Um Multicomputador com Sistema Experimental de Comunicação

Cesar Albenes Zeferino[1]

Hermann Adolf Harry Lücke[2]

Valeria Alves da Silva[3]

Universidade Federal de Santa Catarina

Departamento de Informática e de Estatística

Pós Graduação em Ciência da Computação

C. P. 476 - Campus Trindade Cep 88040 - 900 Florianópolis - SC

Telefone: (048) 231-9738 Fax: (048) 231-9770

RESUMO

Este artigo descreve um multicomputador com um sistema de comunicação de alta flexibilidade, baseado em uma estrutura hierárquica de comunicação. Uma rede de comunicação com canais bipontuais, chaveada por um crossbar, transfere mensagens volumosas entre nós. Um barramento de serviço é utilizado para troca de mensagens pequenas, originárias de nós de trabalho, com informações sobre ligações solicitadas. Um nó centralizado estabelece estas ligações dinamicamente. O presente trabalho descreve a arquitetura do multicomputador e alguns detalhes sobre uma implementação baseada em computadores PC-AT 486 e *transputer link*, bem como algumas estimativas sobre o desempenho da máquina.

ABSTRACT

This paper describes a multicomputer with flexible comunication system, based on an hierarquical comunication structure. A comunication network with bipontual chanels, switched by a crossbar, transfers large messages between nodes. A service bus is used to perform the passing of little messages, that come from the working node with informations about the solicitated connections. A centralized node establishes these connections dinamically. The present paper describes the architecture of multicomputer and some details about an implementation based on PC-AT 486 computers and transputer link, beyond some estimates about the machine's performance.

---

[1] Engenheiro Eletricista (UFSM/RS, 1993); Mestrando do CPGCC/UFSC; Arquitetura de Computadores; Email: zeferino@inf.ufsc.br

[2] Professor UFSC/CPGCC; Dr.ING. em Metrologia e Automação (Universidade Técnica de Hannover, Alemanha, 1976); Arquitetura de Computadores, Automação Industrial; Email: cpgcc@inf.ufsc.br

[3] Bacharel em Ciências da Computação (UCP/RJ, 1993); Bolsista PROTEM - CC/SUL (abril a dezembro/94); Mestranda CPGCC/UFSC (a partir de março/1995); Automação Industrial; Email: valeria@inf.ufsc.br

1. Introdução

Um nó, unidade básica de um multicomputador, é constituído por pelo menos uma unidade processadora, uma memória local e um suporte de *hardware* para comunicação com outros nós. Desta forma, um multicomputador é constituído de um grande número de nós interligados através de um sistema de interconexão [REE87].

O sistema de interconexão pode ser realizado em forma de barramento ou através de ligações ponto-a-ponto. No primeiro modo podem ser usados barramentos paralelos do sistema, redes locais ou redes de longa distância (em todos estes casos somente uma linha de comunicação dedicada a dois processadores pode estar ativa no tempo). Já por meio de ligações ponto-a-ponto é possível obter-se um número maior de comunicações bipontuais simultâneas. Isto possibilita a formação de uma grande variedade de geometrias de comunicação, como por exemplo *pipelines*, grelhas, árvores, hipercubos, entre outras.

Circuitos de chaveamento ou comutadores de conexões, como o *crossbar,* permitem realizar uma configuração dinâmica da rede de interconexão conforme a demanda do próprio programa paralelo.

A máquina T.NODE [NUN93] é um exemplo de sistema com ligações ponto-a-ponto configuráveis dinamicamente. Esta máquina utiliza diversos processadores *transputer* IMS T800 [INM88] interligados através de chaves *crossbar* e canais de comunicação *transputer-link* [INM89].

Como os multicomputadores são sistemas de memória distribuída, onde cada processador possui sua própria memória local, a cooperação entre os processos de um algoritmo paralelo deve ser feita através de troca de mensagens. Em geral uma mensagem é dividida em pacotes, e cada pacote deve conter informações sobre o nó destino (ou sobre o roteamento, se o destino não for um nó vizinho), montagem de sequência e os próprios dados.

