Arquivamento Digital a Longo Prazo Baseado em Seleção de Repositórios em Redes Peer-to-Peer*

Tiago Vignatti[1], **André Vignatti**[2], **Luis C. E. de Bona**[1], **Marcos Sunye**[1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Caixa Postal 19.081 – CEP 81.531-980 – Curitiba – PR – Brasil

[2]Instituto de Computação – Universidade de Campinas (UNICAMP)
Caixa Postal 6.176 – CEP 13.084-971 – Campinas – SP – Brasil

{vignatti, bona, sunye}@inf.ufpr.br, vignatti@ic.unicamp.br

Abstract. *The importance of digital information is increasing constantly in today's society. Digital libraries, Internet applications such email, photo sharing and Web site files often need to be preserved for a long-term and this is a responsability of digital archiving systems. This paper proposes a reliable replication model of immutable digital content to be used in long-term archiving systems. The archiving system is modeled as a set of storage repositories in which independent fail probabilities are assigned. Items are inserted in this system with a reliability that is satisfied replicating them in subsets of repositories. Through simulation, we evaluated three different strategies to create replicas. It is also proposed a totally distributed archiving system using this model over a peer-to-peer (P2P) network. The communication between the nodes (repositories) of such network is organized in a distributed hash table and multiple hash functions are used to select repositories that will keep the replicas of each stored item. The system and its algorithms to create replicas are evaluated through experiments in a real environment. We believe that this model and the proposed algorithms, combined with the structured P2P scalability, make possible to construct a reliable and totally distributed digital archiving systems.*

Resumo. *A importância da informação digital é cada vez maior na sociedade atual. Bibliotecas digitais, aplicações na Internet tais como* email*, compartilhamento de fotos e arquivos de página Web frequentemente precisam ser preservados por um longo período de tempo e cabem aos sistemas de arquivamento digital tal responsabilidade. Este artigo propõe um modelo de replicação confiável de conteúdo digital imutável para ser utilizado em sistemas de arquivamento a longo prazo. O sistema de arquivamento é modelado como um conjunto de repositórios de armazenamento aos quais são atribuídas probabilidades independentes de falha. Itens são inseridos neste sistema com uma confiabilidade que é satisfeita replicando-os em subconjuntos de repositórios. Através de simulação, avaliamos três diferentes estratégias para a criação das réplicas. Também é proposto um sistema de arquivamento totalmente distribuído utilizando este modelo sobre uma rede* peer-to-peer *(P2P). A comunicação dos nodos (repositórios) desta rede é organizada em uma tabela* hash *distribuída e múltiplas*

*Esta pesquisa foi parcialmente suportada pelo CNPq (Proc. 131068/07-5 e 306624/07–9)

funções hashs *são utilizadas para selecionar os repositórios que irão guardar as réplicas de cada item armazenado. O sistema, junto com seus algoritmos para a criação das réplicas, foi avaliado através de experimentos em um ambiente real. Acreditamos que o modelo e os algoritmos propostos, combinados à escalabilidade das redes P2P estruturadas, viabilizam a construção de sistemas de arquivamento digital confiáveis e totalmente distribuídos.*

1. Introdução

Sistemas de arquivamento digital têm a intenção de preservar um grande volume de dados que devem ser armazenados de forma segura por um período de tempo indefinidamente longo. Bibliotecas digitais, aplicações na Internet tais como *email*, compartilhamento de fotos e arquivos de páginas *Web* são alguns exemplos de serviços que precisam destes sistemas. Sistemas de arquivamento podem ser construídos utilizando *hardware* especializado, mas este tipo de solução pode ser economicamente inviável em muitos casos. Uma alternativa é a replicação da informação em múltiplos repositórios de armazenamento, formados por computadores convencionais e de baixo custo [Cooper and Garcia 2001]. As redes *Peer-to-Peer* (P2P) aparecem como uma abordagem promissora para organizar sistemas com tais características, visto que são altamente escaláveis para a distribuição e recuperação de dados. Entretanto, os mecanismos de replicação existentes na maioria das redes P2P não são suficientes para garantir a preservação dos dados por um longo período de tempo.

Neste trabalho propomos um modelo de replicação confiável de dados onde: uma métrica de confiabilidade é associada a cada repositório indicando a probalidade de que os dados armazenados não sejam perdidos ou danificados em um determinado período de tempo, e; cada item (informação digital) é armazenado com uma confiabilidade desejada, permitindo maior durabilidade para os itens considerados mais importantes. O número de réplicas necessárias para preservar um item é determinado pela métrica de confiabilidade dos repositórios, permitindo economia de recursos da rede em relação a outros sistemas onde o número de réplicas é fixo [Maniatis et al. 2005, Risse and Knezevic 2005].

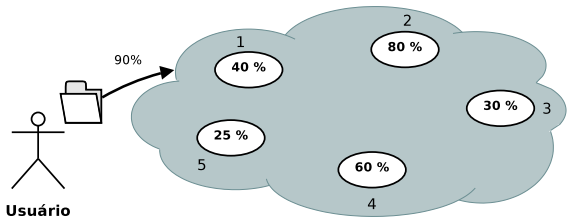

Figura 1. Repositórios rotulados com confiabilidades independentes.

