Avaliação de Desempenho de Protocolos de Ordenação Causal para Comunicação em Grupo*

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

Raimundo José de Araújo Macêdo[++]
macedo@ufba.br

Universidade Federal da Bahia
Laboratório de Sistemas Distribuídos - LaSiD
Prédio do CPD-UFBA, Campus de Ondina, 40.170-110 Salvador, BA, Brasil

Resumo

Para serem eficientes, sistemas distribuídos de larga escala, como os encontrados na Internet, requerem o uso de protocolos de difusão (ao invés da comunicação convencional ponto-a-ponto). Comunicação em Grupo e protocolos de ordenação Causal são mecanismos fundamentais na construção de tais sistemas. No passado, muitos pesquisadores têm devotado suas atenções para o desenvolvimento de protocolos de Comunicação em Grupo com características variadas de ordenação de mensagens e confiabilidade. No entanto, pouquíssimos trabalhos têm se preocupado em descrever experiências de avaliação desses protocolos em ambiente real, e em especial para os de ordenação causal. Neste artigo apresentamos uma metodologia para avaliar o comportamento de tais protocolos sob circunstâncias variadas e aplicamos essa metodologia para avaliar o protocolo BCGcausal, implementado por nós 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

To be efficient, large scale distributed systems such as those present in the Internet, require the use of diffusion protocols (instead of the usual point-to-point communication). Group Communication and causal order protocols are fundamental mechanisms for the construction of such systems. In the past, many researchers have devoted their attention to the development of group communication (multicast) protocols with distinct characteristics of message ordering and reliability. However, only very few works have described experiences on evaluating such protocols on its real environment, in particular for causal order communication. In this paper we present a methodology to analyse the behaviour of such protocols under varied cirscuntances and apply our methodology to evalute the protocol BCGcausal developed by us at LaSiD/UFBA, over a network of Unix workstations. After discussing the methodology proposed, we present the data for the throughput, average delay, and round trip of our protocol against several group configurations (number of member processes) and process activities (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.

Palavras chaves: comunicação na Internet, protocolos de comunicação em grupo, ordenação causal, avaliação de protocolos de comunicação.

---

* Este trabalho recebeu apoio financeiro do CNPq
[+] Pesquisador bolsista DTI-F/CNPq/ProTeM-CC fase III (Projeto Locus), número 38009697-8.
[++] Pesquisador II do CNPq, bolsa PQ número 300013/87-6 (projeto BCG)

1. Introdução

Para serem eficientes, sistemas distribuídos de larga escala, como os encontrados na Internet, requerem o uso de protocolos de difusão (ao invés da comunicação convencional ponto-a-ponto). Um outro aspecto importante é a confiabilidade, visto a nossa dependência crescente desses sistemas. Neste sentido, a abstração de *comunicação em grupo* vem sendo largamente usada como um modelo de comunicação entre processos distribuídos e fortemente recomendada por diversos autores [MR93, Mac94, EMS94, PBS89, BSS91, PRV93, CV95]. Em tal paradigma os processos são agrupados em conjuntos, chamados de grupos e toda a comunicação envolve todos os elementos do grupo. Banco de dados replicados e CSCW são exemplos de aplicações que se beneficiam com tal paradigma..

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 (relação efeito-causa [Lam78]) entre as mensagens quanto o conceito de grupos. 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, pois m2 possui *dependência causal* em relação a m1. 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].

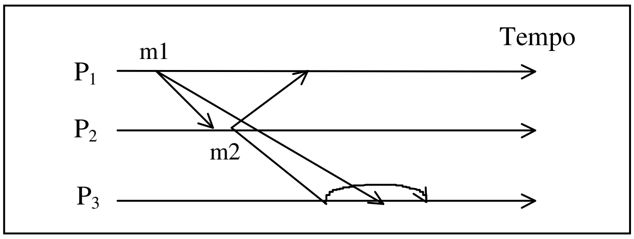

Figura 1: Entrega em ordem causal em comunicação em grupo.

No passado, muitos pesquisadores têm devotado suas atenções para o desenvolvimento de protocolos de comunicação em grupo com características variadas de ordenação de mensagens e confiabilidade. No entanto, pouquíssimos trabalhos têm se preocupado em descrever experiências de avaliação desses protocolos em ambiente real.

