Sistemas Distribuídos e Tolerância a Falhas

José Castanheira m3852
Óscar Pinto m4360

Scalability for Virtual Worlds

Introdução

- Ambientes virtuais em rede (AVR) são considerados a próxima onda de entretenimento digital.
- Exemplo popular são os Massively Multiplayer Online Games (MMOs).
- Arquitecturas actuais MMO são centradas no servidor em que toda a lógica do jogo é executado nos servidores da empresa que hospeda o jogo.
  - Leva a graves problemas de escalabilidade
  - MMOs exigem gráficos e sistemas de física o mais realístico possível.

Introdução
Exemplos

- Second Life
- Microsoft ESP
- World of Warcraft
- Habo Hotel
- Outras áreas de aplicação são o ensino e simulações militares para fins de formação e táctica.

Introdução

- Com a chegada da banda larga, a internet com baixa latência é agora omnipresente.
- Mas isto não é suficiente para resolver o questões de escalabilidade que os AVR começam a ter.
- Estes problemas de escalabilidade surgem em parte devido à necessidade de manter a consistência entre todos os jogadores.
- No melhor dos casos, a incoerência pode ser só passageira sem consequências a longo prazo.
- No entanto, na prática, pode facilmente causar problemas muito mais graves, como objectos perdidos ou duplicados durante uma operação financeira.

Introdução

- A escalabilidade de uma aplicação está fortemente relacionada com a pegada computacional de um único utilizador.

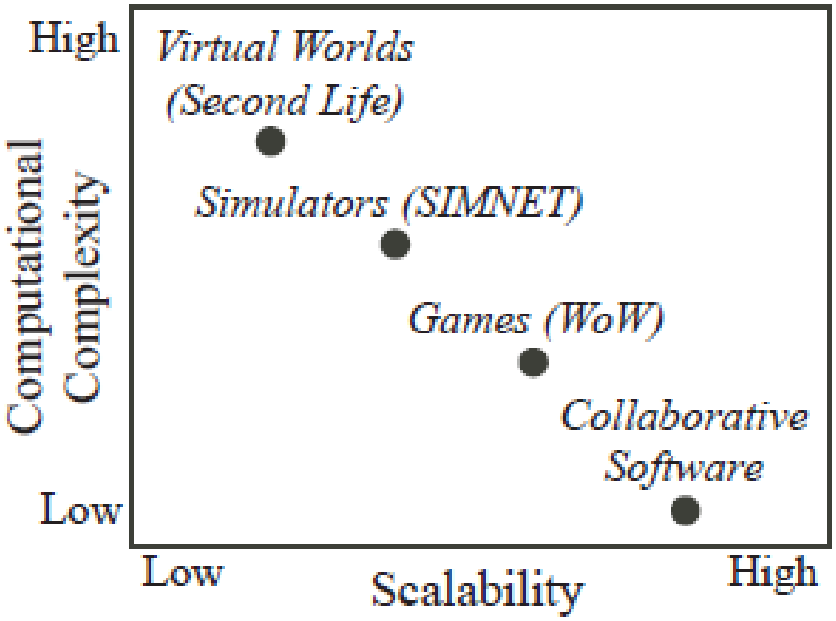


Fig. 1. Scalability versus Complexity

NETWORKED VIRTUAL ENVIRONMENTS (NVE)

- A consistência é importante.
- Para ser realista, todos precisam partilhar de uma visão única do ambiente virtual, “o estado do mundo”.
- Normalmente armazenam o estado do mundo numa base de dados (BD).
- Qualquer interacção com o mundo pode ser pensada como uma transacção
  - Fazer uma observação é uma consulta à BD sobre o estado do mundo.
  - Uma mudança de estado é uma actualização da BD.

NETWORKED VIRTUAL ENVIRONMENTS (NVE)

- Devido a problemas com as BD comerciais, optaram por utilizar BD comerciais apenas para confirmar e ler em *checkpoints* periódicos.
- Para interacções em tempo real, geralmente implementa-se uma camada própria de transacção antes da BD.
- Esta decisão não surgiu porque as transacções de BD são inadequadas para a tarefa!
  - Mas sim, porque as BD não são optimizadas para o tipo de processamento que as NVE necessitam para um bom desempenho em tempo real.

