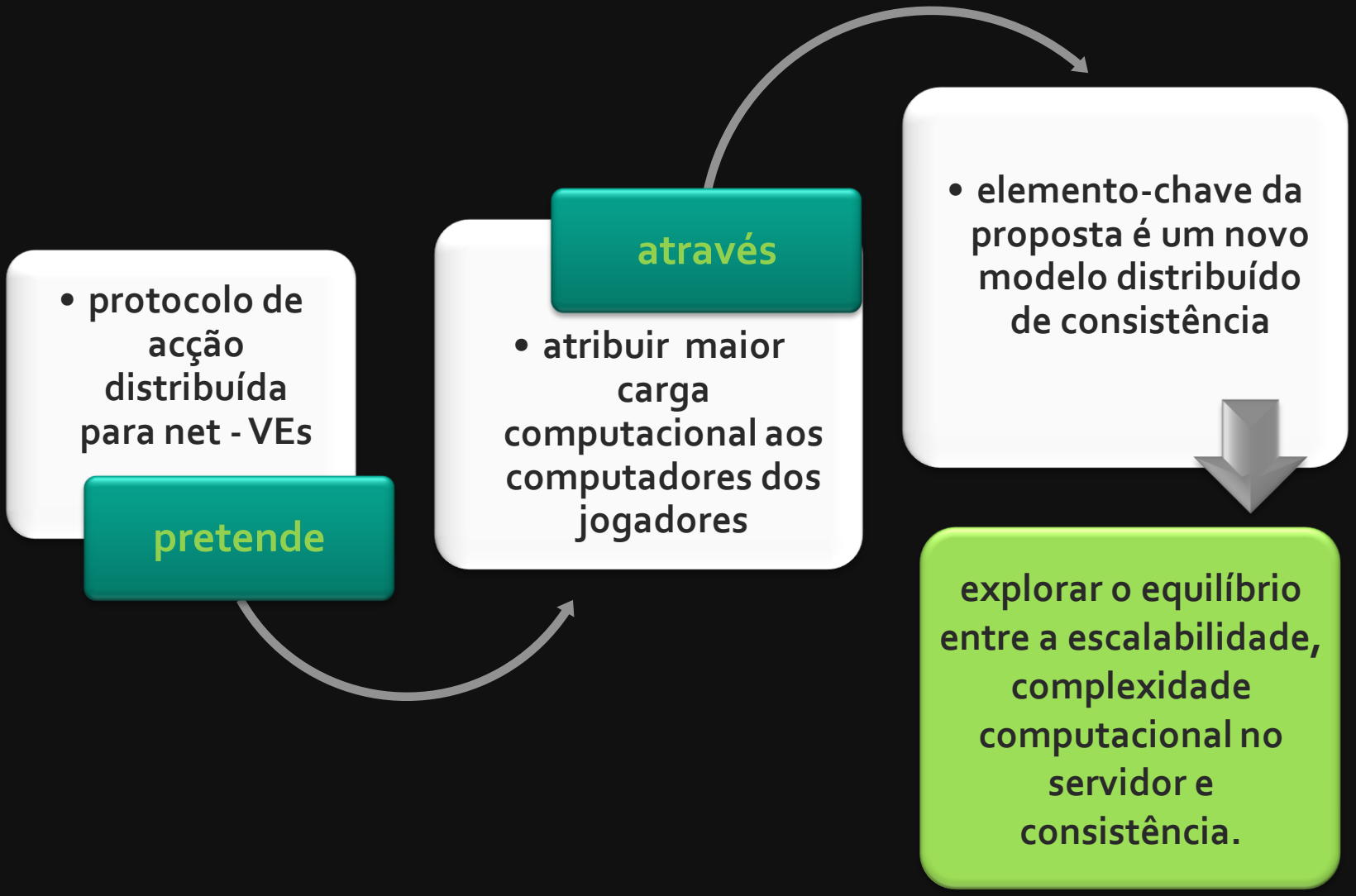
1. Objecto


Networked virtual environments
net-VE
entretenimento digital
Massively Multiplayer Online Games
MMO

2. Problemática



3. Proposta


• protocolo de acção distribuída para net - VEs
pretende
através
• atribuir maior carga computacional aos computadores dos jogadores
• elemento-chave da proposta é um novo modelo distribuído de consistência
explorar o equilíbrio entre a escalabilidade, complexidade computacional no servidor e consistência.

