5
Avaliação experimental

A avaliação experimental foi dividida em duas partes: na primeira comparou-se a utilização da técnica de DHT na comunicação entre os processos com uma comunicação que utiliza conexões sob demanda (uma conexão é aberta para enviar uma mensagem e em seguida é fechada), e na segunda parte foram implementadas aplicações que utilizam as funcionalidades de grupos.

A primeira parte, a avaliação quantitativa, foram medidos o tamanho das rotas, em *hops*, e o tempo de comunicação na troca de mensagens entre processos de uma rede P2P. As medições foram feitas em um ambiente homogêneo (*cluster*) e um ambiente heterogêneo (PlanetLab[5]).

Na parte de aplicações, que é uma avaliação qualitativa, implementou-se uma aplicação que calcula a multiplicação de matrizes de forma distribuída e um esboço de jogo online sem servidor central. A primeira aplicação utiliza mensagens de grupo do tipo *anycast* para distribuir entres os membros de um grupo a multiplicação de matrizes. O jogo *online* enfatiza a cooperação do processos e a utilização de mensagens do tipo *broadcast* para atualizar simultaneamente os membros de um grupo.

5.1
Avaliação quantitativa

No ALua sem a rede sobreposta formada pela técnica de DHT é possível enviar mensagens diretamente entre dois processos, pois como o endereço IP e porta estão contidos nos identificadores dos processos. Basta abrir uma conexão, enviar uma mensagem e fechar a conexão em seguida. Por questões de desempenho, é interessante manter conexões abertas entre os processos para minimizar o custo de estabelecimento de uma conexão, mas neste caso é necessário manter conexões entre *todos* os processos da rede. Já no ALua com a técnica de DHT é mantida uma parcela menor de conexões ativas (ver Subseção 2.4.1), mas as mensagens não são enviadas diretamente entre processos. Assim, nesta avaliação serão comparados os cenários onde processos se comunicam utilizando conexões sob-demanda e utilizando a rede sobreposta formada pela técnica de DHT em um ambiente homogêneo (*cluster*) e outro heterogêneos (PlanetLab). As seguintes métricas foram medidas:

Tempo de comunicação: tempo transcorrido entre um processo P enviar uma mensagem ao processo Q e receber, do processo Q, uma mensagem de

retorno;

Tamanho da rota: tamanho da rota, em *hops*, utilizada na comunicação entre os processos P e Q do item anterior.

O *cluster* utilizado possui 12 máquinas com a seguinte configuração:

- CPU: Intel(R) Core(TM)2 Duo CPU E6750 @ 2.66GHz
- Cache: 4096 KB
- RAM: 1015276 kB
- SO: Linux version 2.6.26.8-57.fc8 (mockbuild@x86-3.fedora.phx.redhat.com)

Na Tabela 5.1 são mostrados os tempos médios de comunicação da solução sob demanda e dos protocolos Chord e Pastry para as rede de 60, 120, 240 e 600 processos. Como esperado, o tempo médio aumenta com o tamanho da rede, pois o tamanho médio da rota também aumenta, como pode ser visto na Tabela 5.2

Tabela 5.1: Tempos médios de comunicação no *cluster*

Protocolo	**N = 60**	**N = 120**	**N = 240**	**N = 600**
Chord	2,80 *ms*	3,11 *ms*	3,48 *ms*	5,40 *ms*
Pastry	2,54 *ms*	3,02 *ms*	4,15 *ms*	7,51 *ms*
Sob demanda	1,80 *ms*	1,80 *ms*	1,80 *ms*	1,80 *ms*

Tabela 5.2: Tamanhos médios das rotas no *cluster*

Protocolo	**N = 60**	**N = 120**	**N = 240**	**N = 600**
Chord	4,24	4,56	5,09	5,78
Pastry	5,63	5,93	6,47	6,76

A Figura 5.1 mostra os tempos médios de comunicação entre processos para rotas de tamanhos entre 3 a 8 *hops* em redes P2P estruturadas formadas pelo protocolo Chord com 60, 120, 240 e 600 processos. Pares de processos foram escolhidos para a medição sem o conhecimento, *a priori*, dos tamanhos das rotas, por isto há casos onde não foram medidas rotas com determinado tamanho, por exempo, para redes com 600 processos não há rotas de 3 *hops*. Na Figura 5.2 pode-se ver a distribuição dos tamanhos das rotas do protocolo Chord também em redes P2P estruturadas formadas pelo protocolo Chord com 60, 120, 240 e 600 processos no *cluster*.