NVE - Arquitecturas - Centralizadas

- Empresas normalmente usam uma arquitectura com múltiplos servidores centrais para alcançar a escalabilidade.
- As técnicas mais utilizadas para atingir escalabilidade além de um único servidor são as seguintes:
  - **Zoning** – Técnica de Particionamento Geográfico.
    - Cerca de 12 Servidores e alguns milhares de utilizadores.
  - **Sharding** – Idêntico à anterior mas preocupa-se com o facto dos utilizadores poderem estar espalhados por todo o mundo, levando a *delays* enormes.
  - **Instancing** – Ao contrários das anteriores, limita -se apenas a pequenas partições do ambiente virtual. Limita severamente ai nteração do jogador.

NVE - Arquitecturas - Distribuídas

- Uma alternativa a vários servidores centrais, é o modelo distribuído.
- Computação distribuída pelos utilizadores = escalabilidade.
- Existe o modelo P2P mas acarreta mais falhas e ineficiências, não havendo escalonamento. Existe apenas uma sincronização virtual que garante a consistência na ordem dos eventos em todos os utilizadores.
- Em geral, o modelo distribuído cliente-servidor traz um equilíbrio entre a preservação dos interesses das empresas em exercer o controle, a escalabilidade do sistema e o atenuar dos problemas de nenhum controlo centralizado em relação a uma arquitectura P2P.
- Extensões para uma arquitectura híbrida que procura um equilíbrio entre P2P e cliente-servidor são o caminho para o futuro.

NVE – Cliente-Servidor

- Consiste num *cluster* de servidores no qual todos os utilizadores se conectam.
- Principal componente é o seu protocolo de consistência.
  - Uma vez que a computação é feita pelos utilizadores, um protocolo deve ser estabelecido entre os clientes e o servidor, garantindo assim a consistência e durabilidade de dados.
  - Existem 3 classes de protocolos.

NVE – Cliente-Servidor
Protocolos

- **Lock Based Protocols:** A fim de processar uma transacção, o cliente deve adquirir bloqueios globais sobre os objectos lidos e gravados pela operação.
  - Cliente contacta o servidor para um bloqueio.
  - Se obtém todos os bloqueios necessários, o cliente executa a transacção no seu estado local e transmite os efeitos da operação para o servidor.
  - O servidor transmite este efeito a todos os outros clientes que actualizam os seus estados locais, prosseguindo logo de seguida para a próxima transacção.
  - Este protocolo tem um grave problema, o tempo que demora a processar a próxima transacção é o dobro do tempo de ida e volta entre cliente e servidor

NVE – Cliente-Servidor
Protocolos

- **Timestamp Based Protocols:** Neste protocolo o controlo é efectuado através de *timestamps*.
  - Associa-se uma versão a cada objecto e um *timestamp* a cada transacção.
  - Os utilizadores executam as suas acções localmente, por vezes com versões "obsoletas" dos objectos.
  - O servidor recebe as diversas "histórias" dos diferentes clientes e cria uma multiversão global da história.

NVE – Cliente-Servidor
Protocolos

- **Object Ownership:** Difere dos protocolos *lockbased*, pois cada objecto é detido e gerido por exactamente um utilizador, conhecido como o dono do objecto.
  - Aos outros utilizadores é permitido armazenar em cache uma versão do objecto, mas não estão autorizados a fazer modificações no seu estado.
  - Embora este protocolo seja altamente escalável, não permite qualquer tipo de contenção do objecto no ambiente.
  - Se dois clientes querem alterar o estado do mesmo objecto, apenas o cliente que possuir o objecto lhe é permitido fazê-lo.
  - O servidor é responsável por garantir a equidade em quem deve ter a ”propriedade” dos objectos.

Protocolos Baseados em Acção – A Proposta

- Protocolos anteriores baseavam-se nos objectos.
- Estes protocolos verificam a consistência ao nível da acção.
  - Ex: Funções para actualizar o estado do jogo.
- Verificam a consistência ao nível da acção.
- São altamente escaláveis.

Protocolos Baseados em Acção - Algoritmos Básicos

- Mensagens passadas entre os clientes e o servidor são constituída principalmente por acções, ao invés de objectos.
- O estado do mundo virtual é uma base de dados de objectos, o ”estado do mundo”.
- Cada programa cliente mantém duas versões do estado do mundo: uma visão optimista (VO) e uma versão estável (VE).
- Para executar uma acção **a**, um cliente aplica **a** na VO, enviando também **a** para o servidor afim de ser serializada.