Arquitecturas Net-VE

- VE Centralizados
- VE Distribuídos
  - VE Cliente - Servidor
    - Protocolo implementado

VE Centralizados

Técnicas para obter escalabilidade num único servidor / central de servidores :

Zoning / Zoneamento

partição geográfica do VE, em áreas pequenas o suficiente, para um único servidor manipular;

Sharding / Fragmentar

o zoneamento funciona bem com um pequeno numero de servidores, ou seja, alguns milhares de jogadores. Mas a partir daqui, a dimensão do VE força as empresas MMO a instanciar mundos completamente separados chamados fragmentos;

Instancing / Instanciação

É essencialmente uma zona privada em que nenhum jogador pode entrar excepto aqueles que originalmente geraram a instância.;

VE Distribuídos

A carga computacional é distribuída entre clientes para obter escalabilidade.

VE Cliente - Servidor

- Consiste num *cluster* de servidores ao qual todos os clientes se conectam;
- É o programa cliente que contém a lógica do mundo virtual;
- O clientes iniciam e processam acções no ambiente;
- Uma **acção** é uma sequência de operações atómicas que actualizam o estado do mundo;

3 Protocolos populares:

Lock Based Protocols

Timestamp Based Protocols

Object Ownership

Protocolo Implementado

Proposta:

***Action Based Protocols* - Protocolo com base em acção**

Verifica-se a coerência ao nível das acções e não ao nível de objectos;

Usa-se semântica aplicacional para fornecer limites teóricos que formalmente provam a escalabilidade da abordagem;

Premissas:

Parte-se do princípio que o VE segue o modelo de um mecanismo de simulação discreto, onde as alterações de estado se dão em intervalos de tempo regulares, *simulation ticks*;

Assume-se o intervalo de tempo entre dois *ticks* consecutivos por **τ**.

Descrição inicial do protocolo:

1. As mensagens passadas entre os clientes e o servidor consistem principalmente de acções;

2. O estado do mundo virtual é um base de dados de objectos, *world state*;

3. Cada programa cliente mantém duas versões do *world state*:
uma versão optimista ζCO e uma versão estável ζCS

4. Para executar uma acção a, um cliente primeiro aplica a a ζCO e também envia a a um servidor para ser serializado;

Protocolo Implementado

5. Simultaneamente, o cliente recebe do servidor um fluxo serializado das acções originárias de todos os clientes e aplica-as em ordem a ζCS;

6. Os resultados da aplicação de acções originadas localmente em ζCO e ζCS são comparadas, e as discordâncias são reconciliadas, se necessário;

7. A única função do servidor é fornecer o *timestamp* e serializar as acções dos clientes;

8. O virtual *timestamp*, juntamente com as posições das acções na fila do servidor, estabelece sincronia virtual entre o servidor e os clientes;

Algoritmo 1 – Protocolo do lado do cliente

Algorithm 1: Client-Side Protocol

1 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.

2 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.

3 Assume that the client receives an action b from the server. There are two possible cases:

4 (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).

5 (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.

Algoritmo 2 – Protocolo do lado do servidor

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 $pos_C = pos(a)$.

Algoritmo 3 – Protocolo de reconciliação

Algorithm 3: Reconciliation Protocol

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

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

$\mathcal{Q} \leftarrow []$

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

apply a_i to ζ_{CO} producing result v

insert $\langle a_i, v\rangle$ into $\mathcal{Q}$

Protocolo Implementado

Este protocolo baseado em acção apresenta 2 vantagens:

- garante a resposta num RTT, permitindo que simultaneamente qualquer tipo de interacção, incluindo contenção de objectos, no ambiente virtual;
- o servidor central não executa quaisquer acções e, por conseguinte, está livre da lógica de jogo;

No entanto, uma grande desvantagem desta primeira abordagem do protocolo baseado em acção é que cada cliente vê e executa todas as acções para todo o mundo, resultando em alta carga computacional para os clientes e exigindo uma largura de banda substancial, tanto para os clientes e o servidor.