Já a Figura 5.3 mostra o mesmo resultado porém utilizando o protocolo Pastry e para tamanhos de rotas entre 4 e 9. Nas duas figuras é mostrado o

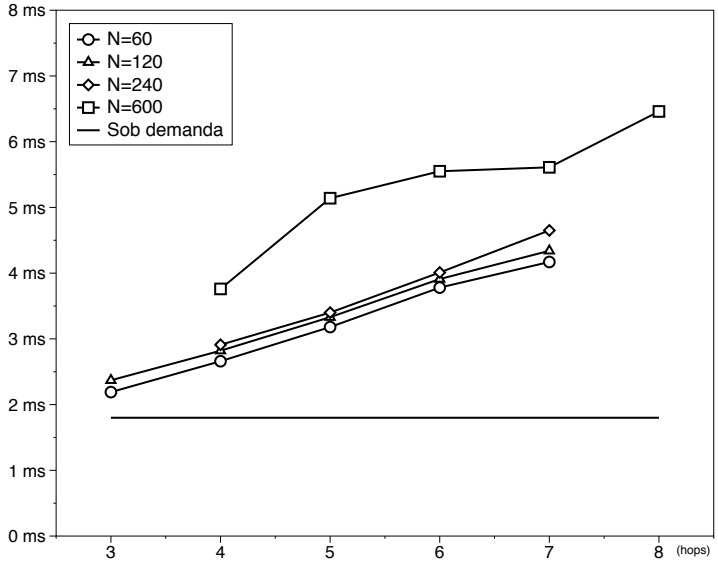

Figura 5.1: Tempo médio de comunicação do protocolo Chord no *cluster*

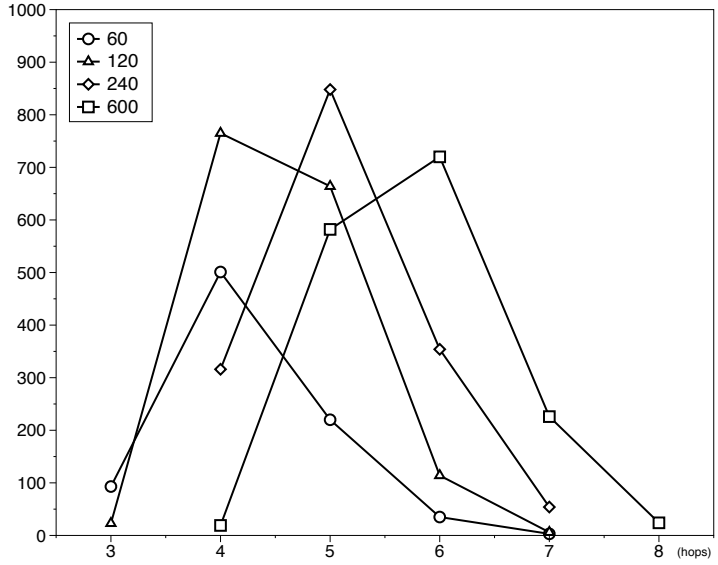

Figura 5.2: Distribuição dos tamanhos das rotas do protocolo Chord no *cluster*

tempo médio de comunicação entre dois processos quando a comunicação é feita sob demanda (com abertura de conexão para o destino) para servir de base na comparação. Na Figura 5.4 pode-se ver a distribuição dos tamanhos das rotas do protocolo Chord também em redes P2P estruturadas formadas pelo protocolo Chord com 60, 120, 240 e 600 processos no *cluster*.