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, 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.

Esse trabalho apresenta uma metodologia, baseada em experimentação em ambiente real, para avaliação de protocolos de ordenação. Como estudo de caso aplicamos essa metodologia para avaliar o protocolo BCGcausal, 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. Vale ressaltar que 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.

A fim de avaliar o desempenho do protocolo BCGcausal estabelecemos 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 observando-o sob várias situações relativas ao nível de atividade e configuração do grupo.

O texto estrutura-se da seguinte forma. A próxima seção apresenta a metodologia proposta A seção 3 descreve a plataforma BCG e o protocolo BCGcausal, objeto de nossa avaliação. A seção 4 descreve os experimentos e apresenta os resultados obtidos. A seção 5 avalia os resultados obtidos e por fim, a seção 6 conclui este trabalho.

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*[1] 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.

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. Essas 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*. 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.

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*[2] do protocolo em relação ao atraso médio, *throughput* e *round trip*. Por outro lado, analisamos também o comportamento do protocolo em função da atividade de transmissão dos processos membros de um grupo, ou seja, analisamos o atraso médio e *throughput* para vários *períodos de transmissão*[3].

Os experimentos responsáveis pelas medições dessas métricas serão descritos, em detalhes, na seção em que apresentaremos e discutiremos os resultados obtidos na avaliação do protocolo BCGcausal. Na seção seguinte descreveremos resumidamente o protocolo BCGcausal.

---

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

[2]Do inglês *escalability.*

[3]Denominamos *período de transmissão* o intervalo de tempo entre a transmissão de duas mensagens consecutivas.

3. Estudo de Caso

O protocolo BCGcausal faz parte da plataforma BCG (Base Confiável de Comunicação em Grupo) e é um protocolo assíncrono[4] e simétrico[5] que usa vetores de relógio lógico para ordenar as mensagens, preservando a relação causal entre as mesmas e tem propriedades similares ao CBCAST [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].

A BCG, 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. Além do BCGcausal estão implementados um protocolo de ordem total, serviços de *membership* e controle de fluxo. A figura 2 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.

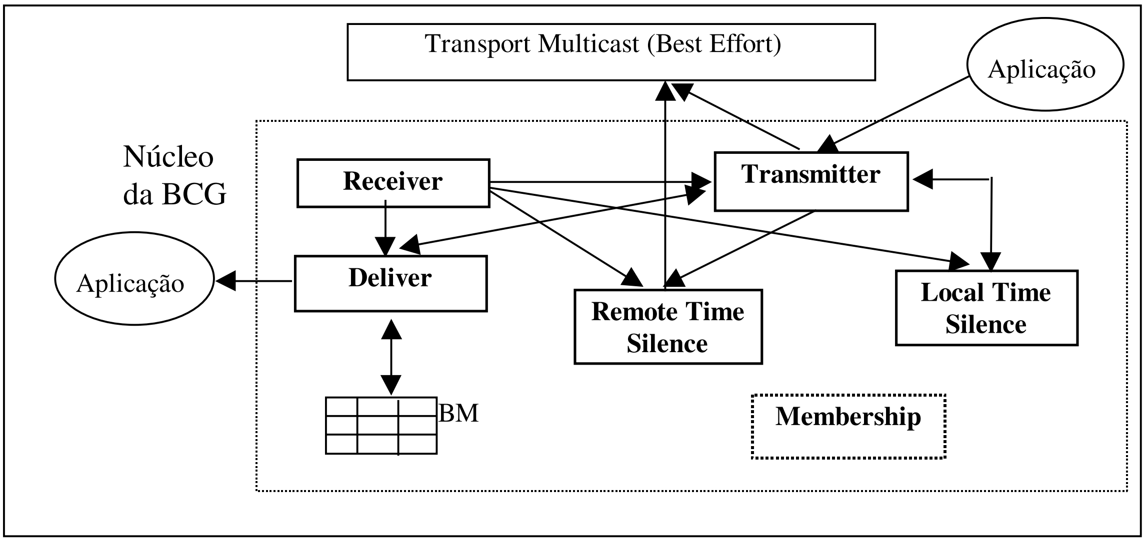

Figura 2: Processos que implementam a BCG.

