Mitigação do *churn* em serviços de armazenamento P2P

Leandro André Siopa Ribeiro

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Orientador: Prof. João Coelho Garcia

Juri

Presidente: Prof. Rui Filipe Fernandes Prada
Orientador: Prof. João Coelho Garcia
Vogais: Prof. Miguel Ângelo Marques de Matos

Janeiro 2021

Agradecimentos

Gostaria de agradecer ao Professor João Coelho Garcia por todo o seu apoio e orientação que me deu durante este último ano, através do qual contribuiu muito para a realização deste trabalho.

Gostaria muito de agradecer a toda a minha família, principalmente aos meus pais que me deram a oportunidade de realizar este curso e o incentivo que me deram para finalizar o curso.

Gostaria de agradecer ao meu irmão, à minha namorada e aos colegas de curso que sempre me motivaram e ajudaram a realizar este trabalho.

Finalmente gostaria de agradecer a todos os familiares, amigos e colegas por todo o apoio que sempre me deram.

Leandro André Siopa Ribeiro

Para os meus pais

Resumo

A saída e entrada de nós de uma rede entre-pares (P2P), o churn, é um problema generalizado neste tipo de redes. Os serviços de armazenamento P2P sofrem também deste problema, pois devido ao churn existe perda e indisponibilidade de dados. Este problema tem levado a que os serviços de armazenamento P2P a serem preteridos face a serviços de armazenamento na nuvem. De forma a mitigar o problema da perda ou da indisponibilidade dos dados, apresentaremos uma solução que consiste no uso de um mecanismo de incentivos para recompensar os nós com melhor reputação e um sistema de reputação que visa avaliar o comportamento dos nós da rede. Com este sistema de reputação pretendemos motivar os nós a estarem ligados à rede para melhorar a sua própria reputação. Dado que quanto melhor for a sua reputação mais recompensas terão, uma vez que estarão a contribuir para um melhor funcionamento da rede, então dado que estas recompensas passarão por mais espaço de armazenamento na rede. Dado que a rede pretende oferecer mais armazenamento para os seus nós. Então é necessário aproveitar eficientemente o espaço de armazenamento disponível na rede, então utilizaremos métodos de compressão de dados para minimizar o armazenamento ocupado pelos dados na rede.

Abstract

The exit and entrance of P2P network's nodes, the churn, is a widespread problem in this type of networks. P2P storage systems suffer from this problem because due to the churn there is data loss and data unavailability, which has led P2P storage systems being less used compared to cloud storage systems. In order to mitigate the problem of data loss or data unavailability, we will present a solution that consists of using an incentive mechanism to reward the nodes based in their reputation and a reputation system to assess the behavior of network nodes. With this reputation system we intend to motivate the nodes to be connected to the network to improve thier own reputation, give better rewards to those with better reputation, since they will be contributing to better network operation. These rewards will go through more storage on the network. Given that the network intends to offer more storage for its nodes, then it is necessary to efficiently use the storage available on the network, so we will use data compression methods to minimize the storage occupied by the data on the network.

Palavras Chave
Keywords

Palavras Chave

Redes entre-pares

Sistema de armazenamento

Incentivos

Reputação

Keywords

Peer-to-peer

Storage system

Incentives

Reputation

Índice

List of Figures

List of Tables

1 Introdução

As redes entre-pares também conhecidas como redes *Peer-to-Peer* (P2P) têm-se popularizado nos últimos anos devido ao facto de não necessitarem de um servidor centralizado para funcionarem, como por exemplo o BitTorrent que é uma rede P2P com muita popularidade e muito usada para a partilha de dados e ficheiros digitais. Nestas redes, cada utilizador é um nó da rede que pode sair e entrar com relativa frequência, o que leva a que as redes P2P possam ter alguns problemas de fiabilidade, pois não é certo que um determinado nó da rede esteja ligado num determinado momento. A organização descentralizada das redes P2P permite distribuir a carga computacional, o tráfego da rede e o armazenamento por todos os nós que constituem a rede. Um exemplo duma rede P2P é o caso dos registos distribuídos baseados em *blockchain* que é uma rede que visa manter transacções seguras e não-anónimas sobre uma rede P2P descentralizada.

Segundo Gantz and Reinsel (2012) tem existido um aumento exponencial de dados ao longo dos últimos anos, o que leva a que exista uma necessidade maior de armazenar os dados produzidos remotamente, devido a um maior número de dispositivos móveis e à mobilidade destes. Os dados devem estar acessíveis em qualquer lugar em redor do globo de forma a que os dados estejam sempre disponíveis quando são necessários. Os serviços de armazenamento de dados (SA) são sistemas que nos permitem armazenar dados remotamente. Os SA vêm resolver os problemas mencionados acima relativamente ao aumento do número de dispositivos móveis e à mobilidade destes, de modo a acedermos aos dados de qualquer dispositivo em qualquer lugar do mundo. Os SA na nuvem (*cloud storage*) são SA georeplicados (espalhados pelo globo) onde estão presentes servidores físicos com dados armazenados e replicados para nos permitir aceder a estes sempre que necessitarmos com elevado grau de garantia de que os nossos dados estão disponíveis no momento em que necessitamos deles. Por norma o fornecedor deste tipo de serviço cobra-nos uma quantia pelo serviço. Os SA em redes P2P são sistemas que nos permitem aceder aos dados em qualquer dispositivo em qualquer lugar do mundo, como os *cloud storage*, mas em vez de serem usados servidores físicos de um fornecedor de serviços na nuvem (*cloud provider*), cada nó de uma determinada rede P2P fornece uma parte do armazenamento do seu dispositivo de forma a poder guardar dados de outros nós. De modo a possibilitar isto, é necessário que

os nós da rede que estejam ligados e conectados à rede sempre que possível, para permitir que quando os dados são acedidos, estejam disponíveis.

Nas redes P2P de não existe a garantia de haver um dispositivo permanentemente contactável dado que este pode sair ou falhar a qualquer momento. Nas redes de armazenamento P2P cada nó fornece uma fracção do seu armazenamento para a rede P2P onde vai guardar dados de outros utilizadores da rede, e espera em troca que os seus dados armazenados na rede estejam disponíveis quando necessitar destes. Nas redes P2P acontece com relativa frequência a entrada e saída de nós da rede o que leva a instabilidade na rede e consequentemente pode levar ao problema de um determinado nó ter dados de outrem e de repente sair da rede. Isto pode levar a que um utilizador que tinha os dados guardados na rede perca os seus dados, pois outro que tinha os seus dados guardados deixou de pertencer à rede.

Existe muita diversidade de nós ligados às redes P2P. Alguns destes nós possuem melhor conexão à Internet ou Internet mais rápida do que outros, enquanto alguns podem passar mais tempo conectados à rede. A qualidade de acesso à rede, o tempo que um nó passa ligado à rede e o números de entradas e saídas do nó da rede são características ligadas ao *churn*, enquanto que a rapidez, a qualidade e o espaço de armazenamento que um nó oferece à rede são características ligadas ao armazenamento.

Por um lado, dado que existe *churn* nos sistemas de armazenamento, necessitamos de replicação para que haja disponibilidade. Por outro lado, os efeitos associados ao churn perturbam o espaço total disponível, e seria importante saber quais os nós da rede que saem e entram desta com alguma frequência, para consequentemente lhes diminuir as garantias e o espaço de armazenamento. O processo de identificar os nós com elevado grau de churn pode fazer-se usando um sistema de reputação que permita aos nós saberem se outros nós são confiáveis relativamente à sua estabilidade na rede, o *churn*. O sistema de reputação vai permitir avaliar o comportamento de outros nós na rede de forma a que a rede saiba o comportamento de um determinado nó. Para auxiliar o sistema de reputação irá existir um sistema de incentivos que irá oferecer melhores recompensas consoante melhor for a reputação de um determinado nó, pois assim os nós vão-se sentir motivados a ter uma melhor reputação para obterem melhores recompensas e assim os nós acabam por reduzir o seu *churn* de modo a que a rede possa confiar mais nestes nós, enquanto nós com menor reputação vão possuir menor espaço de armazenamento na rede.

1.1 Objectivos

Com esta tese pretende-se desenvolver um protocolo para SA's P2P que mitiga o churn e o impacto causado por este nos dados armazenados no sistema, tanto a nível de indisponibilidade como a nível de os dados se manterem guardados no sistema e não se perderem. Na avaliação do protocolo foi implementado um SA P2P que implementa este protocolo, de seguida foram desenvolvidas simulações que exploraram mecanismos de *churn* e outros dados configuráveis que possam simular o comportamento dos utilizadores. Com recurso ao mecanismo de incentivos e de reputação são oferecidas incentivos aos nós e estes são avaliados e são utilizadas técnicas de compressão de dados de forma a reduzir o tamanho destes para se obter melhor eficiência no armazenamento destes e uma qualidade de serviço adequada na rede.

1.2 Estrutura do Documento

O resto do documento é organizada da seguinte forma. A Secção 2 apresentasse o trabalho relacionado, na Secção 3 os conceitos base, na Secção 4 a arquitectura do sistema, o mecanismo de incentivos e reputação, na Secção 4 a metodologia de avaliação e a avaliação do sistema proposto e a Secção 5 apresenta as conclusões e trabalho futuro.

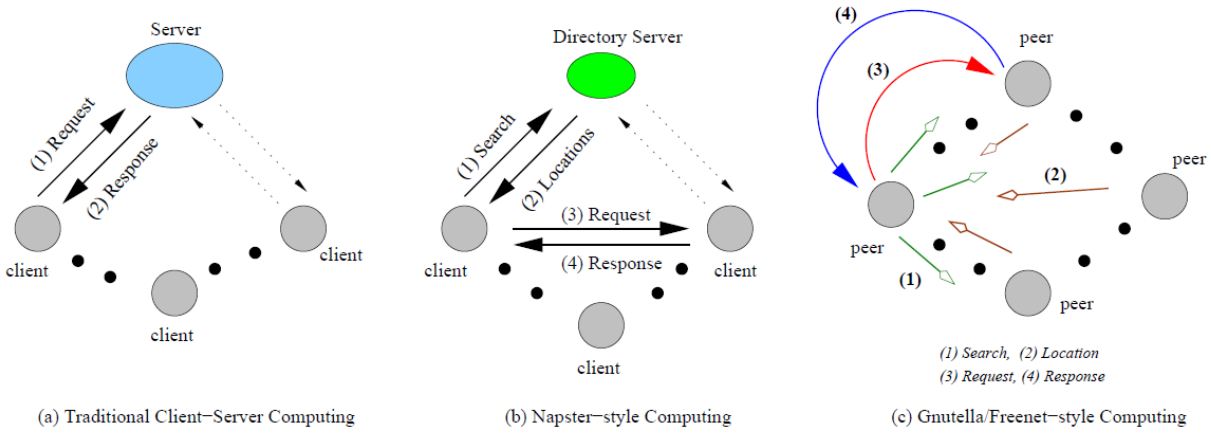
2 Trabalho Relacionado

Nesta secção de trabalho relacionado vamos falar sobre o que já foi feito sobre cada uma das áreas com que este trabalho se relaciona.

A primeira área que vamos explorar são as redes P2P. Esta área é importante pois este trabalho vai ser construído sobre uma rede P2P, portanto é importante perceber o que são redes P2P e perceber como funcionam. A segunda área que vamos abordar no trabalho relacionado é a dos sistemas de armazenamento em redes P2P. Querendo nós neste trabalho mitigar o churn em sistemas de armazenamento em P2P, temos de perceber o que são sistemas de armazenamentos e perceber também como estes funcionam. Outra área importante que vamos abordar no trabalho relacionado é a reputação. Para o nosso sistema ser eficaz no seu objectivo é necessário sabermos como avaliar os intervenientes do sistema de modo a podermos fazer uma distinção e dar melhores recompensas a quem tiver melhor comportamento, o que nos vai eventualmente levar a que possamos confiar mais nesses nós do sistema. Outra área abordada nesta secção é a dos simuladores de redes P2P. É necessário simular o sistema construído de forma a obtermos os resultados para a avaliação, então é necessário saber que simuladores de P2P existem e qual se enquadra melhor para simular o sistema construído. A última área abordada é a do armazenamento comprimido de dados para percebemos que algoritmos existem e quais as vantagens e desvantagens destes. De forma a percebemos que implicações isso vai trazer ao sistema quando se usam esses algoritmos. Estes algoritmos vão ser usados para comprimir os dados no sistema, de forma a aproveitar melhor o espaço que a rede tem ao seu dispor.

2.1 P2P

Nesta secção vamos descobrir o que são sistemas distribuídos entre pares (P2P) e descobrir o que se sabe sobre P2P, como defini-los, os seus modelos e classificações. De seguida vamos ver que tipos de sistemas P2P existem e quais as diferenças entre eles e no fim vamos descobrir qual o comportamentos que os nós têm nas redes P2P de modo a saber o que esperamos destes nós enquanto utilizadores de uma rede P2P.



(a) Traditional Client−Server Computing (b) Napster−style Computing (c) Gnutella/Freenet−style Computing

Figure 2.1: Os três tipos arquitecturas: rede cliente/servidor, rede P2P híbrida e rede P2P pura, respectivamente.

2.1.1 Definição - O que é P2P?

Schollmeier (2001) faz a distinção entre as redes P2P e as redes Cliente/Servidor (Fig 2.1a), que passa sobretudo por nas redes P2P existir o conceito de Servent que é uma palavra que deriva de servidor e de cliente, então um Servent nas redes P2P actua com servidor e como cliente ao mesmo tempo, devido ao facto de não existirem servidores centrais na redes P2P.

Schollmeier (2001) descreve P2P como uma arquitectura de rede distribuída, se os seus participantes partilharem parte dos seus recursos de hardware. Estes recursos partilhados são usados para fornecer o serviço e conteúdo oferecido pela rede. Estes ao serem partilhados numa rede P2P estão acessíveis por outros nós da rede directamente, sem passar por entidades intermédias.