O sistema experimental apresentado neste artigo propõe uma estrutura hierárquica de comunicação através da implementação de duas vias para transferência de

dados. Um barramento de serviço destinado à troca de pequenas mensagens, e uma rede de comunicação com canais bipontuais utilizada para transferência de mensagens volumosas entre os nós. Um processador central, denominado nó de controle, pesquisa através do barramento de serviço as solicitações de conexão dos demais nós, estabelecendo-as na rede de comunicação através do *crossbar*.

Denominamos o sistema de comunicação "experimental" porque, com o uso do barramento de serviço e uma única ligação física entre cada nó e o *crossbar*, pode-se emular, de forma dinâmica e sequencial, as mais diversas arquiteturas estáticas; pipeline ou grelha por exemplo. A sequência das conexões necessárias pode ser estabelecida por um programa no nó de controle e/ou por demanda dos nós de trabalho.

Se a granularidade[4] de programas paralelos executados for muito fina o *overhead* por chaveamento da máquina com uma ligação nó-*crossbar* poderá tornar-se muito alto. Assim sua utilidade seria apenas experimental, a não ser que se aumentasse o número de ligações físicas entre os nós e o *crossbar*. Contudo, para programas com granularidade maior e comunicações preferenciais entre pares de nós, este aumento não se faz necessário. Desta forma, a máquina proposta, com apenas uma ligação nó-*crossbar*, pode ser considerada uma máquina de alto desempenho.

Por tratar-se esta máquina de um sistema "experimental" pretende-se realizar uma implementação econômica utilizando componentes baratos e de boa disponibilidade comercial. A primeira versão usa computadores PCs AT com processadores 486, ou maior, e sistemas de comunicação baseados em *transputer-links*.

É apresentado a seguir, no capítulo 2, a arquitetura do sistema de comunicação, cuja implementação é descrita no capítulo 3. Estimativas de desempenho são mostradas no quarto capítulo e, no capítulo 5, são apresentadas conclusões junto com algumas aplicações do multicomputador proposto.

[4] Qualquer programa paralelo consiste de um coleção de tarefas cooperantes. A granularidade do programa pode ser medida pela quantidade de computação entre interações [REE87].

2. Arquitetura do Sistema de Comunicação

2.1. Estrutura

O modelo geral do sistema proposto consta, conforme a figura 1, de um conjunto de nós de trabalho ligados através de canais de comunicação serial bipontuais, os quais associam pares de nós por meio de um comutador de conexões interposto, formando a rede de comunicação. Em um sistema com "n" nós pode-se transferir dados em grande quantidade simultaneamente e de forma bidirecional em n/2 canais.

Durante o funcionamento normal da máquina o nó com a função de controle utiliza o barramento de serviço e um canal *(link)* de configuração para definir dinamicamente a estrutura da rede de comunicação, efetuando as conexões via *crossbar*. Assim, os nós de trabalho executam o processamento paralelo, enquanto o nó de controle estabele as interligações.

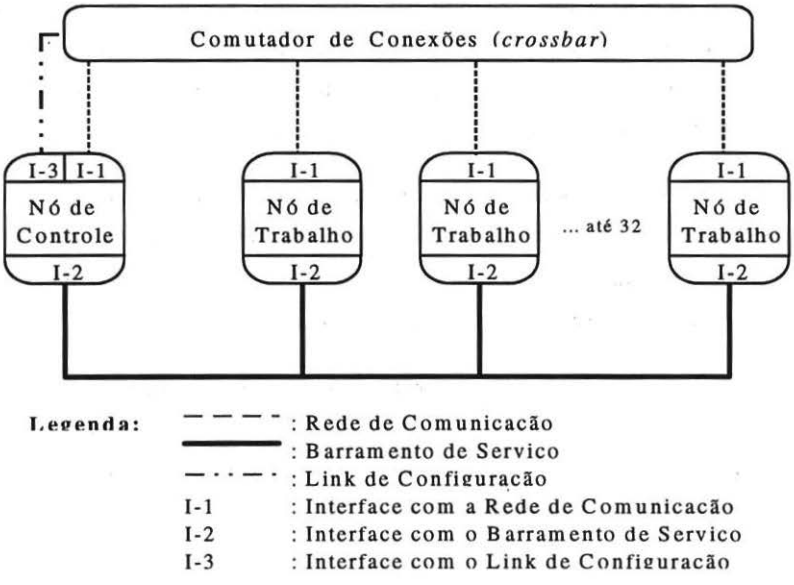


Figura 1 - A Arquitetura do Sistema de Comunicação.