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:

- **Camada de Transporte *Multicast*.** A camada de transporte realiza o mapeamento entre a comunicação ponto-a-ponto e a comunicação multiponto.
- **Mecanismo de *Timesilence*.** É formado pelo processo *Local Time Silence*, cuja a função é gerar mensagens nulas que são transmitidas para o grupo, num período preestabelecido.

[4] Assume-se que as mensagens transmitidas irão, num tempo finito porém indeterminado, chegar a seus destinos.
[5] O protocolo é dito simétrico quando todos os processos executam o mesmo algoritmo.

Essas mensagens são necessárias para construir mecanismos de detecção de falhas e são importantes para aumentar a eficiência da entrega de mensagens pelo protocolo de ordem total [Mac94].

- **Mecanismo de *Membership*.** Esse mecanismo da BCG, 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 os descreveremos em detalhes, 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[6] de mensagens forem satisfeitas [Mac94, MES95].
- **Matriz de Blocos – BM.** É uma matriz N x M (N: tamanho reservado do *buffer* para cada processo do grupo e; M: tamanho do grupo), onde as mensagens são armazenadas. O processo responsável pela manipulação da BM é o *Deliver.*
- **Ordenação de Mensagens.** a ordenação de mensagens é implementada através da cooperação entre três processos, são eles: o ***Transmitter*,** que recebe as mensagens da aplicação e as envia para a *camada de transporte*; o ***Receiver*** que recebe da *camada de transporte* as mensagens da aplicação e enviando-as para o *Deliver*; e o ***Deliver***, responsável por entregar as mensagens recebidas à aplicação. As mensagens somente poderão ser entregues se não violarem na condição de ordem (causal ou total).

A camada de transporte é independente do protocolo BCGcausal, no entanto influencia nos resultados obtidos dos experimentos. Assim, para permitir uma comparação mais precisa definiremos experimentos sobre o protocolo BCGcausal usando a camada de transporte e experimentos apenas sobre a camada de transporte.

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. Por essa razão, nem todos os experimentos sofrerão a influência do *timesilence*, como veremos nas seções posteriores.

4. Resultados de Desempenho

Explicaremos nesta seção como cada experimento foi montado e apresentaremos os resultados obtidos. Uma versão mais detalhada dos dados encontrados em cada experimento pode ser encontrada em [LM97].

---

[6]Uma mensagem é dita estável por um processo de um grupo se ele sabe que já foram entregues por todos os membros do grupo.

As medições de tempos são geralmente 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 pode causar interferências nas medidas. Avaliamos experimentalmente e estimamos que em nossa plataforma 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 operação a ser medida é repetida). No entanto, nem sempre é possível fazê-lo, por exemplo para avaliar o atraso médio (seção 4.2). Portanto, em tais situações as interferências devido ao acesso ao relógio estão presentes nas medições.

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. Para melhorar o entendimento da aplicação de nossa metodologia, a seguir descreveremos resumidamente a plataforma BCG, seus serviços e o protocolo BCGcausal.

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. As seções seguintes os descrevem separadamente.

4.1. Avaliando o *Throughput*

O algoritmo responsável por esse experimento, escrito em pseudocódigo tipo C++, é ilustrado na figura 3 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 3: 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 4 abaixo ilustra o comportamento do *throughput* em função do período de transmissão. Configurou-se grupos com tamanho variando de 2 a 5, 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[7].

