Relatório Técnico Interno 003/97

LaSiD

Avaliação de Desempenho do Protocolo BCGcausal

George Marconi de A. Lima
Raimundo J. de A. Macêdo

Novembro, 1997

Avaliação de Desempenho do Protocolo BCGcausal*

George Marconi de Araújo Lima+
gmlima@lasid.ufba.br

Raimundo José de A. Macêdo++
macedo@lasid.ufba.br

Universidade Federal da Bahia
Centro de Processamentos de Dados - CPD
Laboratório de Sistemas Distribuídos - LaSiD

Resumo

Comunicação em Grupo e protocolos de ordenação Causal são mecanismos fundamentais na construção de sistemas distribuídos confiáveis. Este artigo descreve o projeto e analisa o desempenho do protocolo causal para comunicação em grupo, BCGcausal, implementado numa rede de estações Unix no LaSiD/UFBA. Após uma análise da metodologia de avaliação proposta, apresentamos dados de desempenho relativos ao *throughput*, *atraso médio* e *round trip* do protocolo, contrapostos a várias configurações de grupo (número de processos por grupo) e atividade na transmissão (número de processos transmissores e freqüência de envio de mensagens). No total foram realizados 65 horas de monitoramento em ambiente real relativos a 4 tipos de experimentos. Os dados coletados e analisados indicam a boa expansibilidade do protocolo em relação às métricas utilizadas e o distingue favoravelmente em relação a protocolos similares descritos na literatura.

Abstract

Group Communication and causal order protocols are fundamental mechanisms for the construction of reliable distributed systems. This paper describes the design and analyses the performance of the causal order protocol for group communication named BCGcausal, implemented over a network of Unix workstations at LaSiD/UFBA. After discussing the performance analyse methodology proposed, we present the data of throughput, average delay, and round trip of our protocol against several group configurations (number of member processes) and process activity (number of sender processes and transmission rates). In total we monitored the system for 65 hours from the 4 distinct experiments. The collected data indicates the good scalability of our protocol and allow us to compare it favourably against similar protocols published up to date.

* Este trabalho recebeu apoio financeiro do CNPq/PROTEM.
+ Pesquisador DTI-F/CNPq, número 200811/89.
++ Pesquisador CNPq, número 200811/89.

1. Introdução

Ordenamos os eventos no mundo real fundamentados num conceito de tempo absoluto. Podemos aplicar tal paradigma em sistemas computacionais centralizados e assim, conseguimos estabelecer uma ordem absoluta para os eventos, baseada num relógio único. No entanto, em sistemas distribuídos[1] qualquer troca de informação entre diferentes participantes (processos) é feita através de troca de mensagem, cujo tempo de trânsito entre um ponto e outro é indeterminado. Isso impossibilita a concepção de um relógio único e obriga-nos a pensar em vários *relógios* distribuídos, impedindo que a visão absoluta de tempo seja aplicada. Essa característica nos leva a observar os eventos sob uma visão *relativista de tempo*, onde cada processo possui uma visão particular da computação.

Neste sentido, um importante contribuição foi apresentada por Lamport [Lam78] que introduziu o conceito de relógio lógico, com o qual é possível estabelecer uma ordem de *causa e efeito* (relação "*happened before*":→) dos eventos distribuídos. Tal conceito permite a construção de mecanismos para garantir que processos possam observar diferentes ordens da computação, preservando, contudo, a *causalidade* entre as mensagens.

Essa ordenação *parcial* dos eventos, atrelada a outras dificuldades, tais como, possibilidade de ocorrência de falhas e a imprecisão em detectá-las, dificultam a construção de aplicações distribuídas. A fim de contornar essas dificuldades e diminuir a complexidade no nível da aplicação surgiu o paradigma de comunicação em grupo. Em tal paradigma os processos são agrupados em conjuntos, chamados de grupos e toda a comunicação envolve todos os elementos do grupo. Essa abstração é fortemente recomendada por diversos autores [MR93, Mac94, EMS94, PBS89, BSS91, PRV93, CV95].

Um dos serviços providos pelos protocolos de comunicação em grupo tem a finalidade de *ordenar* os eventos percebidos pelo grupo. Todos os elementos do grupo têm que perceber uma ordem consistente dos eventos. Em [BJ87] é apresentado o conceito de *ordenação causal*, que contempla tanto a causalidade entre as mensagens quanto o conceito de grupos. Protocolos baseados em tal conceito fornecem suporte a diversos tipos de aplicações (CSCW, banco de dados replicados etc.). Como ilustração, observe a figura 1 que representa a entrega em *ordem causal* para um grupo de 3 processos. As linhas horizontais indicam o tempo físico e as linhas diagonais representam o fluxo de mensagens entre processos. Observe que m2 foi recebida antes que m1 no processo p_3. Contudo, sua entrega foi atrasada até que m1 chegasse a p_3. Se além de não violar a ordem causal, todos os processos observarem a mesma ordem dos eventos, diz-se que a ordenação de mensagens é *total* [Mac94].

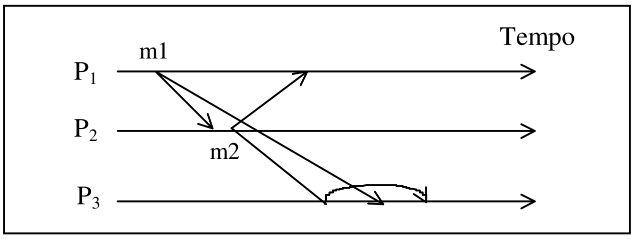


Figura 1: Entrega em ordem causal em comunicação em grupo.

Vários fatores podem caracterizar o desempenho dos protocolos de comunicação. O atraso médio de entrega das mensagens, capacidade do protocolo de processar um certo número de mensagens num intervalo de tempo, são alguns exemplos. Para avaliar os diferentes protocolos

[1]Não estamos nos referindo aos sistemas *fortemente acoplados,* onde os diversos processadores residem na mesma máquina.

existentes, além da complexidade dos algoritmos que os implementam, é necessário realizar medições destes fatores a fim de verificar quão eficiente é sua implementação. Existem algumas avaliações de protocolos, a maioria delas relativa a ordem total [BSS91, CBM94, Mac94, MS95, ACM95]. No entanto, poucos trabalhos se preocupam em medir tais fatores em ambiente real [LRN89, BSS91, Mac94] e, até onde sabemos, nenhum deles apresenta uma análise detalhada para protocolos causais, tal como a realizada neste trabalho.

Não é trivial obter medições precisas em sistemas distribuídos, pois processos concorrentes que não pertencem ao experimento acabam gerando interferências nas medições. Mais especificamente, em comunicação em grupo o problema é agravado [NN95], devido à propagação de mensagens (*replies*), necessárias aos experimentos que interferem nas medições. Em ambiente de simulação, no entanto, as diversas influências podem ser controladas, eliminadas ou até mesmo pré-determinadas, facilitando as medições. Por outro lado, acreditamos que a experimentação em ambiente real provê maior expressividade, pois lida com diversos fatores que não podem ser controlados, e estarão presentes na prática.

O protocolo analisado neste trabalho é um dos protocolos de ordem causal proposto em [Mac94, Mac95] (*protocolo causal relativo*), chamado aqui de BCGcausal por fazer parte da BCG (Base confiável de Comunicação em Grupo), desenvolvida no LaSiD/UFBA. A BCG é uma plataforma distribuída, implementada em C++ numa rede Unix, que provê serviços de comunicação em grupo, com o objetivo de fornecer um ambiente adequado ao desenvolvimento de aplicações distribuídas tolerante a falhas.

A fim de avaliar o desempenho do protocolo BCGcausal estabelecemos algumas métricas e apresentamos uma metodologia, baseada na qual definimos diversos tipos de experimentos. A execução desses experimentos envolveu aproximadamente 65 horas de experimentação em ambiente real, representando uma análise minuciosa do protocolo. Os resultados obtidos descrevem o comportamento do protocolo distinguindo-o favoravelmente em relação a protocolos similares descritos na literatura.

O texto estrutura-se da seguinte forma. A próxima seção apresenta o protocolo BCGcausal. A seção 3 descreve os experimentos e apresenta os resultados obtidos. A seção 4 contém uma comparação com trabalhos correlatos e por fim, a seção 5 conclui este trabalho.

2. O Protocolo BCGcausal

Essa seção descreve resumidamente o protocolo BCGcausal, proposto em [Mac94, Mac95]. Assumimos que os processos estão distribuídos, possivelmente em processadores distintos e não compartilham memória. Todas as mensagens chegam a seu destino num tempo finito, porém indeterminado, livres de falhas de omissão e não são corrompidas. A transmissão de mensagens entre quaisquer pares de processos é FIFO[2]. As falhas consideradas são do tipo *crash*.

Antes de apresentar o algoritmo propriamente dito, na seção 2.3, iremos definir como as relações causais são representadas na BCG, seções 2.1 e 2.2. Por fim, na seção 2.4, apresentamos a arquitetura da BCG, descrevendo alguns detalhes que são relevantes à análise dos experimentos.

2.1 Relógio Lógico

Considere o grupo de processos $g = \{p_1, p_2, \dots, p_n\}$. Cada processo p_i mantém um relógio lógico BC_i (*Block Counter*), o qual é inicializado com zero por todos os processos pertencentes a g,

[2]Essas funcionalidades estão presentes em protocolos de comunicação tal como TCP/IP.

no momento de sua criação. Ao transmitir uma mensagem m, o processo p_i carimba-a com o valor corrente de BC_i, representando assim a causalidade entre as mensagens, semelhantemente a [Lam78]. As regras usadas para incrementar incrementar BC_i são:

R1: ao transmitir uma mensagem ao grupo, evento $send_i(m)$

i. $BC_i = BC_i + 1$

ii. $m.b = BC_i$

R2: ao entregar uma mensagem m à aplicação, evento $deliver_i(m)$

i. $BC_i = \max\{BC_i, m.b\}$

Note que o avanço de BC não ocorre durante o *receive(m)* e sim durante o *deliver(m)*, não contemplando, portanto, as mensagens recebidas e ainda não entregues à aplicação. Tal abordagem aumenta a potencialidade das relações causais entre as mensagens [Mac94].

Baseado nas regras acima, pode-se deduzir as seguintes propriedades [Mac94, Mac95]:

Pr1: $send_i(m) \rightarrow send_i(m') \Rightarrow m.b < m'.b$

Pr2: $\forall$ m, $p_j \in$ m.g: $deliver_j(m) \rightarrow send_j(m') \Rightarrow m.b \rightarrow m'.b$

Pr3: $\forall$ m',m'': m'.b = m''.b $\Rightarrow$ m' e m'' são concorrentes

As propriedades Pr1 e Pr2 seguem diretamente das regras R1 e R2, respectivamente. Juntas elas implicam que para quaisquer mensagens distintas m e m': $send(m) \rightarrow send(m') \Rightarrow m.b < m'.b$. A propriedade Pr3 diz que mensagens com o mesmo número de bloco são concorrentes. Na seção seguinte descreveremos como essas mensagens são representadas na BCG e as implicações desta representação.

2.2 Blocos Causais e Matriz de Blocos

Considere uma matriz bidimensional, BM, denominada *matriz de blocos*, mantida por todo processo p_i pertencente ao grupo g, onde cada linha B, denominada bloco causal BM[B], representa as mensagens transmitidas ou recebidas por p_i, cujo número de bloco é igual a B. O número de colunas da matriz é igual a |g| (tamanho do grupo g).

Ao receber ou transmitir uma mensagem m, cada processo p_i de g marca na sua matriz BM_i [m.b][m.s] = '+', onde m.s é o identificador do processo que enviou m, a fim de representar o recebimento ou a transmissão da mensagem. A figura 2 abaixo ilustra uma BM para um grupo de 6 processos. Suponha que o processo que mantém essa matriz é p_1, então a matriz indica que a última mensagem transmitida possui número de bloco 5 e as últimas mensagens recebidas dos processos p_2, p_3, p_4, p_5 e p_6 possuem, respectivamente, número de blocos iguais a 6, 3, 4, 5 e 2.

Assumimos que, se um processo não tenha nada a transmitir o mecanismo de *time silence* (seção 2.4) se encarrega de transmitir mensagens nulas. Essa característica é fundamental para se detectar falhas e entregar de mensagens em ordem total, serviços também implementados na BCG [Mac94, EMS95].

	p_1	p_2	p_3	p_4	p_5	p_6
1	+			+		
2		+			+	+
3			+			
4	+			+	+	
5	+				+	
6		+				

Figura 2: Ilustração de uma matriz de blocos para um grupo de 6 processos.

Desta representação e das regras apresentadas na seção anterior, deduz-se as seguintes propriedades:

Pr4: Num dado bloco causal, somente mensagens concorrentes são representadas.
Pr5: Sejam m e m', representadas em BM[B] e BM[B'], respectivamente, e $m \rightarrow m'$ então $B < B'$.

Blocos Completos

Um bloco causal, BM[B] ($B \geq 1$), é dito completo se e somente se sabe-se que nenhum processo do grupo transmitirá/receberá mensagens com números de blocos menores ou iguais a B. Mais precisamente: seja o grupo $g = \{p_1, p_2, \ldots, p_n\}$, $\forall\ j\ 1 \leq j \leq |\ g\ |$, um bloco BM[B], $B \geq 1$ está completo se e somente se:

- existe um '+' em BM[B][j], ou
- existe um '+' em BM[B'][j], B' > B

Na figura 2 os blocos 1 e 2 estão completos.

O protocolo de ordem total, implementado na BCG, se baseia em blocos completos para definir a ordem de entrega de mensagens. As mensagens com número de bloco m.B são entregues apenas quando existir um bloco de número B' completo e $B' \geq B$. Mais detalhes ver [Mac94, EMS95].

Pela definição de bloco completo sabe-se que as mensagens com número de bloco B+1 (dado que B está completo) podem ser entregues à aplicação sem violar a ordem causal. No entanto, esperar que um bloco esteja completo introduz uma contenção à entrega causal de mensagens, o que pode ser evitado se mais informações forem anexadas às mensagens transmitidas. A próxima seção define o protocolo de ordem causal baseado num vetor de número de blocos que é anexado a toda mensagem transmitida.

2.3 O Algoritmo de Ordenação Causal

Considere que qualquer mensagem transmitida m, além de carregar seu número de bloco m.b, leva consigo os números de blocos das últimas mensagens entregues à aplicação (*delivered*). Quando m é recebida por um processo, ele pode entregar m imediatamente após entregar todas as mensagens que precedem causalmente m. Para tanto, cada processo $p_i \in g = \{ p_1, p_2, \ldots, p_n \}$ mantém um vetor de números de blocos, chamado de LDV (*Last Delivered Vector*), cujo tamanho é $|g|$. O conteúdo de LDV, juntamente com o número de bloco da mensagem constituirão os *timestamps* de todas as mensagens, m, transmitidas (m.ldv e m.b, respectivamente). Assuma que m.s representa o identificador do processo que enviou m. Assim, o LDV de p_i (LDV_i) satisfaz a seguinte propriedade:

Pr6: $LDV_i[j] = \max\{m.b \mid m.s = p_j$ e *deliver(m)* ocorreu $\}$

Com relação a LDV_i, p_i executará o seguinte:

R3: Antes de transmitir - *$send_i(m)$*:
$m.ldv = LDV_i$
R4: Depois de transmitir:
$LDV_i[i] = m.b$
R5: Depois de entregar - *$deliver_i(m)$*:
$LDV[m.s] = m.b$

Considere os vetores LDV mantidos pelos processos p_i e p_j, LDV_i e LDV_j, respectivamente. Define-se a seguinte relação:

Def3: $LDV_i \leq LDV_j \Leftrightarrow \forall k: LDV_i[k] \leq LDV_j[k]$

Uma mensagem recebida por p_i, é entregue por p_i imediatamente após a seguinte condição tornar-se verdadeira:

Condição de Entrega (CE):
$m.ldv \leq LDV_i$

Pela condição acima definida e pelas regras R3 e R4, qualquer mensagem transmitida por p_i pode ser imediatamente entregue, ou seja, *$send_i(m)$* é imediatamente seguido por *$deliver_i(m)$*.

O vetor LDV tem propriedades similares à *vector clocks* [BSS91]. A diferença é que o segundo mecanismo representa o número de eventos que ocorreram no grupo, enquanto que o primeiro representa as últimas mensagens entregues ao grupo, o que constitui uma representação de causalidade que permite manipular grupos sobrepostos mais facilmente. As propriedades e vantagens podem ser vistas em [Mac94, Mac95].

2.4 Implementação

A BCG (Base Confiável de Comunicação em Grupo), implementada em C++ numa rede UNIX, é um conjunto de processos que fornecem serviços de comunicação em grupo, permitindo a troca de mensagens entre diversas aplicações distribuídas, obedecendo o paradigma de comunicação em grupo. Atualmente estão implementados um protocolo de ordem total e um protocolo de ordem causal, o qual é objeto de avaliação deste trabalho, serviços de *membership* e controle de fluxo. A figura 3 a seguir ilustra os processos que compõem o núcleo da BCG e suas interações. A comunicação interna e a interface entre o núcleo da BCG e a camada de transporte utilizam filas de mensagens. A camada de transporte implementa a comunicação externa utilizando *sockets* TCP/IP.

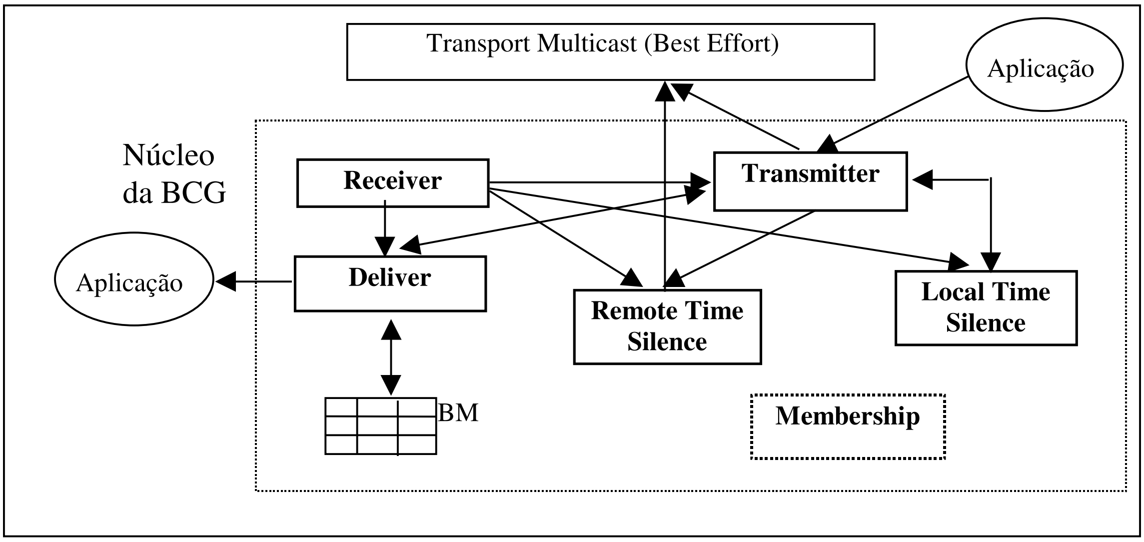


Figura 3: Processos que implementam a BCG.

Cada uma das linhas direcionadas que ligam os processos representam o fluxo de mensagens entre os mesmos, através das filas de mensagens. Descreveremos a seguir, brevemente, as funcionalidades providas pela BCG. Direcionaremos nossa descrição ao protocolo BCGcausal, contudo vale ressaltar que muitos serviços são ortogonais à ordenação de mensagens (camada de transporte, controle de fluxo, mecanismo de *time silence* etc.) e portanto, são idênticos em ambos os protocolos.

Camada de Transporte *Multicast*

A camada de transporte realiza o mapeamento entre a comunicação ponto-a-ponto e a comunicação em grupo, implementando primitivas de comunicação em grupo *send/receive*. A comunicação externa é feita através de *sockets*, usando o protocolo TCP/IP. Desta forma, garante-se que a comunicação entre quaisquer pares de processos obedece a ordem FIFO. A interface entre a camada de transporte e o núcleo da BCG é feita através de filas de mensagens.