2.2. Barramento de Serviço

O nó de controle, através da interface I2 (figura 1), pesquisa os pedidos de cada nó de trabalho utilizando o barramento de serviço. Ele emite um comando POLL(*d_addr*) para o nó de trabalho com o endereço destino *d_addr*, o qual pode responder com um comando CONNECT(*c_addr*) ou DISCONNECT(*c_addr*), solicitando o estabelecimento ou o cancelamento de conexão com o nó de endereço *c_addr*. O nó de controle estabelece a ligação no *crossbar* e então remete um comando ACK para o nó de trabalho, confirmando o atendimento do pedido. Caso o nó de trabalho não deseje nenhum tipo de serviço, deve responder com um comando NOTHING.

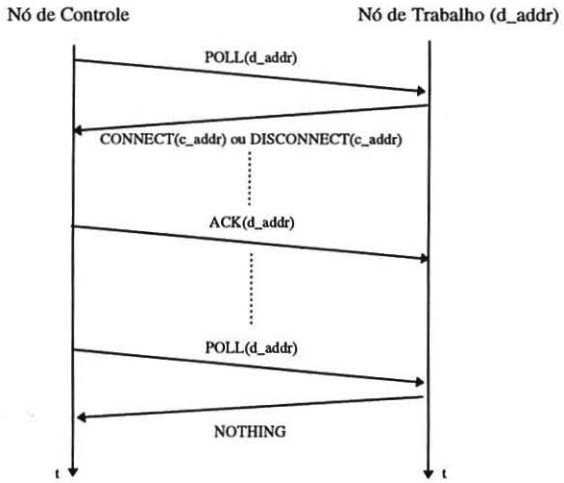


Figura 2 - Exemplo de *polling* com estabelecimento de conexão.

Os comandos apresentados são os comandos básicos do barramento de serviço, e são classificados com relação ao nó de controle em comandos de transmissão (*send*) e de recepção (*receive*). POLL e ACK são do tipo *send*, enquanto CONNECT, DISCONNECT e NOTHING são do tipo *receive*. Cada um destes comandos pode ser constituído por um *byte*, onde os cinco *bits* menos significativos formam o endereço e outros três *bits* identificam o comando. Desta forma podem ser endereçados 32 nós, implementando oito comandos de transmissão e oito de recepção.

Além dos cinco comandos básicos existem outros cinco comandos adicionais. Todos os comandos até agora implementados são listados na tabela 1.

COMANDO	TIPO	ATUAÇÃO
POLL(*d_addr*)	*Send*	O nó de controle pergunta a um determinado nó de trabalho *d_addr* se o mesmo quer algum tipo de servico.
CONNECT(*c_addr*)	*Receive*	O nó de trabalho responde ao nó de controle solicitando o estabelecimento de conexão com o nó *c_addr*.
DISCONNECT(*c_addr*)	*Receive*	O nó de trabalho responde ao nó de controle solicitando o cancelamento de conexão com o nó c_addr.
ACK(*d_addr*)	*Send*	O nó de controle confirma a um determinado nó de trabalho *d_addr* o atendimento de uma solicitação.
NOTHING	*Receive*	O nó de trabalho não deseja nenhum tipo de serviço
BROAD(*Command*)	*Send*	O nó de controle envia um comando para todos os nós. Como por exemplo, um comando de sincronização.
RESET(*d_addr*)	*Send*	O nó de controle reinicializa, por *software*, um determinado nó de trabalho *d_addr*.
INTERRUPT(*d_addr*)	*Send*	O nó de controle solicita ao nó de trabalho *d_addr* o cancelamento da conexão em uso. Este nó responde com DISCONNECT(*c_addr*).
TRANSFER(*d_addr*)	*Send*	O nó de controle deseja enviar uma mensagem, maior que um *byte*, para um determinado nó de trabalho *d_addr* através do barramento de serviço.
TRANSFER(*d_addr*)	*Receive*	O nó de trabalho deseja enviar uma mensagem, maior que um *byte*, para o nó de controle através do barramento de serviço.