Como exemplo, a Figura 1 ilustra uma rede com cinco repositórios, identificados por 1, 2, 3, 4 e 5, e com confiabilidade respectivamente de 40%, 80%, 30%, 60% e 25%. Digamos que um usuário queira inserir um item na rede com uma *confiabilidade desejada* de 90%. Uma estratégia ingênua seria replicar este item na ordem dos identificadores dos repositórios até atingir a confiabilidade desejada. Com uma única réplica no repositório 1, o item teria uma confiabilidade de 40%. Adicionando mais uma réplica no repositório

2, a confiabilidade desejada seria de $1 - (0.6 \times 0.2) = 88\%$ e ainda não atingiria a confiabilidade desejada de 90%. Ao replicar o item no repositório 3, o mesmo estaria com uma confiabilidade de $1 - (0.6 \times 0.2 \times 0.7) = 91.6\%$, chegando na confiabilidade desejada. Uma contribuição deste trabalho é apresentar uma estratégia para a seleção destes repositórios realizando balanceamento de carga e melhorando o uso de recursos.

Este trabalho também propõe um sistema de arquivamente P2P totalmente distribuído implementando este modelo proposto. Neste sistema os repositórios são organizados por uma tabela *hash* distribuída [Rowstron and Druschel 2001a, Ratnasamy et al. 2001, Stoica et al. 2001, Zhao et al. 2004] e múltiplas funções *hash* são utilizadas como mecanismos de replicação. O sistema foi avaliado através de experimentações em um ambiente real.

O restante deste artigo está organizando da seguinte maneira. A Seção 2 discute os trabalhos relacionados. Na Seção 3 apresentamos o modelo de replicação para sistemas de arquivamento e as estratégias de criação de réplicas. Na Seção 4 apresentamos o sistema de arquivamento P2P e a avaliação dos experimentos. A conclusão e trabalhos futuros são apresentados na Seção 5.

2. Trabalhos Relacionados

Em ambientes de preservação digital e arquivamento a longo prazo as informações não são atualizadas e nem removidas [Martins et al. 2006]. Feita uma réplica, ela se torna imutável no sistema. Neste trabalho não estamos interessados em estratégias e mecanismos de atualizações das réplicas.

Soluções tradicionais de *backup* ou armazenamento de dados, tais como sistemas de arquivos replicados e discos RAID, não provêm um grau de autonomia comparável as P2Ps. Diferente do sistema P2P de arquivamento proposto, esses sistemas utilizam algum tipo de controle central para gerenciar o conteúdo a ser replicado. Sistemas P2P de compartilhamento de arquivos como Kazaa, eDonkey e Gnutella [Milojicic et al. 2002] se concentram em buscar recursos, com coleções sempre dinâmicas e não estão focados na confiabilidade da informação. Nestes sistemas um arquivo será replicado toda vez que for copiado em um novo nodo.

O CFS [Dabek et al. 2001] e o PAST [Rowstron and Druschel 2001b] não são capazes de dinamicamente alterar o número de réplicas pois empregam replicação na vizinhança. O número de réplicas não pode exceder o tamanho da lista de vizinhos, dado que o número de vizinhos é atrelado ao protocolo DHT. Um eventual sistema de arquivamento digital construído sobre estes sistemas poderia não alcançar o fator de replicação desejado para preservar suas informações. Outros esquemas de armazenamento de arquivos que utilizam DHT como o OceanStore [Kubiatowicz et al. 2000] e o Glacier [Haeberlen et al. 2005] consideram seus nodos com iguais probabilidades de falhas. Apesar de tratar em primeiro plano da disponibilidade dos itens, o BRICKS [Risse and Knezevic 2005] também padece do mesmo problema ao considerar a probabilidade dos nodos estarem conectados a rede de maneira global.

O LOCKSS (*Lots of Copies Keep Stuff Safe*) [Maniatis et al. 2005] utiliza uma rede P2P de computadores controlados por organizações autônomas para preservar as informações por um longo período de tempo. Ele tem um esquema complexo de auditoria para detectar e reparar eventuais danos nas réplicas. Diferente do nosso modelo, o

LOCKSS trata seus repositórios com uma única probabilidade de falha, o que não modela exatamente ambientes reais.

3. Modelo e Descrição do Problema

Formalmente, o modelo é composto de um conjunto N de $|N| = n$ itens (objetos digitais). Cada item i tem um *tamanho* s_i e um número $0 < r_i < 1$, dito a *confiabilidade desejada* deste item. Além disso, temos um conjunto $M = \{\ell_1, \ldots, \ell_m\}$ de $|M| = m$ repositórios, onde cada repositório ℓ_j tem associado um número $0 < p_j < 1$, dito a *confiabilidade do repositório*. Para determinar esta confiabilidade, podemos considerar alguns parâmetros como a quantidade de *bugs*, as vulnerabilidades no sistema, fatores humanos como a taxa que máquinas são reparadas, dentre outros – está fora do escopo deste trabalho determinar estes parâmetros. Cada repositório também tem associada uma *capacidade de armazenamento* de itens c_j. Definimos a *capacidade de armazenamento livre* de ℓ_j como $f_j = c_j - \sum_{k \in \ell_j} s_k$. A confiabilidade do subconjunto $S \subseteq M$ é definida como $1 - \prod_{\ell_j \in S}(1 - p_j)$, que denota a probabilidade de pelo menos um repositório em S não perder dados em um determinado período de tempo.

Definimos o problema da criação de réplicas da seguinte maneira. Itens vão chegando um a um *on-line*, ou seja, inicialmente a entrada é vazia e os itens vão chegando no decorrer do tempo. Após um item i chegar, devemos escolher um subconjunto de repositórios tal que a confiabilidade desse subconjunto seja pelo menos a confiabilidade desejada de i. Em outras palavras, devemos selecionar $S_i \subseteq M$ tal que $1 - \prod_{\ell_j \in S_i}(1 - p_j) \geq r_i$. Além disso, cada repositório em S_i deve suportar s_i. Ou seja, $\forall \ell_j \in S_i$, $s_i \leq f_j$ tem que valer. Em cada repositório de S_i colocamos então uma cópia do item i (*réplica*) para satisfazer r_i. As réplicas nunca são atualizadas, ou seja, são imutáveis.