Ao se criar um grupo, G, a camada de transporte cria várias conexões TCP/IP, uma para cada par de processos do grupo. Estamos, atualmente, projetando uma nova versão da camada de transporte a fim de torna-la mais eficiente e mais flexível, evitando o excesso de conexões TCP/IP para grupos grandes.

Mecanismo de *Timesilence*

Se não há uma comunicação ativa entre os processos do grupo, é possível que as condições para entrega de mensagens para o protocolo de ordem total (blocos completos) e para detectar falhas não sejam satisfeitas. A função do *Local Time Silence* é gerar mensagens nulas que são transmitidas para o grupo, num período preestabelecido, a fim de completar blocos de mensagens que estejam pendentes e permitir o funcionamento dos serviços de *membership* e o controle de fluxo, indispensáveis numa comunicação em grupo tolerante a falhas.

O processo *Clock Ticks* tem a função implementar um relógio local, enviando[3] mensagens de tempo para os processos que necessitam estabelecer *timeouts*, *Local Time Silence* e *Remote Time Silence*.

Mecanismo de *Membership*

Esse mecanismo, proposto em [Mac94, EMS95], tem a função de estabelecer dinamicamente visões consistentes de grupos, mesmo na presença de falhas. Utiliza dois processos, *Remote Time Silence*, encarregado de suspeitar de processos falhos e *Membership*, responsável por reconfigurar o grupo dinamicamente. Não descreveremos em detalhes a interação, pois os experimentos foram realizados com tal serviço *desligado*.

Mecanismo de Controle de Fluxo

A fim de evitar estouro nos *buffers* de algum processo do grupo, o *Transmitter* apenas transmite mensagens se certas condições de estabilidade[4] de mensagens forem satisfeitas [Mac94, MES95]. Tais condições de estabilidade estão baseadas em blocos completos. Apesar disso, o mecanismo de controle de fluxo é ortogonal à ordem de entrega de mensagens, sendo portanto, utilizado também pelo protocolo de ordem causal.

Matriz de Blocos - BM

As mensagens são armazenadas na BM, pelo *Deliver,* até que sejam consideradas estáveis. A BM está organizada em dois níveis. O primeiro nível é uma tabela *hashing*, endereçada por números de blocos, cuja função *hashing* é h(m.b) = m.b mod N. O tamanho da tabela, N, é conhecido por todos os participantes do grupo. O serviço de controle de fluxo garante que nunca haverá colisão na tabela *hashing*. Cada entrada da tabela contém o número do seu bloco causal e um apontador para uma entrada no segundo nível, chamado de *pool de mensagens*, que estrutura-se como uma lista ligada de espaços livres. Ao se inserir uma mensagem na BM, aloca-se uma entrada na tabela através da função *hashing* e insere-se a mensagem no *pool de mensagens,* removendo um espaço da lista de espaços livres. Quando uma mensagem é considerada estável, o espaço a ela relacionado é novamente inserido na lista de espaços livres. Esse procedimento possibilita uma rápida localização de espaços ao inserir uma mensagem. A figura 3 abaixo ilustra a BM:

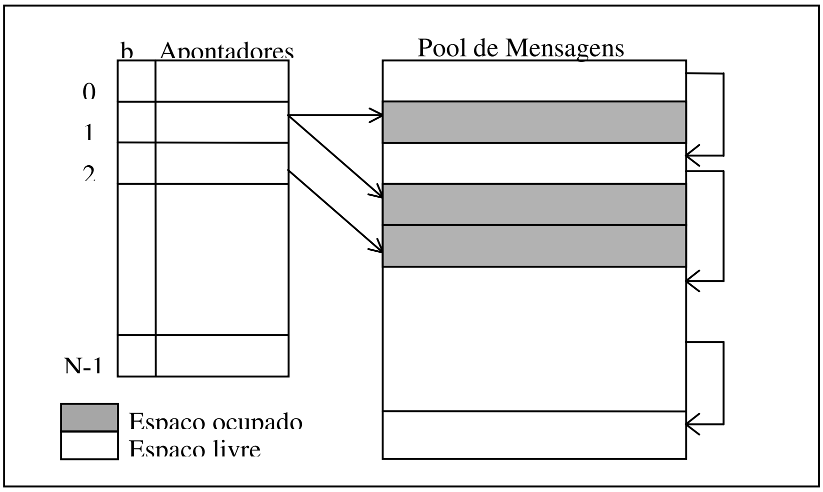


Figura 3: Estrutura em dois níveis da BM.

[3]Usaremos os termos *enviar* e *transmitir* para nos referirmos à comunicação interna via memória (núcleo da BCG - através das filas de mensagens) e externas (através da camada de transporte), respectivamente.
[4]Uma mensagem é dita estável por um processo de um grupo se ele sabe que já foram entregues por todos os membros do grupo.

Ordenação de Mensagens

Os três processos descritos a seguir implementam a ordenação de mensagens na BCG:

- ***Transmitter***: recebe as mensagens da aplicação e as envia para a *camada de transporte.* Atualiza o relógio lógico e, no caso do protocolo de ordem causal, o LDV.

- ***Receiver***: recebe da *camada de transporte* as mensagens da aplicação e passa para o *Deliver*. O cabeçalho das mensagens, no entanto, é propagado a todos os outros processos, a fim de atualizar a visão da última mensagem recebida nesses processos e assim, garantir o funcionamento dos mecanismos descritos anteriormente (*membership*, controle de fluxo e *timesilence*).

- ***Deliver***: entrega as mensagens recebidas pelo *Receiver* à aplicação. Entretanto, as mensagens somente poderão ser entregues quando condição de entrega CE for verdadeira (seção 2.3). As mensagens recebidas são armazenadas localmente na BM.

É necessário que haja uma comunicação bilateral entre o *Deliver* e o *Transmitter,* pois ambos os processos atualizam o LDV. Isso causa um *deadlock* potencial. Para resolver esse problema, sempre que há transmissão de uma mensagem não nula para a camada de transporte, é realizado o seguinte procedimento:

i) o *Transmitter* envia uma mensagem tipo *RequestLDV* para o *Deliver*, a fim de solicitar o atual valor do LDV mantido pelo *Deliver;*
ii) o *Deliver*, recebendo a solicitação do *Transmitter* envia-lhe o LDV atual;
iii) após o *Transmitter* atualizar o LDV, envia a mensagem para a camada de transporte e envia para o *Deliver* o novo valor do LDV (mensagem tipo *UpdateLDV*) e do relógio lógico, BC;
iv) o *Deliver,* recebendo a mensagem do *Transmitter,* atualiza o LDV e insere a mensagem na *BM*, iniciando então o procedimento de entrega.

Observe que essa abordagem evita de o *Deliver* propagar as atualizações do LDV para o *Transmitter* sempre que uma mensagem é entregue à aplicação.

3. Análise de Desempenho

Pode-se avaliar a eficiência de um protocolo de comunicação analítica ou empiricamente. O objetivo da abordagem analítica é avaliar a eficiência, abstraindo-se dos detalhes tecnológicos e de implementação, através do estudo da *complexidade dos seus algoritmos*. Essa análise, em geral, não leva em consideração fatores probabilísticos, tais como, tráfego de mensagens, necessidade de retransmissão de mensagens em caso de falhas de omissão etc.

A abordagem empírica se baseia em colher amostras de dados que representem a execução do protocolo, analisado-o em diversas situações, a fim de traçar o seu perfil de comportamento. Para tanto, é possível utilizar dois métodos: *simulação*, ou realizar a *experimentação em ambiente real.* Existem desvantagens e vantagens em cada um desses métodos. No mundo real existem fatores difíceis de serem determinados, muitas vezes até desconhecidos, o que torna o modelo de simulação uma simplificação do modelo real, facilitando a avaliação do protocolo. No entanto, o preço pago por isso é que as respostas encontradas ao utilizar simulação geralmente são menos precisas. Outro argumento em prol da experimentação em ambiente real é a dificuldade de validação do modelo de simulação, pois sua relação com o problema real é indireta, sendo o modelo seu elo de ligação.

Por considerar os resultados obtidos mais expressivos, utilizamos experimentação em ambiente real para avaliar o protocolo BCGcausal. Medimos vários fatores que o caracterizam em diferentes situações, traçando assim um perfil do comportamento do sistema. Para tanto, apresentamos uma metodologia e estabelecemos algumas métricas.

Esta seção está estruturada da seguinte forma. Primeiramente, na seção 3.1, apresentamos os recursos utilizados para a realização dos experimentos. Em seguida, a seção 3.2 define os experimentos e analisa os resultados obtidos. Finalmente, a seção 3.3 compara os resultados obtidos com trabalhos correlatos.

3.1 Ambiente Experimental

Utilizamos cinco máquinas IBM/RISC 6000 ligadas numa rede local utilizando par trançado (10 MBits/segundo). O sistema operacional em todas as máquinas é AIX 4.1.4 executando ambiente Xwindows. Todas as cinco máquinas exportam/importam arquivos diversos através de NFS.

A comunicação entre pares de processos é realizada através da camada de transporte descrita anteriormente. Essa camada é independente do protocolo de ordem causal, no entanto influencia nos resultados obtidos nos experimentos. Assim, para permitir uma comparação mais precisa iremos apresentar dados relativos ao protocolo BCGcausal usando a camada de transporte e dados obtidos apenas executando a camada de transporte.

3.2 Metodologia de Avaliação

Para avaliar o desempenho de protocolos de comunicação devemos definir métricas e projetar experimentos para coletar dados que representem o comportamento do protocolo em diversas condições de operação. Tais experimentos devem refletir as características do paradigma de comunicação em questão a fim de não obter uma visão distorcida dos dados.

Em comunicação em grupo os processos são agrupados em conjuntos e toda a transmissão pressupõe a difusão das mensagens nos grupos. A fim de preservar a ordenação (ver figura 1) as mensagens devem ficar retidas localmente nos *buffers* para que seja possível ordena-las. Três eventos distintos estão presentes nesse paradigma: *send*, responsável por difundir uma mensagem para o grupo; *receive*, através do qual um processo recebe uma mensagem do grupo e; *delivery*, responsável por entregar uma mensagem recebida quando as condições de entrega forem verificadas (causal ou total, por exemplo).

Observe que em comunicação ponto-a-ponto, onde os processos se comunicam aos pares, o evento *delivery* e *receive* se confundem num único evento, pois toda mensagem recebida pode ser automaticamente entregue. O atraso que uma mensagem leva para ser entregue, portanto, é devido apenas ao tempo de transferência da mensagem do meio físico para o *buffer* da aplicação e à latência do canal de comunicação. Em contrapartida, o tempo de entrega das mensagens em comunicação em grupo depende também da relação (causal ou total) entre as mensagens e de quão eficiente é a representação desta relação. Portanto, usaremos o *atraso médio*[5] como uma das métricas de avaliação. Esta medida representa o tempo médio que uma mensagem, uma vez recebida, leva para ser entregue pelo protocolo à aplicação. No nosso estudo de caso, montamos experimentos para medir o atraso médio na ausência e na presença de dependência causal.

[5]Alguns autores usam o termo *average delivery time,* outros *average delay overhead.*

Geralmente, quando avaliamos um protocolo de comunicação ponto-a-ponto, queremos saber a latência da comunicação, desde a transmissão da mensagem, até sua recepção e, a capacidade máxima de processamento do protocolo. Para tanto, existem duas técnicas que são comumente utilizadas:

- *ping* - um processo transmite rajadas de mensagens para o processo receptor e é medido o número de mensagens processadas por intervalo de tempo. Tal medida é conhecida como *throughput* e representa a capacidade máxima de processamento de mensagens por unidade de tempo;
- *ping-pong* - um processo transmite uma mensagem e espera a resposta do processo receptor. Mede-se o tempo entre a transmissão da mensagem e o recebimento da resposta. Esse tempo é conhecido como *round trip*.

Deve-se ter cuidado ao usar tais técnicas para avaliar o desempenho de protocolos de comunicação em grupo, devido às interferências provocadas pelos *replies* dos membros [NN95, NN97]. As métricas, portanto, passam a ter outros significados. Por exemplo, o *round trip* representará o tempo que uma mensagem transmitida chega a todos os membros do grupo e o transmissor recebe os respectivos *replies*. Observe que todos os membros do grupo também receberão os *replies* (comunicação *muitos-para-muitos*). Esperar os diversos *replies* acaba interferindo nas medições. Pode-se argumentar, contudo, que tal medida ainda é significativa, pois ela caracteriza, por exemplo, o desempenho para implementar serviços de comunicação em grupo, tais como RPC, onde os *replies* são necessários.

Além das medidas de *round trip* e *throughput* adaptadas para comunicação em grupo, medimos o *atraso médio*[6] do protocolo que representa o tempo médio que uma mensagem leva para ser entregue pelo protocolo à aplicação. Esse tempo depende basicamente de duas variáveis: a eficiência da implementação e a dependência causal entre as mensagens. Esta última é influenciada por uma série de outros fatores, tais como latência do canal de transmissão, velocidade de transmissão etc. Montamos experimentos para medir o atraso médio na ausência e na presença de dependência causal.

Um outro aspecto importante é conhecer o comportamento do protocolo para diferentes configurações e diferentes níveis de atividade na transmissão de mensagens. Com esse objetivo, determinamos como o protocolo se comporta para diversos:

- **tamanhos do grupo**, avaliando, portanto a *expansibilidade*[7] do protocolo em relação ao atraso médio, *throughput* e *round trip.*
- **períodos de transmissão**, verificando o *atraso médio* e *throughput*. Denominamos *período de transmissão* o intervalo de tempo entre a transmissão de duas mensagens consecutivas.

Outros experimentos também avaliam a influência que a velocidade do canal exerce na dependência causal entre as mensagens e no atraso médio do protocolo. Para tanto, montamos um experimento que controla a latência do canal de transmissão.

As medições de tempos estão baseadas no acesso ao relógio físico local da máquina. No entanto, a chamada de sistema responsável pelo acesso ao relógio é uma operação custosa, o que causa interferências nas medidas. Avaliamos experimentalmente e estimamos que a cada dois acessos consecutivos ao relógio gasta-se em média 0.5 ms. Pode-se diluir tal influência se executarmos várias vezes o experimento, mantendo o número de acessos ao relógio fixo (apenas a

[6]Alguns autores usam o termo *average delivery time,* outros *average delay overhead*.
[7]Do inglês *escalability.*

operação a ser medida é repetida). No entanto, nem sempre é possível fazê-lo, por exemplo para avaliar o atraso médio (seção 3.2.2). Portanto, em tais situações as interferências devido ao acesso ao relógio estão presentes nas medições.

O serviço de controle de fluxo, mesmo sendo ortogonal à ordenação de mensagens, exerce influência em alguns dos experimentos realizados. Para eliminar seus efeitos aumentamos o tamanho da BM, impedindo que haja contenção na transmissão de mensagens. Com relação ao mecanismo de *timesilence*, como ele é indispensável para a detecção de falhas, mesmo essas não sendo consideradas por nós no experimento, resolvemos deixá-lo ativo, gerando uma mensagem nula a cada 100 ms. Note que uma mensagem nula somente será gerada se o referido processo não transmitir nenhuma mensagem de usuário nesse período (seção 2.4). Por essa razão, nem todos os experimentos sofrerão a influência do *timesilence*, como veremos nas seções posteriores.

Os resultados obtidos para cada tipo de experimento correspondem à média de várias execuções. A mensagem transmitida possui tamanho de 116 bytes em todos os experimentos, os quais foram executados em momentos de pouca utilização da rede. A seção seguinte descreve-os separadamente.

3.3 Resultados de Desempenho

Explicaremos nesta seção como cada experimento foi montado e apresentaremos os resultados obtidos.

3.3.1 Avaliando o *Throughput*

O algoritmo responsável por esse experimento, escrito em pseudocódigo tipo C++, é ilustrado na figura 5 abaixo. O *throughput*, representado pela variável *tp,* é o número de mensagens transmitidas dividido pelo intervalo de tempo compreendido desde o início da transmissão até o recebimento do último *reply.*

Ao invés de transmitir mensagens sem pausa entre duas transmissões consecutivas, como em [BSS91], preferimos variar o período de transmissão, como ilustrado no algoritmo. Desta forma obtemos o comportamento do *throughput* em função do período de transmissão, fornecendo maior expressividade às medidas. Para que o tempo de pausa entre transmissões não seja computado, incluímos um segundo termo na expressão para o cálculo de *tp* (*tpausa).*

Processo 0:

```
        BCGcausal  G(G_NUMBER)
/* Para cada período de trans. faça */
        start = time(); i = 1;
        while (i ≤ Num_mesg) {
            G.send(msg);
            pause(período);
                        i++;
        }
        for(j = 1; j ≤ |G| - 1;j++)
            G.receive(reply);
        tpausa = period * (Num_mes-1);
        tp = Num_mes / (time() - start – tpausa);
```

Demais Processos do grupo:

```
        BCGcausal  G(G_NUMBER)
/*Para cada período de trans. faça */
        i = 1;
        while (i ≤ Num_mesg) {
                G.receive(msg);
                 i++;
        }
        G.send(reply);
```

Figura 5: Ilustração do algoritmo para avaliar o *throughput*.

Os mesmos algoritmos foram utilizados para a camada de transporte, apenas foram substituídas as chamadas *send/receive* da BCG pelas equivalentes chamadas da camada de transporte.

A figura 6 abaixo ilustra o comportamento do *throughput* em função do período de transmissão. Configurou-se grupos com tamanho variando de 2 a 5 (os resultados individuais se encontram no apêndice A), assim como variou-se o período de transmissão, de 1600 a 25 ms. Em cada grupo mediu-se o *throughput* referente à transmissão de 1.000 mensagens para cada um dos períodos de transmissão. O experimento foi repetido 4 vezes. Os pontos do gráfico correspondem à média do *throughput* para os diversos tamanhos de grupo em função do período de transmissão. Os experimentos que avaliam a camada de transporte e o protocolo causal foram submetidos em paralelo a fim de que seus resultados representem o mesmo cenário da rede[8].

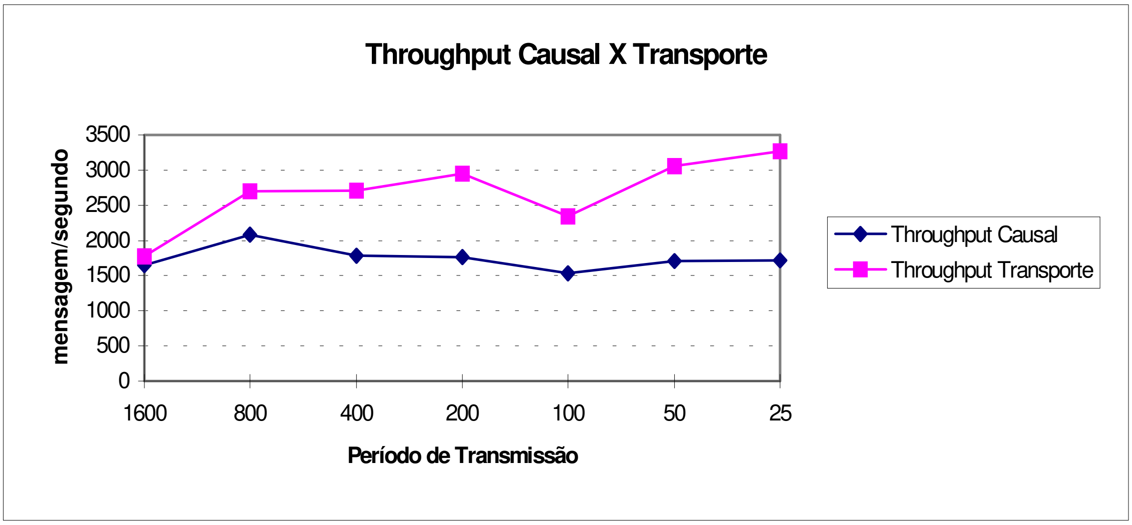


Figura 6: Gráfico do *throughput* em função do período de transmissão.