Como pode-se observar pelos resultados, a solução com conexões sob demanda é mais rápida do que a solução utilizando a técnica de DHT, tanto para o protocolo Chord quanto para o protocolo Pastry. Como o custo de estabelecer uma conexão entre máquinas localizadas no mesmo *cluster* é baixo, este resultado é esperado. Porém não é apenas este fato que explica o tempo de

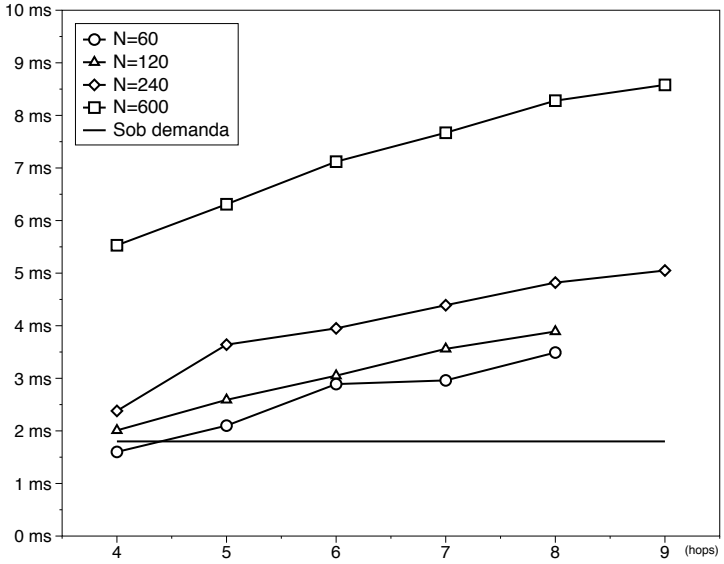

Figura 5.3: Tempo médio de comunicação do protocolo Pastry no *cluster*

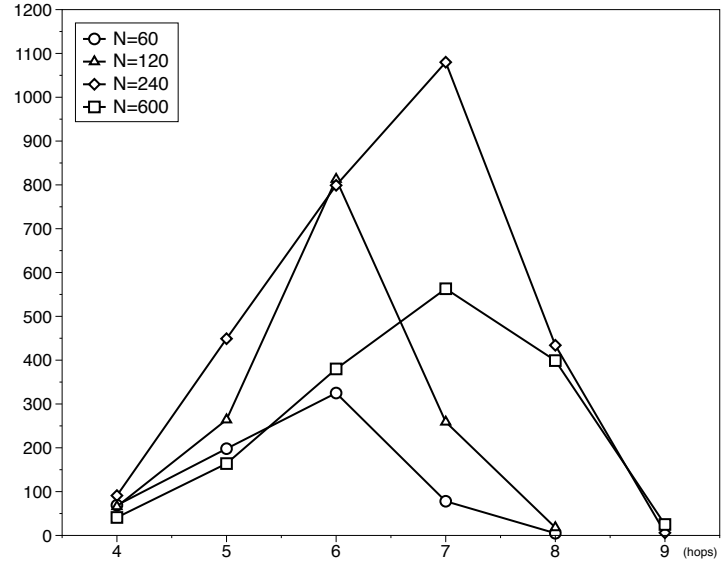

Figura 5.4: Distribuição dos tamanhos das rotas do protocolo Pastry no *cluster*

comunicação ser mais baixo do que utilizando conexões sob demanda. Deve-se levar em conta que, ao utilizar a técnica de DHT, uma mensagem passa por mais de um processo até chegar ao destino. Assim os tempos de *marshalling* e *unmarshalling* de cada processo da rota são adicionados ao tempo de envio de mensagens. O tempo aproximado de *marshalling* e *unmarshalling* é de 0,25 *ms* para ambos protocolos.

Ao mudar de um ambiente de testes homogêneo para um outro heterogêneo, o custo de estabelecimento de conexões aumenta. Na Tabela 5.3 pode-se perceber que este custo maior aumenta o tempo médio de comunicação na solução sob demanda. Pode-se observar também que o protocolo Chord tem

um tempo médio maior que o protocolo Pastry, e isto ocorre porque no protocolo Pastry são escolhidos os vizinhos com menor latência. Na mesma tabela pode-se observar que o tamanho das rotas é maior no protocolo Pastry, porém, mesmo assim, o tempo médio é menor.