Uma rede P2P pode ser classificada com "pura" ou como "híbrida". Uma rede P2P é uma rede P2P "pura" se for possível remover um qualquer nó arbitrário da rede sem que se perca o serviço da rede, (ver Fig. 2.1c). Numa rede P2P "híbrida" é necessário uma entidade central para fornecer partes dos serviços oferecidos pela rede, (ver Fig. 2.1b).

2.1.2 Modelos P2P e sua classificação

As redes P2P são classificadas de acordo com os seus modelos. Kant et al. (2002) define três modelos de redes P2P que são apresentados de seguida.

O modelo de partilha de conteúdo Kant et al. (2002) é muito comum nas redes P2P. Este modelo permite a partilha de conteúdo sobre o controlo de um servidor centralizado que mantém informação da disponibilidade e endereçamento sobre os nós do sistema e os meta-dados sobre

os ficheiros partilhados. Este modelo foi popularizado graças à solução Napster. Nas soluções do estilo do Napster, um nó quando está à procura de um determinado conteúdo/ficheiro faz uma pergunta à directoria do servidor central para saber qual o melhor nó que tem esse ficheiro e de acordo com a resposta obtida faz o pedido do conteúdo ao nó da rede indicado pela resposta que obteve anteriormente. A solução do estilo Napster devido à existência de um servidor central existe a possibilidade de falha centralizada, ou seja, caso o servidor central falhe, o sistema falha consequentemente. Então foi desenvolvido um modelo onde não existe um servidor central, este estilo foi popularizada pela comunidade Gnutella, que se baseia numa abordagem totalmente distribuída. Sendo uma abordagem completamente descentralizada, quando um nó se junta à rede necessita de conhecer pelo menos um nó que esteja dentro da rede, assim que este entra na rede obtém através do seu nó conhecido informação de todos os outros nós da rede. Neste modo de funcionamento, uma procura é iniciada pelo nó aos seus vizinhos e prossegue recursivamente até um determinado número de saltos. Cada procura tem um identificador único que previne que o nó receba respostas duplicadas para o mesmo pedido. Após receber as respostas estabelece uma ligação de modo a obter o conteúdo desejado. As soluções do estilo Gnutella não sofrem do problema de falha centralizada.

Kant et al. (2002) define ainda outro modelo de classificação das redes P2P é o modelo da partilha de recursos de hardware. Uma possível utilização deste modelo é resolver um problema computacionalmente complexo dividido em diversos problemas mais pequenos e depois disso distribuem-se esses pequenos problemas pelos diversos nós na rede para se resolver o problema concorrentemente. Deste modo os nós da rede partilham os seus recursos de hardware de forma a ajudarem-se mutuamente a resolverem os seus problemas.

Por fim, Kant et al. (2002) define o modelo de computação colaborativa que é um modelo com o objectivo de permitir aos utilizadores da rede colaborarem juntos, mesmo não estando juntos geograficamente. Um exemplo deste modelo de rede P2P é o Groove que é uma plataforma descentralizada que permite a interacção entre pequenos grupos. O Groove é um sistema que permite que equipas trabalhem juntas através da rede mesmo não estando juntas fisicamente, com o Groove, os utilizadores podem conversar, trabalhar e partilham informação. No Groove só é necessário recorrer a um servidor para o caso de descontinuidades da rede, como firewalls, proxies ou NAT's.

2.1.3 Tipos de sistemas P2P e exemplos

As redes P2P podem ser divididas em dois grupos consoante a sua arquitectura e o seu modo de funcionamento sendo estes o grupos das redes P2P estruturadas e o grupo das redes P2P não-estruturadas.

Estruturadas. O Pastry apresentado por Rowstron and Druschel (2001a) e o Chord apresentado por Stoica et al. (2001) são dois exemplos de redes P2P estruturadas. As redes P2P estruturadas são modeladas por Distributed Hash Tables (DHT) onde é atribuído um identificador a cada nó da rede. Neste tipo de rede a topologia da rede é controlada e permite que qualquer objecto seja localizado num número baixo de saltos na rede. O Chord é uma rede de sobreposição auto-organizada de nós onde cada nó da rede é posicionado em determinada posição de uma rede circular de acordo com o seu identificador que é atribuído quando o nó entra na rede. O Pastry, como o Chord, posiciona os nós numa rede circular mas o Pastry usa um sistema híbrido entre uma rede circular e em árvore. O Pastry é uma rede completamente descentralizada, resistente a falhas e escalável. Estes algoritmos P2P estruturados têm uma performance de procura de O(log N), onde N é o número total de nós da rede.

Não-estruturadas As redes P2P não-estruturadas ao contrário das redes P2P estruturadas não impõem qualquer estrutura na rede e geralmente, possuem uma fraca eficiência de procura que pode ir até O(N). Este tipo de rede P2P costumam suportar melhor o dinamismo dos nós na rede. Este tipo de redes P2P não impõem qualquer algoritmo para a organização ou optimização da rede. Por norma, os nós juntam-se à rede aleatoriamente sem conhecimento prévio da topologia da rede.

2.1.4 Comportamento de nós em P2P

O comportamento dos nós relativamente ao *churn* é caracterizado pelas entradas e saídas que estes fazem no sistema, isto é útil para a avaliação do comportamento e posteriormente o desenho e avaliação dos sistemas P2P. Stutzbach and Rejaie (2006) definem *churn* como a chegada e partida independente de milhares ou milhões de nós. Neste estudo, é concluído que o comportamento dos nós relativamente ao *churn* é semelhante em diferentes sistemas, o tempo de cada sessão de um nó da rede P2P não tem um tamanho exponencial e o tamanho das sessões dos nós P2P mantém-se correlacionado com entradas consecutivas no sistema.

No estudo realizado por Stutzbach and Rejaie (2006) mostra que em sistemas de distribuição de conteúdo que os nós que saem do sistema têm pouca probabilidade de voltar. No caso do

BitTorrent um nó que tenha saído tem ligeiramente mais probabilidade de reaparecer dentro de alguns minutos, depois disso tem uma probabilidade relativamente idêntica de reaparecer em qualquer momento entre 10 minutos e 1 semana, isto se este nó retornar à rede.

Relativamente ao tamanho da sessão, Stutzbach and Rejaie (2006) afirmam que o tamanho da sessão anterior do nó é um grande indicador do tamanho da próxima sessão do nó nas aplicações de partilha de ficheiros como o Kad e a Gnutella mas isto já não acontece nos sistemas de partilha de conteúdo como o BitTorrent.

Stutzbach and Rejaie (2006) afirmam que a disponibilidade de nós individuais em dois dias consecutivos na rede está fortemente correlacionada e que por norma, os nós costumam entrar uma única vez na rede P2P.

De forma a melhor lidar com o churn, os sistemas baseados em redes P2P devem ser capazes de lidar eficientemente com os nós que entram por alguns minutos na rede. De forma a aumentar a resistência contra o churn, os nós que por norma se mantém na rede por mais tempo, os chamados *long-lived peers*, devem manter estado sobre cada um dos outros. Os nós devem escolher os long-lived peers de forma a possuírem uma melhor conectividade e resistência contra o churn.

Visto que existe uma grande correlação no tamanho das sessões e na disponibilidade, uma aplicação P2P consegue prever/estimar o tamanho da sessão e a disponibilidade por dia baseado no comportamento anteriormente observado do respectivo nó.

2.2 Storage em P2P

Nesta secção do trabalho relacionado vamos ver trabalho anterior sobre armazenamento em P2P. Para isso vamos ver qual é a definição de storage em P2P, ou seja, o que o torna um sistema num sistema de armazenamento em P2P. De seguida, vamos ver o que já existe de storage em P2P, que sistemas foram criados e exemplos destes. Por fim, veremos quais os problemas de armazenamento em P2P e quais os motivos desses problemas.

2.2.1 Definição

Dabek et al. (2001) afirma que um sistema de armazenamento em P2P tem como objectivo levar a que os nós de uma determinada rede P2P disponibilizem armazenamento próprio para a

rede. De modo a que todos os participantes tenham armazenamento para os seus dados, a rede por sua vez partilha esse armazenamento disponível por todos os seus participantes.

2.2.2 Chord

De seguida veremos o protocolo Chord descrito por Stoica et al. (2001), que é um protocolo de tabela de dispersão distribuída que permite fazer procuras distribuídas. Resolve o problema de localizar eficientemente um nó da rede que possui um objecto em particular. Este protocolo suporta uma só operação: dado uma chave, ele mapeia a chave até um nó, dependendo da aplicação que use este protocolo, esse nó pode ser responsável por armazenar um valor associado a essa chave. O Chord simplifica o desenho de aplicações e sistemas P2P baseados no seu protocolo endereçando as seguintes dificuldades:

- **Balanceamento da carga** O Chord actua como uma função de *hash* distribuída espalhando as chaves por todos os nós do sistema.
- **Descentralização** O Chord é totalmente distribuído, dado que nenhum nó é mais importante que qualquer outro, isto aumenta a robustez.
- **Escalabilidade** O custo de procura cresce consoante o número de nós do sistema, mesmo em sistemas grandes o protocolo funciona bem.
- **Disponibilidade** O Chord automaticamente ajusta as suas tabelas internas de modo a reflectir os nós recém-chegados assim como os os nós que falharam assegurando-se que o nó responsável por uma determinada chave é sempre encontrado, mesmo que o sistema esteja sempre no estado de mudança.
- **Nomenclatura flexível** O Chord não impõe qualquer restrição na estrutura de chaves, isto permite oferecer à aplicação flexibilidade de como as chaves são mapeadas.

Neste protocolo não é necessário que cada nó conheça todos os outros nós pertencentes à rede, devido à informação ser distribuída um nó só necessita de manter informação acerca de outros O(log N) nós da rede e uma procura requer O(log N) mensagens. O protocolo Chord devido ao *consistent hashing* oferece as seguintes propriedades:

- Com probabilidade elevada, a função de *hash* equilibra a carga, ou seja, todos os nós recebem aproximadamente o mesmo número de chaves.

- Com probabilidade elevada, quando o n-ésimo nó entra ou da rede, apenas são movidas uma fracção de O(1/N) de chaves para uma localização diferente que corresponde ao mínimo necessário para manter a carga equilibrada.

A função de *consistent hashing* do Chord atribui a cada nó um identificador de m-bit usando uma função como o SHA-1 sobre o endereço de IP do nó enquanto a chave é produzida através do *hashing* da chave fornecida pela aplicação. O identificador m deve ser grande o suficiente para que a probabilidade de dois nós ou duas chaves coincidirem ser insignificante. Os identificadores são ordenados num anel circular com números entre 0 e 2^m-1. O *consistent hashing* é desenhado para deixar os nós entrar e sair da rede com interrupção mínima. Quando um nó entra na rede, algumas das chaves do seu sucessor passam para o nó recém chegado, enquanto que quando um nó sai da rede todas as suas chaves passam para o seu sucessor. Quando um nó armazena ou fornece o identificador a outro nó, este mantém junto ao identificador o endereço do respectivo nó, de forma a que sempre que seja fornecido um dado identificador seja também fornecido o endereço do respectivo nó.

Devido ao *consistent hashing* num ambiente distribuído pouca informação de encaminhamento é utilizada, dado que cada no só precisa de conhecer o seu sucessor na rede. O sucessor na rede de um nó é o nó que aparece imediatamente a seguir no anel. As pesquisas na rede podem ser feitas passando simplesmente de sucessor em sucessor no anel até encontrar o nó que sucede ao identificador, ou seja, até ao nó que a pesquisa mapeia. Esta abordagem é ineficiente pois requer que cada pesquisa tenha de atravessar o anel indo de vizinho em vizinho para encontrar a resposta à procura na rede. Para acelerar este processo é mantida informação adicional. Cada nó mantém um tabela de encaminhamento que possui o número de entradas tal como o número de bits (m) das chaves/identificadores, esta tabela é denominada de *finger table*. Cada entrada i-ésima da tabela contém o identificador do primeiro nó que sucede ao nó em pelo menos $2^i - 1$ no anel. Com a utilização desta tabela o tempo de procura por uma determinada chave é mais reduzido, dado que através do uso da tabela não é necessária que cada procura percorra nó a nó por todo o anel até chegar ao nó que detém essa chave. Esta tabela não possui informação de todos os nós da rede então quando faz uma procura por uma determinada chave na rede, procura o nó que precede o valor que pretende e pergunta-lhe por essa determinada chave e assim sucessivamente até encontrarem o nó responsável por essa chave.

Quando é efectuada uma pesquisa no anel por uma determinada chave ou um determinado identificador é obtido em resposta o primeiro nó com o identificador que sucede à chave/identificador da pesquisa. Primeiro o nó verifica se o identificador/chave se encontra

entre si e o seu sucessor, se não, procura na sua tabela de reencaminhamento pelo identificador do nó mais próximo que antecede a chave/identificador que pretende encontrar, de seguida pede a este nó que faça o mesmo procedimento e assim a pesquisa percorre vários nós sucessivamente até que a chave/identificador se encontre entre o próprio identificador e o identificador do sucessor. Assim que ocorrer este acontecimento, a pesquisa é terminada e é devolvido o identificador do sucessor da chave/identificador da pesquisa.

Numa rede dinâmica, os nós podem entrar a qualquer momento. De modo a que seja possível manter o correcto funcionamento do protocolo, cada nó possui um ponteiro para o nó que o antecede na rede. Após a entrada de um nó da rede este necessita de inicializar o seu predecessor e a sua tabela de encaminhamento, actualizar as tabelas e os predecessores dos nós existentes na rede.

A falha de um nó na rede não deve interromper as pesquisas existentes enquanto o sistema se estabiliza novamente. No pior dos casos, uma pesquisa é feita de sucessor em sucessor até chegar ao nó pretendido, de modo a alcançar isto, cada nó possui uma lista de sucessores que possui os sucessores mais próximos do nó. Assim quando um nó nota que o seu sucessor falha, pode repor a sua primeira entrada com o seguinte sucessor. A aplicação pode usar a lista de sucessores de forma a replicar dados.