Observe no gráfico da figura 6 que para pequenos períodos de transmissão há uma tendência do aumento da diferença entre as respostas do protocolo causal e da camada de transporte. Por exemplo, para o período de transmissão de 25 ms, a camada de transporte apresentou um aumento do *throughput*, mas o protocolo BCGcausal não acompanhou essa tendência na mesma proporção. Esse comportamento é devido ao aumento do fluxo de mensagens nas filas internas, causando maior contenção aos procedimentos de entrega de mensagens e representação destas na BM (processo *Deliver* - seção 2.4). Ao analisarmos, na próxima seção, o comportamento do atraso médio em função do período de transmissão essa explicação ficará mais clara.

A figura 7, que ilustra a variação do *throughput* em função do tamanho do grupo[9], corresponde à média de todos os períodos de transmissão para cada tamanho de grupo. Observe que a curva de expansibilidade do protocolo BCGcausal é aproximadamente constante, o mesmo não ocorrendo para a camada de transporte. Em outras palavras, a diferença entre o *throughput* da camada de transporte e do protocolo BCGcausal cresce com o aumento do tamanho do grupo. Isso também se deve ao aumento do tráfego das mensagens internas, mas aqui esse tráfego corresponde à transmissão de mensagens nulas pelo mecanismo de *timesilence* (seção 2.4).

[8] Acreditamos que é preferível admitir a influência mútua promovida pela execução paralela do que obter respostas que não correspondam ao mesmo cenário do sistema.

[9] Não conseguimos aumentar o tamanho do grupo nem diminuir o período de transmissão, pois há um *bug* no sistema no nível do TCP/IP. Quando há uma excessiva interação entre os processos, mensagens são corrompidas.

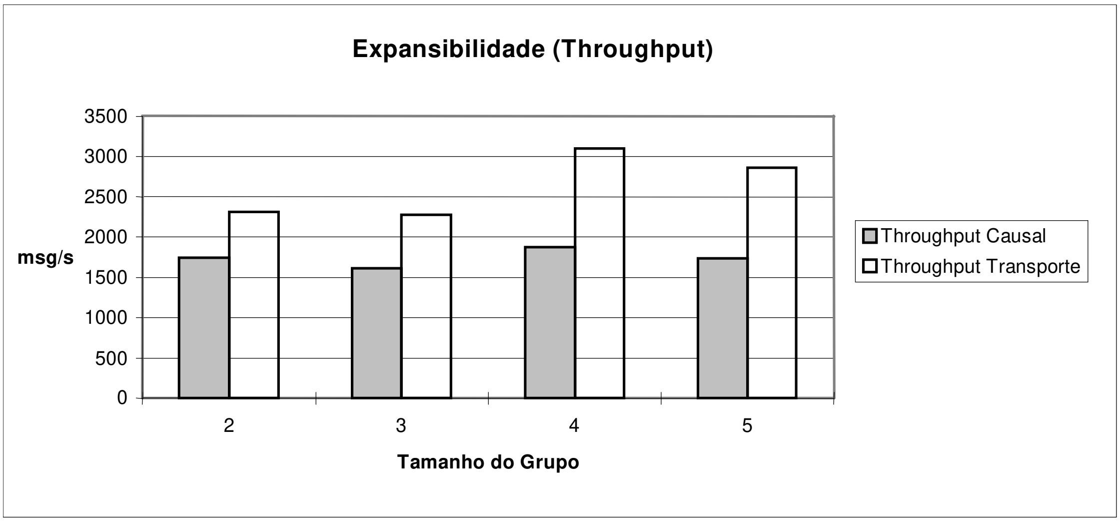


Figura 7: Gráfico do *throughput* em função do tamanho do grupo.

3.3.2 Avaliando o Atraso Médio do Protocolo

O atraso médio, como dito anteriormente, representa o tempo médio que uma mensagem fica retida pelo protocolo até ser entregue à aplicação. No caso do protocolo BCGcausal, se uma mensagem m é recebida, ela é inserida na BM e só é entregue à aplicação quando todas as mensagens m', tal que m' $\rightarrow$ m, forem recebidas (dependência causal). Note que o tempo de trânsito de m' influencia diretamente no atraso médio.

Avaliamos o comportamento do protocolo em função do atraso médio em termos da atividade dos membros do grupo em relação à transmissão de mensagens. Num extremo, apenas um processo transmite mensagens (configuração 1) e portanto, não há dependência causal entre as mesmas, eliminando assim a influência do tráfego na rede. Noutro extremo, todos os processos do grupo são transmissores (configuração 2) e neste caso, tanto a dependência causal entre as mensagens quanto a latência da comunicação passam a influenciar no atraso médio.

Em ambas as configurações cada processo transmissor do grupo transmite 7 blocos de 1.000 mensagens. Cada bloco é transmitido sob um período de transmissão que varia de 1.600 ms à 25 ms por bloco. Variou-se também o tamanho do grupo: de 2 a 6 processos. Cada processo foi executado numa máquina distinta, com exceção do grupo de tamanho 6, onde repetiu-se uma máquina.

O procedimento de medição do atraso médio é da seguinte forma: uma mensagem, ao ser recebida da camada de transporte, é carimbada com o valor corrente do tempo local. Ao entrega-la à aplicação registra-se o tempo local decorrido entre o recebimento e a entrega (observe que isso é feito no núcleo da BCG, processos *Receiver* e *Deliver*, respectivamente)[10]. A média dessas medidas para 1.000 mensagens é então calculada. Todos os experimentos foram executados 4 vezes e os dados apresentados a seguir correspondem a média dessas 4 execuções.

Note que neste experimento é impossível retirar o efeito do atraso devido ao acesso ao relógio. Seja t_i e t_f, os tempos inicial e final, correspondentes ao recebimento e entrega de uma mensagem, respectivamente e suponha que o tempo de interferência da chamada de sistema de acesso ao relógio seja t_Δ. Então, no cálculo final, o tempo que a mensagem leva para ser entregue à aplicação é $T = t_f + t_\Delta - (t_i + t_\Delta)$, significando que t_Δ não é contabilizado. No entanto, vale ressaltar que os procedimentos de recebimento e entrega de mensagens (processos *Receiver* e *Deliver*,

[10] Note que nesse experimento é impossível retirar o efeito do atraso devido ao acesso ao relógio. Os tempos registrados, portanto, estão acrescidos desse valor, estimado experimentalmente em 0.5 ms.

respectivamente), ficam com seus tempos de processamento acrescidos de $2t_\Delta$ (t_Δ em cada um deles), interferindo assim no desempenho como um todo. Estimamos experimentalmente o valor de t_Δ em 0.25 ms.

Configuração 1 - Apenas um Processo Ativo

A figura 8 ilustra as respostas médias dos processos receptores, em função do período de transmissão. Note que o atraso médio não ultrapassa 2 ms. No gráfico está representada a média das respostas de um dos processos receptores, aquele que se repete para os diferentes tamanhos de grupo.

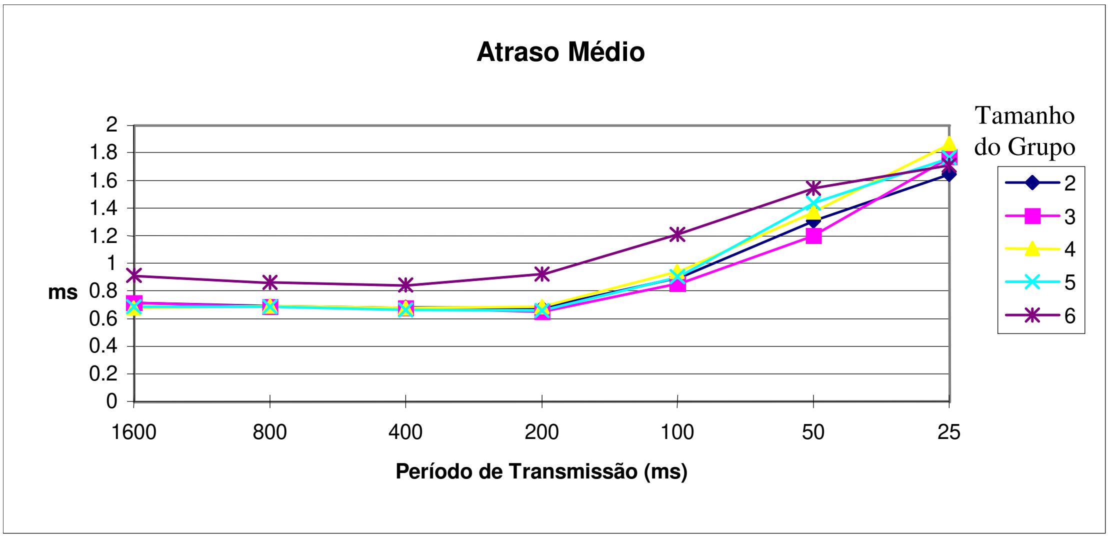


Figura 8: Atraso médio de entrega das mensagens dos processos receptores em função do período de transmissão.

Analisando o comportamento ilustrado na figura 8, verifica-se que o desempenho na entrega das mensagens decresce um pouco com o aumento da freqüência de transmissão, causado pelo tempo gasto pelos procedimentos de representar as mensagens na BM e entregá-las à aplicação. Esse aumento no atraso médio é responsável por provocar, como vimos na seção anterior, um pequeno decréscimo do *throughput* para taxas de transmissão mais altas. Pode-se observar ainda que os diferentes tamanhos de grupo apresentam resposta muito parecidas o que comprova a boa expansibilidade do protocolo. Esse fato será melhor ilustrado no gráfico da figura 9.

Agrupamos os atrasos médios para cada tamanho de grupo. Para tanto, novamente escolhemos um dos processos receptores (aquele que se repete em todas as configurações) e retiramos a média das respostas para todos os períodos de transmissão. A figura 9 representa, portanto, o comportamento do atraso médio em função do tamanho do grupo. Vale ressaltar aqui que, diferentemente do comportamento do *throughput* em função do tamanho do grupo, o mecanismo de *timesilence* não exerce influência sobre atraso médio, pois as mensagens nulas não são inseridas na BM nem entregues à aplicação.

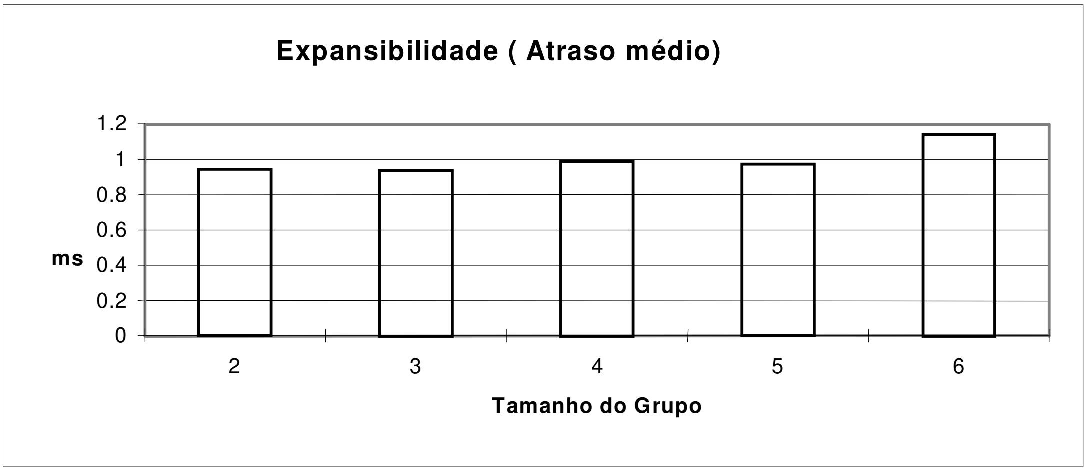


Figura 9: Atraso médio dos processos receptores em função do tamanho do grupo.

Configuração 2 - Todos os Processos do Grupo são Ativos

Ilustraremos aqui como o protocolo se comporta com máxima atividade do grupo, ou seja, quando todos os processos transmitem mensagens (figura 10). Note que quanto menor é o período de transmissão e maior é o tamanho do grupo, maior será o atraso médio. Isso se deve, além do que já foi mencionado no experimento de configuração 1, ao aumento do tráfego na rede e aumento da dependência causal entre as mensagens. A curva de expansibilidade ficará melhor ilustrada pela figura 11.

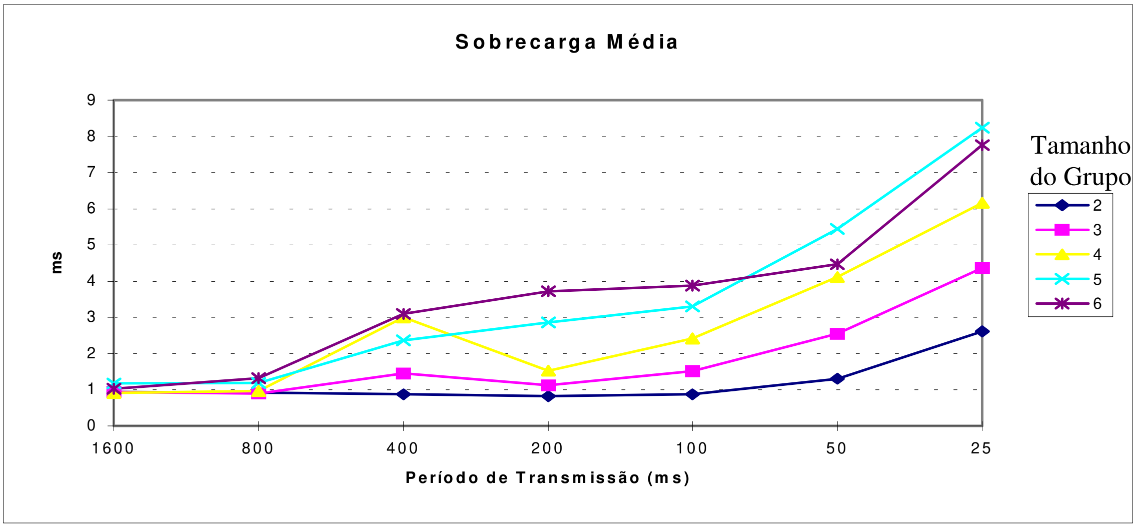


Figura 10: Atraso médio na entrega de mensagens num grupo totalmente ativo.

A figura 11 abaixo ilustra a expansibilidade do protocolo em relação ao atraso médio. Para cada tamanho de grupo, retiramos a média das respostas para todos os períodos de transmissão. Neste gráfico fica clara a influência que o tamanho do grupo exerce sobre o atraso médio. Como o número de mensagens transmitidas neste experimento cresce em função do tamanho do grupo, há maior contenção tanto na comunicação interna quanto na externa e maior probabilidade de crescimento da cadeia causal, fatores que influenciam diretamente no atraso médio.

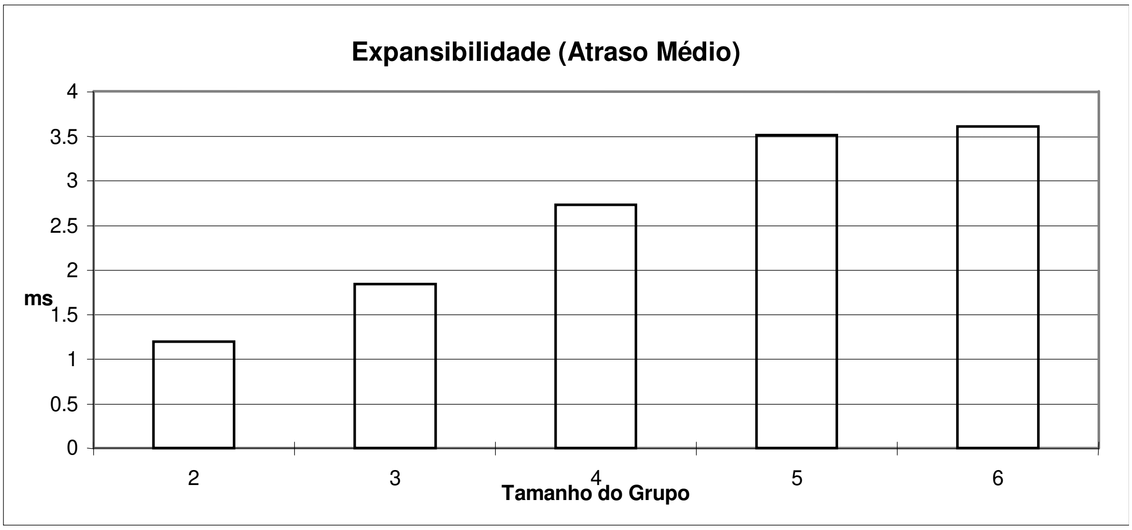


Figura 11: Expansibilidade do protocolo em relação ao atraso médio para um grupo completamente ativo.

3.3.3 Avaliando a Influência da Velocidade do Canal na Dependência Causal e na Entrega das Mensagens.

Se os tempos de transmissão de mensagens fossem todos conhecidos e as mensagens fossem transmitidas em ordem FIFO entre pares de processos, então seria trivial garantir a ordem de entrega causal. No entanto, os diferentes e desconhecidos atrasos de transmissões podem fazer com que a ordem causal seja violada. O objetivo do experimento descrito nesta seção é medir a influência do atraso dos canais na dependência causal e no tempo de entrega das mensagens. Acreditamos que a análise dos resultados apresentados aqui poderá servir de base para ajustar aplicações, dado que se possui conhecimento da latência dos canais.

Configurou-se um grupo de três processos, tal como ilustrado na figura 12, onde um deles é apenas receptor de mensagens, p_2, e os outros são receptores e transmissores. Cada transmissor transmite 1.000 mensagens segundo um período de transmissão, fixado em 100 ms. O atraso do canal de p_0 a p_2 (AC) é ajustado para diferentes valores. Desta forma, provocamos uma variação da dependência causal, pois as mensagens transmitidas por p_0 que provocaram mensagens transmitidas por p_1, demoram de chegar a p_2.

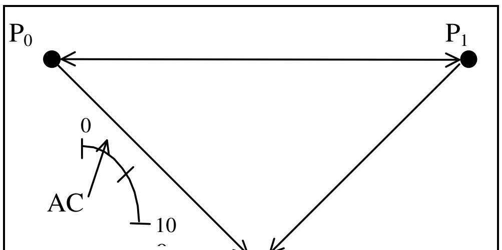


Figura 12: Configuração do experimento para avaliar o grau de influência da velocidade do canal sobre a entrega causal

A fim de variar o atraso do canal, modificamos o código da camada de transporte para conter uma espera ajustável. Assim, para toda mensagem recebida pela camada de transporte para o processo p_2, ao invés da mensagem ser imediatamente entregue para o núcleo da BCG, o processo encarregado de fazê-lo espera um tempo preestabelecido (variando de 0 a 200 ms).

Medimos em relação ao processo p_2 duas respostas: o tempo médio que as mensagens permanecem na BM antes de serem entregues à p_2; e o tamanho médio da dependência causal das mensagens. A segunda medida, a qual chamamos de comprimento médio da cadeia causal, é função de AC. Já a primeira resposta é função de AC e da segunda resposta. Note, portanto, que mesmo que o comprimento médio da cadeia não aumente as mensagens podem ficar mais tempo retidas no protocolo em virtude do tempo de recebimento de outras mensagens que as precedem na cadeia causal. A figura 13 ilustra esse comportamento.

Para medirmos o tempo médio de permanência das mensagens optamos por registrar o intervalo de tempo que a mensagem é armazenada na BM e entregue à aplicação. Desta forma suprimimos o intervalo de tempo que a mensagem chega ao *Receiver* e é passado ao *Deliver* (medidas apresentadas na seção anterior). Isso foi feito para minimizar o efeito da chamada de sistema para acesso ao relógio[11].

