PROTOCOLO "SNOOPY" ADAPTATIVO PARA UM SISTEMA MULTIPROCESSADOR BASEADO EM HIERARQUIA DE BARRAMENTOS

Edward D.M.Ordoñez, Martha X.T.Delgado, Celso A.S.Santos, Sergio T.Kofuji [1]

Laboratório de Sistemas Integráveis
Departamento de Engenharia Eletrônica
Escola Politécnica da Universidade de São Paulo

RESUMO

Este trabalho apresenta e descreve um protocolo adaptativo de coerência de cache baseado nos esquemas "snoopy" para um sistema multiprocessador de memória compartilhada baseado na hierarquia de barramentos. O protocolo permite diminuir o tráfego adicional na rede produzido pelas ações de coerência e se adapta dinamicamente ao compartilhamento e aos padrões de referência que variam amplamente para os diferentes programas aplicativos. A sua avaliação de desempenho é feita com uma análise comparativa com os protocolos Dash e SCI (baseados em esquemas diretórios).

ABSTRACT

This paper shown and depict the adaptative cache coherence protocol based on snoopy protocol for large scale multiprocessors based on hierarquical bus. The protocol allow to dimish the aditional traffic of network caused for coherent actions and adjustment dynamically for shared and standard references that changes for differents workloads. The performance of protocol made with comparative analysis of Dash and SCI systems (based-on directory schemes).

[1] Pesquisadores do Laboratório de Sistemas Integráveis (LSI/EPUSP)
E-mail: (edmoreno, mxtd, saibel, kofuji) @lsi.usp.br

1. INTRODUÇÃO.

Com o objetivo de satisfazer as necessidades computacionais dos próximos anos, os multiprocessadores de memória compartilhada devem ser escaláveis. A importância destas máquinas é que oferecem o aumento necessário na potência de processamento para as tarefas computacionais paralelas enquanto mantêm as vantagens do seu modelo simples de programação. O desafio real é conseguir que o desempenho destes sistemas seja linear com o seu custo.

Existe uma ampla variedade de arquiteturas de multiprocessadores escaláveis. Algumas delas tentam solucionar as limitações de escalabilidade dos sistemas baseados em barramento simples introduzindo arquiteturas de múltiplos barramentos **[Jou92,Wils87],** geralmente na forma de hierarquia de barramentos **[Wins88, Ande92, Good88].** Nestes casos, precisa-se de extensões apropriadas dos protocolos com caches monitoras ("snoopy"). Outros multiprocessadores são orientados à redes de interconexão mais gerais e aos esquemas baseados em diretórios. Também existem soluções que combinam as vantagens das duas propostas anteriores **[Leno90, Leno92a/b]**.

O nosso interesse é estudar multiprocessadores escaláveis que aproveitem as vantagens dos protocolos de coerência de cache baseados em hardware, especialmente dos orientados a barramentos. É bem sabido que as arquiteturas de simples barramento compartilhado foram bem sucedidas no passado e que a limitada largura de banda do barramento restringe o número de processadores que podem ser efetivamente conetados (máximo 30 **[Jou92]).** Então é necessário construir uma rede de interconexão que mantenha a estrutura lógica do barramento simples aliviando os seus problemas de saturação.

Existem várias formas práticas de construir uma rede que trabalhe logicamente como um barramento compartilhado e que permita conectar um número grande de processadores **[Wils87, Wins88, Ande92, Good88].** Usar múltiplos barramentos para conseguir mais de uma transferência no tempo é um mecanismo complexo e custoso de manter coerência num sistema multiprocessador de memória compartilhada que usa a tecnologia de bisbilhoteiar os barramentos.

Este trabalho apresenta a análise de uma implementação de um sistema multiprocessador de memória compartilhada baseado na hierarquia de barramentos e **propõe um protocolo de coerência de cache implementado em hardware**, e baseado no esquema com caches monitoras (protocolos "*snoopy*") para a arquitetura em questão. O protocolo proposto apresenta a idéia de propriedade de bloco o que produz uma minimização do tráfego adicional na rede de interconexão produzido pelas ações de coerência de cache e permite adaptar-se dinamicamente ao compartilhamento e aos padrões de referência que variam amplamente para os diferentes programas aplicativos. Na seção 2 é mostrada a arquitetura de interesse, na seção 3 descreve-se o nosso protocolo onde se mostram os seus estados e o seu funcionamento. A seção 4 fornece exemplos das ações do protocolo, na seção 5 considera-se o desempenho do sistema através da análise comparativa com as ações de outros protocolos e finalmente a seção 6 fornece as conclusões e propõe futuros desenvolvimentos na área.