Neste modelo, temos o objetivo de selecionar subconjuntos de repositórios que minimizem o "desperdício" de suas confiabilidades ao satisfazer a confiabilidade desejada de um objeto inserido. Em outras palavras, para um item i queremos selecionar $S_i \subseteq M$ que forneça a confiabilidade que está mais próxima de r_i. Assim, esperamos que uma quantidade maior de itens possa ser inserida em M. Acreditamos que minimizar o número de réplicas geradas e balancear a criação destas réplicas entre os repositórios sejam boas heurísticas para alcançarmos tal objetivo.

A seguir, reescrevemos a restrição de confiabilidade desejada de forma a facilitar o manuseio do problema de replicação. Mais especificamente, apresentamos uma definição equivalente da restrição de confiabilidade desejada, substituindo o produtório por um somatório. Podemos reescrever a restrição de confiabilidade desejada da seguinte forma $1 - \prod_{\ell_j \in S_i}(1 - p_j) \geq r_i \equiv \prod_{\ell_j \in S_i}(1 - p_j) \leq 1 - r_i \equiv \prod_{\ell_j \in S_i} e^{\ln(1-p_j)} \leq e^{\ln(1-r_i)} \equiv e^{\sum_{\ell_j \in S_i} \ln(1-p_j)} \leq e^{\ln(1-r_i)}$. Mas $e^{\sum_{\ell_j \in S_i} \ln(1-p_j)} \leq e^{\ln(1-r_i)} \iff \sum_{\ell_j \in S_i} \ln(1 - p_j) \leq \ln(1 - r_i)$. Como $0 < p_j, r_i < 1$, então o valor da função logaritmo é negativa. Assim, para clareza de notação, vamos redefinir as variáveis do problema. Seja $a_j = -\ln(1-p_j)$ e $b_i = -\ln(1 - r_i)$, então a restrição de confiabilidade desejada poder ser reescrita de maneira equivalente como

$$\sum_{\ell_j \in S_i} a_j \geq b_i$$

Ou seja, para um item i devemos selecionar $S_i \subseteq M$ tal que a soma de a_j ultrapasse o

valor b_i. Essa definição, além de ser mais fácil de manusear, é equivalente à definição anterior.

3.1. Estratégia de Criação das Réplicas

Para criar as réplicas, inicialmente selecionamos um subconjunto aleatório $S \subseteq M$ de repositórios e então selecionamos o subconjunto $F \subseteq S$ que forneça a confiabilidade que está mais próxima da confiabilidade desejada do item. Ou seja, queremos selecionar $F \subseteq S \subseteq M$ que minimize $1 - \prod_{\ell_j \in F}(1 - p_j) - r_i$. Denominamos tal estratégia como a estratégia do *Subconjunto Ideal*. Usando a definição equivalente do problema, o que queremos é resolver o seguinte PLI

$$\begin{aligned} \min & \sum_{\ell_j \in S} a_j x_j - b_i \\ \text{s.t.} & \sum_{\ell_j \in S} a_j x_j \geq b_i \\ & x_j \in \{0,1\} \qquad \forall \ell_j \in S \end{aligned}$$

Ao resolvermos o PLI, se o valor ótimo for igual a 0 então existe um subconjunto de valores que somados dão exatamente b_i; se o valor ótimo for maior que 0, então não existe tal subconjunto. Desta forma poderíamos resolver o problema de decisão SUBSET-SUM que é NP-COMPLETO [Garey and Johnson 1979]. Portanto, o PLI é NP-DIFÍCIL e, assumindo P $\neq$ NP, não existe algoritmo polinomial que resolva este problema. No entanto, existe um algoritmo de *programação dinâmica* que resolve esse problema em tempo pseudo-polinomial e na prática é satisfatório[1] para a grande maioria das instâncias [Garey and Johnson 1979]. Os detalhes do algoritmo são omitidos por questão de espaço.

A solução do PLI acima não necessariamente irá nos fornecer uma solução boa para minimizar o número de réplicas geradas. Apesar disso, a estratégia do Subconjunto Ideal é motivada pelo fato que em situações práticas é esperado não criar muitas réplicas e ao mesmo tempo selecionar subconjuntos diferentes para cada item, para que a distribuição das réplicas nos repositórios seja feita de modo a balancear a carga.

3.2. Avaliação e Simulação de Experimentos

Através de experimentos avaliamos a estratégia do Subconjunto Ideal comparando com outras duas estratégias simples de criar réplicas: a Aleatorizada e a Gulosa. A estratégia Aleatorizada cria cópias das réplicas em repositórios aleatórios até atingir b_i.

A estratégia Gulosa ordena os valores a_j de maneira não-crescente, incluindo na solução do problema os valores nesta ordem até a soma dos valores selecionados alcançar a confiabilidade desejada b_i. Contudo, note que esta estratégia acumula as réplicas nos repositórios com maiores confiabilidades e isto é ruim para o balanceamento de carga já que as réplicas serão atribuídas, em sua maior parte, ao mesmo subconjunto de repositórios. Para resolver isso, executamos tal estratégia sob um subconjunto de repositórios, de

[1]No nosso simulador a versão do algoritmo por força bruta (computando todos os possíveis subconjuntos) demorou 13.16 **minutos** para ser executado contra 1.1 **segundos** na versão em programação dinâmica para um $|S|$ = 22. Este teste foi realizado num processador Intel Core 2 Duo, 2.00 GHz.

tamanho f, escolhido aleatoriamente ao invés do conjunto total. Note que se $f = |M|$, então recaímos no problema do acúmulo de réplicas nos repositórios com maiores confiabilidades. Por outro lado, se f for pequeno, não existem muitas opções de escolha sobre F ou não há repositórios suficientes para satisfazer a confiabilidade desejada de um item. O desafio é encontrar um valor f que não sobrecarregue os repositórios, mas que ao mesmo tempo criemos poucas réplicas.