Tabela 5.3: Tempos e tamanhos de rotas médios no PlanetLab para N=75

Métrica	**Chord**	**Pastry**	**Sob demanda**
Tempo médio	0,5946 s	0,3048 s	0,4625 s
Tamanho médio da rota	4,8	5,5	1

A Figura 5.5 mostra que o protocolo Pastry possui rotas maiores que o protocolo Chord e seus respectivos tempos médios de comunicação. E na Figura 5.6 está a distribuição dos tamanhos das rotas dos protocolos Chord e Pastry no PlanetLab.

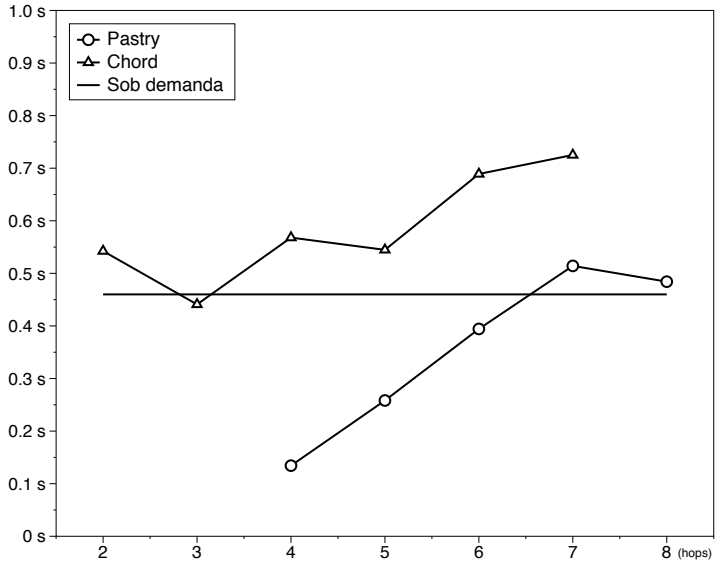

Figura 5.5: Tempo médio de comunicação no PlanetLab para N=75

O protocolo Pastry mostra-se superior em ambientes heterogêneos, já o protocolo Chord se destaca em ambientes homogêneos. Do ponto de vista de desempenho, claramente a solução sob demanda é melhor para ambientes homogêneos, porém ao utilizá-la, os benefícios da rede P2P (replicação de dados e suporte a grupos) não estão disponíveis.