2. ARQUITETURA.

Na figura 1 mostra-se a arquitetura de um sistema escalável, um super-conglomerado *supercluster*. Este super-conglomerado está conformado por vários aglomerados (*clusters*) interconetados através de uma rede de interconexão (rede do super-conglomerado: que pode ser um sistema com barras cruzadas (*crossbar*), uma rede em malha ("mesh"), baseada também em barramento, etc). Cada conglomerado é formado com um número relativamente pequeno de processadores (N=4,8) que são interconetados à uma rede baseada em barramento, a qual é de custo baixo, eficiente (baixa latência) no caso de poucos elementos e permite a implementação de um sistema de coerência baseada nos esquemas bisbilhoteios com caches monitoras. No nível do conglomerado, o sistema de coerência pode ser mantido com o protocolo de invalidação de Illinois ou MESI **[Papa84]** ou qualquer outro protocolo de invalidações, e no nível do super-conglomerado pode ser um esquema baseado no bisbilhoteio (*snooping*), se a rede for barramento.

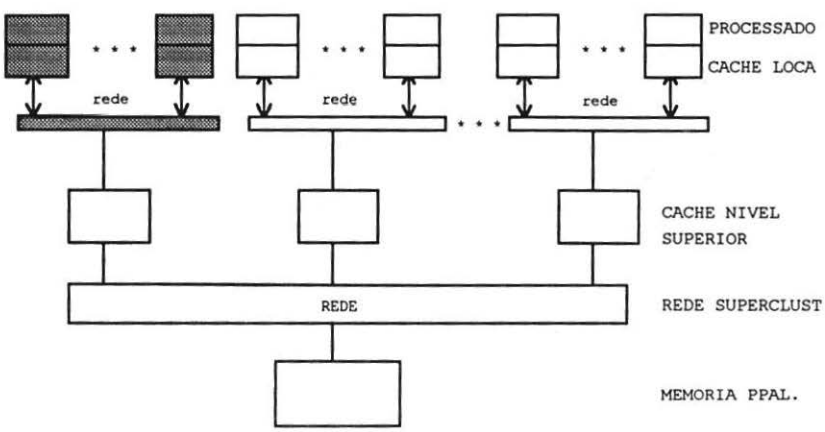

Figura 1. Super-conglomerado de um Sistema Multiprocessador Escalável.

3. PROTOCOLO DE COERÊNCIA.

O protocolo é projetado para executar numa arquitetura que consiste na hierarquia de barramentos. No nível mais baixo estão os processadores com suas respectivas caches locais. Os processadores são agrupados formando conjuntos deles ,conglomerados - *clusters*, e cada conglomerado possui de dois a oito processadores. Cada cache possui uma estrutura dual que permite simultaneamente acessos do processador e inspeções às ações de coerência de cache produzidas por pedidos no barramento. Quando um processador faz um pedido, o cache local diretamente conetado ao processador tenta servi-lo. Se uma falha (*miss*) acontece, o pedido é transmitido ao seguinte nível de cache através do barramento. O pedido pode continuar propagando-se até que o dado solicitado seja obtido. Devido à propriedade de localidade dos programas, uma fração grande das referências a memória feitas pelo processador podem ser satisfeitas no primeiro nível.

Os objetivos desta proposta são:

* Simplificar a implementação usando tecnologia atual de hardware,
* Minimizar o tráfego adicional da rede produzido pelas ações de coerência de cache,
* Adaptar dinamicamente ao compartilhamento e aos padrões de referência que variam amplamente para os diferentes programas.

O protocolo é projetado para minimizar o tempo de resposta e o tráfego na rede, especialmente para os níveis altos da hierarquia. A política de atualização caso uma substituição se faz com uma operação de *write back* ao nível seguinte. Só quando um cache do segundo nível faz uma substituição é que acontece uma migração em direção à memória.

3.1 Estados do Protocolo:

Dividimos os estados do protocolo em dois níveis: em aqueles correspondentes aos caches do nível inferior (caches locais) denominados estados ao nível do processador (mostrados na tabela 1) e estados do não-processador (mostrados na tabela 2, aqui o termo remoto e usado para indicar acessos pedidos pelo nível superior.).