Na avaliação foi utilizado um simulador próprio que implementa os algoritmos e insere uma massa de dados. Todos os experimentos foram executados 10 vezes e a média foi coletada. Em todos os experimentos consideramos um conjunto de 100 repositórios.

Número de Réplicas Inseridas. Para a primeira série de experimentos, o simulador foi executado para repositórios com a confiabilidade média variando entre 50%, 60%, 70% e 80%, com o desvio padrão de 6% em cada um deles. Realizamos a inserção de 1000 itens aleatórios, todos com a confiabilidade desejada de 99%. Neste experimento desconsideramos o tamanho dos itens e a capacidade dos repositórios.

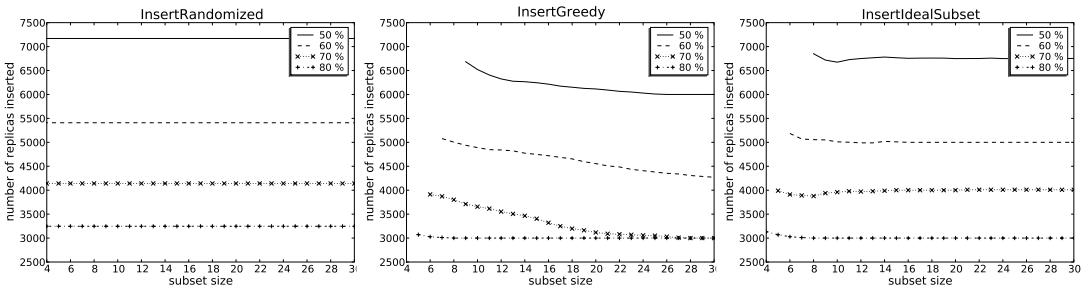

Figura 2. Número de réplicas inseridas em função do tamanho do subconjunto.

A Figura 2 ilustra o primeiro experimento, ao qual as réplicas são geradas em função do tamanho do subconjunto nas três estratégias. Obviamente que para a estratégia Aleatorizada não faz sentido variar o tamanho do subconjunto pois esta estratégia não depende de tal parâmetro, mas o gráfico é ilustrado como forma de comparação com as outras estratégias. Vale observar também que nas outras duas estratégias, dependendo da confiabilidade média dos repositórios, a linha inicia somente após um determinado tamanho do subconjunto de repositórios selecionados. O motivo é que para um tamanho muito pequeno do subconjunto, a confiabilidade desejada não é satisfeita na inserção de todos os itens.

Conforme o tamanho do subconjunto de repositórios cresce, a quantidade de réplicas suficientes para alcançar a confiabilidade desejada diminui. Esta diminuição é mais acentuada na estratégia Gulosa, a qual tem o objetivo explícito de diminuir as réplicas, sempre criando-as em repositórios com maior confiabilidade. A alternativa do Subconjunto Ideal também apresenta um declive, porém menor que na estratégia Gulosa e maior que na Aleatorizada.

Balanceamento da Carga. O desvio padrão da carga dos repositórios (em termos de armazenamento de réplicas) é uma medida suficiente para observar o balanceamento após a inserção das réplicas. Dessa maneira, a Figura 3 ilustra o segundo experimento que mostra o desvio padrão da carga dos repositórios variando em função do tamanho dos subconjuntos. Foi utilizado uma confiabilidade média de 60% para os repositórios com

6% de desvio padrão (o mesmo experimento foi repetido para repositórios com outras confiabilidades e o resultado obtido foi proporcionalmente o mesmo).

Da mesma maneira que no experimento anterior, a alternativa Aleatorizada não sofre influência sobre o tamanho do subconjunto de repositórios selecionados. Para as outras estratégias, podemos observar que quanto maior o tamanho do subconjunto, maior o desvio padrão da carga. A estratégia Gulosa é a que tem o pior balanceamento.

Makespan. A Figura 4 ilustra o terceiro experimento no qual a carga da máquina mais carregada, ou o *makespan*, é plotado em função do tamanho do subconjunto de repositórios. Conforme o tamanho do subconjunto de repositórios selecionados cresce, o *makespan* também cresce nas estratégias Gulosa e do Subconjunto Ideal, porém nesta última, sendo sempre menor o crescimento. Este comportamento de crescimento é justificado pelo fato de alguns repositórios serem sempre os mais requisitados. Na estratégia Gulosa os repositórios mais requisitados serão obviamente os com maior confiabilidade. Os repositórios com menor confiabilidade serão os mais requisitados na estratégia do Subconjunto Ideal, pois repositórios com esta característica são sempre utilizados para evitar o "desperdício" de confiabilidade.

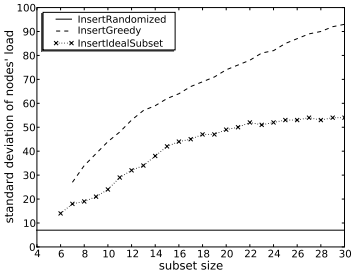

Figura 3. Desvio padrão da carga dos repositórios em função do tamanho dos subconjuntos.