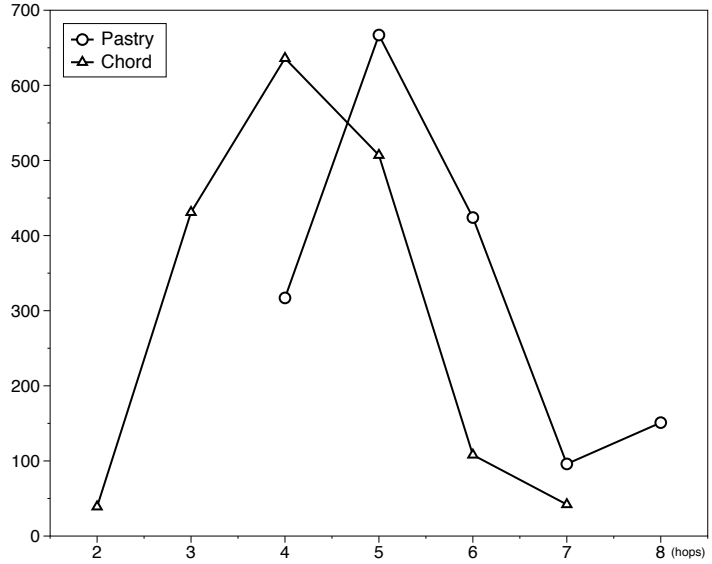

Figura 5.6: Distribuição dos tamanhos das rotas no PlanetLab para N=75

5.2 Avaliação qualitativa

5.2.1 Multiplicação de matrizes

A multiplicação de matrizes é um típico problema que pode ser subdividido em problemas menores e estes resolvidos em paralelo, pois não há interdependência entre os resultados dos sub-problemas. Na solução aqui apresentada, existe um processo *Master* que lê as matrizes A e B e as armazena na rede P2P usando a tabela *hash* distribuida. Os processos *Workers* são os responsáveis por calcular a multiplicação e eles formam um grupo. O *Master* envia uma mensagem do tipo *anycast* ao grupo contendo uma linha da matriz C resultante para que esse a calcule. Essa mensagem é encaminhada a um processo *Worker* que busca na rede P2P a linha da matriz A e a matriz B, e calcula o valor daquela linha na matriz C. Em seguida envia o resultado ao processo *Master*. Quando todas as linhas de C forem calculadas, o processo *Master* tem o resultado da multiplicação.

Com esta solução pode-se avaliar a utilização de grupo de processos como um mecanismo de paralelização de tarefas e também medir o *speed up* de algoritmos que utilizem esta funcionalidade.

Tabela 5.4: Tempos médios da multiplicação de matrizes

N	**Seqüencial**	**Paralelo (P=4)**	**Speed Up (P=4)**	**Paralelo (P=8)**	**Speed Up (P=8)**
100	0,30 *s*	0,15 *s*	2,01	0,09 *s*	3,07
200	2,45 *s*	0,90 *s*	2,70	0,46 *s*	5,34
300	8,35 *s*	2,77 *s*	3,01	1,30 *s*	6,38
400	19,86 *s*	7,02 *s*	2,82	3,06 *s*	6,47
500	38,94 *s*	13,62 *s*	2,85	5,74 *s*	6,77
600	67,24 *s*	22,63 *s*	2,97	9,34 *s*	7,19
700	106,68 *s*	34,89 *s*	3,05	13,86 *s*	7,69
800	160,02 *s*	54,58 *s*	2,93	22,40 *s*	7,14
1000	311,53 *s*	99,99 *s*	3,11	39,73 *s*	7,83
1500	1058,22 *s*	329,56 *s*	3,21	132,31 *s*	7,99

Na Tabela 5.4 estão os tempo de execução dos algoritmos seqüencial e paralelo (para 4 e 8 processos) e o Speed Up dos algoritmos paralelos para matrizes com tamanhos (N) entre 100 e 1500. O algoritmo seqüencial também foi implementado utilizando o ALua: um único nó do tipo *Worker* foi utilizado. Para simplificação, foram utilizadas matrizes quadradas. As medidas foram feita em um ambiente homogêneo (cluster) e foi utilizado o protocolo Chord.

Com a funcionalidade de grupos, novos processos podem facilmente ser adicionados ao *pool* de recursos para aumentar o paralelismo da computação. Pode-se inclusive enviar ao novo processo, assim que ele entrar no grupo, o código que ele deve executar para resolver o problema utilizando o evento `GROUP_NEW_MEMBER`.

5.2.2 Jogo Online

O jogo *online* foi utilizado para analisar a implementação de grupos no ALua. Foi implementado um esqueleto de jogo sem um servidor central para processar todas as operações enviadas pelos jogadores, esse papel será desempenhado pelos próprios jogadores. Para isto, cada processo do jogo funciona como cliente e como servidor. A parte cliente é responsável por: **(i)** exibir ao jogador do processo local a posição do seu personagem no mapa, **(ii)** exibir mensagens de erro e mensagens de outros jogadores, **(iii)** enviar as operações realizadas pelo jogador do processo local para serem processadas pela rede P2P, **(iv)** e receber o resultado do processamento de operações realizadas por outros jogadores para atualizar o seu mapa local. A responsabilidade da

parte servidor é receber as operações dos jogadores, processá-las e enviar os resultados aos jogadores que estão presentes na sua partição do mapa. Nem todos os processos atuam como servidores do jogo, apenas uma certa quantidade, pois dado um mapa de dimensões L x A, este será dividido em partições com dimensões mínimas MIN_L x MIN_A, sendo $MIN\_L \leq L$ e $MIN\_A \leq A$. Cada uma dessas partições será associada a um grupo do ALua e os membros deste grupo serão os jogadores situados nesta parte do mapa. Assim os jogadores desta parte do mapa enviam suas operações para o representante do grupo, que neste caso atuará como servidor, utilizando mensagens do tipo *unicast*.