TABELA 1. Estados dos caches do nível Inferior;

ESTADO	*DESCRIÇÃO*
Inválido (IN)	*O bloco não esta presente no cache ou não contém um dado útil*
Compartilhado (CO)	*O bloco é válido e múltiplas cópias podem existir.* ** Uma leitura pode ser feita localmente* ** Uma escrita não é permitida sem antes ter a propriedade do bloco* ** Não precisa fazer uma operação de write back*
Privado (PR)	*O bloco é de propriedade exclusiva e suporta operações locais de escrita e leitura* ** No caso de uma escrita remota, o bloco é invalidado e concede a propriedade ao cache solicitante.* ** No caso de uma leitura remota, o bloco é fornecido e cede a propriedade.* ** Write back só é necessário numa substituição*

3.2 Pedidos à Rede.

Para manter a coerência do sistema de memória, é preciso transmitir sinais à rede. Estes sinais são gerados dependendo da ação (leitura ou escrita) e do estado do bloco de cache. A seguir se descrevem tais sinais:

- *Leitura*: Um pedido para realizar uma leitura de um determinado bloco,
- *Leitura Descendente*: indica que é necessário ler um dado que está num nível inferior na hierarquia,
- *Leitura Exclusiva*: para carregar um bloco quando acontece uma operação de falha na escrita,
- *Leitura Exclusiva Descendente*: é uma leitura exclusiva ao nível inferior da hierarquia,
- *Invalidação*: um pedido para invalidar outras cópias de um determinado bloco,
- *Invalidacão Descendente*: para invalidar cópias no nível inferior da hierarquia,

- *Write Back*: para indicar que um bloco deve ser levado ao nível superior da hierarquia por ter acontecido uma substituição de um bloco de cache no nível inferior.

TABELA 2. Estados do Nível Hierárquico

ESTADO	*DESCRIÇÃO*
Inválido (IN)	*O bloco não contém o dado válido ou não esta presente*
Compartilhado (CO)	*O cache que possui este bloco tem a sua propriedade. Múltiplas cópias podem existir nos níveis inferiores.* * *Numa leitura, o bloco fornece o dado e concede a propriedade* * *Numa escrita remota, a cópia é invalidada e concede a propriedade* * *Numa escrita local, move a propriedade ao cache solicitante.*
Válido (VA)	*O bloco é válido, uma cópia compartilhada com a propriedade existe num cache superior ou na memória principal.* * *Um pedido de escrita local produz uma invalidação das outras cópias e muda ao estado hierárquico.* * *Uma escrita remota produz invalidações.*
Hierárquico (HI)	*Um cache inferior possui a propriedade do bloco.* * *O bloco não precisa estar fisicamente* * *Este estado serve como guia do caminho para outras referências de outros processadores de outros conglomerados.*

3.3. Ações do Protocolo.

As ações do protocolo são especificadas com base nas operações de escrita e leitura feitas pelo processador. Estas transições de estados são mostradas na figura 2.
1) Sucesso de Leitura : Quando o cache do nível superior recebe um pedido de leitura do nível inferior e encontra o bloco no estado válido ou compartilhado, acontece um sucesso de leitura (*read hit*). O bloco é transmitido ao cache solicitante e o estado não muda. O estado do cache local muda para compartilhado.

2) Falha de Leitura: Uma falha de leitura (*read miss)* acontece quando o cache superior recebe uma mensagem de pedido de leitura desde o barramento inferior. Podem acontecer as seguintes ações:

* Se um dos caches locais em estado compartilhado ou privado tem o dado, então ele impede ao nível superior de fornecê-lo e o transmite diretamente e muda o estado do cache solicitante para compartilhado.

* Se o cache do seguinte nível tem o bloco no estado compartilhado ou privado, então ele fornece o dado ao cache solicitante através do barramento. O cache solicitante muda seu estado para compartilhado.

* Se o bloco do cache superior é inválido, então ele gera um pedido à rede superior. Se algum dos outros caches superiores possui o dado então o fornece diretamente através da rede. Se este dado se encontra no cache superior no estado hierárquico, então se produz um pedido de leitura para o nível superior e

simultaneamente gera um sinal que inibe à memória de fornecer o dado. O cache com o dado transmitirá o dado e todos os caches no caminho com uma cópia do dado mudaram seus estados ao estado compartilhado.