2.2.3 Sistemas fundacionais

Nesta secção sobre a história de storage em P2P, vamos ver alguns sistemas sobre armazenamento em P2P e as suas propriedades. Os seguintes sistemas e as suas propriedades foram abordados no artigo de revisão Hasan et al. (2005).

O primeiro sistema de armazenamento é um sistema de armazenamento exclusivo de leituras na perspectiva dos clientes, é o caso do CFS escrito por Dabek et al. (2001). Só os editores podem escrever neste sistema. Os objectos neste sistema são blocos organizados através de tabelas de dispersão distribuídas para armazenamento destes. Este sistema tem como objectivos a escalabilidade, o balanceamento de carga, a robustez e a eficiência. O sistema de ficheiros é desenhado para um conjunto de blocos distribuído.

Um sistema que além de permitir leituras, também permite escritas é o Ivy de Muthitacharoen et al. (2002), um sistema de armazenamento distribuído e descentralizado baseado em P2P que suporta leituras e escritas. Este sistema é baseado num conjunto de *logs* e numa tabela distribuída de dispersão. Cada nó do Ivy guarda um *log* com as modificações feitas ao sistema que vão sendo sincronizadas entre os vários *logs* dos nós para que cada nó possa ter o

seu *log* actualizado. Este sistema permite em caso de partições da rede ser possível recuperar devido aos *logs* serem guardados indefinidamente. O estado total do sistema é a composição de todos os logs individuais. De modo a evitar leituras inteiras ao log, cada participante realiza cópias do seu estado actual.

Na parte de sistemas de publicação, o Freenet de Clarke et al. (2001) é um sistema de armazenamento distribuído que possibilita publicação, replicação e recuperação de dados enquanto mantém o anonimato dos autores. O Freenet opera como um sistema de ficheiros distribuído independente da localização através dos computadores que permitem que sejam inseridos guardados e pedidos ficheiros anonimamente. Os objectivos do Freenet são: roteamento, armazenamento dinâmico, anonimato e politicas descentralizadas, ou seja, uma rede descentralizada. O Freenet identifica os ficheiros através de chaves que são obtidas usando uma função de dispersão. Quando é recebido um pedido de inserção, o nó verifica se a chave já foi tomada. Em caso de colisões, tenta obter uma nova chave. Se a chave não for encontrada, é feita uma procura pela chave na tabela de roteamento.

O PAST de Rowstron and Druschel (2001b) é um sistema de gestão de armazenamento persistente de larga escala, baseado numa arquitectura P2P. O PAST possui o Pastry Rowstron and Druschel (2001a) como base de implementação, ou seja, é construído com o sistema de procura do Pastry, onde é atribuído um identificador de 128-bit a cada nó. O Pastry trata dos pedidos de procura no sistema. O PAST fornece 3 operações aos clientes que são de inserir um ficheiro e poder replicá-lo no sistema um número de vezes que desejar, fazer a procura de um ficheiro onde é retornada a cópia do ficheiro e ainda a operação de recuperar o armazenamento das cópias de um ficheiro.

Para finalizar esta secção, temos um sistema que fornece acesso distribuído a dados espalhados pelo globo, este sistema é o OceanStore de Kubiatowicz et al. (2000). Dado que este sistema usa sobretudo servidores não confiáveis e de guardar dados em qualquer lugar da rede, este usa encriptação. Além disso o OceanStore fornece alta disponibilidade. Os objectos são identificados através de um identificador global único (GUID) e são localizados por um de dois algoritmos do sistema. As escritas são restritas com uma lista de acessos de controlo (ACL) enquanto as leituras são efectuadas por quem possua uma chave. As actualizações do sistema são realizadas usando um protocolo bizantino entre as réplicas primárias e secundárias.

2.2.4 Sistemas recentes

Ao longo dos últimos anos foram desenvolvidas várias soluções para armazenamento de dados em P2P que veremos de seguidas essas mesmas soluções.

2.2.4.1 Eyo

O Eyo - Device-Transparent Personal Storage de Strauss et al. (2011) é um sistema de armazenamento pessoal que pretende fornecer transparência ao utilizador face a dispositivos desconectados fornecendo, para isso, o que é chamado de transparência de dispositivo, ou seja, o utilizador pensa no ficheiro X em vez de pensar no ficheiro X no dispositivo Y. O Eyo sincroniza actualizações entre os vários dispositivos, ou seja, assim que existe alguma actualização num determinado dispositivo, é propagada essa actualização para todos os dispositivos alcançáveis. Estas actualizações podem causar conflitos. De forma a resolvê-los o Eyo suporta resolução automática de conflitos, as aplicações necessitam de examinar o histórico de modificações de um simples objecto de forma a resolver os conflitos. No caso de existir modificações concorrentes no mesmo objecto o Eyo fornece à aplicação ambas as versões de cada objecto assim como as duas versões ancestrais em comum para que esta decida qual versão quer manter.

O Eyo separa os metadados do conteúdo do objecto e replica os metadados por todos os dispositivo de forma a que cada dispositivo tenha a capacidade de gerir cada objecto de qualquer dispositivo mesmo não tendo o conteúdo desse objecto. Isto tem como objectivo que o Eyo use os metadados como forma de procura do conteúdo do objecto. Na implementação do Eyo, um objecto pode ser um ficheiro de texto, uma fotografia ou qualquer outro tipo de dados, este sistema de armazenamento pessoal abstrai-se do tipo especifico de dados e vê tudo como um objecto.

Como já vimos, o Eyo separa os metadados do conteúdo dos ficheiros. Para o Eyo uma versão dos metadados consiste num conjunto de pares chave/valor. Uma aplicação pode tentar ler o conteúdo do objecto mas se este não existir, gera um erro. Mesmo que o conteúdo do objecto não exista, a aplicação pode fazer operações nos metadados. Se o utilizador quiser o conteúdo do objecto, o sistema usa os metadados de forma a ajudar a encontrar um dispositivo com o conteúdo do ficheiro. O Eyo suporta um sistema de consultas (*queries*) de forma a que possam ser feitas procuras pelos ficheiros que tenham metadados que correspondem à procura efectuada. Esta solução de armazenamento P2P tem um sistema que permite a aplicação definir regras para controlar que conteúdos de que objectos têm mais prioridade para serem armazenados em dispositivos de armazenamento limitado, mas devido a esta medida não funcionar como é

esperado, pois os utilizadores não são bons a preverem que objectos vão necessitar no futuro. A abordagem do Eyo que guarda os metadados em todos os dispositivos facilita o encontro dos objectos que faltam e corrige as regras definidas pelos utilizadores que o sistema ache que deveriam ser melhoradas. Este sistema suporta um histórico de versões dos objectos, permitindo que quando os dispositivos sincronizam um dado objecto e aparecem várias versões distintas, seja possível, através deste método, verificar a versão que se sobrepõe às restantes. Esta solução permite sincronização continua dos objectos de modo a ajudar a reduzir os conflitos derivados da concorrência propagando as mudanças do objecto rapidamente. De forma a suportar a sincronização contínua o Eyo possui um mecanismo de gestão automático de conflitos para quando várias modificações são efectuadas no mesmo objecto em dispositivos diferentes, para isso a aplicação necessita do acesso ao histórico de informação. De forma a sincronizar dois objectos entre os dispositivos, os metadados e o conteúdo, quando é feita a sincronização dos metadados, que por norma são pequenos, esta deve ser feita o mais rapidamente possível de forma a permitir a transparência entre dispositivos, de modo a produzir metadados idênticos após a sincronização dos dois dispositivos. Na sincronização do conteúdo que consiste em objectos maiores que alteram com menor frequência e leva algum tempo a enviar através de ligações redes lentas, o objectivo é mover os objectos para o melhor lugar de acordo com as regras de posicionamento. O principal objectivo da sincronização de metadados do Eyo é que esta decorra com a maior rapidez possível de forma a que existam cópias idênticas em todos os dispositivos. Além disso o processo de sincronização deve ser eficiente, ou seja, as actualizações devem fluir imediatamente.

2.2.4.2 WheelFS

Outra solução desenvolvida foi o WheelFS de Stribling et al. (2009), que é um sistema de armazenamento distribuído para ajudar aplicações a partilhar dados e obter tolerância a falhas. O objectivo do WheelFS é que seja usado por aplicações distribuídas. Um sistema de armazenamento distribuído de larga escala como o WheelFS deve possuir algumas propriedades-chave de forma a ser prático. Essas propriedades passam por ser útil para a construção de blocos de aplicações maiores, deve apresentar uma interface fácil de usar e deve permitir que diversos sites acedam aos dados quando necessários. Quando não for possível chegar aos dados, o sistema de armazenamento deve retornar uma falha ou encontrar uma cópia com rapidez e ainda deve permitir que as aplicações controlem que dados são guardados para obter tolerância a falhas e os objectivos de desempenho.

O WheelFS foi desenhado para ser corrido por vários sites distribuídos espalhados pela Internet. Este não protege contra falhas bizantinas. O WheelFS com a sua interface de sistema de ficheiros POSIX fornece uma hierarquia independente da localização física de directorias e de ficheiros. Neste solução, cada objecto pode ser uma directoria ou um ficheiro, então cada objecto tem um servidor de armazenamento primário que é responsável para armazenar o seu conteúdo. Os clientes quando pretendem aceder ou modificar um objecto consultam o primário. O WheelFS replica os ficheiros usando replicação primário/backup, ou seja, mantém cópias do objecto noutros servidores para caso o primário falhar ou esteja indisponível no momento que o cliente acede ao objecto. Para permitir um melhor desempenho no acesso a objectos populares que raramente são actualizados, o WheelFS permite que os clientes guardem em cache de forma a que outros clientes possam ler directamente sobre a cache dos clientes que tiverem o ficheiro em cache.

O WheelFS fornece *semantic cues* embutido no sistema de ficheiros do POSIX, os *cues* são úteis no contexto ao nível do armazenamento. O uso de *cues* permite à aplicação reduzir latência colocando dados perto de onde estes são necessários. O objectivo de um *cue* é fornecer informação adicional sobre o objecto a qual o *cue* pertence. Existem 4 tipos de *cues* que são as seguintes: localização, durabilidade, consistência ou de leituras grandes. A ideia passa por a aplicação especificar os *cues* no caminho do ficheiro, por exemplo, wfs/**.Cue**/data referia ao caminho /wfs/data com o *cue* **.Cue**, isto permitiria que a interface POSIX se mantenha inalterada, mas o uso de *cues* pode fazer com que quebre o software que reconhece os caminhos de ficheiros e assuma que a *.cue* seja uma directoria. é possível que o caminho tenha várias *cues* separadas por barras "/".

2.2.4.3 PRACTI

O PRACTI de Belaramani et al. (2006) é uma solução para a replicação de larga escala. O PRACTI fornece 3 propriedades às aplicações que o utilizam como sistema de replicação. A primeira propriedade é a propriedade de replicação parcial que consiste no sistema conseguir pôr parte dos dados ou metadados em qualquer nó do sistema, ao contrário de certos sistemas que requerem que o nó mantenha cópias de todos os objectos. Outra propriedade é a propriedade de consistência arbitrária o que permite à aplicação escolher se pretende consistência forte ou fraca dado que o sistema fornece ambas e permite que as aplicações escolham entre consistência forte e fraca. Por outro lado, as aplicações que não necessitam de consistência forte, não precisam de ter o custo de possuir esta consistência. A última propriedade do PRACTI é a propriedade da

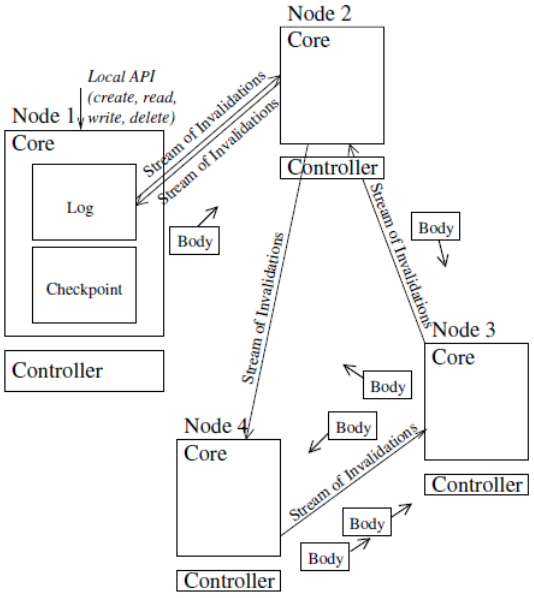


Figure 2.2: Arquitectura do PRACTI, retirado de Belaramani et al. (2006)

topologia independente que permite a qualquer nó trocar actualizações com qualquer outro nó da rede tal como outros sistemas P2P existentes.

Na Fig. 2.2 temos uma representação da arquitectura do PRACTI, onde o *Node 1* ilustra a estrutura de dados local de cada nó. O *Core* do *Node 1* contém os mecanismos do protocolo. As aplicações acedem aos dados guardados no *core* local através de uma *API* local de cada nó para ler, criar, apagar e escrever objectos. Essas funções operam através do *Log* e do *Checkpoint*. As modificações no nó local são juntas ao *log* e actualizadas no *Checkpoint* de forma a suportar as politicas de replicação parcial. O *Checkpoint* é o local onde ficam guardadas as alterações de modo a assegurar a vista consistente causal dos dados do sistema. Este mecanismo permite a cada nó escolher que subconjunto de dados pretende para armazenar localmente e trocá-lo a qualquer momento. De forma a suportar falhas nas leituras e actualizações entre os nós, os *cores* usam dois tipos de comunicação como ilustrado na figura 2.2: *Streams of Invalidations* ordenadas por ordem causal e *Body messages* desordenadas. O protocolo para enviar *Streams of Invalidations* assegura que o *log* e o *Checkpoint* de cada nó reflecte sempre uma vista causal consistente dos dados do sistemas. De modo a suportar replicação parcial dos dados existe separação das *invalidations* e dos *bodies*. As *invalidations* notificam o receptor que ocorreu uma escrita e as mensagens de *body* contêm o conteúdo da escrita. O *core* coordena a separação destas fontes de informação para manter consistência local. Os nós podem enviar *imprecise invalidations* em vez de enviar várias *invalidations* dado que uma simples *imprecise invalidation* pode substituir várias *invalidations* e ainda assim manter a replicação parcial.