Protocolos Baseados em Acção - Algoritmos Básicos

- Paralelamente, o cliente recebe do servidor um fluxo serializado das acções originadas por todos os clientes, e aplica-as na VE.
- Os resultados da aplicação de acções originadas localmente no VO e VE são comparados e as discordâncias são conciliadas, se necessário.
- A única função do servidor é fazer o timestamp e serializar as acções dos clientes. Este timestamp virtual, juntamente com as posições de acções na fila no servidor, estabelece uma sincronia virtual entre o servidor e os clientes.

Protocolos Baseados em Acção
- Vantagens

- Garante uma resposta em cada comunicação (ida e volta), permitindo qualquer tipo de interacção em simultâneo.
- Uma segunda vantagem é que o servidor central não executa qualquer acção e, portanto, está livre da lógica do jogo.
- O servidor apenas acções faz timestamp as acções, colocando-as em filas de entrega para os clientes e gerindo o tráfego de rede.
  - Permite ao servidor lidar com um número muito grande de clientes.

Protocolos Baseados em Acção - Desvantagens

- Cada cliente vê e executa todas as acções para todo o mundo.
  - Resulta numa carga computacional elevada nos clientes.
  - Requisitos de largura de banda considerável tanto no cliente como o servidor.
- Escalabilidade muito limitada, tempo de resposta comunicação é uma viagem de ida e volta.
- Para conseguir uma melhor escalabilidade há que explorar a semântica da aplicação.

Utilizando a Semântica da Aplicação

- O servidor restringe o conjunto de mensagens de actualização enviadas para um cliente através de uma restrição sintáctica.
  - Ex: A visibilidade de um ”avatar” no mundo virtual.
- O servidor restringe o conjunto de mensagens de actualização enviadas para um cliente através de uma restrição sintáctica.
- Parece ser bom, mas só se aplica em acções que se saiba à partida o movimento que o cliente irá efectuar, gera problemas em acções arbitrárias.
  - Ex: Arquitectura *Ring* implica que o designer crie uma camada de obstrução que representa o objecto que bloqueia a visibilidade do cliente.

Utilizando a Semântica da Aplicação - Problema

- O uso de restrições sintácticas, tais como a visibilidade restrita, têm um problema mais profundo:
  - **Não se consegue manter a consistência.**
  - Não é possível abranger a transitividade das acções, as personagens podem interagir facilmente umas com as outras, mesmo sem se "verem".
- A área real que pode influenciar um avatar é muito maior do que a região visível.
- Para um cliente determinar o efeito de uma acção **a**, precisa de ter bastante informação sobre o mundo virtual, para assim determinar todas as acções com potencial de influenciar **a**.
- Esta dependência causal das acções depende da sua semântica e frequentemente não podem ser capturadas por restrições sintácticas.

Modelo Incompleto do Mundo

- Resolve o problema de consistência do modelo anterior.
- O cliente envia para o servidor uma mensagem de conclusão, sempre que é produzido um resultado estável de uma acção.
- O servidor utiliza essas mensagens para construir um estado estável do mundo.
- O servidor executa uma análise de leitura e escrita para determinar independentemente para cada cliente
- A vantagem deste modelo é que um cliente não tem (necessariamente) de avaliar cada acção, apenas aquelas que o afectam.
  - Poupa-se assim tempo de execução no cliente, bem como largura de banda na rede.

Modelo Incompleto do Mundo

- Pode ser tolerante a falhas do cliente, com um custo razoável de largura de banda.
  - Fazendo com que cada cliente envie mensagens de conclusão para cada acção realizada, e não apenas as suas próprias acções.
  - Com esta mudança, o único caso em que o servidor não recebe uma resposta a alguma acção é quando todos os clientes avaliam que acção falhou.
  - Em tais casos, é aceitável supor que a acção nunca foi submetida.
  - O cliente também pode ser optimizado em termos de memória.
    - O servidor pode informar o cliente periodicamente da última acção realizada, permitindo ao cliente eliminar as acções mais antigas e que já não interessam.

First Bound Model

- Com o Modelo Incompleto, cada cliente avalia apenas um subconjunto preciso de acções, as acções que realmente o podem afectar.
- Surgindo assim o The First Bound Model para prever um limite do número de acções que cada cliente pode avaliar.
- Dá assim um limite no número de acções gerais que podem afectar directamente todo o conjunto de acções de um cliente.
  - São exactamente essas acções que devem ser enviadas para o cliente.