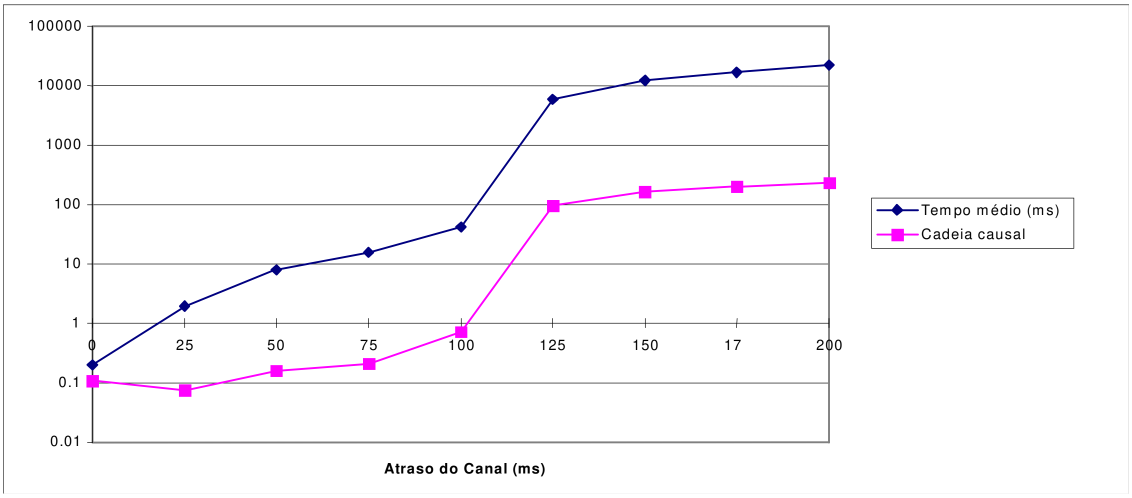


Figura 13: Tempo médio de permanência das mensagens na BM e tamanho da cadeia causal em função da velocidade do canal AC.

Os dados apresentados no gráfico da figura 14 representam a média de três execuções, onde 1.000 mensagens foram transmitidas para cada ponto, totalizando portanto 27.000 mensagens (apêndice C).

3.3.4 Avaliando o *Round Trip*

Esse experimento visa medir o *round trip*, descrito anteriormente (seção 3.2) e foi montado da seguinte forma: um dos processos do grupo, com tamanho N, transmite uma mensagem e espera receber N-1 *replies* dos demais processos do grupo. O *round trip* foi medido após a transmissão de 1.000 mensagens. Configurou-se grupos de 2 a 6 processos e repetiu-se o experimento 10 vezes para cada tamanho de grupo a fim de atenuar as variações das medições. O algoritmo abaixo (figura 14) ilustra o procedimento realizado:

[11] Como ambas as chamadas de acesso ao relógio do sistema são feitas no mesmo processo (Deliver) e o intervalo de tempo entre elas é pequeno, há uma otimização do sistema operacional diminuindo o tempo de execução de tal rotina à dezenas de microsegundos.

```
Processo 0:
/* Para cada tamanho de Grupo */
        BCGcausal  G(G_NUMBER);
        start = time(); i = 1; Num_mesg = 1000;
        while (i ≤ Num_mesg) {
                G.send(msg);
                for (j = 1; j< |G|; j++)
                        G.receive(reply);
                i++;
        }
        rtrip = (time() - start) / Num_mesg
```

```
Demais Processos do grupo:
/* Para cada tamanho de Grupo */
        BCGcausal  G(G_NUMBER);
        i = 1; Num_mesg = 1000;
        while (i ≤ Num_mesg) {
                G.receive(msg);
                G.send(reply);
        i++;
        }
```

Figura 14: Algoritmos para medir round trip.

Esses mesmos algoritmos foram utilizados para as medições da camada de transporte, apenas substituindo a operação *send/receive* da BCG pelas operações equivalentes da camada de transporte. Os resultados do experimento estão representados na figura 15 a seguir. Os dados obtidos com o experimento encontram-se no apêndice D.

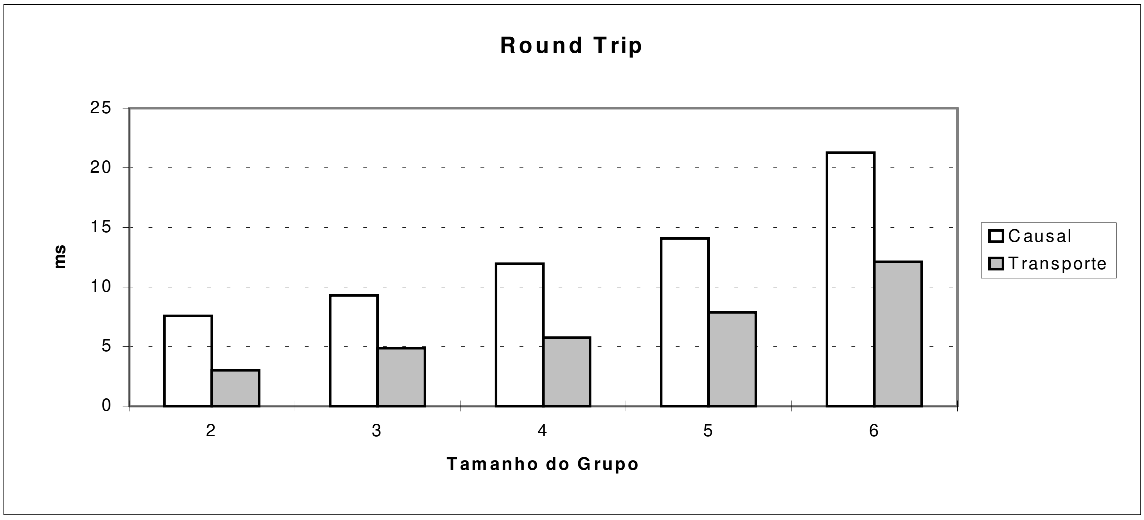


Figura 15: Expansibilidade segundo *round trip*.

O tempo de *round trip* aumenta ligeiramente com o tamanho do grupo. Observe que a curva de expansibilidade é linear para as configurações apresentadas (até 6 processos), devido aos fatores discutidos na seção 3.2. Ressaltamos aqui que não há influência do mecanismo de *timesilence*, pois todos os processos estão ativos, reduzindo portanto o número de mensagens nulas transmitidas para zero. Contudo, é importante salientar que há a influência da camada de transporte sobre os resultados obtidos, causado pelo excesso de conexões TCP/IP (ver seção 2.4).

4 Avaliação e Comparação com Outros Trabalhos

Por causa das diferentes condições de experimentação que envolvem: ambiente experimental, *hardware*, decisões de implementação, tráfego na rede, etc., torna-se difícil comparar trabalhos distintos, considerando apenas os valores absolutos apresentados. Desta forma, analisaremos o comportamento de diferentes protocolos em situações semelhantes.

Até onde sabemos existem dois protocolos de ordem causal que apresentam dados relativos à experimentação, O CBCAST [BSS91] e o Psync[PBS89]. O primeiro assemelha-se bastante com o BCGcausal, pois também ordena as mensagens baseado em vetores que registram a causalidade entre as mensagens (seção 2.3). Já o segundo é baseado na construção de grafos de contextos, mantidos por cada processo do grupo. Cada vértice do grafo representa uma mensagem e as arestas as dependências causais.

Analisando o *round trip* do CBCAST, observa-se a mesma curva linear de crescimento encontrada por nós para o BCGcausal. O CBCAST apresenta, em média, 12.14 ms, enquanto que o protocolo BCGcausal 12.83 ms, valores que podem ser considerados iguais. Vale ressaltar que a camada de transporte implementada pela BCG utiliza TCP/IP, enquanto que o CBCAST usa pacotes UDP. Com relação ao *throughput* encontramos valores variando entre 1.610 a 1869 mensagens/segundo, enquanto que o CBCAST apresenta resultados inferiores, de 305 a 1019 mensagens/segundo para os mesmos tamanhos de grupo.

Em [PBS89] são apresentados dois tipos de experimentos para avaliar o *round trip* do Psync. O primeiro tipo se assemelha com o nosso experimento para um grupo de 2 processos, onde foram encontrados tempos de *round trip* de 4.0 ms, enquanto nós encontramos 7.56 ms. Vale ressaltar que o Psync foi implementado sobre a camada IP, o que pode explicar a diferença encontrada. O segundo tipo de experimento tenta medir o *round trip* para grupos maiores. Configurou-se um anel lógico, onde circula um *token* implícito. Quem detém o *token* transmite M mensagens para os demais. O processo que se segue no anel começa a transmitir após receber o *token* e. cada processo mede os tempos após receber M mensagens. Tal abordagem é inadequada aos seus objetivos e incompatíveis com os experimentos apresentados por nós neste trabalho, pois na verdade, as medições são realizadas considerando a comunicação entre pares de processos.

Os demais trabalhos que temos conhecimento que também apresentam dados relativos à implementação, referem-se a ordem total. Ao comparamos com esses trabalhos devemos ter em mente as diferenças entre a ordenação total e causal. Para garantir a entrega em ordem total é necessário que as mensagens recebidas pelos processos sejam retidas por mais tempo, até que todo o grupo concorde em entregá-las na mesma ordem, o que pode requerer maior número de mensagens para sincronização. Tais características influem diretamente no desempenho, como pode ser observado nos protocolos de ordem total descritos em [Mac94, MS95, ACM95, CBM94].

Gostaríamos de destacar dois protocolos de ordem total: Newtop [Mac94, EMS95], por fazer parte da BCG (seção 2.2); e o *On-Demand* [ACM95], por apresentar valores de *throughput* relativamente altos, quando comparados com os demais protocolos de ordem total.

O Newtop apresenta uma sobrecarga média 100 vezes maior do que os valores encontrados por nós para o BCGcausal. A diferença entre os valores se deve ao fato do Newtop ter que esperar blocos completos para entregar mensagens, o que torna seu desempenho dependente do mecanismo de *timesilence*.

Os maiores valores de *throughput*, em torno de 1000 mensagens/segundo, são encontrados por simulação pelo protocolo *On-Demand* [ACM95]. Tal protocolo, baseia-se na passagem de um *token* que circula entre os membros do grupo. Tal política de ordenação exige um número menor de mensagens trocadas pelos membros do grupo, influindo no desempenho. No entanto, há maior esforço para reconfigurar o grupo caso haja falhas.

5 Conclusão

Existem poucos trabalhos na literatura que preocuparam-se em apresentar dados relativos à implementação e ao desempenho de protocolos de comunicação em grupo e, em particular, de ordem causal. Em parte isso se deve à dificuldade de realizar medições, devido às interferências inerentes ao experimento, tais como manipulação de eventos probabilísticos (contenção na comunicação, escalonamento de processos pelo sistema operacional etc.). Para contornar essas dificuldades muitos autores preferem usar simulação, onde é possível controlar ou até mesmo evitar a presença desses fatores. Contudo, acreditamos que com experimentação em ambiente real pode-se obter de forma mais expressiva o comportamento do sistema analisado, pois na prática esses fatores estão presentes.

Neste trabalho apresentamos detalhes da implementação do protocolo BCGcausal, descrevendo sua arquitetura e a interação entre os processos que compõem a plataforma BCG. Validamos e avaliamos, através de experimentação em ambiente real, a implementação do protocolo BCGcausal. Os experimentos visaram colher dados que pudessem descrever o protocolo sob diversas situações, traçando assim um perfil do sistema. Para tanto, montamos quatro tipos de experimentos, onde variamos o tamanho do grupo e o período de transmissão das mensagens. Os experimentos foram realizados sobre um conjunto de seis máquinas Unix interconectadas por uma rede local

Os experimentos obtiveram medidas de *throughput*, *round trip* e *atraso médio.* Avaliamos também a expansibilidade do protocolo em relação a essas medidas e medimos a variação do *throughput* e do atraso médio para diferentes períodos de transmissão. Quanto ao atraso médio, não temos conhecimento de uma semelhante avaliação feita para um protocolo de ordem causal.

Comparando os dados encontrados para o BCGcausal com o dos protocolo mais referenciado na literatura, o CBCAST do sistema ISIS [BSS91], encontramos resultados bastante parecidos no que se refere ao *round trip*. Com relação ao *throughput,* este apresenta valores de *throughput* inferiores ao encontrados por nós para o protocolo BCGcausal.

Outro experimento, ainda não explorado, é a verificação da influência da velocidade do canal no atraso da entrega das mensagens à aplicação e na dependência causal. Essa medida apresenta resultados interessantes: demonstra a relação entre a velocidade dos canais de comunicação com o atraso médio da entrega de mensagens e com o tamanho da cadeia causal.

Os resultados obtidos apresentam uma descrição detalhada do comportamento do protocolo BCGcausal, servindo de base para que uma aplicação possa ser parametrizada adequadamente. Acreditamos também que a metodologia apresentada, juntamente com a descrição detalhada dos experimentos, possam fornecer subsídios para futuros trabalhos no que se refere à avaliação de desempenho de protocolos de comunicação em grupo, tema, até então, pouco explorado na literatura.

Bibliografia

[ACM95] G. A. Alvarez, F. Cristian e S. Mishra. *On-Demand Asynchronous Atomic Broadcast*. 5th IFIP Working Conf. on Dependable Computing for Critical Applications, 1995.

[BJ87] K. Birman, T. A. Joseph. *Reliable Communication in the Presence of Failures*. ACM TDCS, (5,1), pp. 47-76, 1987.

[BSS91] K. Birman, A. Schiper e P. Stephenson. *Lighweight Causal and Atomic Group Multicast*. ACM Transaction on Computing Systems, (9,3), pp. 272-314, agosto de 1991.

[CBM94] F. Cristian, R. de Beijer e S. Mishra. *A Performance Comparation of Asynchronous Atomic Broadcast Protocols*. Distributed Systems Engineering Journal, (1,4), pp. 177-201, 1994.

[CV95] F. J. N. Cosquer e P. Veríssimo. *The Impact of Group Communication Paradigms on Groupware Support*. Proc. of the 5th Workshop on Future Trends of Distributed Computing Systems. Korea, 1995.

[CZ85] D. Cheriton, W. Zwaenepoel. *Distributed Process Groups in V Kernel*. ACM TOCS, (3,2), pp. 97-107, 1985.

[EMS95] P. D. Ezhilchelvan, R. A Macêdo e S. K. Shrivastava. *Newtop: A Fault-Tolerant Group Communication Protocol.* Proc. Of the International Conference on Distributed Computing Systems, IEEE Computer Society, pp. 296-306, Canadá , maio 30 - junho 2, 1995.

[Lam78] L. Lamport. *Time, Clocks and Ordering of Events in a Distributed System*. CACM, (21, 7), pp. 558-265, julho 1978,.

[Mac94] R. J. Macêdo. *Fault Tolerant Group Communication Protocols for Asynchronous Systems*. Tese de Doutorado, agosto de 1994.

[Mac95] R. A Macêdo. *Causal Order Protocols for Group Communication*. SBRC 95, pp. 265-283, maio de 1996.

[MES96] R. A Macêdo e P. D. Ezhilchelvan, S. K. Shrivastava. *Buffer Overflow Avoidance Techniques for Groups Communication Protocols*. SBRC 96, pp. 633-652, de 1996.

[MR93] A. Mostefaoui e M. Raynal. *Causal Multicast in Overlapping Groups: Towards a Low Cost Approach*. Relatório Técnico INRIA, nº 710, março de 1993.

[MW97] S. Mishra e L. Wu. *Flow Control in High Performance Atomic Multicast Services*. Proc. of the International Symposium on High Performance Computing. pp. 295-306, Canadá, julho de 1997.

[NN95] N. Nupairoj e L. Ni. *Benchmarking of Multcast Comunication Services*. Relatório Técnico MSU-CPS-ACS-103, Departament of Computer Science, Michgan State University, 1995.

[NN97] N. Nupairoj e L. Ni, *Peformance Metrics and Measurement Techniques of Collective Comunication Services*. First International Workshop on Communication and Architectural Support for Network-Based Parallel Computing (CANPC 97), San Antônio, Tx, pp. 212-226, fevereiro 1997.

[PBS89] L. L. Peterson, N. C. Buchholz e R. D. Schlichting. *Preserving and Using Context Information in Interprocess Comunication*. ACM Transaction on Computing Systems, (7,3), pp. 217-246, agosto de 1989.

[VRV93] P. Veríssimo, L. Rodrigues e Werner Vogels. *Group Orientation: a Paradigm for Modern Distributed Systems.* ESPRIT Basic Research Project First Year Report, volume 1, outubro 1993.

Apêndice A

Este apêndice apresenta o comportamento relativo ao *throughput*, tanto do protocolo BCGcausal, quanto da camada de transporte para grupos de 2, 3, 4 e 5 processos respectivamente.

Figura A1: Dados e gráfico relativos a quatro execuções para avaliação do *throughput* para um grupo de 2 processos.

Protocolo Causal - 2 processos					
Transmissão	Execução1	Execução2	Execução3	Execução4	Throughput Causal
1600	1514.42413	1447.52993	1513.19124	1751.87801	1556.755831
800	1792.44556	2094.2847	2254.0748	1878.95755	2004.940654
400	2186.55705	2252.96772	1979.70799	2086.98556	2126.55458
200	755.264191	2480.38019	2476.61457	2264.36456	1994.155878
100	1324.68227	993.660446	1421.94719	1648.53281	1347.205678
50	2126.53296	2562.59129	1880.31777	2058.36706	2156.952272
25	1919.64371	1718.07727	2097.6894	2191.6175	1981.756969
Média	1659.9357	1935.64165	1946.22042	1982.95758	1881.188837

Camada de Transporte - 2 processos					
Transmissão	Execução1	Execução2	Execução3	Execução4	Throughput Transp.
1600	2203.86646	777.574744	2292.72084	2601.62602	1968.947016
800	842.240022	2641.56127	3550.37989	3221.21362	2563.848701
400	3248.38799	2955.2312	3622.74213	3102.60308	3232.2411
200	2900.93033	3947.2178	2551.52811	2933.27389	3083.237532
100	2798.40827	1113.40472	3049.09655	1686.46862	2161.84454
50	3258.27357	2480.13413	3462.53194	3205.22066	3101.540076
25	3300.29736	3360.29409	3414.89235	2324.75422	3100.059503
Média	2650.34343	2467.91685	3134.84169	2725.02287	2744.53121

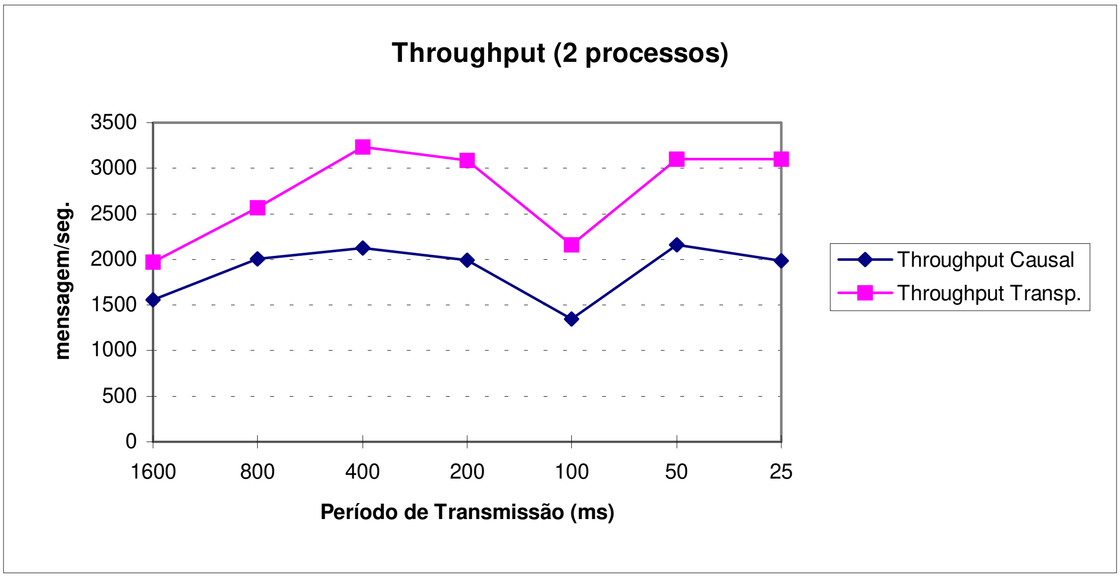


Figura A2: Dados e gráfico relativos a quatro execuções para avaliação do *throughput* para um grupo de 3 processos.