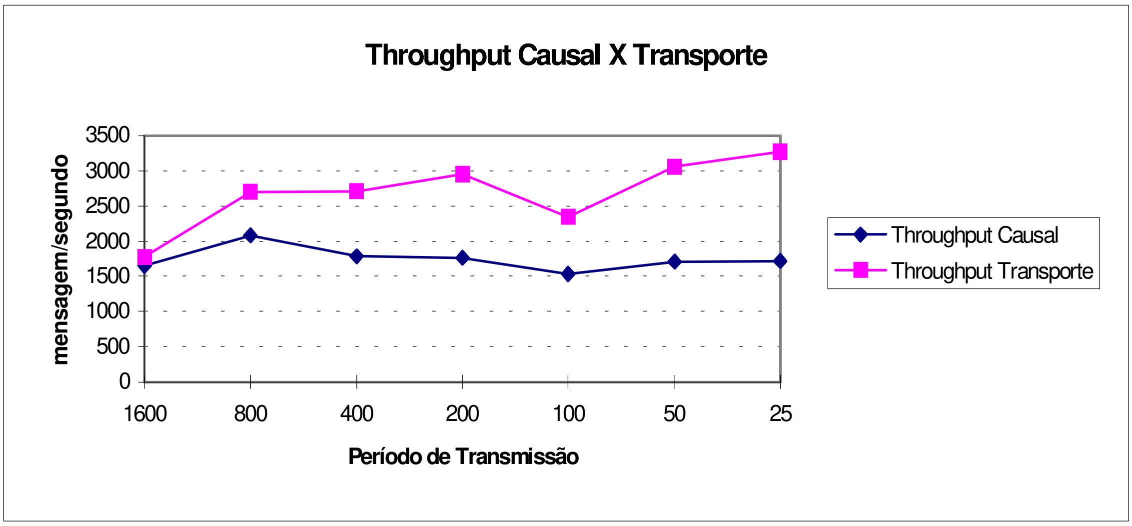

Figura 4: Gráfico do *throughput* em função do período de transmissão.

Observe no gráfico da figura 4 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 3). Ao analisarmos, na próxima seção, o comportamento do atraso médio em função do período de transmissão essa explicação ficará mais clara.

A figura 5, que ilustra a variação do *throughput* em função do tamanho do grupo, 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 3).

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

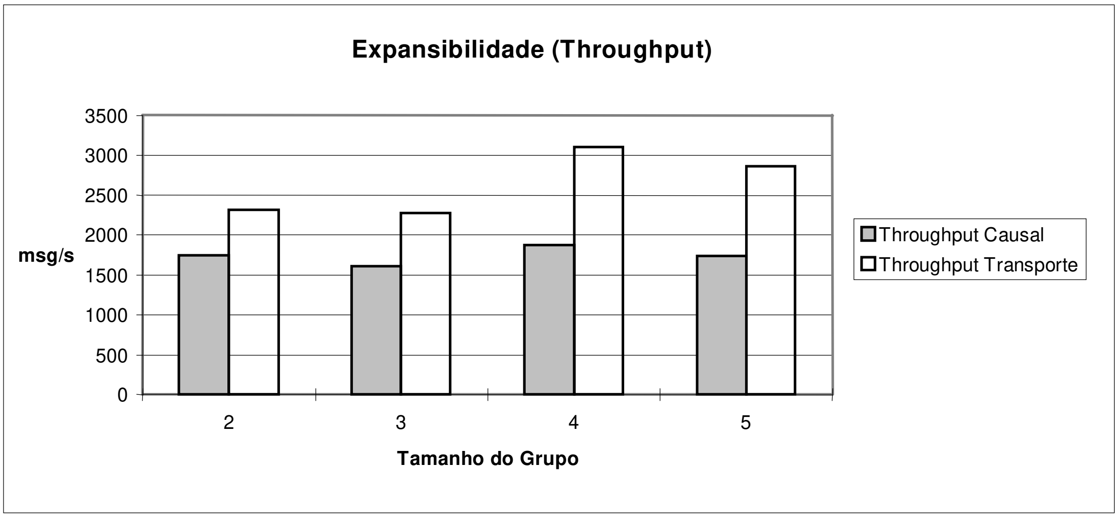

Figura 5: Gráfico do *throughput* em função do tamanho do grupo.

4.2. 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 (o símbolo$\rightarrow$ indica que m é causalmente dependente de m'). 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). 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*, respectivamente), ficam com seus tempos de processamento acrescidos de $2t_\Delta$ (t_Δ em cada um deles), interferindo assim no desempenho do sistema como um todo. Estimamos experimentalmente o valor de t_Δ em 0.25 ms.

Configuração 1 - Apenas um Processo Ativo

A figura 6 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.

Analisando o comportamento ilustrado na figura 6, 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. 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.