Tabela 1 - Comandos do barramento de serviço.

Os comandos TRANSFER (*send* e *receive*) permitem a transmissão de mensagens maiores (até 255 *bytes*) através do barramento de serviço. O TRANSFER foi introduzido para viabilizar mensagens mais complexas, por exemplo sobre o *status* de um nó, e para possibilitar uma extensão da máquina além de 32 nós.

Uma vez detectada uma solicitação de conexão ou desconexão, é possível imaginar as mais variadas estratégias para a execução do chaveamento via *crossbar*.

Dependendo de um esquema de prioridades estabelecido, pode-se por exemplo executar um POLL com chaveamento imediato ou um POLL em todos os nós com chaveamento posterior. Caso o nó com o qual foi solicitada uma conexão já esteja ligado a um outro nó de trabalho, pode-se esperar que esta conexão seja cancelada ou então, interromper um dos nós, suspendendo ordenadamente a conexão.

2.3. Rede de Comunicação

Uma vez estabelecida a conexão, ocorre a transferência de mensagens de dados entre os nós de trabalho. Não há limitações quanto ao comprimento da mensagem. Contudo, esta deve conter um cabeçalho com informações, como por exemplo, sobre o tipo e o tamanho da mensagem.

3. Implementação

3.1. Componentes Principais

Os nós de trabalho e de controle constituem-se em microcomputador do tipo IBM-PC-AT compatível [HAA92], contendo cada qual um processador **i486** [INT90], ou maior, um barramento ISA [KLO94], uma memória privativa de 4 M*bytes* e uma placa de comunicação. Esta placa é ligada com o PC através do barramento ISA e possui módulos que implementam as interfaces com a rede de comunicação (figura 1 - I1) e com o barramento de serviço (figura 1 - I2). A placa de comunicação do nó de controle possui além destas interfaces uma interface com o *crossbar*, denominada *link* de configuração (figura 1 - I3) e uma CPU própria, que será responsável pelo gerenciamento do barramento de serviço e do *crossbar*.

A comunicação das interfaces I1 e I2 utiliza o protocolo do *transputer link* e são implementados via *link adaptor* IMS C011 [INM88] que realiza a conversão paralela-serial e vice-versa.

A opção pelo uso de máquinas PC-AT completas com slot ISA, foi devida a fácil disponibilidade do equipamento e sua boa documentação. Ao invés destes computadores completos, também podem ser usadas somente suas placas-mãe. O projeto também pode ser transferido para outros computadores mais poderosos, como por exemplo o SPARC, com a substituição do barramento ISA para o do S-BUS.

O *transputer link* permite uma implementação comprovada e confiável do sistema de comunicação, com as características pricipais de uma transmissão assíncrona, e *full-duplex*, com dois *bits* de *start*, oito de dados e um de *stop*. O recebimento de cada pacote de onze *bits*, é confirmado por dois *bits* de *acknowledgement*. Podem ser escolhidas taxas de transmissão de 10 ou 20 Mbit/s.

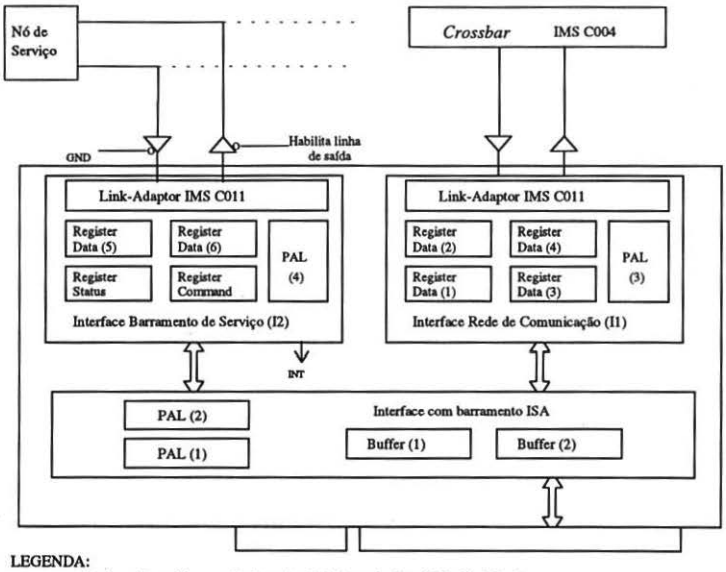
Para alcançar distâncias de transmissão em torno de dois metros, os quais são necessários para interligar dois PCs completos, por exemplo dentro de um laboratório, são usados *line drivers* de série 74F.... ou *drivers* conforme a RS422. No caso de transmissão no barramento de serviço, estes *drivers* devem possuir saídas de três estados.