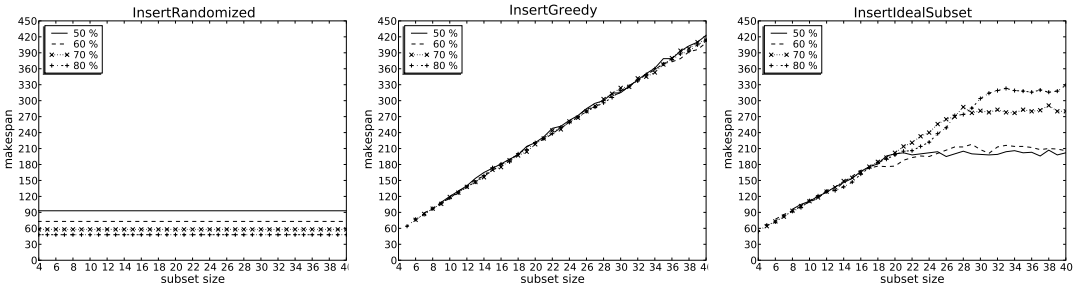

Figura 4. Makespan em função do tamanho do subconjunto.

Taxa de Sucesso na Inserção de Itens. Para conjuntos de repositórios com capacidades de armazenamento homogêneas, o *makespan* é um importante índice pois avalia a taxa de sucesso na inserção de itens. A estratégia que apresentar o maior *makespan* implicará na menor taxa de sucesso para a inserção de itens. Entretanto, em ambientes reais é comum encontrarmos redes nas quais a capacidade de armazenamento dos repositórios diferem em várias ordens de magnitude. Neste tipo de rede, o *makespan* não necessariamente vai indicar o mesmo comportamento que nos repositórios com capacidade igual de armazenamento. O principal objetivo deste experimento é medir a taxa de sucesso na inserção de itens em repositórios heterogêneos em termos de capacidade. Inserimos itens até que algum deles falhe na criação das réplicas por falta de capacidade de armazenamento em um repositório. Os itens têm tamanho de 35 MB e confiabilidade desejada de 99%. A capacidade de armazenamento dos 100 repositórios varia entre 100 MB e 100.000 MB, e confiabilidade média deles é de 67% (desvio padrão de 17%).

Dependendo da estratégia utilizada, alterar o tamanho do subconjunto de repositórios selecionados causa diferentes taxas de inserção: quanto maior for o subconjunto selecionado de repositórios, maior o número possível de soluções e conseqüentemente maior a taxa de inserção de itens. A Tabela 1 mostra a inserção de itens em função do tamanho do subconjunto de repositórios para as três estratégias de criação de réplicas. A estratégia do Subconjunto Ideal tem os melhores resultados.

	10	20	40	100
Aleatorizada	18569	30228	32464	34138
Gulosa	17544	28182	30891	33899
Subconjunto Ideal	20535	31094	35399	39107

Tabela 1. Inserção de itens variando o tamanho do subconjunto.

Discussão do Resultados. Dentre as três estratégias, a Gulosa sempre vai gerar o menor número de réplicas. Por outro lado, ela sempre apresenta o *makespan* mais alto. A estratégia Aleatorizada é a que apresenta o melhor balanceamento das réplicas, porém apresenta o pior número de réplicas geradas. O meio termo entre diminuir o número de réplicas e espalhá-las é obtido com a estratégia do Subconjunto Ideal. Além disso, em um conjunto de repositórios de capacidades de armazenamento heterogêneas, a estratégia do Subconjunto Ideal demonstrou preencher melhor os espaços dos repositórios ao inserir uma quantidade maior de itens.

4. Um Sistema Peer-to-Peer para Arquivamento Digital

O modelo proposto na seção 3 está definido de maneira genérica para ser aplicado sobre qualquer mecanismo distribuído para organizar repositórios de armazenamento. Em particular, as redes P2P estruturadas apareceram como canditadas naturais para organizar esses repositórios, visto que são altamente escaláveis para a distribuição e recuperação de dados. Entretanto, é indispensável a utilização de estratégias de replicação que garantam a confiabilidade da informação da forma que o ambiente de arquivamento digital exige. Além disso, existe uma dificuldade inerente das redes P2P estruturadas que é a seleção exata dos nodos (repositórios) que armazenarão as réplicas. Selecionar um subconjunto específico de nodos não é trivial utilizando a maneira determinística de rotear informações das DHTs. A seguir apresentaremos e defenderemos a arquitetura utilizada para a implementação do sistema P2P de arquivamento digital.

P2P Estruturada e DHT. O sistema faz o roteamento das mensagens na rede através das P2P estruturadas, utilizando as DHTs. Acreditamos que a arquitetura P2P não-estruturada não satisfaça bem os requisitos de um sistema para arquivamento digital por algumas razões. Primeiramente, no ambiente de arquivamento a longo prazo a frequencia de busca por arquivos é normalmente uniforme, ou seja, todos os objetos são igualmente populares [Lv et al. 2002]. Redes P2P não-estruturadas usam algoritmos de força bruta, através de inundação, para realizar a busca e portanto são mais apropriados para conteúdos populares. Além disso, algoritmos de busca das redes não-estruturadas geralmente não conseguem localizar itens raros, o que é inadmissível para o contexto de preservação e arquivamento digital. E por fim, a escalabilidade das não-estruturadas deixam a desejar quando comparada às estruturadas.

A principal crítica em relação a P2P estruturadas é com populações extremamente transientes. Manter a estrutura das tabelas de roteamento é relativamente custoso quando

acontece *churn*. Entretanto, a transiência das máquinas em organizações que têm a intenção de preservar documentos digitais não acontece tão frequentemente quando comparada com máquinas utilizadas em aplicações tradicionais na arquitetura não-estruturada. Portanto, os ajustes necessários na topologia da redes estruturadas não devem sobrecarregar o sistema de arquivamento no caso de *churn*.

Replicação. No ambiente de arquivamento digital é importante salientar que não estamos interessados em atualizações das réplicas, uma vez que estamos falando de objetos *somente de leitura* e *estáticos* [Martins et al. 2006]. Feita uma réplica, ela se torna imutável no sistema. Além disso, para armazenamento a longo prazo a *integridade* e a *autenticidade* das réplicas são requisitos básicos que devem ser tratados.