Assim, este primeiro protocolo, enquanto garante um tempo de resposta em um RTT e alcança coerência entre os clientes, tem uma limitada escalabilidade.
Para obter melhor escalabilidade é explorada a semântica de aplicação, ou semântica aplicacional .

Descrição da Semântica Aplicacional:

No domínio de protocolos baseados em objectos, várias optimizações foram propostas. A maioria dessas optimizações são variantes do **paradigma de área de interesse**. Nestes modelos o servidor restringe o conjunto de actualização de mensagens (e dados de objecto) enviado a um cliente através de uma restrição sintáctica (ex: a visibilidade de um Avatar num VE é uma aproximação popular)

Problemas do paradigma de área de interesse:

- restrições à visibilidade são aplicáveis apenas a acções de tipo movimentos, não generalizando bem acções arbitrárias;

- este tipo de restrições não são suficientes para garantir consistência (ex: transitividade de acções – personagens podem interagir umas com as outras, mesmo não estando no campo de visão)

Descrição da Semântica Aplicacional:

Para um cliente determinar o efeito de uma acção **a**, necessita possuir suficiente informação sobre o VE para determinar todas as acções que potencialmente influenciam **a**.
Esta dependência causal de acções depende da sua semântica e frequentemente não consegue ser abrangida por restrições sintácticas.

Uma acção na perspectiva da BD :

- uma acção **a** consiste num conjunto de leitura **RS(a)**, e um conjunto de escrita **WS(a)** e a execução de código necessário para computar valores do **WS(a)** dados valores do **RS(a)**;
- assumindo que **RS(a)** está contido em **WS(a)**, quebra-se a distinção entre conjuntos de leitura e de escrita;
- os algoritmos ocasionalmente usam escrita incondicional (blind write actions), denotada por: **a = W(S,v)** ;
- ou seja, **WS(a) = S** e por convenção **RS(a) = S** também.

Algoritmo 4 – Mundo Incompleto Protocolo do lado do cliente

Algorithm 4: Incomplete World Client-Side Protocol

1 The client maintains a queue

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

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.

2 Whenever the client executes an action a, it is 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 $\mathcal{Q}$, and the action a is sent to the server.

3 Assume that the client receives an action b from the server. There are three possible cases:

4 (Action b originated at some other client, or is a blind write created by the server): 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) $x \notin WS(\mathcal{Q})$. (This has the effect of updating items in the state that are not awaiting permanent values from the server).

5 (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 $\mathcal{Q}$. Otherwise, ζ_{CO} is reconciled with ζ_{CS} using Algorithm 3. In either case, a *completion message* $\langle a_i, u\rangle$ is sent to the server.

Algoritmo 5 – Mundo Incompleto Protocolo do lado do servidor

Algorithm 5: Incomplete World Server-Side Protocol

1 The server maintains the *authoritative state* ζ_S. It also maintains a global queue of ordered actions. For each action a in the queue it maintains the set $sent(a)$ of clients to which the action has been sent.
For any i, let $\zeta_S(i)$ be the state of the virtual world at the server after applying the effects of actions $a_1 \ldots a_i$. Then at time t, the server holds: (i) $\zeta_S(j)$ for the least j such that no response for a_{j+1} has yet been received, and (ii) $a_{j+1} \ldots a_n$.

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. It also sets $sent(a) \leftarrow \emptyset$

4 (b) It computes a reply to a using Algorithm 6.

5 When a completion message arrives at the server for a_i (Step 5 in the client-side protocol), the server holds it until $\zeta_S(i-1)$ is available. It then installs the values into ζ_S, resulting in $\zeta_S(i)$, and discards a_i from the action queue.