2.2.5 Problemas

Nesta subsecção vamos ver quais os problemas que existem nos serviços de armazenamento em P2P. Hasan et al. (2005) apresenta os seguintes problemas nos serviços de armazenamento em P2P.

O primeiro problema apresentado é o facto de muitos dos sistemas serem projectos académicos, ou seja, os dados de performance analisados foram obtidos através de simulações em vez de serem obtidos de estudos realizados. Outro problema passa pelo facto de as aplicações de armazenamento P2P não serem capazes de fornecer uma alta largura de banda necessária aos requisitos necessários para certas aplicações. O *survey* escrito por Hasan et al. (2005) fala no problema de os mecanismos de atribuição de um número de identificação global serem, em alguns sistemas, ambíguos.

Na parte da segurança dos dados é apresentado o problema de os dados estarem armazenados em nós não confiáveis da rede espalhados pelo globo, o que pode levar a que existam acessos não autorizados aos dados por parte dos nós que estão a armazenar-los. Cifrar os dados fornece protecção mas isto não é capaz de bloquear os ataques de dicionário ou força bruta aos dados dos nós potencialmente maliciosos que pretendam obter acesso aos dados, comprometendo a sua segurança. É possível que algum utilizador malicioso possa tentar a realização de ataques de negação de serviço distribuído (DDOS) num nó da rede, podendo inserir ficheiros de lixo num determinado nó de forma a reduzir a performance deste. Estes utilizadores maliciosos podem ainda usar o *churn* de forma a baixar a performance do sistema, ou seja, saindo e entrando no sistema de forma a que leve que os dados estejam indisponíveis ou se percam mesmo, no pior caso.

Em serviços de armazenamento P2P, não é possível fornecer garantias de desempenho nem a garantia de atomicidade das operações.

2.3 Incentivos e reputação em sistemas P2P

Os sistemas P2P baseiam-se em contribuições dos nós pertencentes ao sistema, nestes sistemas é muito comum alguns dos seus participantes consumam recursos do sistema sem contribuírem, este comportamento é conhecido como *free-riding* segundo Piatek et al. (2007). Nestes sistemas é esperado que todos os participantes contribuam para um melhor desempenho do mesmo. Devido ao facto de as pessoas no geral se guiarem pelos seus próprios interesses leva a que exista *free-riding* nos sistemas P2P o que leva a uma degradação do serviço fornecido.

Uma solução para ajudar a mitigar o *free-riding* passa pela criação de incentivos que recompensem os participantes cooperativos do sistema. Deste modo os participantes são motivados a contribuírem mais para o sistema e assim o desempenho deste não é degradado devido à fraca ou mesmo à falta de contribuições de alguns participantes do sistema.

Piatek et al. (2007) conclui que o mecanismo de incentivos incorporado no BitTorrent desencoraja o *free-riding* por parte dos participantes. Buragohain et al. (2003) mostra que através de um mecanismo de incentivos que os participantes do sistema têm um comportamento mais altruísta e assim diminui o comportamento de *free-riding* destes.

Através do uso de incentivos, o sistema deve conseguir motivar os nós para terem melhor comportamento, de forma a que haja mais nós com mais contribuições para tornar o sistema mais robusto e com menos falhas.

Haddi and Benchaïba (2015) apresenta os objectivos que os incentivos devem promover uma melhor disponibilidade do serviço. Para isso o mecanismo de incentivo deve distinguir os utilizadores que colaboram com o sistema dos que não colaboram. O mecanismo deve prevenir, detectar e punir os utilizadores egoístas ou maliciosos, compensar os utilizadores altruístas e deve condicionar o acesso ao serviço, se necessário.

Um sistema de reputação deve resolver os seguintes problemas apresentados por Kamvar et al. (2003), de forma a ser eficiente e funcionar correctamente:

- O sistema deve ter auto-policiamento, ou seja, a ética dos utilizadores do sistema ser definida pelos próprios utilizadores em vez de uma entidade central.
- O sistema deve manter o anonimato. A reputação de um utilizador deve ser associada a um identificador do utilizador em vez de ser possível associar a reputação ao próprio utilizador.
- O sistema não deve fornecer nenhuma vantagem aos novos utilizadores, estes devem construir a sua reputação com base no bom comportamento e só após possuírem uma boa reputação, podem obter alguma recompensa.
- O sistema de reputação deve minimizar a sobrecarga causada em termos de cálculos, infraestrutura, armazenamento e complexidade das mensagens.
- O sistema deve ser robusto contra utilizadores maliciosos que possam actuar em grupos, de forma a que em conjunto consigam corromper o sistema.

No survey de Haddi and Benchaïba (2015) são apresentados mecanismos de incentivo sendo os seguintes baseados em reputação. Um esquema de reputação é um mecanismo onde é especificado como a reputação é calculada, guardada e disseminada. Neste esquema existem nós com mais privilégios que são entidades da rede que calculam, guardam e disseminam os valores de reputação dos restantes nós. Neste sistema após dois nós interagirem, ambos enviam a um dos nós com mais privilégios a sua avaliação do nó com quem interagiu. O valor da reputação é obtido através de uma função de distribuição beta. O valor da reputação de um nó bem comportado tem um pequeno aumento de cada vez enquanto o nó mau comportado tem o seu valor de reputação diminuído com uma valor maior.

Kamvar et al. (2003) explica o exemplo do sistema de reputação do eBay, neste sistema os compradores e os vendedores classificam-se um ao outro depois de cada transacção e a reputação geral de um participante é a soma das classificações dos últimos 6 meses. Um exemplo da maneira de funcionamento do sistema de reputação é um nó votar positivamente (valor=1) caso a experiência com o outro utilizador correu bem ou votar negativamente (valor=-1), caso a experiência tenha corrido mal.

De seguida veremos os aspectos mais importantes do algoritmo de reputação apresentado por Kamvar et al. (2003) chamado EIGENTRUST. Neste algoritmo é atribuído um valor único chamado "valor de confiança global" a cada nó da rede. Este valor reflecte a experiência de todos os nos da rede com esse determinado nó. Para um melhor funcionamento do algoritmo, todos os nós da rede participam no calculo desses valores. A reputação global de cada nó é atribuído pelos outros nós. De modo a que os nós maliciosos não corrompam o sistema, neste algoritmo todos os valores de votação devem ser compreendidos em 0 e 1. Neste algoritmo o "valor de confiança global" é dado pelo calculo do *left principal eigenvector* de uma matriz dos valores de confiança locais de cada nó. Através deste cálculo, Kamvar et al. diz que se houver um número n de nós elevado, então o vector de confiança vai convergir para o mesmo vector para cada nó i do sistema. Por outras palavras, o vector de confiança global quantifica quanto o sistema confia como um todo em cada nó.

3 Conceitos Base

Nesta secção de conceitos base vamos apresentar conceitos que servem de base a este trabalho. Primeiro exploramos os simuladores P2P e de seguida a compressão de dados.

3.1 Simuladores P2P

Desenvolver protocolos para sistemas P2P é uma tarefa complexa pois requer o teste da infraestrutura e a avaliação dos protocolos. Em caso de redes P2P de larga escala, torna-se uma tarefa não trivial de realizar pois devido às dimensões das redes P2P de larga escala. O teste de algoritmos distribuídos é uma tarefa complexa pois é necessário o monitoramento individual de cada elemento do algoritmo. De forma a mitigar estes problemas são usados simuladores de redes P2P que permitem simular o comportamento de uma rede P2P sem a necessidade de montar uma rede P2P física.

3.1.1 Propriedades

De modo a avaliar simuladores P2P, Surati et al. (2017) apresenta as seguintes propriedades de simuladores P2P que são úteis para avaliar os simuladores e permitir uma escolha adequada para cada situação.

3.1.1.1 Arquitectura do simulador

A arquitectura especifica as características no desenho e no funcionamento do simulador e funcionalidades básicas apresentadas a seguir.

- Estrutura da rede P2P - Esta característica indica qual ou quais as estruturas de rede P2P suportadas pelo simulador podendo ser estruturadas, não-estruturas ou ambas.
- Modo do simulador - O modo de simulador indica se o simulador foi projectado para dar suporte à simulação de eventos discretos ou à simulação baseada em ciclos ou em ambos. Na

simulação de eventos discretos, a operação do sistema é representado como uma sequência de eventos, ou seja, cada evento ocorre num instante de tempo e marca a mudança do estado do sistema. Enquanto uma simulação baseado por ciclos é uma simulação sequencial onde cada acção do protocolo é executada por cada nó sequencialmente em cada ciclo.

- Simulação da rede subjacente - Os simuladores P2P podem ter diferentes formas de simular a rede subjacente, podendo ser baseados em pacotes ou baseados em fluxo. A simulação baseada em pacotes simula os pacotes assim como as ligações e calcula a largura de banda, atraso e roteamento dos pacotes. Enquanto os simuladores baseados em fluxo abstraem os detalhes do nível de simulação, não tendo em atenção os detalhes anteriormente referidos. Devido a todos os detalhes simulados pelos simuladores baseados em pacotes, estes demoram mais para completar as simulações.

- Simulação do protocolo subjacente - Os simuladores P2P podem ser classificados como simuladores genéricos, específicos de protocolos ou específicos de domínio. O primeiro tipo de simuladores pode ser usados para simular qualquer tipo de aplicação P2P, enquanto os restantes são desenhados para simular um protocolo especifico ou um domínio especifico, respectivamente.

- O suporte à simulação paralela distribuída - Um critério a ter em conta é o facto de os simuladores suportarem simulações a correr em várias máquinas para obter uma maior escalabilidade e um menor tempo de simulação versus simulação feita sequencialmente numa única máquina.

- O suporte ao churn - O comportamento de um nó num ambiente dinâmico é simulado com uma taxa de churn, ou seja, a taxa com que os nós saem e entram da rede numa unidade de tempo. Esta é uma propriedade importante para este trabalho, pois a ideia é mitigar o churn em rede P2P, então é importante que o simulador suporte churn.

3.1.1.2 Usabilidade

A propriedade da usabilidade é usada para medir a facilidade de aprender a usar o simulador. Os simuladores que tenham melhores documentações que facilitem a sua aprendizagem são preferíveis para permitir uma melhor adaptação do utilizador ao simulador.

3.1.1.3 Escalabilidade

O tamanho da rede, ou seja, o número de nós que podem ser simulados pelo simulador define a escalabilidade deste, por norma, os protocolos de aplicações P2P são geralmente grandes com recursos a muitos nós na rede, o que torna a escalabilidade uma propriedade importante para avaliar um simulador.

3.1.1.4 Estatísticas

Outra propriedade chave são os resultados produzidos pelo simulador e a forma que são armazenados para referências futuras O resultado produzido deve ser expressivo e fácil de manipular de forma a obter análises estatísticas.

3.1.2 Apresentação e comparação dos simuladores

Surati et al. (2017) mostrou os resultados de uma pesquisa sobre vários simuladores de redes P2P, onde são apresentados vários simuladores e as suas características. Ainda neste survey, é afirmado que os simuladores genéricos são preferíveis aos simuladores específicos de protocolos e de domínio. O survey Surati et al. (2017) apresentou uma tabela com o resumo das vantagens e das desvantagens dos vários simuladores P2P estudados, ver Fig. 3.1.

Após a análise da Fig. 3.1, podemos retirar os pontos fortes e os pontos fracos de cada um dos simuladores apresentados, e assim percebemos qual o simulador mais indicado para a simulação deste trabalho. No entanto, existe um simulador chamado Corten escrito por Sequeira (2019), que não é apresentado neste survey. O Corten é um simulador escrito em Rust que traz algumas características interessantes que não estão presentes nos simuladores apresentados na Fig. 3.1.

O Corten permite a modelação de assincronia, ou seja, permite que um método local possa ser executado antes ou depois do tempo, simulando a falta de sincronização entre nós. O Corten ainda permite guardar estado de uma determinada simulação, guardando todo o estado do sistema de modo a permitir que seja possível retornar à simulação mais tarde, ou então repetir várias vezes a simulação a partir do estado guardado. Na Fig. 3.2 é possível comparar as diversas características pertencentes a cada simulador.

Após uma analise das características mais importantes dos simuladores para a simulação do trabalho realizado, deve ser escolhido o simulador mais adequado para tal.