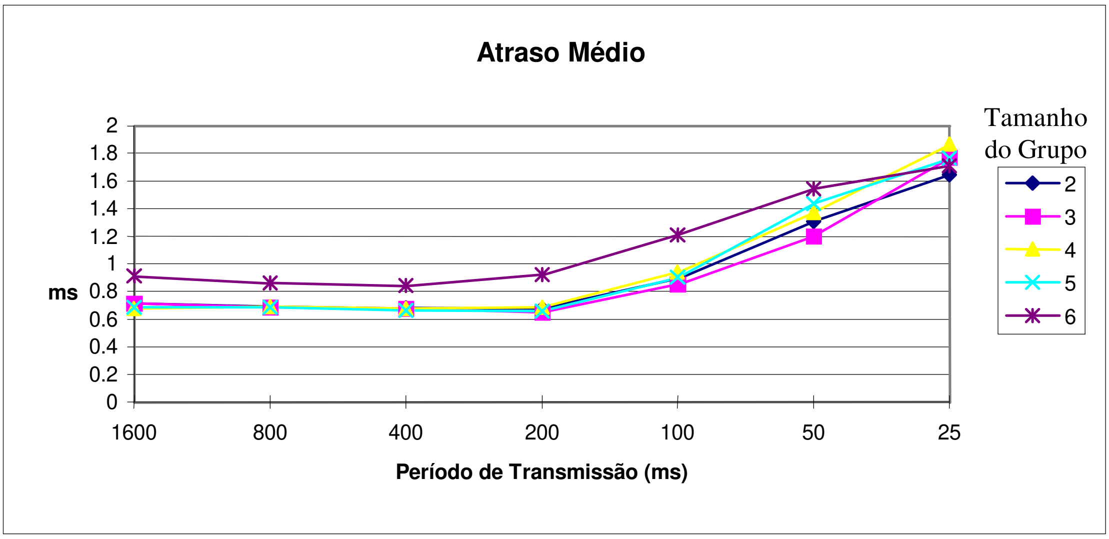

Figura 6: Atraso médio de entrega das mensagens dos processos receptores em função do período de transmissão.

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 7). 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.

Observe que neste experimento o tamanho do grupo influencia diretamente no atraso médio. A explicação para isso é a seguinte. Como o número de mensagens transmitidas 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.

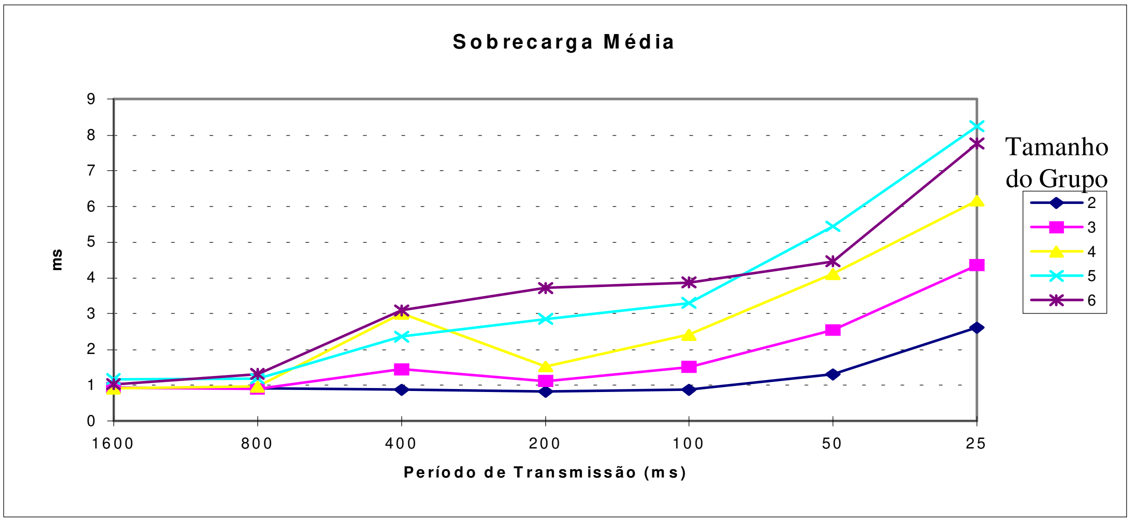

Figura 7: Atraso médio na entrega de mensagens num grupo totalmente ativo.