Information Bound Model

- O número de acções não autorizadas que podem directa ou indirectamente causar um conflito é ilimitado.
- Este modelo decide se existe ou não uma acção que deve ser abandonada.
- Como todos os clientes não enviam as acções exactamente ao mesmo tempo, a ordem aleatória de chegada das acções no servidor vai garantir a equidade, ou seja, a probabilidade de uma acção recebida ser abandonada é a mesma para todos os clientes.
- Algoritmo é computacionalmente barato e, portanto, o modelo pode ser utilizado em ambientes de tempo real.

First Bound Model & Information Bound Model

- Os dois modelo juntos, geram dois limites.
- O primeiro limite é o número máximo de acções que precisam de ser enviadas a um cliente devido a conflitos directos.
  - Representado como uma função do tempo e distância no hiperespaço de atributos.
- O segundo limite é o número máximo de acções que podem fazer parte das acções de fecho (final)
  - Representado como uma função da distância.
- Combinando estes dois limites, temos o número limite de acções enviadas para um cliente a cada instante, representada como uma função de tempo e distância.

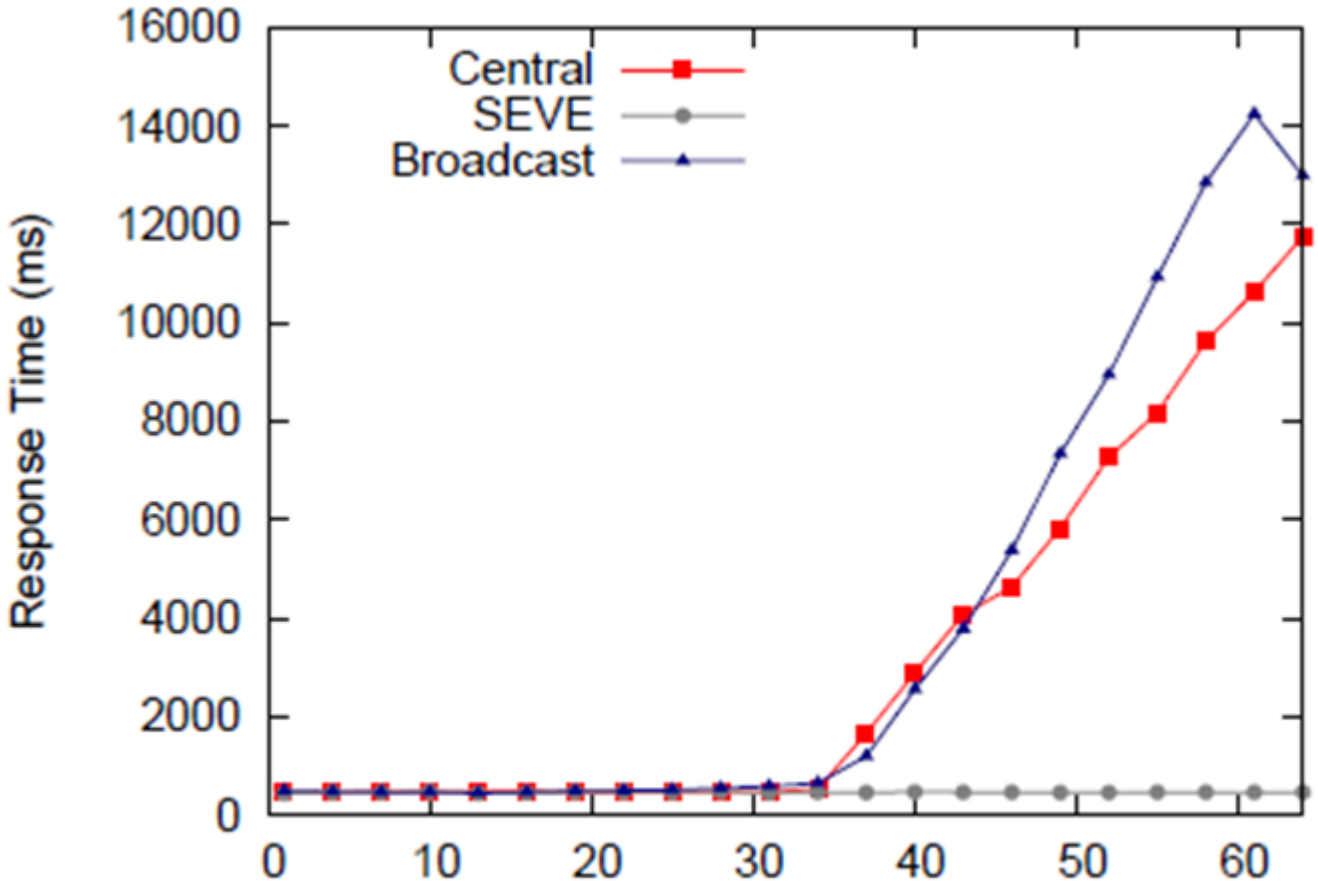
Testes

Virtual world size	1000 x 1000
Number of walls	0 – 100,000
Number of clients	0 – 64
Average latency	238ms
Maximum bandwidth	100Kbps
Moves per client	100
Move generation rate	Every 300ms per client
Move effect range	10units
Avatar visibility	30units
Threshold	1.5 × Avatar visibility