Protocolo Causal - 3 processos					
Transmissão	Execução1	Execução2	Execução3	Execução4	Throughput Causal
1600	1514.77971	1790.88867	1696.33845	1731.06817	1683.268752
800	2085.94946	2105.23657	2077.46881	2028.3194	2074.243559
400	1918.04206	2023.81217	1128.45802	1851.76269	1730.518737
200	1554.29865	2696.43531	1566.67126	1395.9754	1803.345156
100	2273.49716	2154.85197	1761.67373	691.352562	1720.343857
50	729.293533	2235.766	1962.05773	1649.63691	1644.188543
25	2050.43244	1799.53536	1251.44072	1767.70891	1717.279356
Média	1732.32757	2115.21801	1634.87268	1587.97486	1767.59828

Camada de Transporte - 3 processos					
Transmissão	Execução1	Execução2	Execução3	Execução4	Throughput Transp.
1600	746.809257	2271.97339	2588.12568	2569.61736	2044.131421
800	2453.85525	3019.76132	3095.61104	3037.15965	2901.596815
400	793.15979	2401.75616	3245.69945	3483.11906	2480.933616
200	1697.03799	3621.67937	3025.05044	2928.15479	2817.98065
100	2812.71799	3078.05012	1037.08728	3089.57608	2504.357865
50	2625.64361	2991.13726	3259.64366	3537.13103	3103.38889
25	3604.91566	4095.28919	2323.54424	3279.95749	3325.926646
Média	2104.87708	3068.52097	2653.5374	3132.10221	2739.759415

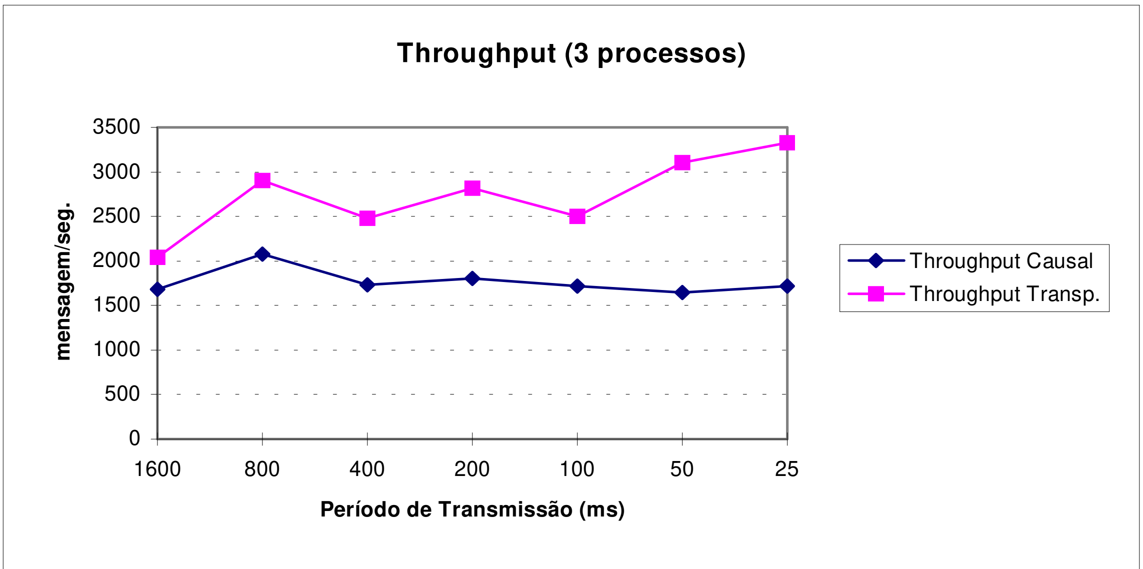


Figura A3: Dados e gráfico relativos a quatro execuções para avaliação do *throughput* para um grupo de 4 processos.

Protocolo Causal - 4 processos					
Transmissão	Execução1	Execução2	Execução3	Execução4	Throughput Causal
1600	1774.90007	1648.62521	1556.60678	1683.27215	1629.501381
800	2074.10787	2228.42715	2082.73455	2258.44262	2189.868108
400	2194.30096	2085.85374	2031.92562	1760.08927	1959.289542
200	2499.0004	2083.60247	2212.04191	2050.64688	2115.430419
100	1903.83356	2205.99103	1475.62271	1590.11837	1757.244037
50	1878.3117	1756.66655	1737.85802	1813.86811	1769.464227
25	1726.20938	1695.18533	2104.72258	1858.02462	1885.977511
Média	2007.23771	1957.7645	1885.93031	1859.20886	1900.967889

Camada de Transporte - 4 processos					
Transmissão	Execução1	Execução2	Execução3	Execução4	Throughput Transp.
1600	1293.8117	2629.32326	2798.02459	2866.33971	2396.874818
800	3279.86067	3611.34685	3240.87128	3751.00808	3470.771721
400	3836.533	3422.33691	3547.48306	3550.32947	3589.17061
200	3270.83851	3162.24532	3344.35857	3892.74703	3417.547359
100	3334.48929	3173.47483	2697.27898	2720.46661	2981.427429
50	3798.93022	2877.42459	2737.5285	3120.96226	3133.711393
25	3703.48424	3176.90265	3889.15897	4157.50284	3731.762173
Média	3216.84966	3150.43634	3179.24342	3437.05086	3245.895072

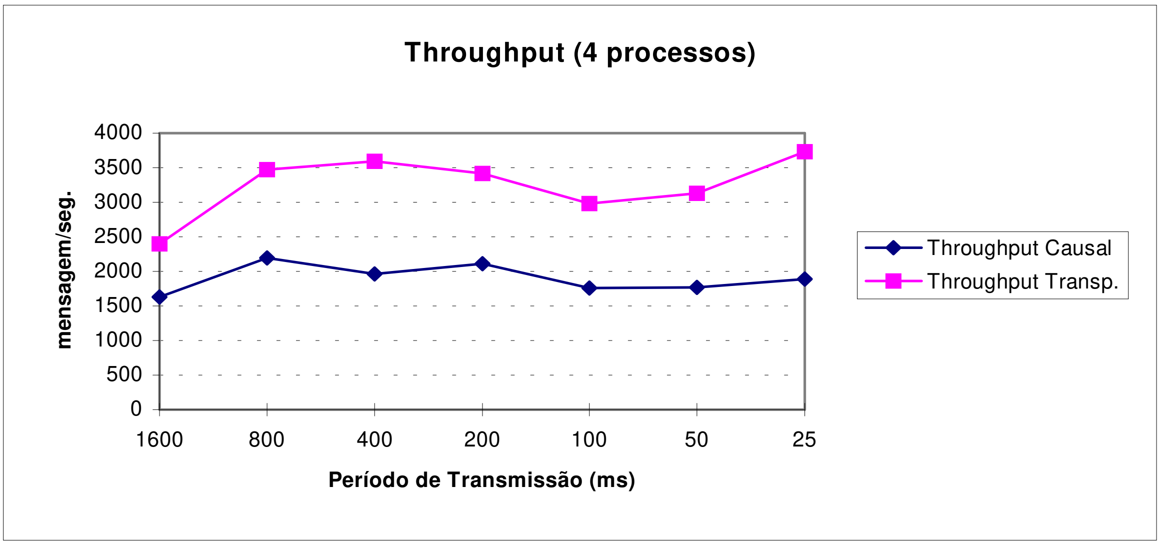
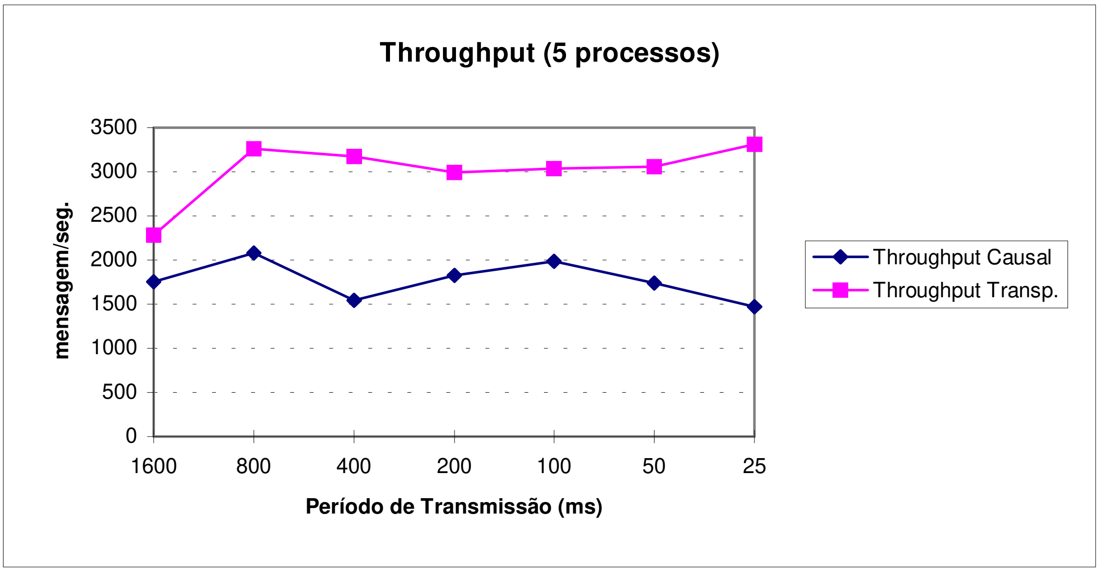


Figura A4: Dados e gráfico relativos a quatro execuções para avaliação do *throughput* para um grupo de 5 processos.

Protocolo Causal - 5 processos					
Transmissão	Execução1	Execução2	Execução3	Execução4	Throughput Causal
1600	1745.64244	1733.27047	1795.82865	1739.7628	1756.287308
800	2008.13697	2001.53718	2062.38721	2176.82336	2080.249252
400	1775.73529	1651.24677	1543.27418	1444.26856	1546.263173
200	2008.16923	2174.12573	1475.45289	1826.69421	1825.424278
100	1636.8407	2343.67676	1571.7957	2037.11211	1984.194855
50	1478.58346	1853.55094	1680.49716	1675.81663	1736.621575
25	1390.60203	1634.5959	1356.30553	1418.10782	1469.669749
Média	1720.53002	1913.14339	1640.79162	1759.79793	1771.244313

Camada de Transporte - 5 processos					
Transmissão	Execução1	Execução2	Execução3	Execução4	Throughput Transp.
1600	1091.03724	2633.61653	2783.15855	2635.65728	2285.867401
800	3110.29414	3087.90652	3338.88702	3497.59191	3258.669897
400	3546.26277	2829.74286	2861.03386	3460.72253	3174.440506
200	3162.40533	3196.93095	2211.60651	3395.14765	2991.522609
100	2843.87415	2941.85137	2875.06685	3482.79153	3035.895973
50	3004.97925	2858.81731	3492.53471	2871.39317	3056.931109
25	3279.34439	3458.77144	3849.26287	2649.51196	3309.222666
Média	2862.59961	3001.091	3058.79291	3141.83086	3016.078594

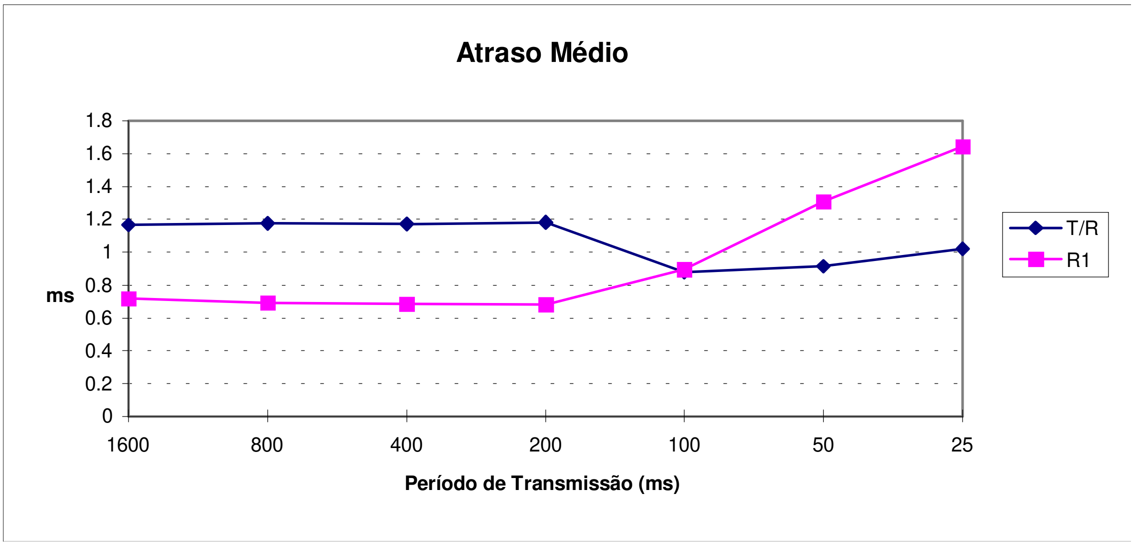


Apêndice B

Este apêndice apresenta o comportamento relativo ao atraso médio na entrega de mensagens dos processos transmissores e receptores para as diferentes configurações do grupo: 2, 3, 4, 5 e 6 processos, tanto para grupos com um processo ativo, quanto para grupos totalmente ativos.

Figura B1: Dados relativos ao atraso médio para um grupo de dois processos com um transmissor.

Processo transmissor/receptor (T/R)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	1.17858	1.16644	1.16183	1.1715	1.1695875	0.00717886
800	1.17846	1.17259	1.16293	1.18653	1.1751275	0.00993882
400	1.16578	1.15614	1.16674	1.19107	1.1699325	0.01488245
200	1.17767	1.17554	1.18389	1.19609	1.1832975	0.00923484
100	0.872368	0.877355	0.880336	0.889953	0.880003	0.00740312
50	0.931414	0.890853	0.920739	0.915399	0.91460125	0.01717514
25	1.0396	0.979801	1.06319	1.00133	1.02098025	0.03746207
Processo Receptor 1 (R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.730143	0.697299	0.736126	0.700212	0.715945	0.02003378
800	0.729752	0.686558	0.684439	0.670507	0.692814	0.02563394
400	0.697132	0.670961	0.676458	0.689827	0.6835945	0.01200893
200	0.689969	0.675402	0.673981	0.674669	0.67850525	0.00766449
100	1.0681	0.833652	0.858456	0.814127	0.89358375	0.11774983
50	1.22098	1.61662	1.04388	1.34985	1.3078325	0.24105979
25	1.92599	1.27435	1.80174	1.57968	1.64544	0.28587302


Atraso Médio
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
ms
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R
R1

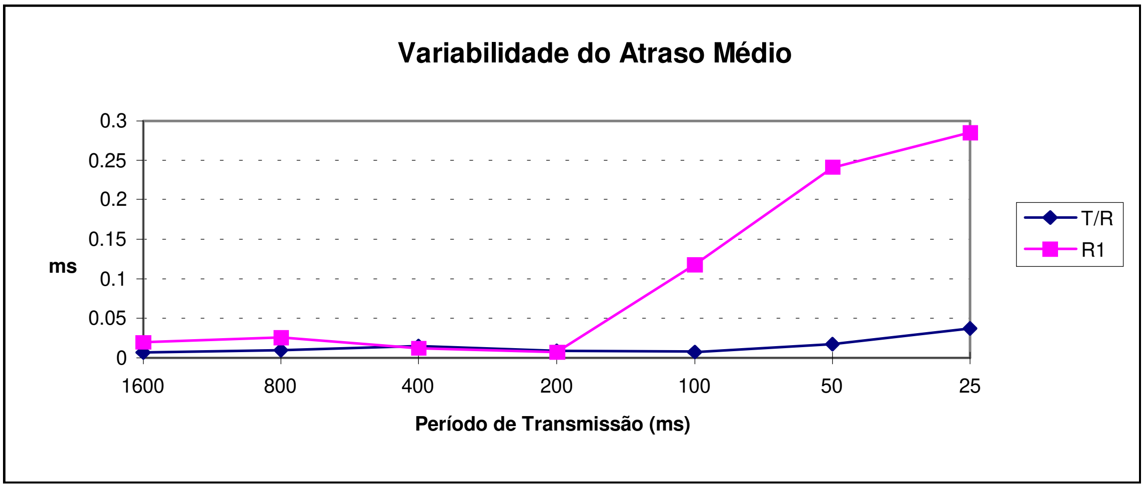
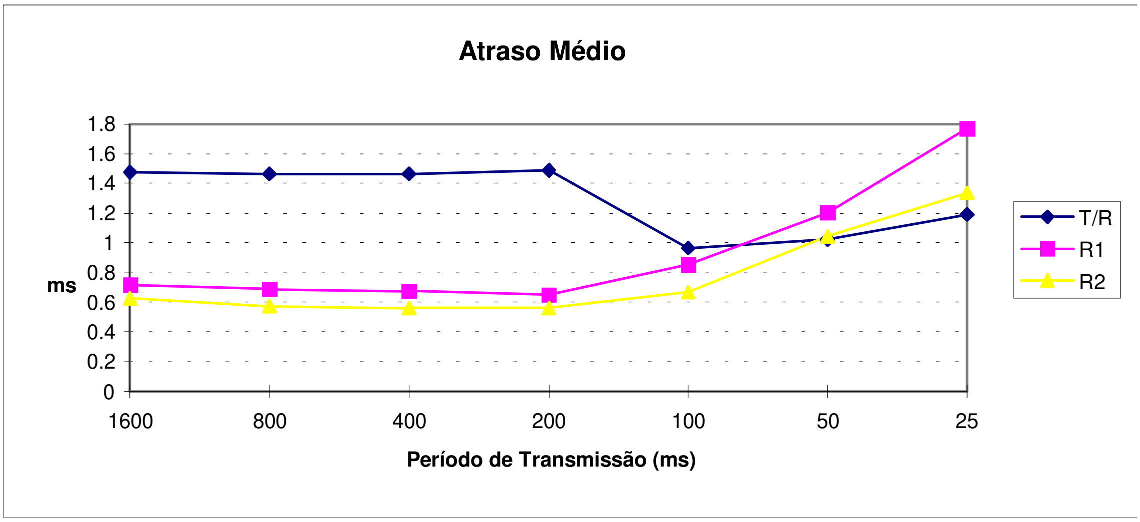

Variabilidade do Atraso Médio
0.3
0.25
0.2
0.15
0.1
0.05
0
ms
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R
R1

Figura B2: Dados relativos ao atraso médio para um grupo de três processos com um transmissor.