Foi escolhida a estratégia de múltiplas funções *hash* para replicar os dados no sistema P2P. Utilizamos uma maneira muito simples para criar as *hashs*. O número da réplica a ser criada i é anexado no final da chave do objeto e então passado como argumento para uma função h hipotética. Suponha que o nome do objeto é *foo*, então $h(foo1), h(foo2), \ldots, h(foor)$ nos daria r *hashs* para este objeto. Esta maneira simula muito bem a criação de múltiplas funções *hash* e recuperação das mesmas. Enumeramos os principais pontos que nos motivaram a empregar a estratégia de múltiplas funções *hash*: (i) *a independência por um protocolo DHT*. A estratégia de múltiplas *hash* permite que possamos utilizar qualquer protocolo DHT para rotear as mensagens na rede. Outras estrátégias de replicação, como replicação por vizinhança ou por caminho, não se beneficiam de tal característica; (ii) *número independente de réplicas para cada objeto*. Uma das principais características do nosso modelo de replicação é a independência pelo número de réplicas que cada objeto digital possui. A estratégia de replicação P2P simétrica não tem esta flexibilidade e não poderia ser utilizada como mecanismo de replicação para o nosso modelo. Podemos citar o BRICKS como um exemplo de um sistema de armazenamento confiável de dados ao qual define o grau de replicação fixo dos itens; (iii) *recuperação fácil do objeto*. A facilidade de recuperar uma determinada réplica sem a dependência de recuperar outra previamente é uma grande motivação da estratégia com múltiplas *hash*. A independência na hora de recuperar uma réplica facilita o algoritmo de busca. Na estratégia do *hash* correlacionado, todas as chaves de um determinado objeto são correlacionadas a primeira chave, não possibilitando tal característica.

Seleção dos Repositórios. Nas redes P2P estruturadas a topologia da rede é bem definida e o conteúdo é colocado em locais determinados – a maneira que a DHT indexa uma chave é sempre determinística. Assim, escolher exatamente o nodo que um conteúdo deve ser guardado não é uma característica deste tipo de rede. Entretanto se estamos interessados em salvar o conteúdo em repositórios com confiabilidades determinadas, então devemos prover um mecanismo que simule este processo de seleção. Ao aplicar a chave de um objeto em r funções *hash*, temos um subconjunto com r parâmetros de confiabilidade dentro dele. Desta maneira, obtemos um subconjunto de confiabilidades ao qual podemos selecionar as quais melhores se adequarão nas estratégias de criação das réplicas. É importante notar que selecionar *todos* os nodos da rede não é uma boa abordagem pois, em eventuais algoritmos executados no sistema, uma instância teria o tamanho da rede e neste caso a complexidade computacional seria inviável. Por outro lado, executar estratégias a partir de um subconjunto de tamanho constante é viável mesmo em face da escalabilidade da rede.

Esta maneira de selecionar os repositórios implica em não colocar nenhuma informação centralizada sobre o local em que as réplicas estão armazenadas. Uma alternativa seria existir um super-nodo que tivesse um "diretório" ao qual pudesse ser feitas consultas sobre a informação exata do lugar das réplicas de determinado objeto. Porém, este supernodo seria um ponto de contenção. O nosso sistema evita super-nodos, adotando uma abordagem totalmente distribuída onde nenhuma informação é centralizada. Assim, para fazer a recuperação de uma réplica por exemplo, todos os repositórios do subconjunto determinado pelas múltiplas funções *hash* são considerados. A Figura 5 ilustra a seleção de repositórios realizada por três objetos diferentes. Na figura, os subconjuntos de repositórios determinados pelas chaves `objeto_a`, `objeto_b` e `objeto_c` são representados pelos traços pontilhado. Note que não necessariamente estes subconjuntos precisam ser disjuntos. Os círculos pretos são repositórios que contêm réplicas enquanto os brancos não as contêm.

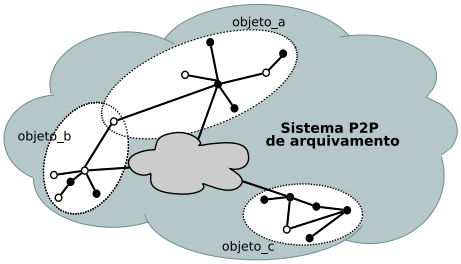

Figura 5. Subconjunto de repositórios determinado pelos objetos digitais.

4.1. Algoritmos de Operação do Sistema

Uma grande característica do sistema proposto é que nenhuma modificação no protocolo DHT é necessária para permitir sua utilização. Simplesmente é realizado um "invólucro" nas operações básicas da DHT, `put` e `get`, chamando-as respectivamente de algoritmos `insert` e `retrieve`. É importante notar que ambos algoritmos são processados localmente em cada nodo. Por exemplo, um usuário que desejar fazer a inserção ou recuperacão de um objeto contacta qualquer nodo da rede, que através da DHT inicia todo o processo de roteamento da mensagem.

input: $name$, $value$, desired reliability r_i
begin
  $R =$ select subset of repositories using multiple hash functions; use $name$
  $S =$ get subset of repositories by the insertion strategy; use r_i and R
  foreach *s in S* **do**
    $j \leftarrow$ get hash function number of s
    put($h_j(name), value$)
end

Algoritmo 1: insert (name, value, reliability)