O *link adaptor* C011 é utilizado no modo 1 de operação, apresentando duas interfaces paralelas de oito *bits*, uma de entrada e outra de saída. Além disso, dispõe de um protocolo *handshake* (Valid e Ack) completo, facilitando o uso deste componente com circuitos complementares.

Para atuar como comutador de conexões foi escolhido o *crossbar* C004 [INM89a], também da Inmos, que contém 32 multiplexadores.32x1.

3.2. As Interfaces

As interfaces foram projetadas e implementadas com o objetivo de aproveitar o máximo da capacidade de transferência nos *transputer links*, e gerar uma carga mínima de comunicação nos nós de trabalho. Assim, a interface com o barramento ISA foi desenvolvida com a largura de 16 *bits* (figura 3 - *buffers* 1 e 2; PALs 1 e 2).



LEGENDA:

Register Data (1)	: Registrador de Leitura do *Byte* Mais Significativo.
Register Data (2)	: Registrador de Leitura do *Byte* Meno Significativo.
Register Data (3)	: Registrador de Escrita do *Byte* Mais Significativo.
Register Data (4)	: Registrador de Escrita do *Byte* Menos Significativo.
Register Data (5)	: Registrador de Leitura.
Register Data (6)	: Registrador de Escrita.
PAL (1)	: Circuito Combinacional do Chip Select do Barramento.
PAL (2)	: Circuito Combinacional do Chip Select da Placa de Interface.
PAL (3)	: Circuito Sequencial da Interface da Rede de Comunicação.
PAL (4)	: Circuito Sequencial da Interface do Barramento de Serciço.
Buffer (1)	: Buffer de Dados Bidirecional (D8 - D15).
Buffer (2)	: Buffer de Dados Bidirecional (D0 - D7).

Figura 3 - Arquitetura da placa de interfaces do Nó de Trabalho.

Na rede de comunicação é necessário realizar a serialização das palavras de 16 *bits* para *bytes* requeridos pelo *link adaptor*. Isto é feito através dos registradores de escrita e de leitura (figura 3 - Registradores 1, 2, 3 e 4) e por um circuito implementado em um dispositivo lógico programável (figura 3 - PAL 3).

O nó de controle executa permanentemente no barramento de serviço o comando POLL, perguntando aos nós de trabalho se os mesmos necessitam algum tipo de serviço. Para otimizar este processo foram tomadas duas medidas principais. Primeiro, todos os comandos do barramento de serviço, com execeção do comando TRANSFER, têm o comprimento de um *byte*. Segundo, a resposta ao POLL é gerada por um autômato (figura 3 - PAL 4), que transfere o *byte* de comando previamente escrito pelo nó de trabalho em um registro da interface I2 (figura 3 - Registrador 6).

Dos comandos básicos, somente o comando ACK interrompe o processo ativo no nó de trabalho, informando-o do atendimento de pedidos de conexão e desconexão com outro nó de trabalho.

4. Estimativas de Desempenho

As placas das interfaces dos nós de trabalho e de controle ainda encontram-se em fase de teste. Assim, no presente momento, não é possível determinar medidas concretas de desempenho. Porém, levando-se em conta as características de chaveamento dos componentes utilizados neste sistema de comunicação, pode-se chegar a algumas estimativas quanto ao desempenho do mesmo.

Para uma frequência de 10 MHz (T_{CLK} = 0.1 μs) aplicada nos *transputer-links* e nas máquina de estado dos circuitos sequenciais, implementados na placa de interfaces, valem as relações que seguem.