Simulator	Favorable aspects	Limitations
PeerfactSim. KOM	• Offers existing implementations for a variety of P2P protocols • Consists of a modular architecture and flexible tools to configure simulation setups and to create scenarios • Provides logging and statistics architecture for debugging and collection of simulation states • Provides an add-on visualization component for understanding of the simulation of P2P-networks • Supports large scale P2P systems	• May be complex to understand as multiple layers with many interfaces are provided
D-P2P-Sim	• Supports distributed environment • Powerful integrated GUI for statistic data observation, collection and analysis • Implemented as close as possible to an application level P2P software • Extremely high scalability • Enhanced as D-P2P-Sim + to include multi-million node simulation support, failure-recovery models simulation capabilities and more statistics	• No visualizer to visualize the topology
ProtoPeer	• Easy switching between the simulation and live network deployment without changing a line of code • Defines peerlets for re usability of code and unit testing	• Documentation is not provided in detail • No information is provided for the interactive visualizer
PeerSim	• Very high scalability • Cycle based and Event based simulation model • Support some well known models • Supports dynamic network	• Do not have details of underlying communication network • No support of distributed simulation • Only cycle based engine is documented
RealPeer	• Extensible and reusable elements in the framework • Follows the new concept of simulation-based development of P2P systems	• No interactive visualizer provided • Scalability is lower • No support of distributed simulation
OMNeT++	• Vast GUI support • Support for parallel distributed simulation execution • Reusability of simulation models • Multitier topologies are supported • Has a powerful GUI execution environment	• Low scalability • Less protocols implemented for P2P systems
OverSim	• Good GUI interface for validation and debugging new or existing overlay protocols • Highly scalable • Supports IPv4 and IPv6 as well as UDP and TCP [35]	• Overlay protocol has to provide at least a key-based routing interface (KBR) to the application
Overlay weaver	• Capabilities for distributed simulation • API is clean and well designed so that source is quite readable • Provides multiple routing algorithms	• Statistics gathering need a lot of work • Visualizer reduces the performance of the emulator • Documentation is quite scattered
PlanetSim	• Output network topology graph in GML and pajek formats • Provides high quality software, layered design and the entities that can easily be replaced • Available as P2P overlay, latency-aware overlay, bandwidth-aware overlay, multi-overlay simulation as well as Ant algorithms for suitability of researchers • API is clean and well designed so that source is quite readable • The framework is extensible at all levels • Complete transparency to services running either against the simulator or the network	• Possible to visualize the overlay topology at the end of a simulation run, but there is no interactive GUI • A very simplified underlying network layer without consideration of bandwidth and latency costs, hence, its difficult to simulate heterogeneous access networks and terminal mobility
Dnet	• Supports parallel execution using multiple simulators	• Not very well documented • P2P protocol implementation is not given • Statistics gathering is not well defined
3LS	–	• Supports only unstructured overlays • No support of dynamic network • Very low scalability
Optimal-sim	• Simulates churn model of nodes • Supports dynamic network • Provides BRITE interactive visualizer tool	• No support of distributed simulation • Scalability is not good
NS-2	• Supports visualization using NAM (Network AniMator) • Runs parallel with other machines • Provides extensive documentation	• Not a specific P2P simulator as works at network layer • Very low scalability

Figure 3.1: Esta tabela retirada de Surati et al. (2017) mostra algumas propriedades de alguns simuladores genéricos.

	simulation	emulation	real execution	packet loss	latency	jitter	reproducibility	number of nodes	churn	logging	asynchrony	checkpointing
PeerSim	✓			✓	✓	✓	✓	10^6 * 2.5×10^5 **	✓	✓		
Optimal-sim	✓			?	?	?	?	10 000	✓	✓		
PlanetSim	✓						✓	100 000	✓	✓		✓
NDP2PSim	✓			✓	✓	?	?	480	✓	✓		
D-P2P-Sim	✓						✓	400 000†	✓	✓		
PeerfactSim.KOM	✓			✓	✓	✓	✓	50 000‡	✓	✓		
DistAlgo	✓			?	?	?	?	?	?	✓		
ProtoPeer	✓		✓	✓	✓	✓	✓	50 000	✓	✓		
Neko	✓		✓	?	?	?	?	§	?	✓		
RealPeer	✓		✓				✓	20 000	?	✓		
OverlayWeaver		✓	✓	✓	✓	✓		4 000	✓	✓		
SPLAY		¶	✓				✓	500†	✓	✓		
P2			✓							✓		
Mace			✓							✓		
Corten	✓		✓‖	✓	✓	✓	✓	10^6	✓	✓	✓	✓

Figure 3.2: Esta tabela retirado de Sequeira (2019) mostra as características dos vários simuladores apresentados.

3.2 Armazenamento comprimido de dados

Nos sistemas de armazenamento P2P os nós participantes da rede vão ter direito a um certo espaço para poderem guardar dados na rede. Este espaço varia de acordo com o espaço que o próprio nó oferece para outros nós da rede guardarem dados. A rede só tem disponível a quantidade de espaço que os nós oferecerem, ou seja, o espaço para armazenar dados na rede é um recurso limitado. Sendo o espaço de armazenamento um recurso limitado, então é necessário aproveitar este espaço da melhor maneira possível, sem desperdiçar espaço de armazenamento. Para isso, iremos usar duas técnicas que vão permitir reduzir armazenar dados comprimidos.

3.2.1 Deduplicação

A deduplicação é uma técnica que permite poupar armazenamento para armazenar o mesmo conjunto de dados, evitando que existam várias cópias dos mesmos dados, para isso são identificados pedaços únicos de dados ou simplesmente padrões de bytes durante o processo de análise. Ainda durante o processo de análise, sempre que é encontrado um pedaço de dados igual a outro já guardado anteriormente, simplesmente é guardado uma referência que aponta para o pedaço que já estava armazenado anteriormente. Os sistemas de deduplicação guardam uma única cópia do pedaço juntamente com metadados sobre como reconstruir o ficheiro original com base nos pedaços. Reduzir a sobrecarga que esta técnica introduz num sistema é um grande desafio que é

Programme	Total CPU time/s compress	decompress	Total compressed size (bytes)	mean bits/char
compress	9.6	5.2	1246286	3.63
gzip	42.6	4.9	1024887	2.71
Alg-C/D	51.1	9.4	856233	2.43
comp-2	603.2	614.1	848885	2.47

Figure 3.3: Esta tabela retirada de Burrows and Wheeler (1994) mostra a comparação entre vários algoritmos de compressão.

necessário resolver, pois em implementações mais básicas, pode levar a que cada *fingerprint* de cada pedaço seja comparado com todos os outros pedaços. A *fingerprint* é o resultado de um procedimento que que tem como objectivo identificar exclusivamente os dados originais, ou seja, *fingerprint* de um ficheiro é algo que permitir identificar esse ficheiro entre todos os outros.

Christen (2011) apresenta quatro passos no processo de deduplicação. O passo de limpeza dos dados e a uniformização destes é o primeiro passo e o passo crucial neste processo, o foco deste primeiro passo é a conversão dos dados em bruto para uma forma de dados melhor estruturados, assim como a resolução de inconsistências na forma como os dados estão codificados. O segundo passo, passa pela indexação que gera pares de registos candidatos que irão ser comparados em detalhe no passo seguinte, o passo da comparação. No passo da comparação vários campos são comparados para cada par de registos, que resulta num vector que contém um valor de semelhanças calculado para esse par. Estes valores são usados no passo seguinte que classifica os pares de registos candidatos anteriormente comparados como correspondentes, não-correspondestes e possível correspondência.

3.2.2 Algoritmo de compressão

Como já vimos, aproveitar da melhor forma possível o armazenamento disponível neste sistema é um desafio a enfrentar. Outra técnica que vamos ver é a compressão de dados que consiste em transformar os dados armazenados por um utilizador de forma a ocuparem um menor armazenamento, ou seja, os dados comprimidos. deve ser possível recuperar dos dados comprimidos para os dados originais de forma a que o utilizador tenha acesso aos seus dados que armazenou tal como os armazenou na rede.

Na Fig. 3.3 é apresentada um tabela com a comparação de vários algoritmos de compressão. Nesta tabela estão presentes os resultados após testes entre 4 algoritmos, Burrows and Wheeler (1994) realizou esses testes. De seguida apresentamos um desses testes. Neste teste podemos

observar que o algoritmo comp-2 possui uma melhor taxa de compressão mas a sobrecarga que este impõe é enorme e o algoritmo Alg-C/D possui uma boa taxa de compressão e a sobrecarga imposta não é exageradamente grande, por isso acreditamos ser um bom algoritmo. O Alg-C/D que são dois algoritmos apresentados por Burrows and Wheeler (1994).

4 Sistema

O objectivo desta dissertação é desenvolver um SA P2P que mitigue o churn e o impacto causado por este nos dados armazenados no sistema, tanto a nível de indisponibilidade como a nível da persistência dos dados. Para isso, iremos utilizar mecanismos de incentivos e reputação para avaliar os nós do sistema e aliar a estes mecanismos, técnicas de compressão de dados de forma a reduzir o tamanho dos dados para obter melhor eficiência no armazenamento destes. Neste sistema, cada utilizador vai poder armazenar dados na rede e, em contrapartida, cada participante vai oferecer armazenamento à rede de forma a que exista armazenamento disponível para que seja possível para outros armazenarem os seus dados. No sistema, os participantes podem fazer diversas operações como armazenar objectos (escritas), obter os seus objectos (leituras) e remover os seus objectos na rede (remoções).

O mecanismo de reputação e de incentivos que apresentamos de seguida têm o objectivo de motivar os utilizadores a passarem mais tempo conectados à rede e assim mitigar o churn e, consequentemente, os efeitos causados por este, tais como a indisponibilidade de dados e/ou a perda definitiva destes. No sentido de motivar os participantes a não abandonarem o sistema será medida a sua permanência no sistema bem como as operações realizadas com sucesso e oferecido mais espaço de armazenamento aos nós que melhor cooperarem no serviço de armazenamento. Para a utilização destes mecanismos num determinado sistema é necessário que este sistema seja implementado numa rede P2P e que seja um sistema de armazenamento de dados. Este sistema tem de permitir escritas, leituras e remoções de objectos na rede. Para melhorar a eficiência da rede e dos mecanismos propostos aconselha-se o uso de compressão de dados de forma a reduzir o tamanho dos objectos para uma melhor eficiência no armazenamento destes.

O modelo de falhas considerado é o modelo de falha por paragem. Neste modelo os nós podem falhar por paragem, deixar de ser possível contactá-los através da rede ou os próprios nós deixarem de poder contactar outros nós através da rede. Estas falhas podem ser permanentes, caso um nó falhe e nunca mais recupere da falha e consequentemente não volta à rede ou pode ser temporária, ou seja, o nó eventualmente recupera da falha e volta à rede. Assumimos que todos os participantes do sistema não têm falhas bizantinas o que não é realista mas no contexto deste trabalho, isto foi assumido.

4.1 Mecanismo de reputação e incentivos

De forma a diminuir o *churn* queremos incentivar os participantes a passarem mais tempo conectados à rede para isso vamos oferecer incentivos para que estes estejam mais tempos conectados à rede. Nesta subsecção vamos apresentar o mecanismo de reputação e incentivos proposto.

O sistema de armazenamento baseia-se em contribuições por parte dos nós participantes e como o espaço total da rede é o espaço oferecido por todos os nós do sistema, é necessário aproveitar o espaço de armazenamento eficientemente. Por isso, o sistema de reputação é complementado com técnicas de compressão de dados de forma a tornar o armazenamento eficiente.

Um nó, quando se junta à rede, compromete-se a armazenar e a fornecer um determinado objecto quando este é solicitado pelo seu proprietário e, em troca, espera também ele poder armazenar objectos na rede e obtê-los de volta quando assim o entender. As operações executadas pelos nós da rede podem ser concluídas com sucesso ou podem falhar; no caso de uma escrita efectuada, esta pode ser concluída com sucesso se o objecto ficar armazenado ou pode falhar se o objecto não ficar armazenado. No caso das leituras, caso seja concluída com sucesso, o objecto pretendido é retornado ou no caso de falhar não é retornado nenhum objecto. O resultado destas operações podem ser medidas consoante o sucesso destas e assim medir o comportamento dos seus participantes.

De forma a avaliar a reputação de um nó do sistema, é usado um algoritmo de reputação em parte, baseado no algoritmo *Eigentrust* proposto por Kamvar et al. (2003). No algoritmo *Eigentrust* são contabilizadas as transacções positivas e negativas entre todos os nós do sistema, enquanto que no algoritmo proposto é contabilizado o tempo que um nó passa ligado à rede e o tempo que este se ausenta da rede. Esta adaptação é feita pois pretende-se avaliar a disponibilidade dos nós no sistema em vez das transacções efectuadas entre estes. Isto é contabilizado por um mecanismo de *pings*, onde cada nó comunica periodicamente de forma a mostrar que está ligado e presente na rede. Após falhar um conjunto de *pings* é assumido que o nó falhou. Os *pings* do nó i efectuados com sucesso *suc(i,j)* e os *pings* que não foram efectuados são denominados sem sucesso *insuc(i,j)*. Os *pings* são contabilizados a partir do momento em que o nó entra na rede e envia o seu primeiro *ping*. Tal como no algoritmo acima identificado, o valor s_{ij} define as diferenças entre os sucessos e insucessos e é definido por:

$$s_{ij} = suc(i,j) - insuc(i,j) \tag{4.1}$$

De forma a agregar os valores de confiança locais, tal como é sugerido no algoritmo, é

necessário normalizar estes valores. Os valores normalizados são denominados de valores de confiança normalizados e são dados pela seguinte formula:

$$c_{ij} = \frac{max(s_{ij}, 0)}{\sum_j max(s_{ij}, 0)} \tag{4.2}$$

A soma de todos os c_{ij} de um nó j é 1. Periodicamente cada nó faz o calculo de reputação baseados nos valores c_{ij} que possui. O valor de reputação é sempre um valor positivo. Para o calculo da reputação é necessário calcular a média ($\bar{x}$) e o valor do desvio padrão (σ) dos valores de c_{ij}. O valor de reputação do nó i ,r_i, é dado pela seguinte formula:

$$r_i = \frac{max(c_{ij} - \overline{x}, 0)}{\sigma} * 100 \tag{4.3}$$

Os valores de reputação são armazenados pelo respectivo nó que efectuou o cálculo. Os valores de reputação são valores reais maiores ou iguais a zero. São oferecidos incentivos a quem possui melhor reputação. Estes incentivos passam pela quantidade de armazenamento oferecida a cada nó. Esta quantidade de armazenamento oferecida a cada nó é relativa ao armazenamento que este ofereceu à rede. A tabela 4.1 devolve a quantidade de armazenamento relativo a que um nó tem direito devido à sua reputação. Não existem valores de reputação negativos devido à formula 4.3. Na tabela 4.1 a coluna da esquerda representa os valores possíveis de reputação que os nós podem tomar enquanto a coluna da direita representa as recompensas que obterão em função da reputação que possuírem.

Os *pings* efectuados pelos nós são enviados a qualquer nó que esteja sempre contactável e que seja responsável pelo cálculo da reputação ou então a um conjunto de nós que fiquem responsável pelo cálculo (que poderão ser todos os nós da rede). Se for um conjunto de nós responsáveis pelo cálculo, estes, por sua vez, partilham entre si os *pings* recebidos de forma a assegurar que os valores estão consistentes entre si. Após a agregação dos valores de *pings* recebidos, cada um calcula o valor de reputação de cada nó. Dado que os valores estarão todos sincronizados, os valores de reputação dos participantes será igual entre todos os nós do conjunto de cálculo da reputação.