A partição do mapa é feita quando é detectada a entrada de um novo jogador através do evento `NEW_NODE_JOIN`. O jogador que recebeu este evento irá particionar a sub-partição do mapa da qual é responsável com o novo jogador, caso o mapa ainda não tenha atingido o tamanho mínimo.

Esta arquitetura permite a distribuição do processamento entre os processos da rede, e como as informações dos grupos estão replicadas em outros processos através da replicação de dados oferecida pelo protocolo Pastry, falhas em processos atuando como servidor não interrompem a operação do jogo. Na Figura 5.7 é possível ver uma partição do mapa do jogo e a distribuição das partições entre os processos da rede.

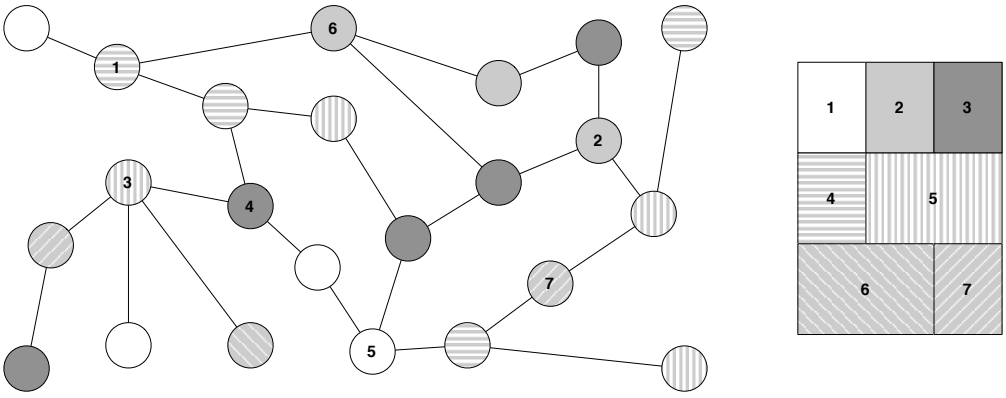

Figura 5.7: Exemplo de partição do mapa do jogo

Além da lista dos jogadores localizados em uma partição do mapa, também são armazenadas em cada representante de grupo as dimensões da partição do mapa que este grupo representa, as posições de cada jogador dentro destas dimensões e referências para as outras partições fronteiriças. Através dos eventos `GROUP_NEW_MEMBER` e `GROUP_MEMBER_LEAVE`, quando um jogador entra ou sai de uma partição do mapa, é feita a notificação aos jogadores dentro desta partição que eles devem redesenhar seus mapas com a nova lista de jogadores.

As operações disponíveis aos jogadores são: **(i)** mover, **(ii)** atacar, **(iii)** enviar mensagem local (aos jogadores da mesma partição) e **(iv)** enviar mensagem global (a todos os jogadores). A operação mais simples é envio de mensagens, pois basta o servidor enviar uma mensagem do tipo *multicast* para todos os jogadores do grupo contendo a mensagem do jogador solicitante. Esse procedimento é realizado tanto para mensagens locais quanto para mensagens globais. No segundo caso existe um grupo global que tem todos os jogadores como membros. As operações de mover e atacar devem ser validadas pelo servidor. Caso não sejam válidas, o servidor envia uma mensagem de erro ao jogador solicitante. Se for uma operação válida, o servidor envia uma mensagem, via *multicast*, a todos os jogadores do grupo contendo a nova posição de todos os jogadores. Quando a nova posição do jogador é fora dos limites da partição da qual o servidor é responsável, o mesmo envia a operação ao servidor responsável pelo nova posição, e este a processa, valida e envia o resultado aos jogadores localizados nesta partição.

Esta aplicação demonstra a possibilidade de se manter informações de estado de grupos mais complexas do que somente a sua lista de membros e também auxiliou na definição de quais informações da camada DHT devem ser visíveis pelas aplicações.