O tempo T_B gasto para transferir um *byte* no *transputer-link*, segundo seu próprio protocolo, é dado pelo produto do somatório dos *bits* dos pacotes de dado e reconhecimento com o período do *clock* utilizado.

$$T_B = (start + data + stop + wait + Ack) \times T_{CLK} \quad (1)$$

$$T_B = (2 + 8 + 1 + 1 + 2) \times 0.1\ \mu s = 1.4\ \mu s$$

A máquina de estados do circuito sequencial utilizado na interface com o barramento de serviço introduz um atraso médio de dois ciclos de *clock* para gerar a resposta a um *byte* recebido. Assim, resulta:

$$T_{mBS} = 2 \times T_{CLK} \quad (2)$$

$$T_{mBS} = 2 \times 0.1\ \mu s = 0.2\ \mu s$$

Já a máquina de estados do circuito sequencial da interface com a rede de comunicação introduz um tempo de atraso T_{mRC} relativo a três ciclos de *clock*, gastos na transferência de um *byte*.

$$T_{mRC} = 3 \times T_{CLK} \quad (3)$$

$$T_{mRC} = 3 \times 0.1\ \mu s = 0.3\ \mu s$$

O atraso para a transferência de um *byte* via barramento ISA (16 *bits*) é, conforme [KLO94]:

$$T_{ISA} = 0.2\ \mu s$$

O tempo T_{RC} para transferência de um *byte* na rede de comunicação é:

$$T_{RC} = T_B + T_{mRC} + T_{ISA} \quad (4)$$

$$T_{RC} = 1.4\ \mu s + 0.3\ \mu s + 0.2\ \mu s = 1.9\ \mu s$$

Para o estabelecimento, ou cancelamento, de uma conexão é necessário enviar quatro *bytes* de configuração para o *crossbar* via *transputer-link*. Assim, o tempo $T_{CB_{CFG}}$ gasto na configuração deste componente é:

$$T_{CB_{CFG}} = 4 \times T_B \quad (5)$$

$$T_{CB_{CFG}} = 4 \times 1.4\ \mu s = 5.6\ \mu s$$

Desconsiderando os atrasos nos *line drivers* (menores que 50ns) obtém-se, para os ciclos básicos do barramento de serviço, os seguintes tempos:

- Ciclo POLL - NOTHING:

$$C_{PN} = 2 \times T_B + 2 \times T_{mBS} \quad (6)$$

$$C_{PN} = 2 \times 1.4\ \mu s + 2 \times 0.2\ \mu s = 3.2\ \mu s$$

- Ciclo POLL - CONNECT - Conexão - ACK

$$C_{PCA} = 3 \times T_B + 3 \times T_{mBS} + T_{CB_{CFG}} \quad (7)$$

$$C_{PCA} = 3 \times 1.4\ \mu s + 3 \times 0.2\ \mu s + 5.6\ \mu s = 10.4\ \mu s$$

Considerando, por exemplo, um sistema com 32 nós, onde o nó de controle (endereço 0) realiza um ciclo de *polling* endereçando sequencialmente os nós com endereço 1 a 31, pode-se calcular os tempos a seguir.

O tempo mínimo $\mathbf{Tcon_{MIN}}$ para estabelecer uma conexão solicitada pelo nó com endereço N (1 a 31) é obtido considerando que todos os nós anteriores a N não querem nenhum tipo de serviço.

$$Tcon_{MIN(N)} = (N - 1) \times C_{PN} + C_{PCA} \quad (8)$$

assim, para o primeiro(N=1) e último (N=31) nós da fila de *polling* obtém-se:

$Tcon_{MIN(N=1)} = 10.4\ \mu s$ (primeiro nó da fila de *pollling)*

$Tcon_{MIN(N=31)} = 106.4\ \mu s$ (último nó da fila de *polling)*

O tempo máximo $\mathbf{Tcon_{MAX}}$ para estabelecer uma conexão é determinada assumindo que todos os nós estão interconectados, formando 16 pares de ligações bipontuais, e que há 16 novos pedidos de conexão. Primeiramente deve-se cancelar as ligações existentes para que as novas possam ser estabelecidas .

$$Tcon_{MAX} = 16 \times T_{CB_{CFG}} + 16 \times C_{PCA} \quad (9)$$

$$Tcon_{MAX} = 16 \times 5.6\ \mu s + 16 \times 10.4\ \mu s = 256\ \mu s$$

Considerando que um nó de trabalho qualquer deseje estabelecer uma conexão com outro nó para transferir uma mensagem de tamanho $\mathbf{M_T}$ igual a 200 *bytes*, e utilizando-se os tempos acima determinados, obtém-se o tempo total gasto na comunicação.