TABLE I

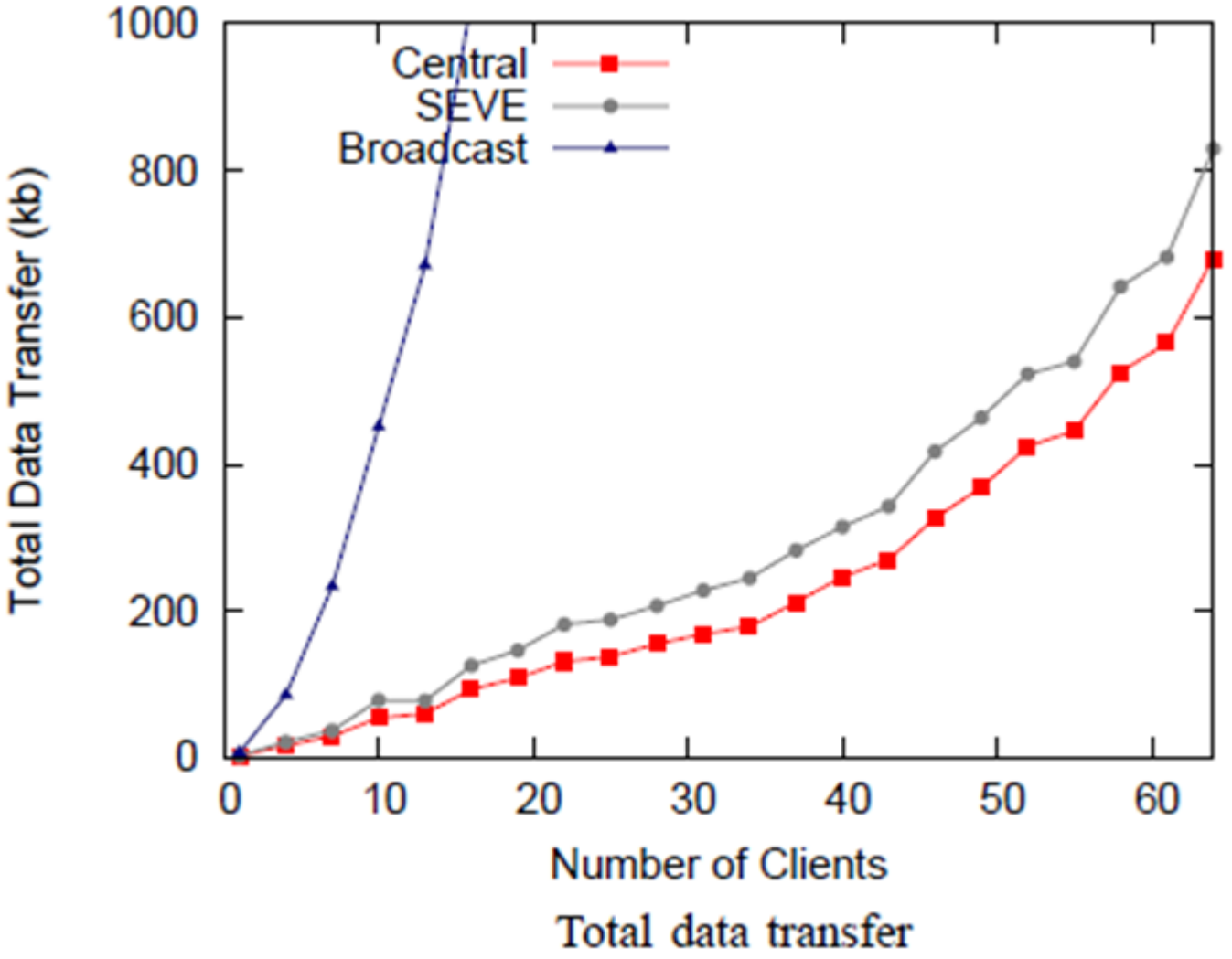
SIMULATION SETTINGS

Testes



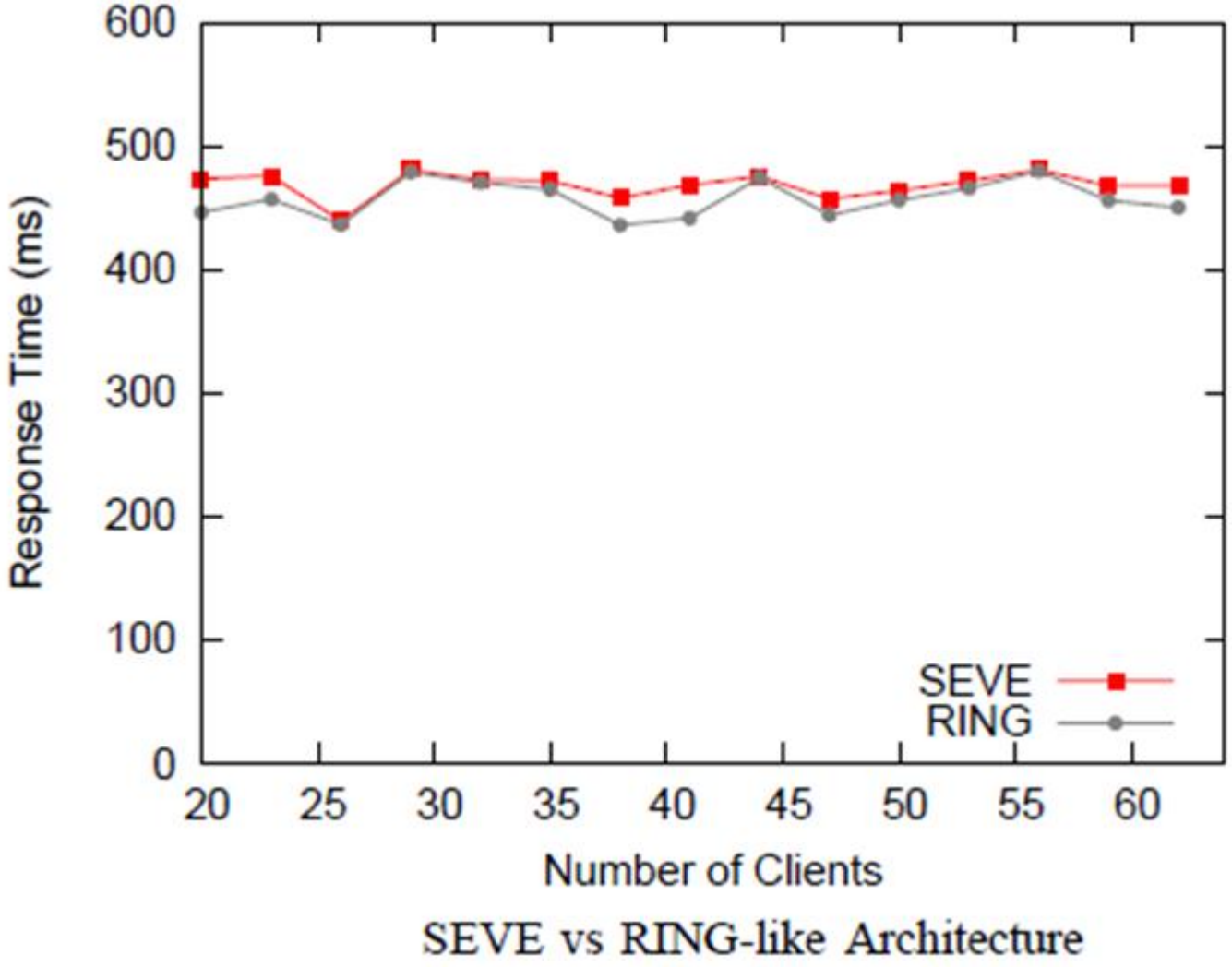
Scalability of SEVE vs. Central architecture

Testes



Total data transfer

Testes



SEVE vs RING-like Architecture

FIM