Para inserir um objeto, o usuário do sistema de arquivamento executa a rotina `insert(name, value, reliability)`, como visto no Algoritmo 1. Ao inserir um item na rede, o usuário previamente escolhe a confiabilidade desejada deste item (`reliability`). Dependendo da estratégia de inserção utilizada, é feita a busca das confiabilidades de cada repositório utilizado (que poderia ser armazenada na própria DHT

por exemplo). Várias funções *hash* irão mapear o item em diferentes nodos da rede utilizando `put(name, value)`, satisfazendo a confiabilidade desejada. O algoritmo cuida para não atribuir mais de duas réplicas de um mesmo objeto em único nodo.

```
input: name
begin
    counter ← r
    while counter ≠ 0 do
        Let i chosen uniformly at random in {1, ..., r}
        value ← get(h_i(name))
        if value is not null then
            return value
        counter ← counter − 1
    return −1 /* not found                                    */
end
```

Algoritmo 2: retrieve (name)

Para recuperar o objeto, o usuário executa a função `retrieve(name)`, onde `name` é o nome do objeto requerido, visto no Algoritmo 2. A idéia do algoritmo é escolher um número aleatório de maneira uniforme entre 1 e r, onde r é o tamanho do subconjunto de repositórios selecionados. Se o número sorteado corresponde a uma função *hash* existente (réplica), o algoritmo termina com sucesso e o conteúdo do objeto é retornado para o usuário. Outras dois fatores podem acontecer para que a DHT não retorne o objeto: quando o nodo está ausente ou quando o valor sorteado é uma função que aponta para um nodo que não contém a réplica do objeto. Para ambos os casos, o algoritmo sorteia outro número. Um contador (`counter`) existe no algoritmo para garantir o término caso nenhuma réplica exista após percorrer todos os possíveis valores de r.

4.2. Resultados Experimentais

O sistema P2P de arquivamento foi implementado e avaliado através de experimentos executados em um ambiente real. A implementação foi feita utilizado o ambiente de construção de redes *Overlay Weaver* [Shudo et al. 2008]. O *Overlay Weaver* proporciona grande flexibilidade na escolha dos protocolos DHT e outros tipos de serviço alto-nível implementados como redes de sobreposição. Em particular, este ambiente de construção suporta o Chord, Pastry, Tapesty e Kademlia. Em nossos experimentos utilizamos o Chord. É importante notar que o *Overlay Weaver* já apresenta um mecanismo próprio de replicação ao qual foi desligado para a avaliação dos nossos experimentos. Os experimentos foram conduzidos em uma rede de 12 nodos, quantidade suficiente para validar a capacidade de arquivamento a longo prazo das informações e também o mecanismo de criação das réplicas.

Consideramos a confiabilidade de cada nodo como a probabilidade dele não perder nenhuma informação no período de 1 ano. Assim, podemos simular qual será o estado da rede em relação a preservação das informações com o passar dos anos. Por exemplo, se a confiabilidade de um nodo é 70%, então ele tem 70% de chances de iniciar o segundo ano com todas as suas informações intactas. Da mesma maneira, este mesmo nodo tem 70 % de probabilidade de iniciar o terceiro ano, e também os outros próximos, sem nenhuma perda de informações. Note que o passar dos anos não implica em uma probabilidade maior de perda, pois tais probabilidades são independentes. Nos experimentos não foi avaliado a mudança de confiabilidade dos nodos ao passar dos anos.

O experimento é iniciado com uma determinada rede de nodos na qual são inseridos diversos objetos. Objetos são inseridos com diferentes confiabilidades. Para a replicação dos objetos é utilizado a estratégia do Subconjunto Ideal com tamanho dos subconjuntos igual a 6. No primeiro ano todos os nodos estão disponíveis (*online*) e nos próximos anos forçamos a desconexão de alguns nodos. Uma vez que o nodo é desconectado da rede (*offline*), ele perde todas as informações que tinha. Naturalmente, nodos que têm confiabilidade baixa, são os mais prováveis a serem desconectados. Ao voltar à rede, o nodo é considerado como um novo. Entretanto, nos experimentos não tratamos de novos nodos entrando na rede. No final, espera-se que objetos com maior confiabilidade desejada sejam preservados por mais tempo.

A Figura 6 mostra o experimento. Do total dos 12 nodos, 2 deles continham 30% de confiabilidade de não falhar, 2 continham 50%, 2 continham 70%, 3 continham 80% e os 3 restantes continham 90% de confiabilidade. Inicialmente, no primeiro ano, foram inseridos um total de 25 objetos, separados em 5 grupos, onde cada grupo continha objetos com confiabilidade desejada respectivamente de 30%, 50%, 70%, 90% e 99%. Após o primeiro ano de existência do sistema, 1 nodo de confiabilidade igual a 30%, 1 nodo de 50% e outro de 90% falharam. Mesmo com estas falhas, todos os 25 objetos foram capazes de serem recuperados. No final do quinto ano do sistema, mais dois nodos (30% e 70% de confiabilidade) se desconectaram da rede. Neste quinto ano não foi possível recuperar na rede 3 objetos de 30% de confiabilidade desejada, 4 objetos de 50%, 1 de 70% e 1 de 90%. Ao passar o décimo de existência do sistema mais três nodos ficaram *offline*. Foram nodos de 70%, 80% e 90%, respectivamente. Não foi possível localizar um total de 3 objetos de 30% de confiabilidade desejada, 4 objetos de 50%, 4 de 70% e 2 de 90%. No final do décimo quinto ano mais um nodo de 80% de confiabilidade se desconectou e os mesmos objetos do décimo ano foram possíveis de se localizar. No vigésimo ano, 1 nodo de 50% se desconectou, sobrando apenas um nodo de 80% e outro de 90%. Nesta idade do sistema era possível localizar todos os objetos de 99% de confiabilidade desejada e 1 de 90%. Os outros todos já tinham sido perdidos. No vigésimo quinto ano o nodo de 80% falhou e ainda era possível recuperar 4 objetos de 99% e 1 de 90%. No trigésimo ano o último nodo se desconectou da rede e nenhuma réplica mais existia.