Algoritmo 6 – Fecho Transitivo (A)

Algorithm 6: Transitive Closure(A)

Require: $a_i, \ldots, a_n$ is the action queue
Require: a_{n+1} has just arrived from client C
Require: $+$ denotes prepending an action to a sequence
  $A \leftarrow \{a_{n+1}\}$
  $S \leftarrow RS(a_{n+1})$
  for $(j = n; j > i; j = j - 1)$ **do**
    if $WS(a_j) \cap S \neq \emptyset$ **then**
      if $C \in sent(a_j)$ **then**
        $S \leftarrow S \setminus WS(a_j)$
      else
        $S \leftarrow S \cup RS(a_j)$
        $A \leftarrow a_j + A$
        $sent(a_j) \leftarrow sent(a_j) \cup \{C\}$
  $A \leftarrow W(S, \zeta_S(S)) + A$
  return A

Modelo Primeira Barreira -First Bound Model

Segundo o Modelo de Mundo Incompleto, cada cliente avalia apenas um subconjunto "necessário" de acções (que efectivamente afectam o cliente).

Supondo que um cliente poderia avaliar um conjunto de acções **AS** num tempo constante **γ** independente do tamanho de **AS**.

Então o tempo para o servidor receber uma resposta para qualquer acção de um cliente seria no máx. **RTT+γ** .

Significando que o servidor poderá necessitar de enviar ao cliente todas as acções que "viu" no anterior **(RTT+γ) / τ** , fornecendo assim a **primeira barreira.**

Modelo Primeira Barreira -First Bound Model

A maior parte dos net-VE possuem propriedades estritas de localização que podem ser exploradas.

Cada participante pode ser representado por um tuplo, normalmente de grande dimensão, esta tuplo tem uma taxa máxima finita de mudança de posição.

Como resultado, muitas das acções estão restritas a uma esfera de raio fixo determinada pelo participante.

Por exemplo, quando um combatente está à procura dum alvo para atacar, esta é esfera é relativa ao poder de ataque e posição espacial.

Modelo Primeira Barreira -First Bound Model

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

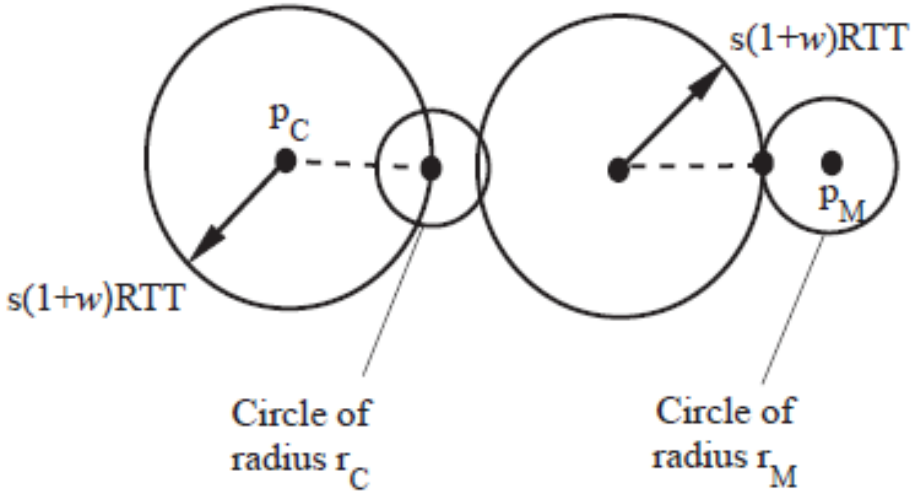


Fig. 4. The worst-case in First Bound Model

A equação fornece o **first-bound** ao número de acções que podem directamente entrar em conflito com as acções do cliente, representada como uma esfera centrada na posição do cliente no mundo virtual

Modelo Barreira de Informação – Information Bound Model

Algorithm 7: Information Bound Model

```
1  global actionCount, previousCount, lastCommitted,
   numClients

2  function onActionSubmission(action)
3  begin
4      A_actionCount ← action
5      let i = actionCount
6      for (j = 0; j < clientCount; j+ = 1) do
7          if |p_Ai − p_Cj| ≤ (2s × (1 + ω) RTT) + r_C + r_A
           then
8              clientConflicts_{i,clientConflictCount_i} ← j
9              clientConflictCount_i+ = 1
10         end
11     end
12     actionCount+ = 1
13 end

14 function onNextTick()
15 begin
16     for (i = previousCount; i < actionCount; i+ = 1)
       do
17         let S = RS(A_i)
18         let invalid = false
19         for (j = i − 1; j > lastCommitted; j− = 1) do
20             if isValid_j and S ∩ WS(A_j) ≠ ∅ then
21                 if |p_Ai − p_Aj| > threshold then
22                     invalid ← true
23                     break
24                 end
25                 S ← (S − WS(A_j)) ∪ RS(A_j)
26                 conflicts_{i,conflictCount_i} ← j
27                 conflictCount_i+ = 1
28             end
29         end
30         isValid_i ← not invalid
31     end
32     previousCount ← actionCount
33 end
```