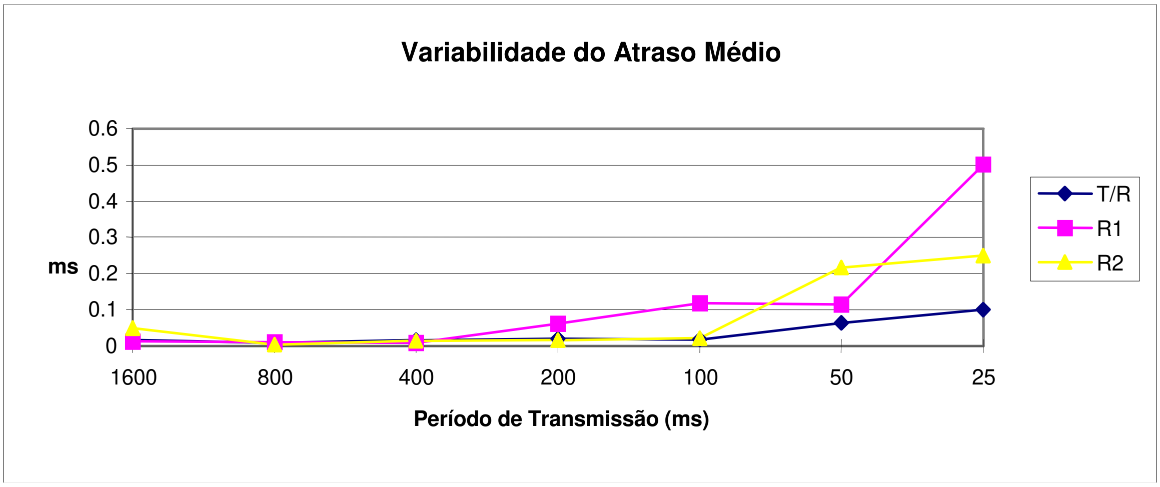
Pocesso transmissor/receptor (T/R)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	1.47297	1.45714	1.49638	1.46581	1.473075	0.01683091
800	1.47229	1.4518	1.4677	1.46461	1.4641	0.00878609
400	1.48466	1.4444	1.45361	1.4643	1.4617425	0.01730754
200	1.47102	1.48723	1.47335	1.51514	1.486685	0.02027477
100	0.940261	0.96392	0.96179	0.985914	0.96297125	0.01865852
50	0.969854	1.09902	0.965646	1.04608	1.02015	0.06427341
25	1.10282	1.15358	1.33545	1.16706	1.1897275	0.10100894

Pocesso receptor 1 (R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.716946	0.731728	0.711374	0.703274	0.7158305	0.01199307
800	0.674667	0.694976	0.698291	0.687642	0.688894	0.01047647
400	0.663367	0.681774	0.678931	0.680507	0.67614475	0.00859751
200	0.665988	0.6757	0.702915	0.561522	0.65153125	0.06200828
100	0.958769	0.908724	0.85626	0.685751	0.852376	0.11870628
50	1.1981	1.34455	1.06531	1.21114	1.204775	0.11412363
25	2.43905	1.79173	1.24726	1.59141	1.7673625	0.50107344

Pocesso receptor 2 (R2)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.577439	0.673368	0.658364	0.586988	0.62403975	0.04883951
800	0.578112	0.574066	0.571491	0.571872	0.57388525	0.00303775
400	0.556692	0.540789	0.561553	0.574831	0.55846625	0.014059
200	0.54366	0.551272	0.583556	0.561522	0.5600025	0.01732412
100	0.679625	0.636041	0.669887	0.685751	0.667826	0.02217398
50	0.733012	1.17057	1.06132	1.21114	1.0440105	0.2167712
25	1.01616	1.46085	1.27645	1.59141	1.3362175	0.249443



Atraso Médio
ms
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R
R1
R2



Variabilidade do Atraso Médio
ms
0.6
0.5
0.4
0.3
0.2
0.1
0
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R
R1
R2

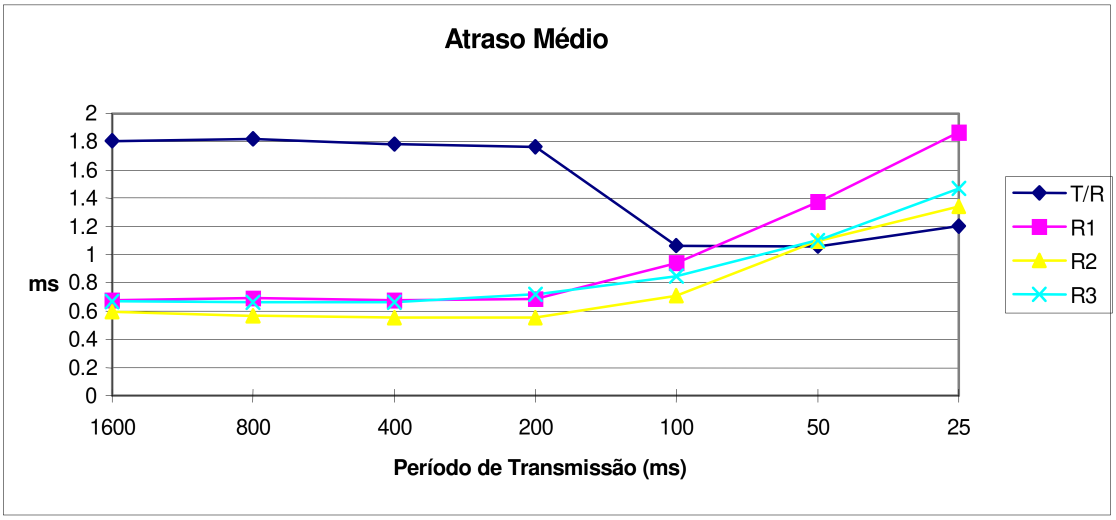
Figura B3: Dados relativos ao atraso médio para um grupo de quatro processos com um transmissor.

Processo transmissor/Receptor (T/R)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	1.82137	1.80709	1.80613	1.80021	1.8087	0.00897786
800	1.82637	1.82705	1.82786	1.80845	1.8224325	0.00934154
400	1.79246	1.77042	1.791	1.78164	1.78388	0.01017349
200	1.7937	1.78092	1.72316	1.77126	1.76726	0.03080303
100	1.03404	1.06289	1.01628	1.13921	1.063105	0.05425057
50	1.04661	1.0721	1.03457	1.09304	1.06158	0.02616641
25	1.25365	1.18698	1.21673	1.15804	1.20385	0.04094341

Processo Receptor 1 (R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.686239	0.672687	0.673866	0.682328	0.67878	0.00657006
800	0.712575	0.680579	0.682979	0.687562	0.69092375	0.01472199
400	0.676732	0.677627	0.676361	0.67638	0.676775	0.00059306
200	0.696837	0.679169	0.687084	0.687802	0.687723	0.00722612
100	0.969751	0.892833	0.973892	0.924784	0.940315	0.03868525
50	1.24229	1.24572	1.41587	1.58241	1.3715725	0.16224214
25	2.20217	2.7338	1.44628	1.076	1.8645625	0.74528378

Processo Receptor 2 (R2)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.599696	0.591067	0.588085	0.592858	0.5929265	0.00492371
800	0.555231	0.57375	0.564099	0.567527	0.56515175	0.00772661
400	0.566373	0.552511	0.544629	0.550554	0.55351675	0.00920262
200	0.57213	0.552914	0.545771	0.534736	0.55138775	0.01572046
100	0.670529	0.662047	0.672574	0.831839	0.70924725	0.08185481
50	1.22953	1.02646	0.845837	1.28283	1.09616425	0.20012741
25	1.28541	1.16567	1.13086	1.77607	1.3395025	0.29847759

Processo Receptor 3 (R3)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.669376	0.681618	0.661625	0.664726	0.66933625	0.00878561
800	0.662414	0.66941	0.660715	0.66678	0.66482975	0.00398117
400	0.660895	0.671744	0.662518	0.662741	0.6644745	0.00491567
200	0.674126	0.673052	0.65922	0.873396	0.7199485	0.10252328
100	0.885927	0.846935	0.835472	0.828108	0.8491105	0.02573779
50	0.92772	1.06768	1.03537	1.38391	1.10367	0.19617427
	1.45579	1.46097	0.926278	2.02861	1.467912	0.4501647


Atraso Médio
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
ms
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R
R1
R2
R3

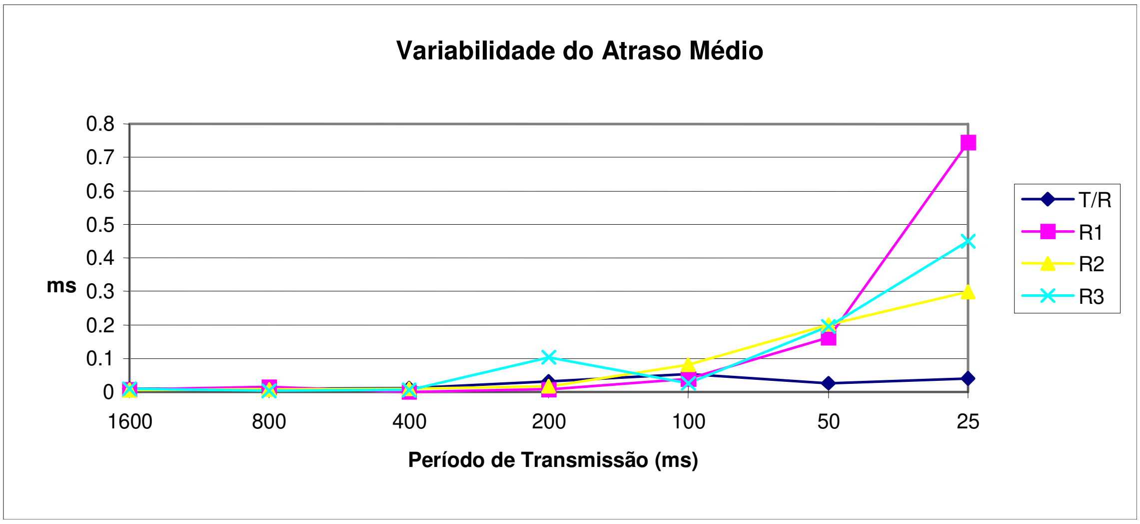

Variabilidade do Atraso Médio
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
ms
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R
R1
R2
R3

Figura B4: Dados relativos ao atraso médio para um grupo de cinco processos com um transmissor.

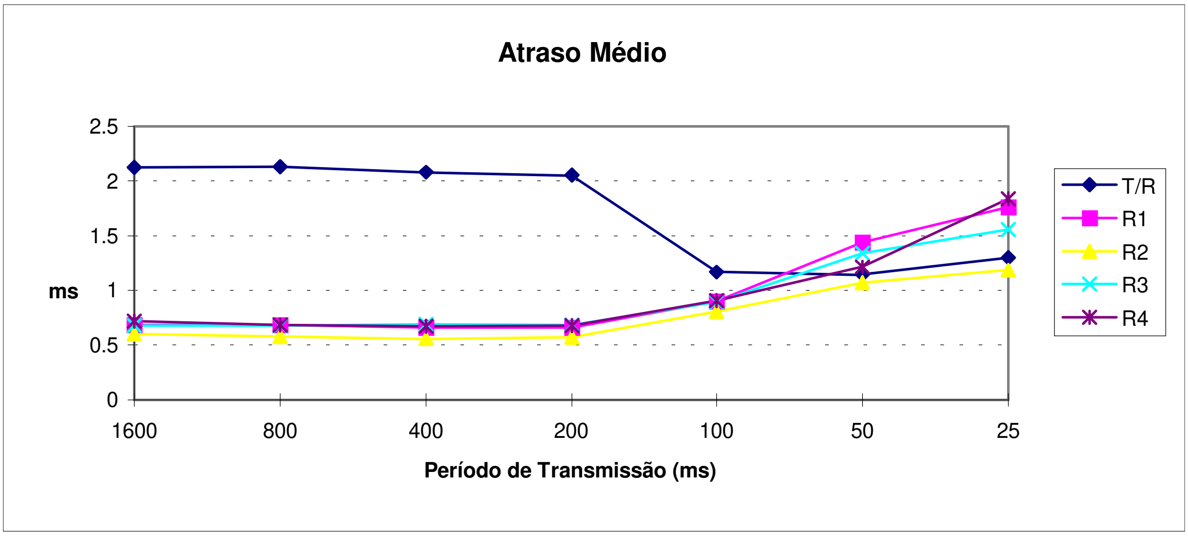
Processo transmissor/receptor (T/R)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	2.15897	2.1284	2.10363	2.12766	2.129665	0.02267322
800	2.14322	2.16876	2.12655	2.10707	2.1364	0.02614672
400	2.07539	2.09405	2.06927	2.08797	2.08167	0.01134538
200	2.16096	2.06244	2.0289	1.952	2.051075	0.0866246
100	1.23397	1.24519	1.09714	1.09888	1.168795	0.08186681
50	1.14875	1.14893	1.16563	1.11694	1.1450625	0.02035069
25	1.22121	1.40244	1.22822	1.35234	1.3010525	0.09053403

Processo receptor 1 (R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.6754	0.686277	0.696967	0.697042	0.6889215	0.00895121
800	0.725529	0.663917	0.676237	0.678985	0.686167	0.02342336
400	0.666227	0.650819	0.668128	0.663926	0.662275	0.00677941
200	0.662885	0.658693	0.663836	0.656654	0.660517	0.00295267
100	0.943562	0.812106	0.978669	0.8755	0.90245925	0.06400756
50	1.37333	1.24877	1.27107	1.86355	1.43918	0.24947097
25	1.39934	2.38652	1.79515	1.47003	1.76276	0.3898367

Processo receptor 2 (R2)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.603128	0.598249	0.606506	0.612535	0.6051045	0.00600225
800	0.577945	0.585249	0.582071	0.586553	0.5829545	0.00383361
400	0.553732	0.549553	0.572483	0.557618	0.5583465	0.00998316
200	0.581707	0.563682	0.594748	0.559763	0.574975	0.01628116
100	0.863837	0.764994	0.819164	0.794331	0.8105815	0.04184139
50	1.15596	1.13789	1.02383	0.960921	1.06965025	0.0931443
25	1.04264	0.831498	1.47381	1.40298	1.187732	0.30338462

Processo receptor 3 (R3)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.68237	0.680882	0.686248	0.672375	0.68046875	0.00585076
800	0.679396	0.680611	0.676765	0.67372	0.677623	0.0030573
400	0.69051	0.692468	0.698065	0.689551	0.6926485	0.00380958
200	0.690874	0.674513	0.688585	0.691284	0.686314	0.00795646
100	0.874864	0.957949	0.91789	0.823291	0.8934985	0.05780758
50	0.978109	1.51445	1.10707	1.78251	1.34553475	0.37029126
25	1.28495	1.32928	2.42223	1.19352	1.557495	0.57925449

Processo receptor 4 (R4)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.707489	0.71211	0.730793	0.736981	0.72184325	0.01425985
800	0.681635	0.700793	0.679679	0.690943	0.6882625	0.0096919
400	0.685351	0.677534	0.674382	0.671832	0.67727475	0.00586755
200	0.682068	0.679559	0.671857	0.695153	0.68215925	0.00969095
100	0.983473	0.853755	0.930851	0.859279	0.9068395	0.0619925
50	1.07578	1.25316	1.36569	1.16627	1.215225	0.12372049
25	2.76106	1.6596	1.3312	1.58583	1.8344225	0.6335748


Atraso Médio
2.5
2
1.5
1
0.5
0
ms
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R
R1
R2
R3
R4

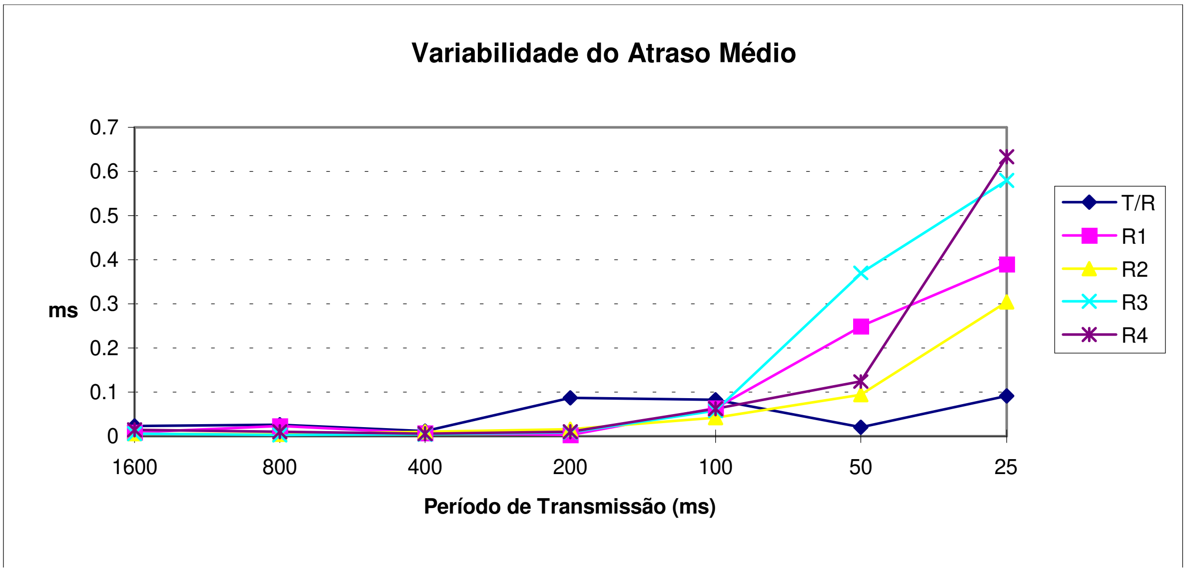

Variabilidade do Atraso Médio
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
ms
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R
R1
R2
R3
R4

Figura B5: Dados relativos ao atraso médio para um grupo de seis processos com um transmissor.

Processo transmissor/receptor (T/R)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	2.43955	2.45501	2.43466	2.44542	2.44366	0.00875237
800	2.44779	2.42478	2.46744	2.45139	2.44785	0.01759288
400	2.35228	2.36365	2.36181	2.36733	2.3612675	0.00641612
200	2.16048	2.02183	2.12914	2.02947	2.08523	0.07004615
100	1.21188	1.25384	1.24443	1.2083	1.2296125	0.02291429
50	1.28844	1.32389	1.27026	1.26648	1.2872675	0.02622948
25	1.31466	1.35231	1.52518	1.39665	1.3972	0.09166451

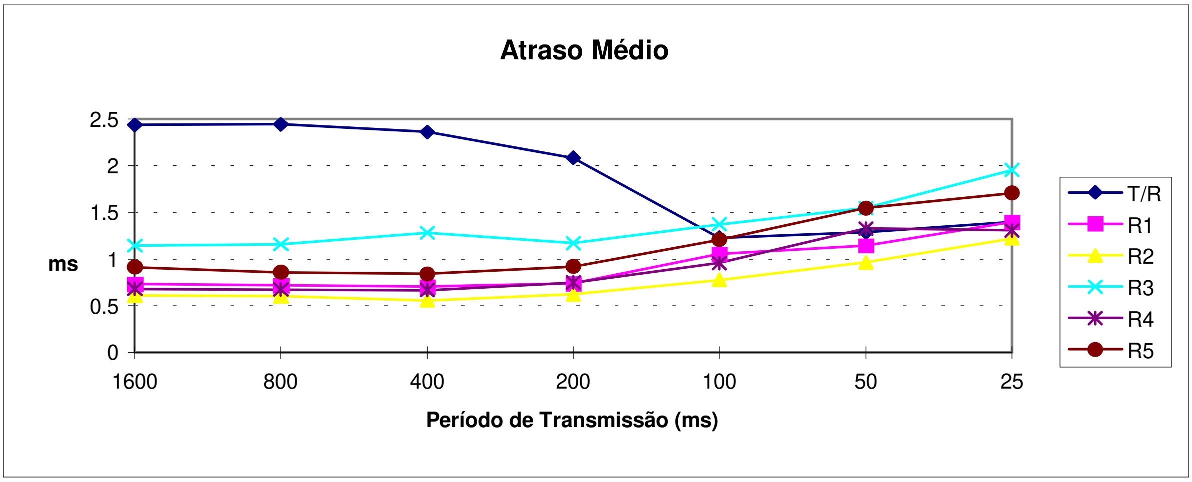
Processo receptor 1 (R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.720653	0.722107	0.726218	0.748379	0.72933925	0.01291008
800	0.744085	0.717566	0.711049	0.704846	0.7193865	0.01726528
400	0.707799	0.719919	0.689217	0.701615	0.7046375	0.01278602
200	0.715634	0.743321	0.782772	0.718862	0.74014725	0.03098873
100	1.32544	0.936841	0.931968	1.01725	1.05287475	0.18587022
50	1.27984	1.26273	1.06197	0.9939	1.14961	0.14339033
25	1.7743	1.06001	1.68415	1.07881	1.3993175	0.38279501

Processo receptor 2 (R2)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.665909	0.597126	0.602729	0.579496	0.611315	0.03771826
800	0.604775	0.605205	0.591096	0.596429	0.59937625	0.00684031
400	0.5574	0.559602	0.565103	0.558727	0.560208	0.00338657
200	0.693106	0.610231	0.596361	0.594978	0.623669	0.04680092
100	0.774674	0.91668	0.686957	0.741023	0.7798335	0.09812557
50	1.14491	0.882515	0.924848	0.906392	0.96466625	0.12140568
25	1.40221	1.24111	0.837092	1.4228	1.225803	0.2715743

Processo receptor 3 (R3)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	1.15132	1.17671	1.13726	1.1274	1.1481725	0.02140775
800	1.15129	1.17686	1.15234	1.14784	1.1570825	0.01332438
400	1.26677	1.31638	1.29786	1.25928	1.2850725	0.02673272
200	1.16111	1.1694	1.24176	1.12252	1.1736975	0.04976128
100	1.40912	1.10577	1.55945	1.41112	1.371365	0.19054533
50	1.54411	1.43536	1.67351	1.52592	1.544725	0.09815003
25	1.39209	2.19307	2.35156	1.8786	1.95383	0.42294062