Segundo Piatek et al. (2007) os incentivos desencorajam o comportamento de *free-riding*. Com a utilização de incentivos os utilizadores tendem a melhorar o seu comportamento o que leva a que exista uma redução do *churn* e dos efeitos causados por este.

Para um armazenamento mais eficiente dos dados no SA P2P é recomendado o uso com-

Reputação	Recompensa
0	50%
0-25	75%
25-50	100%
50-100	125%
100 - 150	150%
150 - 200	175%
200+	200%

Table 4.1: Tabela reputação/beneficio

pressão de dados de forma a que se consiga um melhor aproveitamento do espaço disponível e assim uma melhor qualidade do serviço fornecido.

4.2 Implementação

Para a posterior avaliação do mecanismo de incentivos e reputação vamos implementar o algoritmo num sistema de armazenamento P2P descrito nesta secção.

O sistema vai possibilitar o armazenamento de objectos na rede e posterior acesso aos mesmo quando for necessário. O sistema vai funcionar sobre uma rede P2P híbrida. Apesar de de uma rede P2P híbrida ser mais centralizada em relação a uma rede P2P pura, a utilização destas é preferível pois existe a necessidade de, quando um nó pretende juntar-se à rede, ter conhecimento de outro nó pertencente à rede. Para isso é necessário que esteja sempre um nó contactável. Para possibilitar a entrada de nós na rede e de forma a não existir um único ponto de entrada na rede que se poderá tornar um ponto de estrangulamento do sistema, existe um conjunto de nós que vai permitir a entrada de outros nós na rede. Este conjunto de nós, denominado de *ring service*, vai permitir dar estabilidade ao sistema pois estes nós vão estar a maioria do tempo conectados à rede. Caso algum dos nós do *ring service* falhe, existirão outros que possibilitarão a entrada de nós na rede.

Cada nó do sistema vai correr um cliente. Este cliente vai permitir ao utilizador juntar-se à rede, seguir o protocolo interino de disseminação de mensagens e realizar as seguintes operações:

- Escrever - Esta operação permite que o utilizador armazene um dado objecto no sistema.
- Modificar - Esta operação permite que o utilizador modifique um dado objecto já armazenado no sistema.

- Ler - Esta operação permite que o utilizador obtenha um dado objecto que anteriormente armazenou no sistema.
- Apagar - Esta operação permite que o utilizador remova um dado objecto anteriormente armazenado no sistema.

O protocolo de mensagens usado para o envio destas entre nós na rede é o TCP (*Transmission Control Protocol*). Este protocolo fornece entrega confiável, ordenada e com verificação de erros de pacotes entre aplicações. Com a utilização deste protocolo pode-se assumir que não existe perda de mensagens na rede, o que quer dizer que, se algum nó da rede envia uma mensagem a outro, ambos estiverem ligados e não ocorrer nenhuma falha, então a mensagem é entregue com sucesso e o remetente obtém a confirmação que a mensagem foi entregue.

4.2.1 Protocolo de entrada e disseminação de mensagens

O sistema é baseado no protocolo Chord descrito por Stoica et al. (2001). Devido a este protocolo permitir encontrar um determinado objecto num ambiente distribuído dada uma chave. As propriedades oferecidas pelo Chord como o balanceamento de carga por todos os participantes do sistema, a descentralização, a escalabilidade, a disponibilidade e a nomenclatura flexível são propriedades que interessam na construção do sistema. A disseminação das mensagens no sistema, a entrada de nós do sistema e a organização interna dos nós pelo sistema são baseados neste protocolo. Este protocolo dispõe os nós pertencentes em forma de um anel organizados de acordo com o seu identificador.

Na inicialização do sistema, o primeiro nó vai inicializar o sistema e o anel. Para inicializar o anel, este nó começa por criar um ponteiro do seu sucessor e do seu predecessor para si próprio dado que é o único elemento na rede. De seguida preenche cada entrada da sua tabela de reencaminhamento com um ponteiro para si próprio. Os restantes nós pertencentes ao *ring service* entram na rede seguindo o processo de entrada que é descrito de seguida contactando o primeiro nó que iniciou o serviço.

Sendo que, para um nó entrar na rede, tem que ter conhecimento de um nó sistema, qualquer nó que pretenda entrar tem de contactar um nó do *ring service* que esteja ligado à rede. No processo de entrada o nó envia mensagem a um elemento do *ring service* com o seu identificador. O nó contactado faz uma pesquisa na rede pelo sucessor do identificador fornecido, e assim que obtiver essa resposta, envia ao nó que pretende entrar o identificador e o respectivo endereço do sucessor.

Dado que, após a entrada de um nó da rede, o anel não fica consistente, o sistema possui um mecanismo que permite corrigir essas inconsistências. Cada nó do sistema precisa de seguir o protocolo Chord para a estabilização do anel, que consiste nas seguintes operações:

- Estabilizar: cada nó periodicamente envia uma mensagem ao seu sucessor e este responde-lhe com o seu antecessor. Após receber a resposta verifica se esse nó se encontra entre si e o seu sucessor, se sim, actualiza o seu ponteiro do sucessor para o identificador do nó que recebeu como resposta e de seguida notifica o seu sucessor actualizado a indicar que é o seu antecessor. O nó ao receber a notificação verifica se esse nó é efectivamente o seu antecessor. Se for, então actualiza o seu ponteiro do antecessor.

- Verificar o antecessor: cada nó regularmente precisa de verificar se o seu antecessor não falhou. Caso detecte que o seu antecessor tenha falhado, remove o ponteiro.

- Verificar o sucessor: cada nó regularmente verifica se os sucessores na sua lista não falharam. Caso detecte que algum nó da sua lista tenha falhado remove-o da lista e volta a organizar a lista.

- Corrigir as entradas na tabela de reencaminhamento: periodicamente, cada nó do anel actualiza uma entrada aleatória da sua tabela. Para actualizar a i-ésima entrada, faz uma pesquisa na rede pelo sucessor do identificador $n + 2^i - 1$ onde n é o identificador do próprio nó. Após obter a resposta da pesquisa efectuada, guarda o ponteiro para o nó que possui esse identificador na respectiva entrada da tabela.

O sistema encara dois tipos de saídas de nós da rede. Existe a saída abrupta da rede devido a uma falha em que um nó simplesmente sai da rede sem avisar o sistema, o sistema eventualmente vai perceber que o nó falhou. Por outro lado existe a saída ordeira da rede em que o nó que pretende sair avisa os restantes da sua saída e pode fazer uma acção antes de sair, como será explicado mais à frente. Após a saída de um nó da rede, este pode voltar a entrar na rede a qualquer momento. O processo de reentrada é idêntico ao processo de entrada inicial da rede.

Quando um nó pretende armazenar um objecto é atribuído um identificador a esse objecto obtido através da função de *hash* SHA-1. Para armazenar um objecto na rede é efectuada uma procura na rede pelo nó que possui o identificador que sucede ao identificador do objecto que se pretende armazenar. Após o resultado desta pesquisa, é transferido o objecto para esse nó obtido na pesquisa e este nó armazena o objecto. Quando se pretende fazer uma leitura de um dado objecto na rede, é efectuada a pesquisa pelo nó que possui o identificador que sucede ao identificador do objecto que se pretende encontrar. Após se obter o identificador do nó que tem

o objecto armazenado, inicia-se a transferência do objecto para o nó que efectuou a pesquisa. No caso das remoções de objectos na rede, o processo é idêntico às leituras. Inicia-se a pesquisa pelo objecto e após a pesquisa retornar o identificador do nó onde o objecto está armazenado, o nó que efectuou a pesquisa envia um pedido ao nó que possui o objecto armazenado para remover o objecto.

Quando um dado objecto é armazenado, este fica armazenado no nó com o identificador que sucede o identificador do objecto. Este nó é o responsável por armazenar este objecto e por replicar este objecto pela sua lista de sucessores. Assim que recebe um objecto para armazenar, envia uma cópia deste objecto para os nós que pertencem à sua lista de sucessores. Este fica também responsável por periodicamente verificar se cada um dos seus sucessores possui uma cópia de cada um dos objectos do qual é responsável. No caso de algum dos seus sucessores não possuir uma cópia de algum dos objectos, o responsável envia-lhe uma cópia do objecto em falta. Este mecanismo permite repor as cópia em falta nos elementos da lista de sucessores, devido a alguma saída que ocorra.

Qualquer nó pertencente à rede pode sair a qualquer momento da rede. Isto faz com que os objectos que este possui armazenados fiquem indisponíveis ou no pior caso, os objectos são perdidos permanentemente quando os nós saem da rede definitivamente e não voltam mais a entrar. De forma a mitigar este efeito é necessário que exista replicação dos objectos, para que estes não sejam perdidos permanentemente no caso da saída definitiva do nó que os possui armazenados e que não estejam indisponíveis quando o nó se ausenta da rede por um determinado momento. A replicação dos objectos é feita guardando cópias extra dos objectos nos nós que sucedem o nó onde fica armazenado o objecto. Quando se armazena um ficheiro, além de armazenar o ficheiro no nó pretendido, é também armazenado uma cópia nos sucessores do nó onde fica armazenado o objecto. Quando se efectua uma pesquisa por um determinado objecto, a procura é realizado no nó que devia ser responsável pelo objecto. Se não possuir o objecto, a pesquisa é efectuada nos seus sucessores, pois estes possuem uma cópia extra. Quando é efectuada uma remoção da rede, é enviado um pedido para o nó responsável por esse objecto e para os seus sucessores. Cada nó ao receber esse pedido, apaga esse objecto do seu armazenamento.

Quando um nó entra ou sai da rede é necessário proceder à mudança de alguns objectos na rede de forma a que os objectos se mantenham armazenados de acordo com o seu identificador. Quando um nó entra na rede e após ter conhecimento do seu sucessor, envia um pedido ao seu sucessor indicando o seu identificador e o sucessor transfere-lhe os objectos que possuem uma chave menor que o identificador do nó. Estes objectos ficam da responsabilidade do nó

que acabou de entrar. Para finalizar será removida a cópia do último sucessor pois este já não pertencerá à lista de sucessores do nó que se juntou à rede. Quando acontece uma saída de um nó da rede, os objectos da responsabilidade deste terão de ser transferidos para a responsabilidade do seu sucessor. No caso de uma saída ordeira, o nó que abandona a rede, envia uma mensagem ao seu sucessor a informá-lo que vai sair da rede e os objectos do nó que vai abandonar a rede passarão a ser da responsabilidade do sucessor deste. Como o sucessor do nó que vai sair já possui cópias dos objectos que eram da responsabilidade do nó que vai sair, não é necessário transferir os respectivos objectos. Quando ocorre uma saída abrupta da rede, o nó que abandona a rede não envia mensagem a avisar que abandonou a rede, eventualmente a rede com recurso às funções do Chord irá perceber que o nó abandonou a rede e então o seu sucessor ficará responsável pelos objectos do nó que abandonou a rede tal como acontece quando ocorre uma saída ordeira.

4.2.2 Compressão de dados

Para armazenar dados eficiente na rede, os nós irão comprimir os dados de forma a que o armazenamento destes seja eficiente. O algoritmo utilizado no sistema para a compressão de dados é o algoritmo *Shannon Fano*. Os testes realizados por Kodituwakku and Amarasinghe (2010) mostram que este algoritmo oferece um rácio de compressão médio de 63.85%.

5 Avaliação

Neste capitulo apresentamos a avaliação dos mecanismos propostos. O objectivo desta avaliação é tirarmos resultados acerca dos mecanismos propostos que visam reduzir o churn e consequentemente mitigar o impacto deste. Para a avaliação do sistema iremos realizar algumas simulações usando dados que simulam o comportamento dos utilizadores de redes P2P para produzir padrões de acesso ao sistemas de armazenamento. Na Secção 4.1 descrevemos os dados que usamos nas simulações, na Secção 4.2 apresentamos uma única simulação realizada sem o apoio do protocolo de reputação que serve para apresentar o sistema e mostrar alguns dos resultados obtidos. Na Secção 4.3 avaliamos o sistema com o uso do protocolo de reputação e incentivos face a um sistema sem estes mecanismos. Na Secção 4.4 avaliamos o sistema com o uso do protocolo de incentivos e reputação desta vez sem o uso da compressão de dados e a comparação do sistema sem uso deste mecanismo. Na Secção 4.5 concluímos este capítulo com os resultados retirados da avaliação realizada.

5.1 Descrição das simulações

Para a realização das simulações é utilizado o simulador Corten de modo a conseguir-se obter resultados que permitam tirar conclusões acerca do desempenho dos mecanismos propostos. Para a realização das simulações foi desenhado o sistema descrito no Cap.3 usando a linguagem Rust. As simulações usam parâmetros retirados de estudos reiais que observaram o comportamento de utilizadores de sistemas de ficheiros.

Para enquadrar as entradas na rede com dados reais, cada nó entra na rede depois de outro nó com um intervalo temporal retirado de uma distribuição de Weibull onde o parâmetro shape é k = 0.62. A Fig. 5.1 mostra dados que são retirados de um estudo realizado por Stutzbach and Rejaie (2006) onde o tempo entre a entrada de nós na rede corresponde à distribuição referida.

O tamanho das sessões dos nós conectados à rede P2P são também obtidos do mesmo estudo e são retirados de uma distribuição de Weibull com k = 0.59, lam = 41.9, apresentada graficamente na Fig. 5.2, como é mostrado em Stutzbach and Rejaie (2006).