- a função **onActionSubmission()** é chamada quando um cliente submete uma acção (adicionada à File global de acções), a função avalia o conjunto de clientes **(clientConflits**) que podem ter interesse na acção num futuro próximo;
- a função **onNextTick()** é invocada a cada tick, **τ**;
- o intervalo **[previousCount, actionCount]** fornece os identificadores de todas as acções submetidas no tick anterior

Para cada acção submetida **A**, a função **onNextTick()** avalia em conflitos um fecho transitivo de todas as acções não confirmadas conflituosas.

Se algumas das acções conflituosas se encontra a uma distância maior de **A** ,num dado threshold, então **A** é descartada.

Modelo Barreira de Informação – Information Bound Model

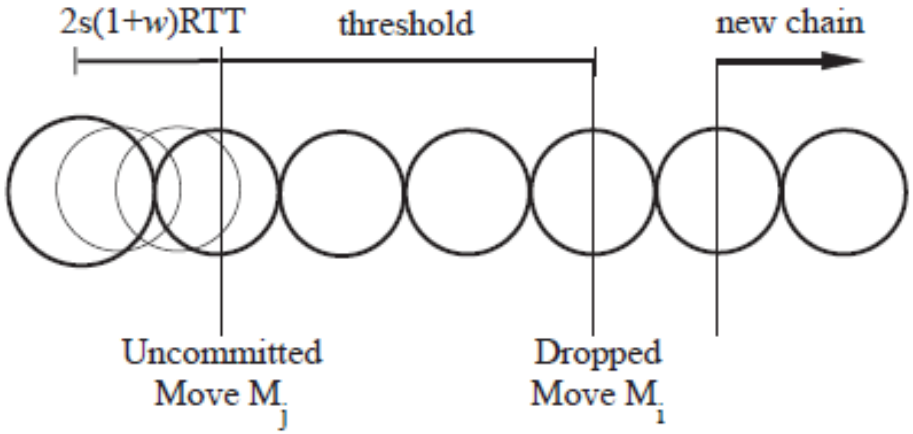


Fig. 5. Chain breaking in the Information Bound Model

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

First Bound Model + Information Bound Model

O **First Bound Model** e o **Information Bound Model**, fornecem dois limites.

- o primeiro limita o nº máx. de acções que necessitam ser enviadas a um cliente devido a conflitos directos, representada em função do tempo e distância no hiperespaço de atributos;
- o segundo limita também o nº de acções que podem tomar parte de quaisquer acções de fecho transitivo, representada em função da distância.

Combinando ambos, obtemos um limite "relaxado" sob o nº de acções enviadas a um cliente a cada tick, representados em função da distância e do tempo.

Optimizações ao Protocolo

Eliminação de acções inconsequentes

Assumido que uma acção apresentada por qualquer participante pode afectar futuras acções de todos os outros participantes que satisfaçam um certo valor limite de distância entre suas posições.

Afirma-se que o número de tais conflitos pode ser drasticamente reduzido com a integração não trivial de semântica MMO no sistema.

Podemos, portanto, alargar o sistema a fim de
permitir que os clientes especifiquem exactamente que tipo de acções e informações estão interessados , em vez de assumir absoluta uniformidade.

Optimizações ao Protocolo

Área de abate

Outra suposição feita é que a área de influência de qualquer acção é uma esfera centrada em seu ponto de ocorrência.

No entanto, a maioria das acções como o disparo de uma flecha, ou mesmo a deslocação, normalmente tem um vector velocidade associados.

Podemos portanto, integrar esse vector velocidade no cálculo para prever eventuais futuros conflitos.

$$\| \bar{p}_M + (\bar{v}_M \times (t_M - t_C)) - \bar{p}_C \| \leq (2s \times (1+\omega)RTT) + r_C,$$

Questões ???