4.3. Avaliando o *Round Trip*

Esse experimento visa medir o *round trip*, descrito anteriormente (seção 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 8) ilustra o procedimento realizado:

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 8: 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 9 a seguir.

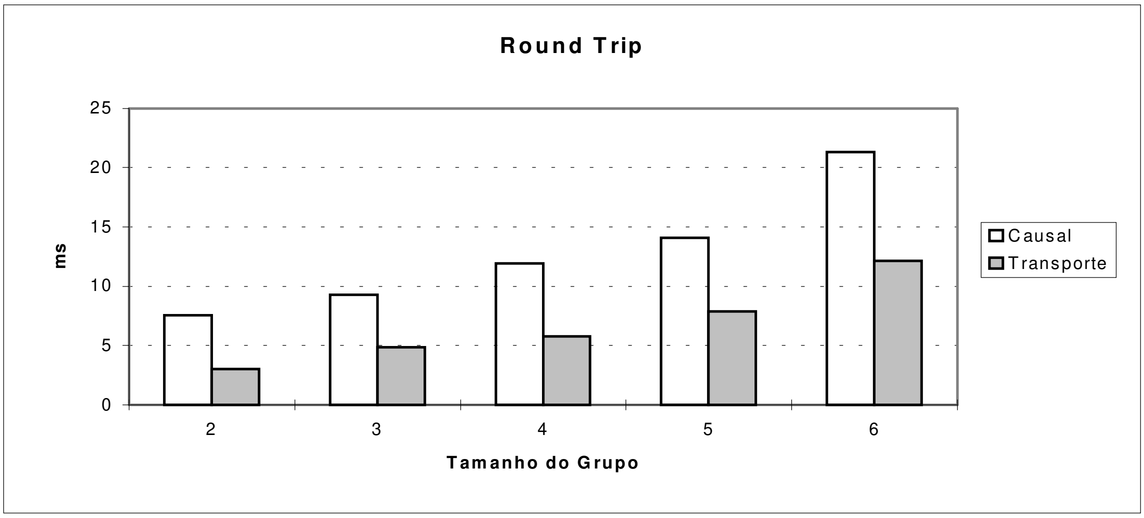

Figura 9: Expansibilidade segundo *round trip*.

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 2 e 3. 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 3).

5. Considerações Sobre os Dados Obtidos na Avaliação do BCGcausal

Podemos analisar os resultados obtidos comparando os dados apresentados aqui com dados conseguidos por outros autores em outros protocolos de comunicação. Entretanto, 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 dos dados. 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. 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].

6. 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 uma metodologia para avaliar protocolos de comunicação em grupo. Tais protocolos são importantes na construção de sistemas distribuídos em larga escala, como os encontrados na Internet, pois esses necessitam de disseminação confiável de mensagens. Aplicamos tal metodologia ao protocolo BCGcausal e assim descrevemos seu comportamento em diversas condições de operação. Descrevemos também a plataforma BCG e seus serviços.

Os experimentos visaram colher dados que pudessem descrever o protocolo sob diversas situações, traçando assim um perfil do sistema. Para tanto, propomos 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.

Relacionamos nosso trabalho aos similares existentes na literatura. Comparando os dados encontrados para o BCGcausal com os do protocolo causal 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.

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 15th 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,.

[LM97] G. M. de A. Lima e R. J. A. Macêdo. *Avaliação de Desempenho do protocolo BCGcausal*. Relatório Técnico Interno do LaSiD, RTI-003/97. Novembro 1997.

[Mac94] Raimundo J. de A. Macêdo. *Fault Tolerant Group Communication Protocols for Asynchronous Systems*. Ph.D. Thesis, Computing Science Department, University of Newcastle upon Tyne, UK, 1994.

[Mac95] Raimundo J. de A. Macêdo. *Causal Order Protocols for Group Communication*. SBRC95, Belo Horizonte-MG, pp. 265-283, maio de 1995.

[MES96] Raimundo J. A. Macêdo e P. D. Ezhilchelvan, S. K. Shrivastava. *Buffer Overflow Avoidance Techniques for Groups Communication Protocols*. SBRC96, pp. 633-652, Fortaleza-CE, maio de 1996.

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

[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.