Processo receptor 4 (R4)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.695959	0.666588	0.684836	0.675435	0.6807045	0.01260704
800	0.666344	0.672954	0.671747	0.673105	0.6710375	0.00318747
400	0.653174	0.66589	0.678418	0.661047	0.66463225	0.01057945
200	0.684448	0.688351	0.684071	0.939875	0.74918625	0.12714056
100	1.10985	0.921154	0.97346	0.831216	0.95892	0.11651303
50	1.31003	1.19469	1.50021	1.29972	1.3261625	0.12719672
25	1.61757	1.21932	1.18279	1.21073	1.3076025	0.20723263

Processo receptor 5 (R5)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.953397	0.924023	0.876276	0.896229	0.91248125	0.03357711
800	0.863458	0.867819	0.859963	0.8564	0.86191	0.00488069
400	0.831944	0.816817	0.868114	0.861232	0.84452675	0.02423207
200	0.979563	0.919179	0.871461	0.907788	0.91949775	0.0449169
100	1.08572	1.15868	1.38873	1.21858	1.2129275	0.12918076
50	1.40798	1.43717	1.71797	1.61775	1.5452175	0.14788823
25	1.108	1.45859	1.92874	2.34098	1.7090775	0.53900851



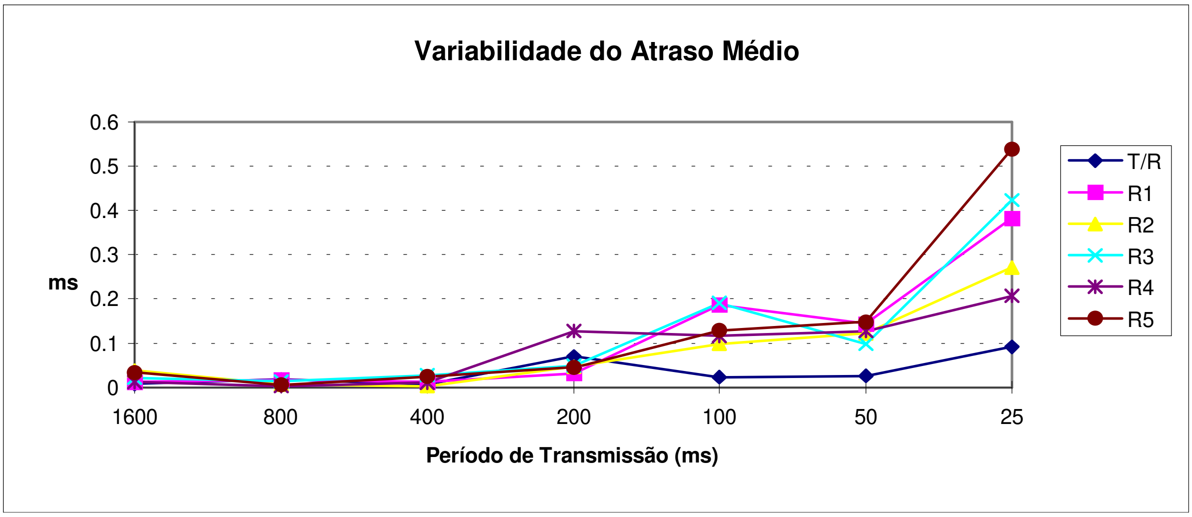
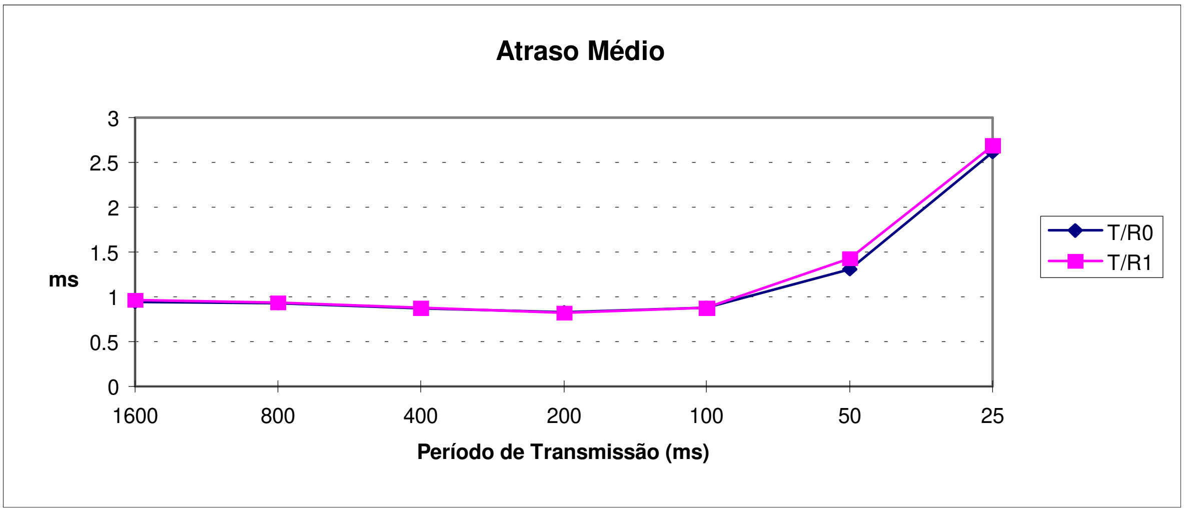


Figura B6: Dados relativos ao atraso médio para um grupo completamente ativo de 2 processos.

Processo transmissor/receptor 0 (T/R0)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.986434	0.911809	0.926015	0.93449	0.939687	0.03253919
800	0.936404	0.946294	0.914697	0.914432	0.92795675	0.01598281
400	0.919781	0.924548	0.828797	0.807184	0.8700775	0.0608198
200	0.815102	0.876428	0.865228	0.76075	0.829377	0.05295448
100	0.881514	0.882169	0.889282	0.857588	0.87763825	0.01382195
50	1.5199	1.18626	1.27287	1.26257	1.3104	0.14491055
25	2.54757	2.43428	2.83921	2.65388	2.618735	0.17217482

Processo transmissor/receptor 1 (T/R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.932124	1.09862	0.928273	0.901381	0.9650995	0.09005803
800	0.929184	1.05462	0.879703	0.876333	0.93496	0.08335134
400	0.923833	0.90376	0.849076	0.839112	0.87894525	0.04126961
200	0.798626	0.766783	0.794295	0.93431	0.8235035	0.07520488
100	0.884241	0.861473	0.902666	0.862464	0.877711	0.01967686
50	1.3269	1.51581	1.36757	1.51422	1.431125	0.09828261
25	2.84207	2.80499	2.56594	2.53932	2.68808	0.15751041


Atraso Médio
3
2.5
2
1.5
1
0.5
0
ms
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R0
T/R1

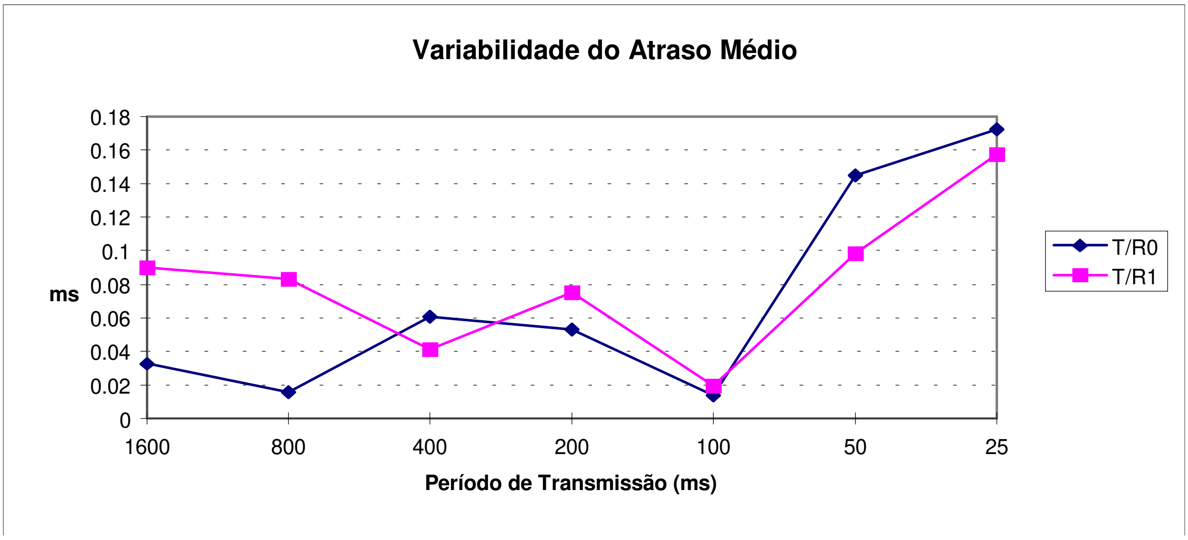
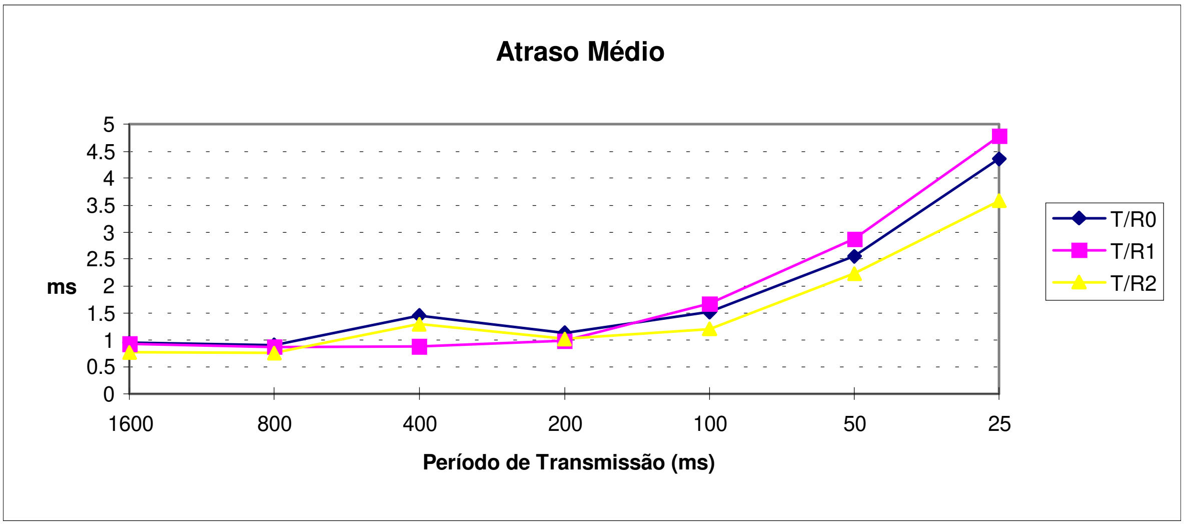

Variabilidade do Atraso Médio
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
ms
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R0
T/R1

Figura B7: Dados relativos ao atraso médio para um grupo completamente ativo de três processos.

Processo transmissor/receptor 0 (T/R0)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.905789	0.969642	0.944011	0.957421	0.94421575	0.02767387
800	0.884282	0.959123	0.855415	0.908134	0.9017385	0.04391082
400	0.83288	1.08614	0.859024	3.03787	1.4539785	1.06203447
200	1.27799	1.04223	1.09666	1.09302	1.127475	0.10337341
100	1.58388	1.44912	1.4675	1.5785	1.51975	0.07137432
50	2.79444	2.46128	2.43092	2.518	2.55116	0.16615336
25	4.33197	3.77978	5.24165	4.07546	4.357215	0.63131599

Processo transmissor/receptor 1 (T/R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.959198	0.933534	0.943644	0.877189	0.92839125	0.03572964
800	0.861718	0.848956	0.943812	0.806489	0.86524375	0.05745379
400	0.951052	0.815887	0.895968	0.87889	0.88544925	0.05566655
200	0.989998	0.971954	0.928121	1.03895	0.98225575	0.0458653
100	1.60636	1.61019	1.53326	1.91173	1.665385	0.1680013
50	2.98851	2.88072	2.69261	2.91391	2.8689375	0.12589721
25	4.85864	5.21745	4.38551	4.68996	4.78789	0.34690407

Processo transmissor/receptor 2 (T/R2)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.775627	0.792641	0.732875	0.750136	0.76281975	0.02652608
800	0.806734	0.720374	0.725554	0.778157	0.75770475	0.04183068
400	0.732238	0.709318	2.04246	1.67766	1.290419	0.67447897
200	0.840892	0.973902	1.47735	0.796347	1.02212275	0.31271732
100	1.28368	1.2926	1.08519	1.14017	1.20041	0.1038226
50	2.46216	2.14898	2.14444	2.1665	2.23052	0.1547193
25	4.10331	3.7341	3.37247	3.11266	3.580635	0.43169199


Atraso Médio
ms
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R0
T/R1
T/R2

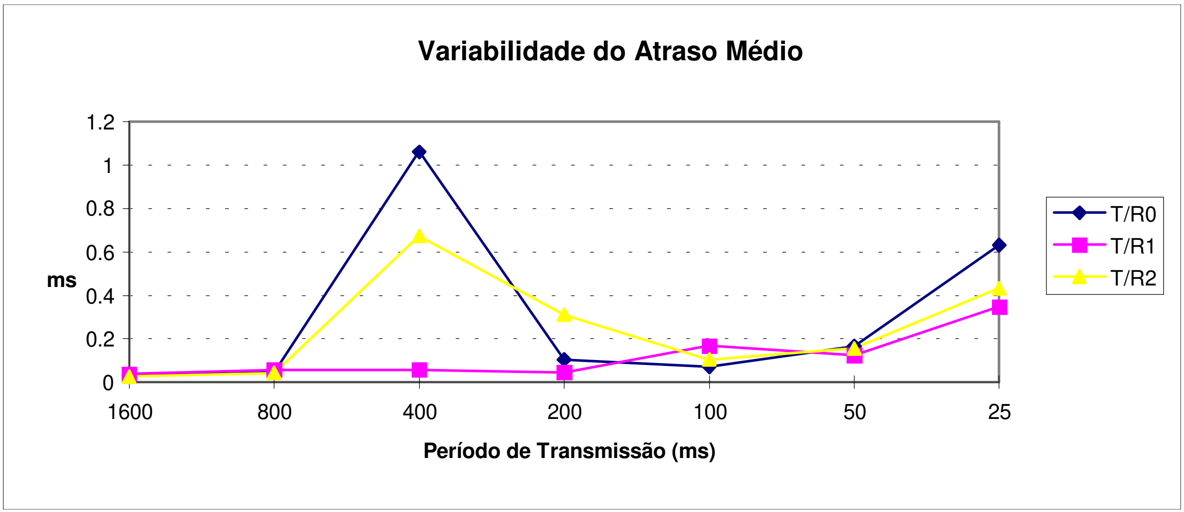

Variabilidade do Atraso Médio
ms
1.2
1
0.8
0.6
0.4
0.2
0
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R0
T/R1
T/R2

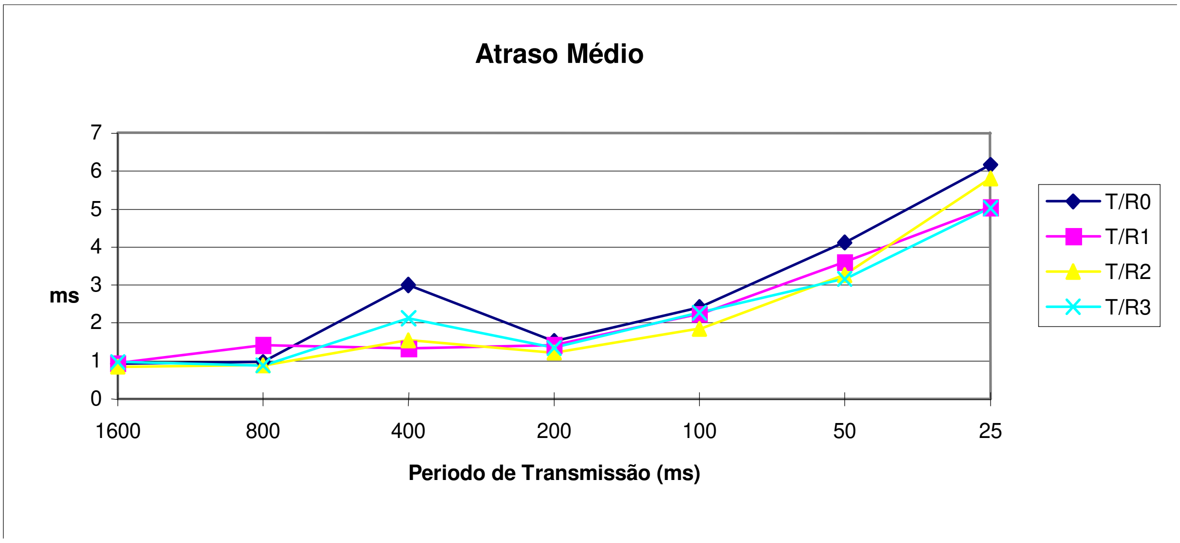
Figura B8: Dados relativos ao atraso médio para um grupo completamente ativo de quatro processos.

Processo transmissor/receptor 0 (T/R0)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.996237	0.947588	0.870208	0.868625	0.9206645	0.06242341
800	0.948546	0.981301	0.979467	0.972776	0.9705225	0.01510215
400	1.33983	1.10832	5.26973	4.31594	3.008455	2.0990239
200	1.68096	1.20378	1.61638	1.60759	1.5271775	0.21806594
100	2.41257	2.58779	2.40119	2.2635	2.4162625	0.13291461
50	3.84873	4.07097	4.70305	3.85305	4.11895	0.40298741
25	6.06305	5.99775	5.79793	6.78663	6.16134	0.43184759

Processo transmissor/receptor 1 (T/R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.887502	0.990445	0.957279	0.897786	0.933253	0.04898719
800	0.862328	0.883268	1.83682	2.11596	1.424594	0.64732633
400	1.142	0.970924	1.06899	2.16063	1.335636	0.55444415
200	1.31917	1.27081	1.19725	1.90128	1.4221275	0.32334411
100	1.72636	2.39948	2.3315	2.49654	2.23847	0.34805918
50	3.2158	3.85187	3.54079	3.78599	3.5986125	0.28817734
25	4.27334	4.60742	5.8448	5.47206	5.049405	0.73232918

Processo transmissor/receptor 2 (T/R2)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.750205	0.820401	0.79005	1.08249	0.8607865	0.15057152
800	0.94776	0.8992	0.736992	0.94193	0.8814705	0.09872187
400	1.49045	1.18769	1.9985	1.52509	1.5504325	0.33495609
200	1.63971	1.17157	1.01654	1.06589	1.2234275	0.28495662
100	2.02432	1.85996	1.86322	1.62589	1.8433475	0.16402205
50	2.99366	3.66444	3.26763	3.15623	3.27049	0.28571255
25	4.75414	7.12414	5.59314	5.76255	5.8084925	0.98167798

Processo transmissor/receptor 3 (T/R3)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.963205	0.920287	0.949966	1.04009	0.968387	0.05105917
800	0.874142	0.865491	0.887842	0.944928	0.89310075	0.03575589
400	0.888796	0.961955	4.68531	1.97138	2.12686025	1.77572957
200	1.52512	1.34317	1.19618	1.34215	1.351655	0.13469091
100	1.86214	2.28871	2.35709	2.56363	2.2678925	0.29466666
50	2.50893	2.44274	3.98249	3.68543	3.1548975	0.7938968
25	3.10542	5.12265	6.56444	5.30106	5.0233925	1.43066386


Atraso Médio
ms
Periodo de Transmissão (ms)
7
6
5
4
3
2
1
0
1600
800
400
200
100
50
25
T/R0
T/R1
T/R2
T/R3

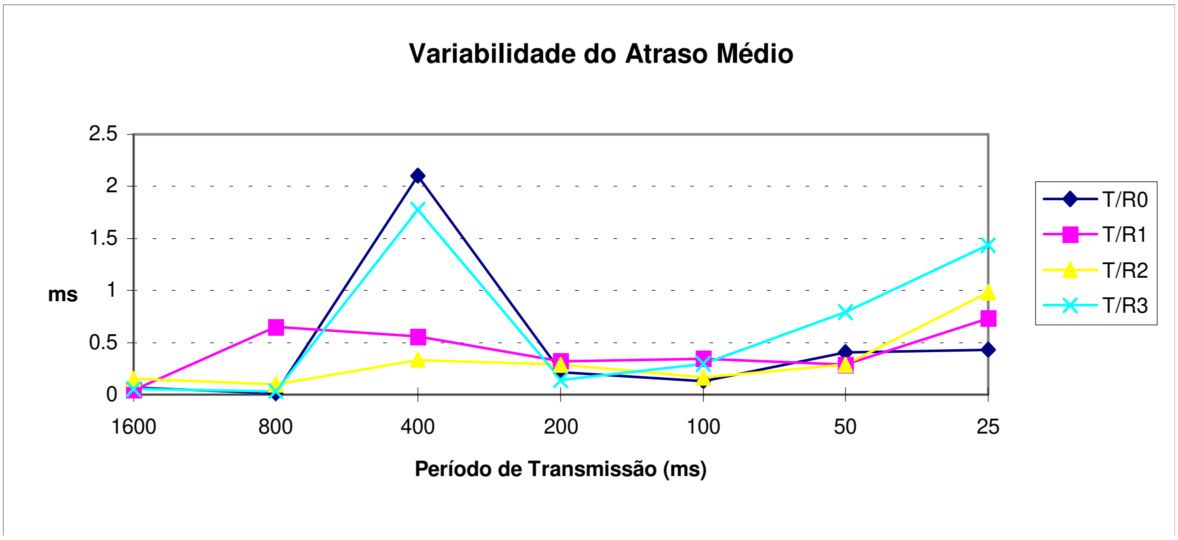

Variabilidade do Atraso Médio
ms
Período de Transmissão (ms)
2.5
2
1.5
1
0.5
0
1600
800
400
200
100
50
25
T/R0
T/R1
T/R2
T/R3

Figura B9: Dados relativos ao atraso médio para um grupo completamente ativo de cinco processos.