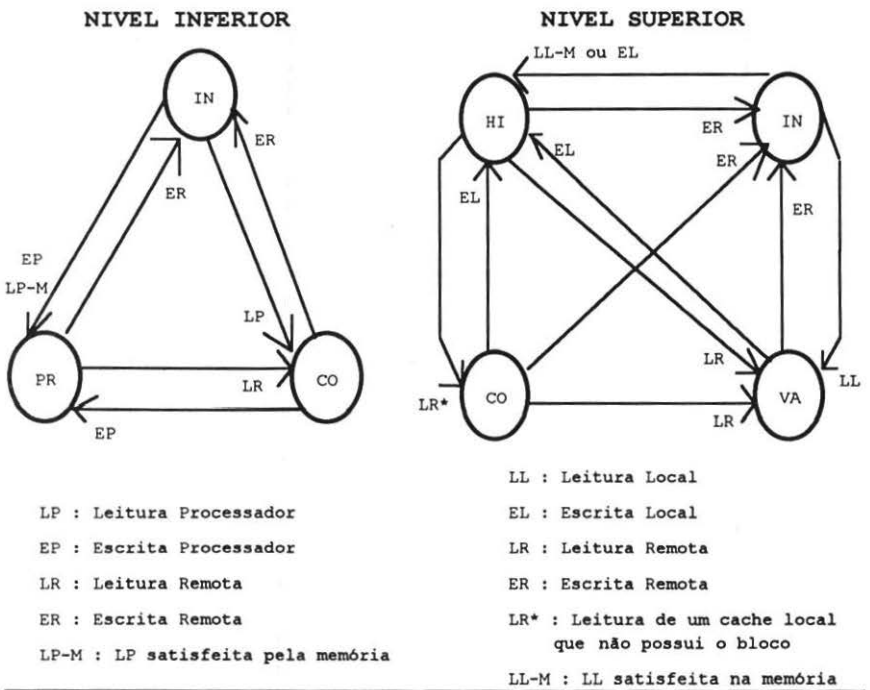

Figura 2. Diagrama das Transações do Protocolo Proposto

3) Sucesso de Escrita: Dependendo do estado do bloco, acontecem as seguintes ações no caso de um sucesso de escrita (*write hit*):

* Se um processador faz um pedido de escrita e um cache do nível inferior possui o bloco em estado privado, então a escrita se faz localmente e sem mudança de estado.

* Se o bloco está em estado compartilhado, deve-se procurar ter a propriedade dele. Um sinal de invalidação é transmitido ao barramento e todos aqueles caches com uma cópia do dado mudam seu estado para inválido. Se a cópia estiver também no outro nível da hierarquia, então ele também fica com uma cópia inválida, mas para indicar que a cópia existe sob ele, muda seu estado para hierárquico.

* Se o cache superior estiver em estado inválido, o dado sera procurado num outro conjunto. Se este cache tiver o dado em estado compartilhado ou

hierárquico, então gera sinais de invalidação sob ele e transmite o dado ao nó solicitante. O cache superior deste conglomerado recebe o dado e muda seu estado para o estado hierárquico e transmite o dado sob ao cache solicitante que coloca seu estado em privado.

* Se o cache superior do outro conglomerado estiver com o bloco no estado inválido, então a memória fornece o dado.

4) Falha na Escrita: No caso de uma falha de escrita (*write miss*) um bloco é carregado da mesma forma que no caso de falha de leitura exceto que o bloco em questão esteja no estado compartilhado. Neste caso, todos os caches que possuem cópia do bloco ficam com o estado inválido e o cache superior muda para o estado hierárquico.

5) Substituição; Somente se gera uma operação de *write back* quando o bloco selecionado para substituição pertence ao cache superior e este se encontra no estado compartilhado. A operação *write back* se faz já que é possível que o estado do bloco tenha mudado no tempo que ele estava com a propriedade. Além de fazer a atualização deste bloco à memória se transmite um sinal de invalidação aos caches inferiores que possuem cópia do bloco.