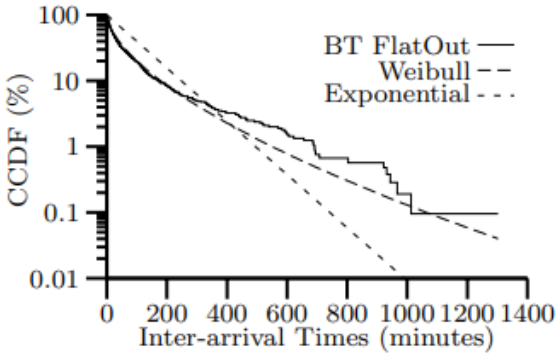


Figure 5.1: Tempo entre a entrada de nós nas redes P2P segundo o estudo de Stutzbach and Rejaie (2006)

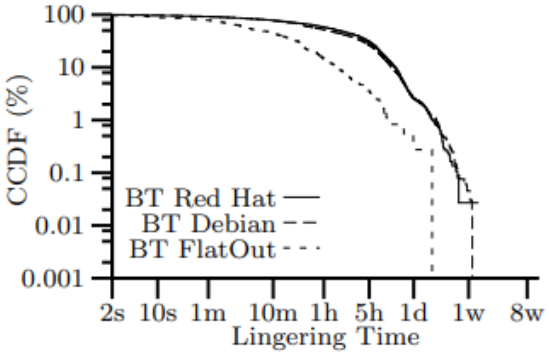


Figure 5.2: Tamanho das sessões dos nós nas redes P2P segundo o estudo de Stutzbach and Rejaie (2006)

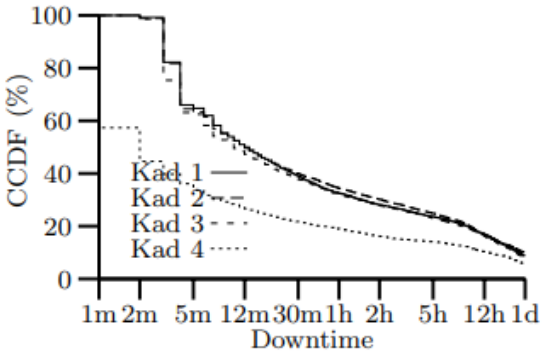


Figure 5.3: Tempo que um nó está ausente das redes P2P segundo o estudo de Stutzbach and Rejaie (2006)

A Fig. 5.3 retirada do estudo realizado por Stutzbach and Rejaie (2006) mostra o tempo que os nós estão ausentes da rede. Nas simulações realizadas são utilizados valores desta tabela representado na Fig. 5.3 de forma a que a simulação tenha esta aproximação ao estudo efectuado. No gráfico estão representados os valores do vários estudos.

Nas simulações que efectuámos cada nó tem uma quantidade de armazenamento disponível para armazenamento da rede. Quando um nó entra na rede, define a quantidade de armazenamento, este valor é retirado de uma distribuição log-normal que é a distribuição que podemos aproximar o mais possível ao estudo referido. Os valores da log-normal são os seguintes: μ é 16.9 e o valor σ é 1.0 o que leva a um armazenamento médio de 36GB oferecido por cada nó do sistema tal como sugere o estudo realizado por Anderson and Fedak (2006).

Um estudo realizado durante 5 anos em sistemas de ficheiros por Agrawal et al. (2007) mostra que o tamanho dos ficheiros aumenta em média 15% por ano. No ano de 2004, o ano do estudo, o tamanho médio dos ficheiros era de 189KB. À data da realização das simulações tendo em conta a evolução proposta por Agrawal et al. (2007), o tamanho médio de ficheiros será de aproximadamente 1769KB. Nas simulações realizadas, os objectos criados e armazenados pelos nó do sistema são retirados de uma distribuição log-normal onde o valor μ é 6.98 e o valor σ é 1.0. Esta distribuição leva a que os ficheiros criados tenham em média 1769KB.

5.2 Simulação do sistema

Nesta secção vamos apresentar uma simulação com o objectivo de apresentar o sistema e de seguida apresentamos os respectivos resultados.

Na primeira simulação apresentada são utilizados 10 nós na rede. Durante o decorrer desta

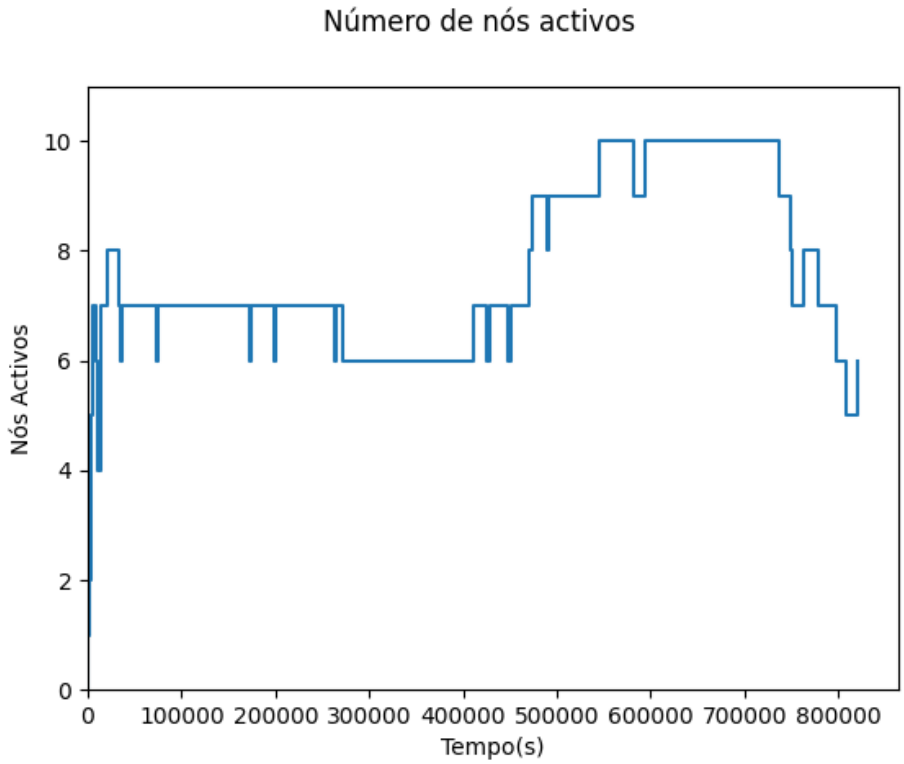


Figure 5.4: Número de nós activos na rede durante a primeira simulação

simulação foram escritos 9550 objectos na rede. A Fig. 5.4 apresenta o número de nós que está vivo a cada momento. Podemos observar o comportamento geral dos nós da rede durante o decorrer da simulação, sendo que podemos observar alguma instabilidade na rede com algumas saídas e entradas de nós o que pode levar a que exista perda ou indisponibilidade de dados o que acaba por não acontecer nesta simulação pois é uma simulação curta e com pouco nós. Na Fig.5.5 podemos observar como é a fase inicial da rede enquanto os nós se estão a juntar e observamos que enquanto alguns nós ainda estão a entrar na rede outros já saíram e voltam a entrar.

5.3 Simulações entre sistemas e comparação de resultados

Nesta secção vamos apresentar os resultados das simulações de um sistema de armazenamento P2P baseado no protocolo Chord bem como do mesmo sistema com a adição dos mecanismos de reputação propostos. Ambos os sistemas correram a mesma simulação, ou seja, os nós do sistema têm exactamente o mesmo comportamento, cada nó sai e entra na rede exactamente no mesmo momento em ambas as simulações. Adiante, nos gráficos apresentados, o sistema sem mecanismo de reputação e incentivos será denominado de Sistema1 enquanto o sistema com mecanismo de reputação e incentivos será denominado Sistema2. Devido a um problema apre-

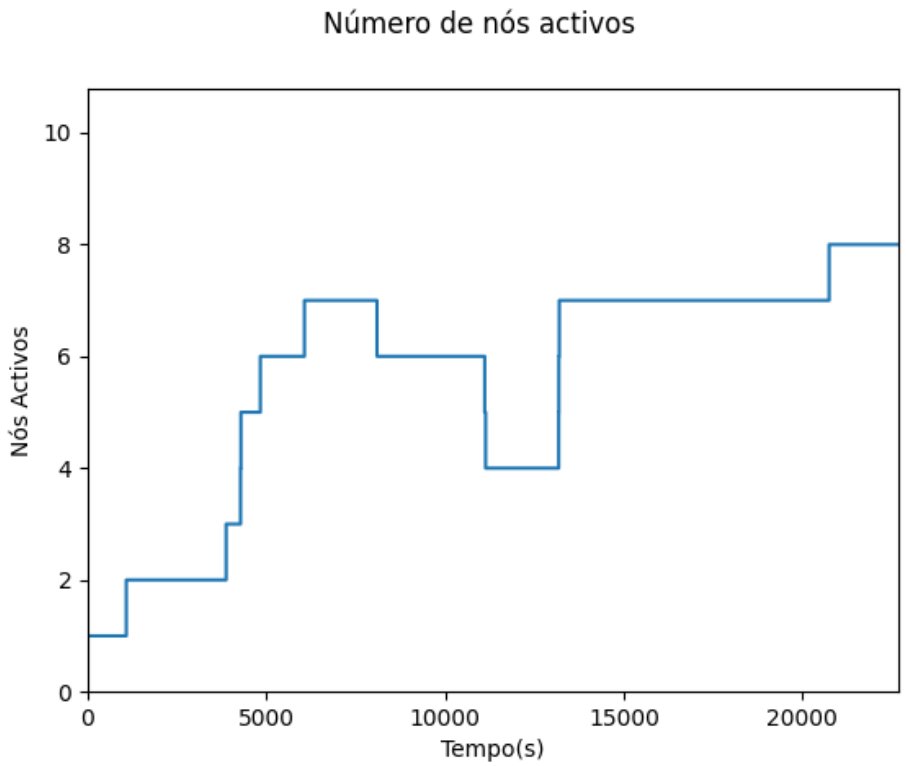


Figure 5.5: Entrada de nós no inicio da primeira simulação

sentado pelo simulador utilizado que falha quando o número de eventos é muito elevado então não é possível efectuar simulações com mais eventos do que as simulações aqui apresentadas pois este apresenta uma falha se as simulações forem muitos extensas e/ou se estiver a simular um número elevado de nós.

Durante a realização das simulações nesta secção, o tempo entre as entradas dos nós na rede foi reduzido, foi utilizada a distribuição de Weibull tal como explicado acima mas com intervalos de entrada menores para possibilitar realizar simulações com um maior número de nós no tempo que é possível antes de o simulador apresentar a falha. A redução deste intervalo entre a entrada de nós na rede pode levar a exista alguns erros de leitura de objectos devido a um ritmo de entrada de nós na rede o que pode levar a que o Chord demore algum tempo a estabilizar o anel e consequentemente a que os objectos ainda estejam a ser transferidos entre os nós. Este problema pode ocorrer sobretudo na fase inicial das simulações que é quando existe uma maior entrada de nós face aos nós existentes na rede. Este problema pode ser resolvido refazendo o pedido de leitura do mesmo objecto passado um pequeno intervalo de tempo.

Em ambas as simulações efectuadas entraram 6333 nós na rede. A Fig.5.6 mostra o número de nós que estão na rede ao longo da simulação. Neste gráfico podemos reparar que os nós vão entrando na rede à medida que as simulações avançam no tempo e estes podem sair e voltar

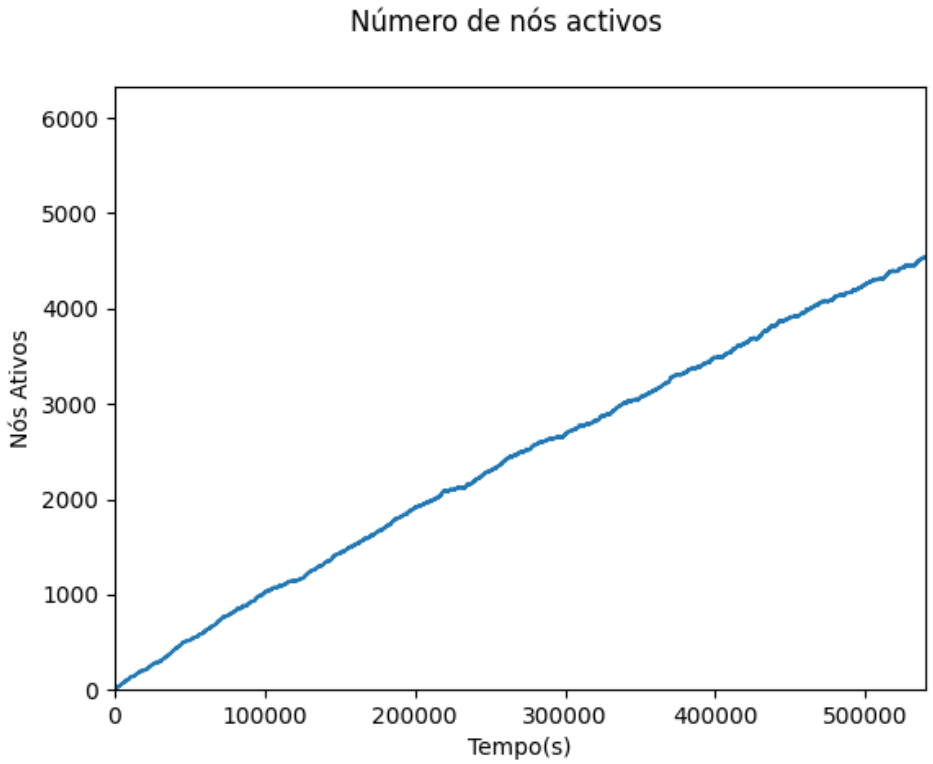


Figure 5.6: Número de nós activos na no cenário 1

à rede, este comportamento é de acordo como o descrito na Secção 4.1, no momento final da simulação estão aproximadamente 70% dos nós activos na rede.

Durante ambas as simulações todos os nós fazem uma operação a cada 10 segundos. Esta operação tanto pode ser uma escrita ou leitura de um qualquer objecto, sendo que a escrita ou a leitura têm a mesma probabilidade de ocorrer. A ideia destas simulações é saturar a rede com objectos e assim encher rapidamente a rede, daí não serem feitas remoções de objectos por parte dos nós na rede.