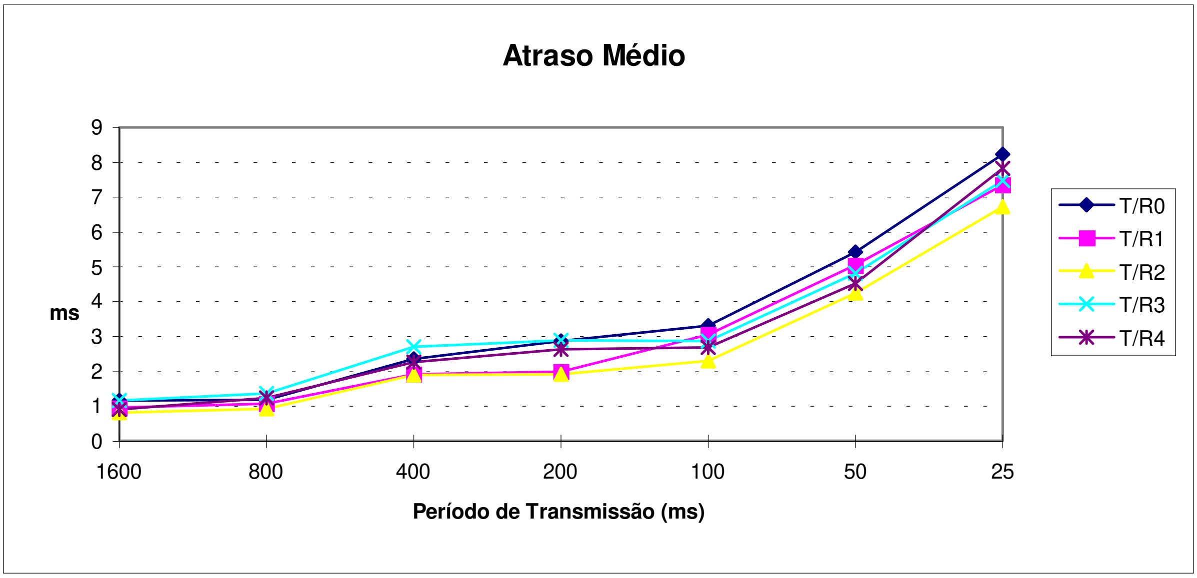
Processo transmissor/receptor 0 (T/R0)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.968565	1.16636	0.872355	1.67838	1.171415	0.35945518
800	1.53689	1.08989	0.90934	1.22635	1.1906175	0.264855
400	2.14916	3.81535	1.51615	1.95153	2.3580475	1.00687842
200	4.76049	1.60006	1.51468	3.59178	2.8667525	1.58582434
100	3.64556	3.48144	3.09381	2.99602	3.3042075	0.30939701
50	6.13169	6.02809	4.67726	4.92698	5.441005	0.74593179
25	9.92139	8.40445	7.26866	7.36048	8.238745	1.23439144

Processo transmissor/receptor 1 (T/R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.963841	0.948704	0.960675	0.905471	0.94467275	0.02693524
800	0.950723	1.18347	0.835684	1.36809	1.08449175	0.23806444
400	1.94259	1.02458	3.70067	1.03568	1.92588	1.25896207
200	1.60949	2.51184	2.22785	1.66273	2.0029775	0.43973835
100	2.99261	3.46643	3.13869	2.62493	3.055665	0.34886942
50	5.03424	5.63253	4.85725	4.66786	5.04797	0.41743528
25	7.64333	7.94071	7.05477	6.76438	7.3507975	0.53699466

Processo transmissor/receptor 2 (T/R2)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.779208	0.780605	0.761366	0.972864	0.82351075	0.09995332
800	0.893876	0.895013	0.83728	1.13333	0.93987475	0.13175619
400	1.34208	0.974297	3.8805	1.39425	1.89778175	1.33495873
200	1.15619	1.2288	2.16505	3.16569	1.9289325	0.94386388
100	2.42611	2.30687	2.05528	2.40002	2.29707	0.16912369
50	4.30674	4.92377	3.92519	3.85338	4.25227	0.48988903
25	6.41575	7.67376	7.03385	5.80846	6.732955	0.80228047

Processo transmissor/receptor 3 (T/R3)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.954109	0.970908	0.977108	1.78956	1.17292125	0.4112073
800	0.935268	1.47232	1.00311	2.07355	1.371062	0.52568928
400	1.44563	3.01487	2.15085	4.21317	2.70613	1.19215297
200	3.57058	1.59936	3.58487	2.81826	2.8932675	0.93396806
100	2.51205	2.84567	3.01523	3.14099	2.878485	0.27261664
50	4.38076	4.33973	5.59605	4.90768	4.806055	0.58672952
25	7.75296	6.05223	8.11192	8.01779	7.483725	0.96635394

Processo transmissor/receptor 4 (T/R4)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.972571	0.88147	0.868505	0.961426	0.920993	0.05357915
800	1.13437	1.0563	1.23552	1.57304	1.2498075	0.22763593
400	1.35416	2.54107	2.46277	2.73157	2.2723925	0.62247519
200	4.21779	1.90726	2.51531	1.92672	2.64177	1.08790818
100	2.60412	2.57073	3.11021	2.47103	2.6890225	0.28642695
50	4.99443	3.64877	4.91268	4.49813	4.5135025	0.61607157
25	8.05011	6.20005	7.80478	9.28965	7.8361475	1.26967823


Atraso Médio
ms
9
8
7
6
5
4
3
2
1
0
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R0
T/R1
T/R2
T/R3
T/R4

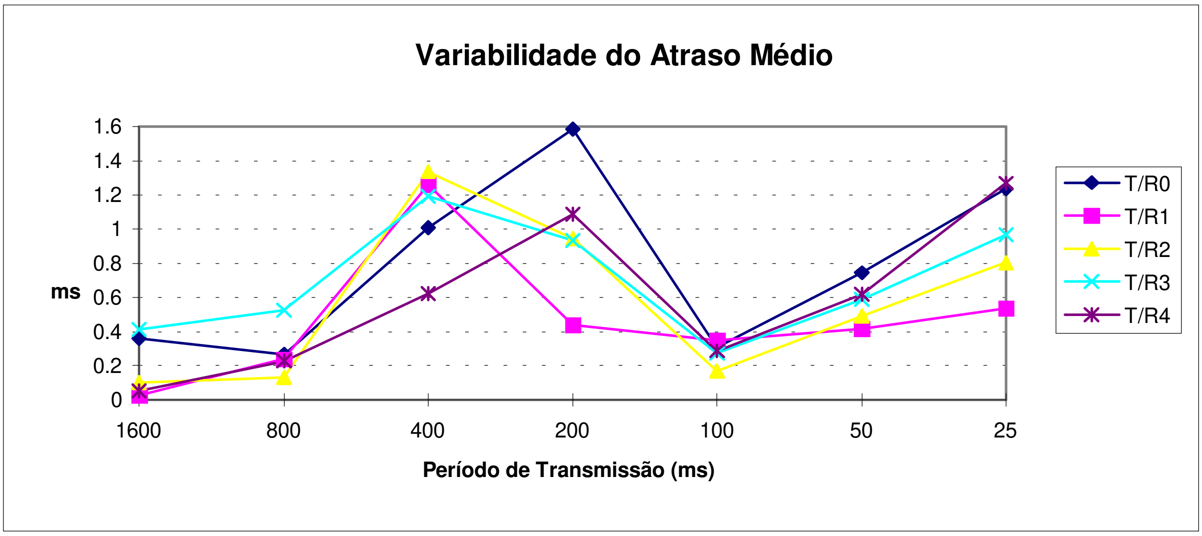

Variabilidade do Atraso Médio
ms
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1600
800
400
200
100
50
25
Período de Transmissão (ms)
T/R0
T/R1
T/R2
T/R3
T/R4

Figura B10: Dados relativos ao atraso médio para um grupo completamente ativo de seis processos.

Processo transmissor/receptor 0 (T/R0)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.915913	1.29418	0.930308	0.961864	1.02556625	0.18010117
800	1.2472	1.10365	1.98262	0.966962	1.325108	0.45302829
400	2.08745	1.77129	5.0022	3.49613	3.0892675	1.4793637
200	2.47973	2.75365	2.37611	7.28327	3.72319	2.37872478
100	3.14306	4.3582	3.12266	4.89322	3.879285	0.88918172
50	3.33313	4.98316	5.92308	3.63396	4.4683325	1.20639958
25	5.31414	7.89035	9.90968	7.90126	7.7538575	1.88327285

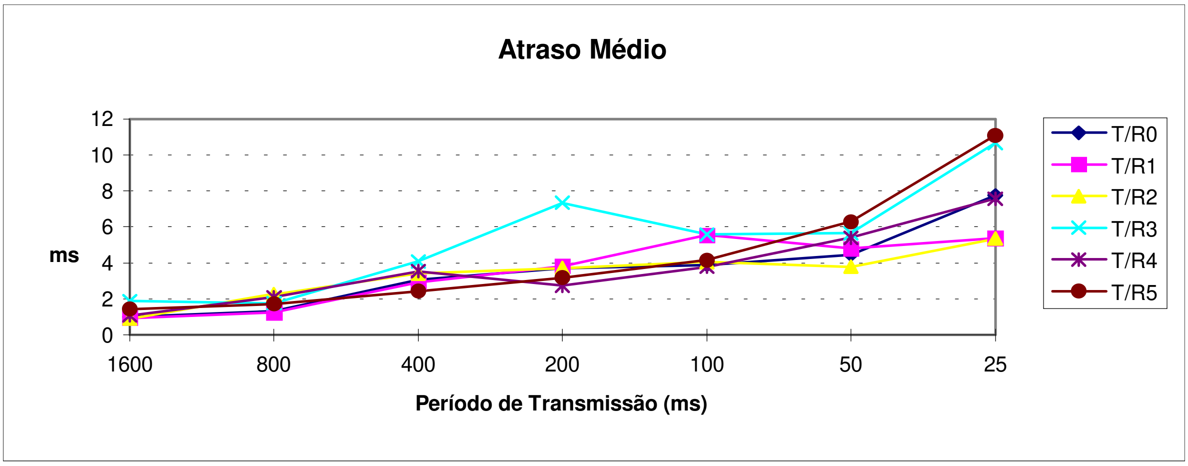
Processo transmissor/receptor 1 (T/R1)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	1.04901	0.846216	0.919105	0.933042	0.93684325	0.0839122
800	1.29223	1.23827	1.3188	1.11276	1.240515	0.09152297
400	1.63172	2.81455	2.12885	5.24491	2.9550075	1.60177067
200	2.96122	3.42083	4.44338	4.47891	3.826085	0.75706818
100	4.29443	5.27319	5.74356	6.91488	5.556515	1.08831259
50	3.56958	5.11168	3.58971	6.92444	4.7988525	1.59050513
25	3.40092	7.00474	4.28672	6.76017	5.3631375	1.79402069

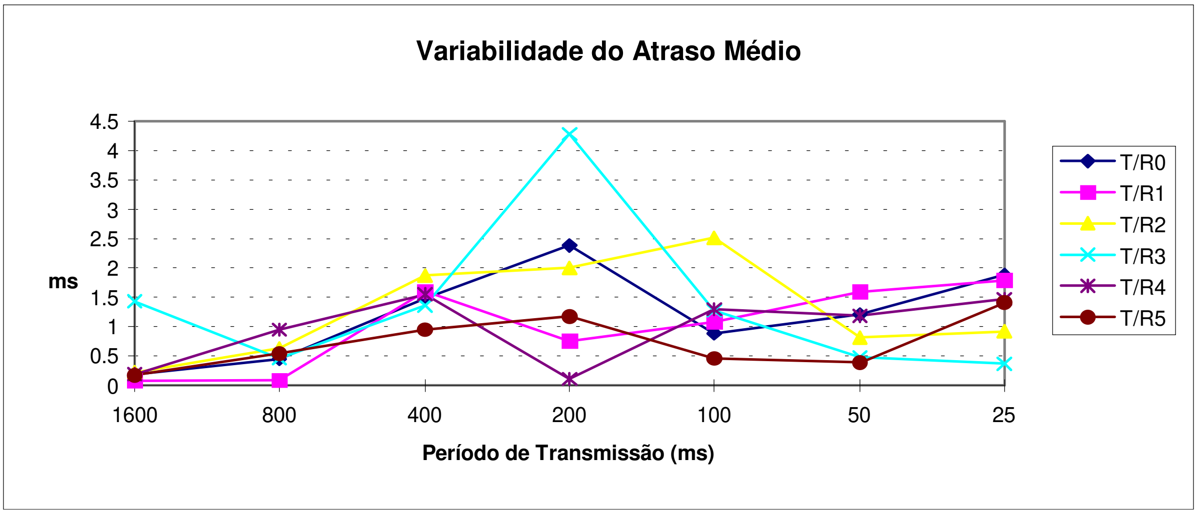
Processo transmissor/receptor 2 (T/R2)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.917951	1.24534	0.7468	0.806732	0.92920575	0.22236547
800	2.21816	1.58154	2.15403	3.12183	2.26889	0.63658543
400	4.11424	2.01122	1.74649	5.71878	3.3976825	1.87524782
200	5.17725	5.6705	2.23241	1.75289	3.7082625	2.0008267
100	2.94448	7.81648	2.64109	2.77692	4.0447425	2.51755143
50	3.45895	4.965	3.11815	3.53933	3.7703575	0.81708703
25	4.94376	6.7237	4.76078	4.96209	5.3475825	0.92190268

Processo transmissor/receptor 3 (T/R3)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	1.17446	1.1909	4.0431	1.19069	1.8997875	1.42889575
800	2.30497	1.58713	1.24984	1.95337	1.7738275	0.45598491
400	3.52151	2.51121	4.67302	5.63101	4.0841875	1.35772454
200	4.20992	3.10242	10.9192	11.1477	7.34481	4.28422083
100	4.03719	5.88525	5.27878	7.1211	5.58058	1.28311621
50	5.12425	5.44268	6.25117	5.78027	5.6495925	0.48227572
25	10.4979	10.4486	11.1917	10.4478	10.6465	0.36422113

Processo transmissor/receptor 4 (T/R4)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	0.811959	1.27645	1.07849	1.10716	1.06851475	0.19205111
800	1.19016	1.56154	3.36245	2.23171	2.086465	0.95361466
400	3.13633	2.24726	2.98314	5.80919	3.54398	1.55920697
200	2.72839	2.91505	2.63407	2.74094	2.7546125	0.1171114
100	2.81357	3.41556	5.69364	3.19633	3.779775	1.29993487
50	3.72451	5.99934	6.41425	5.55036	5.422115	1.18544367
25	5.83359	6.97347	9.17402	8.29389	7.5687425	1.4683163

Processo transmissor/receptor 5 (T/R5)						
Transmissão	Execução1	Execução2	Execução3	Execução4	Atraso	Desvio Pad.
1600	1.56701	1.61339	1.24276	1.35277	1.4439825	0.17573169
800	2.49048	1.47864	1.74931	1.23081	1.73731	0.5449347
400	2.59593	1.75464	1.67226	3.7245	2.4368325	0.95452559
200	2.65585	2.45449	2.71157	4.939	3.1902275	1.17106607
100	4.40605	4.12222	3.50441	4.54392	4.14415	0.46121429
50	6.17303	5.86559	6.24341	6.79967	6.270425	0.38910463
25	11.8849	12.6445	9.73959	10.0374	11.0765975	1.41175915





Apêndice C

Este apêndice apresenta os dados relativos à influência do canal de comunicação no protocolo BCGcausal. O período de transmissão está fixado em 100 ms.

Fgura C1: Dados de três execuções do experimento para avaliar a influência da velocidade do canal na entrega de mensagens em ordem causal.

Atr. do Canal	Espera na BM				Cadeia Causal			
	Execução1	Execução2	Execução3	Média	Execução1	Execução2	Execução3	Média
0	0.14899	0.28086	0.173252	0.201034	0.0709291	0.182318	0.0769231	0.1100567
25	0.511344	0.684286	4.61673	1.937453333	0.0174825	0.0244755	0.183317	0.0750917
50	4.7716	8.47574	10.9136	8.053646667	0.0949051	0.167333	0.21978	0.1606727
75	14.4675	15.0223	17.3515	15.61376667	0.195305	0.200799	0.232767	0.2096237
100	43.4634	38.547	43.2827	41.76436667	0.848152	0.584915	0.734765	0.7226107
125	6171.22	5230.68	6451.28	5951.06	98.3322	85.9356	102.15	95.4726
150	12318.3	12328.7	12203.9	12283.63333	164.138	164.047	162.476	163.55367
175	16807.2	16739	16743.8	16763.33333	200.926	199.902	199.929	200.25233
200	20820.5	20850.6	24595.5	22088.86667	225.488	226.368	245.198	232.35133

Apêndice D

Este apêndice apresenta os dados relativos à *round trip* referente a grupos de tamanho 2, 3, 4, 5 e 6 processos. Os experimentos foram repetidos 10 vezes.

Fgura D1: Dados de dez execuções do experimento para avaliar o *roud trip* do protocolo BCGcausal e da camada de transporte.

Tamanho do Grupo									
0 2		0 3		0 4		0 5		0 6	
Causal	Transporte	Causal	Transporte	Causal	Transporte	Causal	Transporte	Causal	Transporte
7.68599	3.02585	9.71258	4.83037	12.0707	5.31925	14.1812	6.50127	19.772	12.041
7.99312	3.02293	9.1362	5.05895	12.1076	5.67276	14.153	14.3525	21.3912	13.5145
7.25977	3.29967	9.16212	4.67112	11.547	8.26475	14.0113	6.51149	22.7026	11.0688
7.78599	3.00431	9.0276	4.97465	11.8731	5.04716	14.5779	9.64402	22.9298	12.0785
7.6324	3.35023	9.75796	5.39537	11.9811	6.00027	14.5703	7.35487	23.2208	12.0197
7.60526	3.07499	9.13768	4.60747	11.9369	5.42726	13.9917	6.71032	20.4405	11.4929
7.4376	2.79046	9.23467	4.73395	12.1823	5.79581	14.1577	8.77512	20.7972	11.4798
7.62975	2.78418	9.04316	5.08639	11.89	5.53789	13.4053	6.40841	21.9387	13.2715
7.28179	2.82334	9.27278	4.90942	11.7916	4.76714	13.8396	6.22902	18.8476	13.8084
7.3161	2.74194	9.26886	4.33895	11.8701	5.53461	13.9468	6.33423	20.7726	10.4317