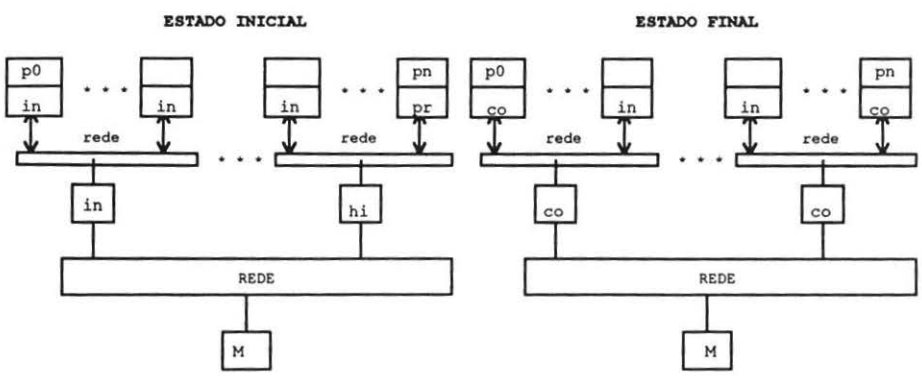

Figura 3. Exemplo do Protocolo: Caso Leitura

4. EXEMPLO DAS AÇÕES DO PROTOCOLO.

4.1 Caso Leitura.

O processador P1 faz um pedido de leitura ao bloco X que se encontra no estado privado no processador Pn, como é indicado na figura 3. Como o bloco pedido no está no cache local, é gerado um pedido de leitura à rede (ao barramento inferior) o qual é tranmitido ao barramento quando se ganha o seu acesso. Como nenhum outro cache do conglomerado inferior possui o dado, gera-se um pedido de leitura ao seguinte nível (ao cache superior). Similarmente, como

o bloco não está neste cache, é gerado um pedido de leitura à REDE (barramento superior). O cache superior do conglomerado N se encontra no estado hierárquico (hi), indicando que neste nível da hierarquia se encontra o bloco de interesse. Neste momento, este cache gera simultaneamente um sinal de reconhecimento para desativar a resposta da memória principal e um sinal de pedido de leitura para a parte inferior da hierarquia solicitando o bloco em questão. Uma vez o bloco alcançado no cache do processador N, é transmitido pela rota anterior e muda o estado de todos os caches no seu caminho para o estado Compartilhado. A memória fica também com uma cópia do bloco (operação write back).

4.2 Caso Escrita.

Supomos que o estado inicial do sistema é o estado inicial do caso da leitura. Ao invés de fazer uma leitura, se faz uma operação de escrita, como é indicado na figura 4. Acontece uma sequência de ações muito sililares ao caso da leitura. Ao invés de gerar um pedido de leitura, se faz um pedido de leitura exclusiva. Quando se obtem acesso ao bloco de interesse, o cache que possui o bloco muda o seu estado a inválido (in), o cache superior muda de hierárquico (hi) ao estado inválido (in), o cache superior do conglomerado que contém o cache ativo (que faz o pedido de leitura) muda de inválido para hierárquico (indicando que este conglomerado contém o bloco), e o cache ativo fica com a propriedade do bloco e muda de inválido para privado (pr). Neste caso, não se faz write back à memória.

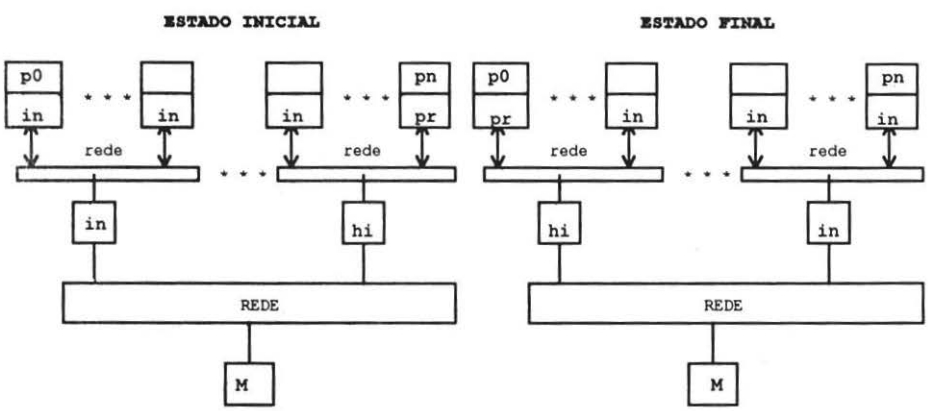

Figura 4. Exemplo do Protocolo: Caso Escrita

5. DESEMPENHO ESPERADO.

Os protocolos com caches monitoras e os esquemas diretório são as duas propostas básicas para manter coerência de cache. Atualmente os primeiros apresentam o problema de que poucos processadores podem facilmente saturar o barramento (devido ao melhoramento rápido na sua tecnologia) e os esquemas diretório podem ser aplicados a qualquer rede de interconexão e são relativamente mais escaláveis.