$$Tc = Lc + Tc_{NOM} \quad (10)$$

onde **Lc** é a latência da comunicação, dada em função do tempo necessário para o estabelecimento da conexão; e $\mathbf{Tc_{NOM}}$ é o tempo de comunicação nominal, ou seja, é o tempo gasto na transferência da mensagem entre os nós conectados.

$$Lc = T_{CON} \quad (11)$$

$$Tc_{NOM} = M_T \times T_{RC} \quad (12)$$

$$Tc_{NOM} = 200 \times 1.9\ \mu s = 380\ \mu s$$

Assim, a partir das equações de 8 a 12, obtém-se o *overhead* associado à transmissão de uma mensagem de 200 *bytes*. Quando o nó de trabalho é o primeiro da

fila de *polling* (N=1) têm-se um *overhead* de 2.7%. Contudo, se o nó é o último da fila (N=31), e todos os nós anteriores não querem nenhum tipo de serviço, o *overhead* associado é de 21.9%. Finalmente, para o tempo máximo de conexão (equação 9) obtém-se um *overhead* de 40.3%.

4. Conclusão

Foi apresentado um multicomputador com um sistema de comunicação de alta flexibilidade. Ele permite a emulação dinâmica das mais variadas arquiteturas paralelas. Como um multicomputador paralelo de alto desempenho, ele parece adequado para programas com integração ou granularidade média. Desta forma ele vai ser usado no Projeto Nó// (Nó paralelo) [COR95], e nos trabalhos realizados em conjunto entre as Universidades Federais de Santa Catarina, Rio Grande do Sul e Santa Maria, no Projeto Charrua [NAV94]. Vê-se aplicações adequadas também na área de automação industrial, onde ocorrem muitas vezes processos bem definidos e com pouca comunicação. Para estas aplicações, o *hardware* empregado até o momento (CPU 486, *transputer links*) é mais do que suficiente para atender a performance exigida.

Neste momento estão fase de implementação e teste um conjunto de placas de interface. Este conjunto baseia-se em um primeiro projeto que ficou relativamente complexo, devido ao grande número de componentes usados. Em versões futuras pretende-se usar lógica programável de escala maior e/ou microprocessadores dedicados, prevendo também aumentar o número de canais na rede de comunicação, bem como no barramento de serviço. Um sistema de interrupção poderá diminuir o tempo de latência causado pelo ciclo de *polling*.

5. Bibliografia

[COR95] CORSO, T.B. **Um Sistema Operacional para um Multicomputador,** a publicar,1995.

[HAA92] HAAS, J. **The 486 - Book,** Abacus, 1992.

[INM88] INMOS, **IMS C011 - Link Adaptor**, Inmos Limited,1988

[INM89a] INMOS, **IMS C004 - Programable Link Switch**, Inmos Limited,1989

[INM89b] INMOS, **Transputer aplications notebook**, Inmos Limited,1989

[INT90] INTEL, **i486 Microprocessor Programer's Reference Manual,** Intel Corporation,1990.

[KLO94] KLOTH, A., **Bussysteme des PC**, Franzis-Verlag, Poing, Germany, 1994.

[LÜC90] LÜCKE, H.A.H. **Computadores Pessoais Para Automação de Laboratórios,** IV Congresso Nacional de Automação Industrial (CONAI), São Paulo - SP, 1990.

[NAV94] NAVAUX, P.O.A. **CHARRUA: Um Ambiente Multicomputador com Rede de Conexão Dinâmica**, Projeto submetido ao PROTEM II, 1994.

[NUM93] NUNES, R.C.; NAVAUX, P.O.A.; JANSCH - PORTO, I. **Algoritmo de Reconfiguração na Máquina T.NODE em Caso de Falhas.** Anais do V SBAC-PAD, pp. 344-357, Florianópolis, 1993.

[REE87] REED, D.A.; FUJIMOTO, R.M. **Multicomputer Networks: Message-Based Parallel Processing.** MIT Press, Massachusetts, 1987.