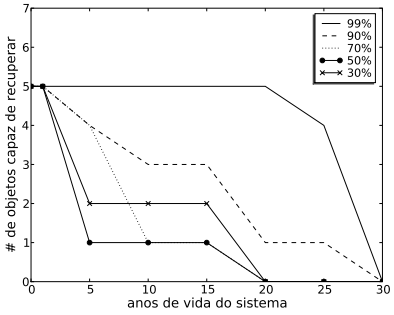

Figura 6. Quantidade de Objetos capaz de recuperar em função da idade do sistema.

Encaramos a confiabilidade de cada nodo como a probabilidade dele não perder nenhuma informação no período de 1 ano. Ao simular o passar dos anos, os experimentos comprovaram a capacidade do sistema em preservar informações por um longo período de tempo. A importância pela preservação de cada objeto digital – medido no modelo pela confiabilidade desejada – impacta em diferentes tempo de vida do mesmo. Objetos que foram inseridos com uma confiabilidade desejada maior tiveram um tempo de vida maior. Assim, através dos experimentos, podemos concluir duas principais características do sistema: **Independência de preservação dos objeto**. Diferentes informações requerem diferentes tempo de armazenamento. Algumas coleções de fotos, revistas e artigos podem precisar de poucos anos de armazenamento enquanto outras informações, como objetos

digitais de museus e de bibliotecas, necessitam de centenas de anos. Nosso sistema permite total flexibilidade na escolha do tempo de vida dos objetos a serem preservados; **Otimização dos recursos de armazenamento**. Repositórios de armazenamento podem sofrer várias ameaças em seus conteúdos, portanto cada repositório tem uma diferente confiabilidade. Assim, modelar cada repositório de armazenamento com um parâmetro capaz de medir sua probabilidade independente de falha é a maneira mais próxima de modelar redes reais. Tal maneira permite a flexibilidade no tempo de preservação de cada objeto e conseqüentemente diferente números de réplicas para eles, impactando em uma melhor utilização dos recursos de armazenamento da rede.

5. Conclusão

Este artigo apresentou um modelo de replicação confiável de dados para sistemas de arquivamento digital com dados imutáveis. A principal característica deste modelo é na associação de probabilidades independentes de falha para cada repositório de armazenamento as quais possibilitam que itens sejam guardados conforme suas necessidades pelo tempo de preservação. Através de simulações constatamos que dentre as três estratégias, a estratégia do Subconjunto Ideal mostrou-se a mais equilibrada nos quesitos de minimizar as réplicas e o *makespan* e ainda teve a melhor taxa de inserção de objetos. Tal estratégia apresenta uma complexidade computacional que não poderia ser possível de ser executada na prática caso não fosse cuidadosamente formulada. Devido a alta escalabilidade para a distribuição e recuperação de dados, as redes P2P estruturadas aparecem como candidatas naturais para implementar o modelo do sistema de arquivamento proposto. As P2P estruturadas têm uma dificuldade inerente ao modo de roteamento das informações que dificulta a seleção de nodos (repositórios) específicos para o armazenamento das réplicas. Entretanto, esta dificuldade foi suprida através da utilização de múltiplas funções *hash*. A implementação deste sistema P2P de arquivamento digital foi avaliada em um ambiente real.

Como trabalho futuro está a implementação de um sistema completo de preservação digital. Muita pouca ênfase foi dada para a recuperação dos itens. Além disso o sistema deveria se preocupar com a auditoria das réplicas e também com outras ameaças tais como obsolescência de *software* e formato, não tratadas neste trabalho. Eventualmente utilizar o nosso modelo de arquivamento digital sobre sistemas como o BRICKS e LOCKSS e avaliar sua viabilidade também é interessante.

Referências

Cooper, B. and Garcia, H. (2001). Creating trading networks of digital archives. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 353–362, New York, NY, USA. ACM.

Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. (2001). Wide-area cooperative storage with CFS. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, Chateau Lake Louise, Banff, Canada.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*.

Haeberlen, A., Mislove, A., and Druschel, P. (2005). Glacier: highly durable, decentralized storage despite massive correlated failures. In *NSDI'05: Proceedings of the*

2nd conference on Symposium on Networked Systems Design & Implementation, pages 143–158, Berkeley, CA, USA. USENIX Association.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Wells, C., and Zhao, B. (2000). Oceanstore: an architecture for global-scale persistent storage. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 190–201, New York, NY, USA. ACM.

Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. (2002). Search and replication in unstructured peer-to-peer networks. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 258–259, New York, NY, USA. ACM.

Maniatis, P., Roussopoulos, M., Giuli, T., Rosenthal, D., and Baker, M. (2005). The lockss peer-to-peer digital preservation system.

Martins, V., Pacitti, E., and Valduriez, P. (2006). Survey of data replication in P2P systems.

Milojicic, D. S., Kalogeraki, V., Lukose, R., and Nagarajan, K. (2002). Peer-to-peer computing. Technical report, HP Labs.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. (2001). A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA. ACM.

Risse, T. and Knezevic, P. (2005). A self-organizing data store for large scale distributed infrastructures. In *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*, Washington, DC, USA. IEEE Computer Society.

Rowstron, A. and Druschel, P. (2001a). Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*.

Rowstron, A. and Druschel, P. (2001b). Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility.

Shudo, K., Tanaka, Y., and Sekiguchi, S. (2008). Overlay weaver: An overlay construction toolkit. *Comput. Commun.*, 31(2):402–412.

Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H. (2001). Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. D. (2004). Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53.