Considerando a desvantagem anterior dos esquemas baseados em barramentos, a seguir apresentam-se razões que explicam porque trabalhar com uma arquitetura escalável baseada em hierarquia de barramentos.

Pode-se comparar o esquema proposto com dois esquemas diretórios escaláveis e mostrar as vantagens e as desvantagens. O primeiro esquema diretório é implementado no multiprocessador DASH, que utiliza um vetor de bits completo (esquema diretório completamente mapeado) no diretório local, onde cada bloco mantêm informações da localização de todos os blocos **[Leno90].** E o outro esquema é usado no sistema IEEE SCI standard (*Scalable Coherent Interface*), onde usa-se um diretório encadeado **[P1596, Gust92].**

Vantagens e Desvantagens:

- Os esquemas com caches monitoras não tem a sobrecarga de procura e manutenção dos esquemas diretório. O protocolo com diretórios antes de realizar alguma operação tem que transmitir o pedido ao diretório (isto pode produzir latências consideráveis em sistemas escaláveis.

Por exemplo, no caso de uma falha de leitura (*read miss*) num bloco compartilhado o pedido é levado ao barramento e o proprietário do bloco solicitado fornece o dado e o estado do bloco é mudado simultaneamente em todos os caches. Num esquema diretório, o nó solicitante transmite o pedido ao diretório e logo o controlador de diretório faz o pedido ao proprietário real do bloco.

- O esquema com caches monitoras proposto gera menos comunicação entre os conglomerados.

Por exemplo; quando acontece uma falha de escrita (*write miss*) no DASH todos os caches que possuem uma cópia do bloco devem transmitir um sinal de reconhecimento da invalidação ao nó solicitante. No SCI são necessários mais mensagens para percorrer do princípio ao fim a lista encadeada do bloco.

- Contrariamente aos esquemas diretórios que precisam de um espaço de memória para manter os diretórios, os esquemas com caches monitoras só necessitam de uns poucos bits que estabelecem os estados de um bloco.

- A capacidade de transmissão atômica dos barramentos pode fazer que uma mensagem seja transmitida a todos os nós no mesmo tempo, o que pode reduzir o tempo de manutenção da coerência. Contrariamente, nos esquemas diretório o fato de transmitir um sinal aos nós pode só ser executado serialmente.

Por exemplo: no caso de um sucesso de escrita (*write hit*) a um bloco compartilhado o sinal de invalidação pode-se transmitir atomicamente pelo barramento a todos os outros caches. Nos esquemas diretório o processo de invalidação tem que ser feito serialmente.

- A principal desvantagem de nosso esquema é a sua complexidade em hardware e os custos, os quais podem limitar a sua escalabilidade. E a bem conhecida limitada banda passante da rede ("*bandwidth*").

É importante ressaltar que os esquemas diretório são efetivamente mais escaláveis que os esquemas com caches monitoras mas apresentam o custo de acessos com uma latência possivelmente maior. Por exemplo, considerando o caso do SCI com uma lista encadeada de miles de nós interconetados. O Dash tenta limitar esta possível alta latência considerando uma certa complexidade no hardware.

Finalmente, ressaltamos que estes esquemas melhorarão no futuro com os avanços rápidos alcançados com a melhoria da tecnologia.

6. CONCLUSÕES.

Este trabalho apresentou e descreveu um protocolo de coerência de cache para um multiprocesador de memória compartilhada baseado numa hierarquia de barramentos. O protocolo possui propriedade de bloco o que produz uma diminuição do tráfego pela rede, produzido pelas ações de coerência.

Os esquemas de coerência baseados em hardware com caches monitoras tem a desvantagem de suportar poucos processadores de alto desempenho. Os esquemas baseados em diretórios são muito mais escaláveis mas podem apresentar latências de acessos a memória muito altas. Nós comparamos a nossa proposta com estes últimos e com esquemas *snoopy* tradicionais e mostramos que o desempenho da arquitetura apresentada é promissor.

Com o fim de estabelecer uma validação real do protocolo e do sistema hierárquico, estamos construindo um simulador da arquitetura. Este simulador contém o sistema de memória e o protocolo de coerência. A avaliação será feita com simulação comandada por execução. O padrão das referências a memória e os tempos de execução dos processadores para aplicações reais serão obtidos usando o Tango-Lite, que é um ambiente de simulação desenvolvido pela Universidade de Stanford **[Herr93, Gold93].** As aplicações consideradas serão o conjunto de programas do Splash **[Sing91],** "benchmark" para sistemas multiprocessadores de memória compartilhada. (O Splash contém os seguintes programas aplicativos: water, ocean, mp3d, locus route, cholesky e barnes). Os resultados das simulações permitirão obter o número de processadores que podem formar um conglomerado e o número destes que podem ser interconectados.

7. REFERÊNCIAS.

[Ande92] ANDERSON, C.; BAER, J.L. A Multi-Level Hierarchical Cache Coherence Protocol for Multiprocessors. **Technical Report**, University of Washintong, 1992

[Gold93] GOLDSCHMIDT, S.R. Simulation of Multiprocessors: Accuracy and Performance. **PhD. Thesis**, DEE, University of Stanford, Jun. 1993.

[Good88] GOODMAN, J.R.; WOEST, P.J. The Wisconsin Multicube: A New Large Scale cache Coherent Multiprocessor. In: Ann. Int. Symp. on Computer Architecture, IEEE, 15th. **Proceedings**, p. 422-31, 1988.

[Gust92] GUSTAVSON, D.B. The Scalable Coherent Interface and Related Standards Projects. **IEEE Micro Chips, Systems, Software and Applications**, p. 10-22, Feb. 1992.

[Herr93] HERROD, S.A. Tango-Lite: A Multiprocessor Simulation Enviroment - Introduction and User's Guide. **Technical Report**, Stanford University, Sep. 1993.

[Jou 92] JOU, T.S.; ENBODY, R. A Scalable Snoopy Coherence Scheme on Distributed Shared-Memory Multiprocessors. In: IEEE Supercomputing. **Proceedings**, p. 652-60, 1992.

[Leno90] LENOSKY, D. et al. The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor. In: Ann. Symp. on Computer Architecture, IEEE, 17th. **Proceedings**, p. 148-59, 1990.

[Leno92a] LENOSKY, D. et al. The Stanford DASH Multiprocessor. **IEEE Computer**, p. 63-79, Mar. 1992.

[Leno92b] LENOSKY, D. et al. The DASH Prototype: Implementation and Performance. **ACM Transactions on Computer Systems**, p. 92-103, 1992.

[Papa84] PAPAMARCOS, M.; PATEL, J. A Low Overhead Coherence Solution for Multiprocessors with Private Cache Memories. In: Ann. Int. Symp. on Computer Architecture, IEEE, 11th. **Proceedings**, p. 348-54, 1984.

[P1596] P1596 Working Group, SCI (Scalable Coherent Interface) -Cache Coherence Overview. **Technical Report** Revision 0.33, IEEE Computer Society, Nov. 1989.

[Sing91] SINGH, J.P.;WEBER, W.D.; GUPTA, A. Splash: Stanford Parallel Applications for Shared Memory. **Technical Report CSL-TR-91-469**, Stanford University, Apr. 1991.

[Tvin94] TVING, Ivan. Simulation Environment for Multiprocessor Cache Based Supercomputer using the Scalable Coherent Interface (SCI-IEEE-P1596) in a Ring-Based Topology. **Master's Thesis, DTH ID-E 579**, University of Denmark, p. 199, Feb. 1994.

[Wils87] WILSON, A.W. Hierarchical cache/Bus Architecture for Shared Memory Multiprocessors. In: Ann. Int. Symp. on Computer Architecture, IEEE, 14th. **Proceedings**, p. 244-52, 1987.

[Wins88] WINSOR, D.C.; MUDGE, T.N. Analisys of Bus Hierarchies for Multiprocessors. In: Ann. Int. Symp. on Computer Architecture, IEEE, 15th. **Proceedings**, p. 100-7, 1988.