Durante a simulação sem os mecanismos propostos foram feitas 6 466 046 tentativas de escrita de objectos na rede dos quais aproximadamente 79,96% destas foram realizadas com sucesso. Durante a simulação com os mecanismos propostos foram feitas 6 296 083 tentativas de escritas de objectos na rede dos quais aproximadamente 92,20% destas foram realizadas com sucesso. Os erros aqui referidos são sempre o mesmo erro que é quando um nó tenta escrever um objecto e não existe espaço de armazenamento disponível no nó pretendido. A Fig. 5.7 mostra o gráfico com a evolução do número de erros que ocorreram durante ambas as simulações. Neste gráfico podemos observar que a existe um menor número de erros de escritas no sistema com os mecanismos propostos em relação à simulação sem esses mecanismos, isto deve-se sobretudo a um menor espaço ocupado dos discos na simulação com os mecanismos propostos que é obtido

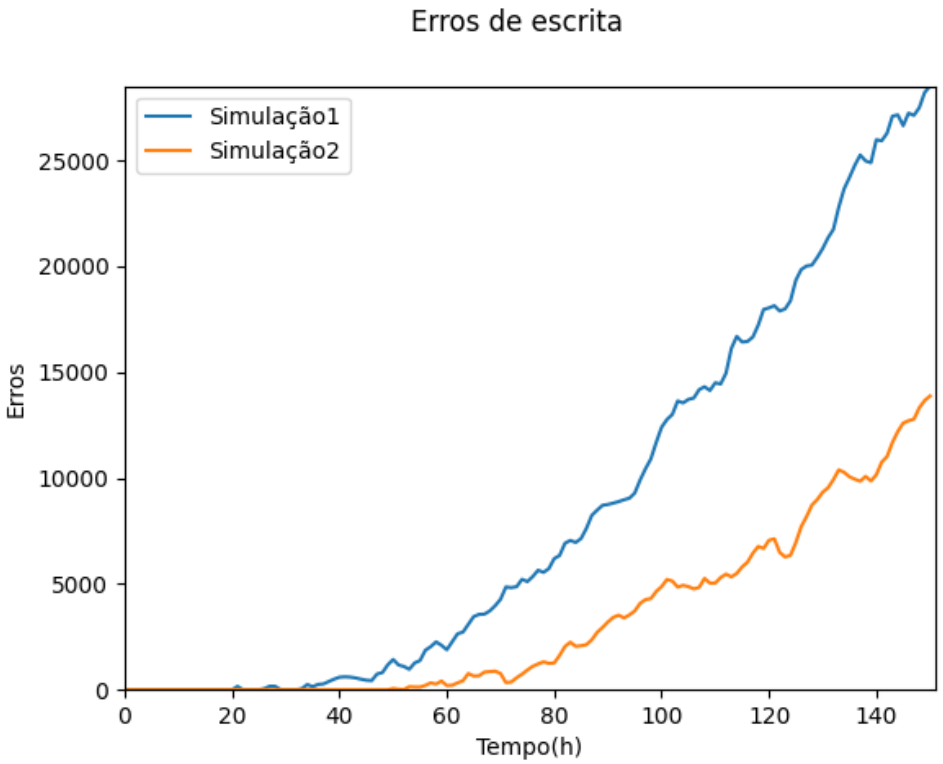


Figure 5.7: Erros de escrita ocorridos em ambos os sistemas no cenário 1

devido à compressão dos objectos utilizada.

Durante a simulação sem os mecanismos propostos foram feitas 5 743 644 tentativas de leitura de objectos na rede dos quais aproximadamente 99,95% tiveram sucesso. Durante a simulação com os mecanismos propostos foram feitas 5 827 433 tentativas de leitura de objectos na rede dos quais aproximadamente 99,96% tiveram sucesso. Na Fig. 5.8 é mostrado o gráfico com o número de erros de leitura que ocorreram durante a simulação sem os mecanismos propostos e durante a simulação com os mecanismos propostos. Alguns destes erros podem acontecer devido à mudança de objectos de nós que ocorre após a entrada ou saída de nós da rede. Este problema pode ser mitigado se for feito o mesmo pedido de leitura após um intervalo de tempo. No entanto também podem existir casos onde objectos sejam perdidos permanentemente ou fiquem efectivamente indisponíveis, caso as 3 réplicas que mantém cópias de um dado objecto se ausentem da rede e não seja possível mover essas cópias para outros nós ou criar cópias adicionais. A diferença entre a quantidade de erros é quase nula. Isto deve-se ao facto de ter sido utilizado o factor de replicação 3, ou seja, além da cópia original do objecto são armazenadas duas réplicas adicionais. Devido à utilização deste factor de replicação, existe um maior armazenamento utilizado da rede para guardar o objecto mas, por outro lado, a probabilidade de um objecto se perder permanentemente ou que exista uma perda temporária é relativamente inferior.

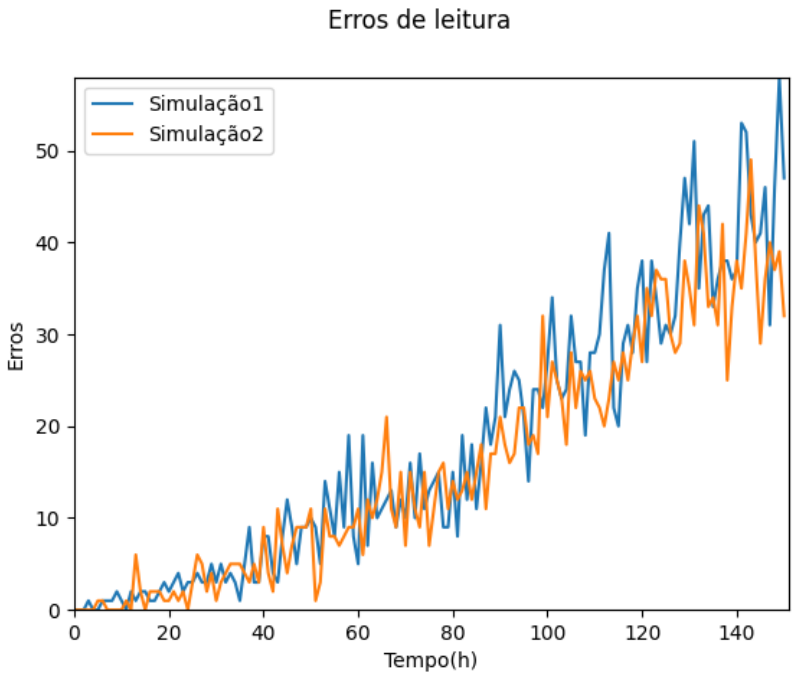


Figure 5.8: Erros de leitura ocorridos em ambos os sistemas no cenário 1

5.4 Simulações entre sistemas sem compressão e comparação de resultados

Nesta secção vamos apresentar os resultados das simulações de um sistema de armazenamento P2P baseado no protocolo Chord e o resultados das simulações do mesmo sistema com a adição dos mecanismos de incentivos e reputação e sem recorrer ao uso da compressão de dados. Tal como na secção anterior ambos os sistemas correram a mesma simulação, ou seja, os nós do sistema têm exactamente o mesmo comportamento, saindo e entrando na rede no mesmo momento da simulação. Nos gráficos presentes nesta secção o sistema com o uso de mecanismo de incentivos de reputação é denominado de Sistema3 enquanto que o sistema sem recurso a este mecanismo é denominado Sistema4. Por sua vez a simulação com o Sistema3 é denominada de Simulação3 enquanto que a simulação com o Sistema4 é denominada Simulação4. As simulações nesta secção diferencia-se da anterior pelo facto de não ser utilizada compressão de dados e noutros campos configuráveis das simulações tal como explicado abaixo.

Nas simulações apresentadas nesta secção foram utilizados 3000 nós. Durante a realização das simulações nesta secção o tempo entre as entradas dos nós na rede foi reduzido, foi utilizada a distribuição de Weibull tal como explicado acima mas com intervalos de entrada menores desta vez 50 vezes mais pequenos para possibilitar a distinção da fase de entrada dos nós para a fase onde os nós já se encontram na rede. Devido ao encurtamento do tempo de entrada realizado

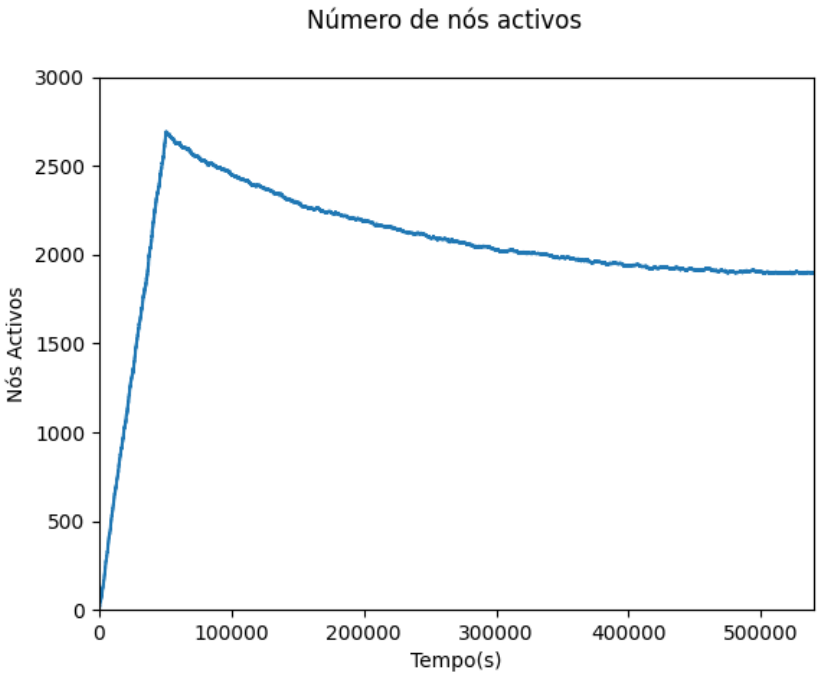


Figure 5.9: Número de nós activos na rede no cenário 2

podemos observar uma taxa de entrada inicial grande até ao pico do número de nós. Após este momento ao longo da simulação o número de nós activos na rede vai tender para um número a rondar os 2000 nós activos que iria ser o mesmo número para o qual se iria tender caso não fosse usado este encurtamento.

Tal como nas simulações da secção anterior, durante ambas as simulações todos os nós fazem uma operação a cada 10 segundos. Esta operação tanto pode ser uma escrita ou leitura de um qualquer objecto, tendo a escrita ou a leitura têm a mesma probabilidade de ocorrer. A ideia destas simulações é saturar a rede com objectos e assim encher rapidamente a rede, daí não serem feitas remoções de objectos por parte dos nós na rede.

Durante a simulação com recurso aos mecanismo propostos foram realizadas 4 711 189 tentativas de leitura de objectos na rede dos quais aproximadamente 99.91% tiverem sucesso, na simulação sem os mecanismos propostos houve 4 091 759 tentativas de leitura de objectos dos quais 99.87% tiveram sucesso. Na Fig. 5.10 está representado graficamente o número de erros de leitura por hora que ocorreram nas simulações. Nesta simulação foi utilizado um factor de replicação 2, ou seja, além do objecto original armazenado foi armazenado uma cópia extra do objecto. Podemos observar que existiu uma maior taxa de erros de leitura nas simulações no cenário 2 em relação às simulações do cenário 1. Isto deve-se devido ao facto de ter sido utilizado um factor de replicação 2. o que implica menos réplicas dos objectos armazenados. Mas apesar de este ligeiro aumento dos números de erros, é consumido menos espaço para

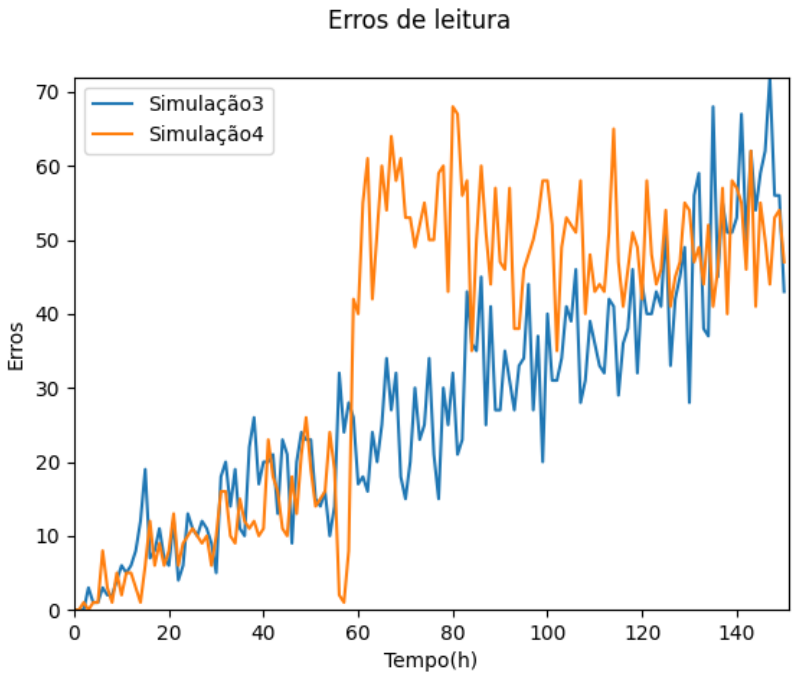


Figure 5.10: Erros de escrita ocorridos em ambos os sistemas no cenário 2

armazenar objectos e as respectivas réplicas o que liberta armazenamento extra na rede para que pode ser utilizado pelos nós da rede.

Durante a simulação com recurso aos mecanismo propostos foram realizadas 5 120 896 tentativas de escrita de objectos na rede dos quais aproximadamente 69,69% tiveram sucesso, Na simulação sem os mecanismos propostos foram realizados 5 438 844 escritas de objectos na rede dos quais 63.34% aproximadamente tiveram sucesso. Na Fig 5.11 estão representados graficamente os erros de escrita ocorridos durante a execução das simulações. Podemos observar uma menor taxa de erro e um menor número de erros na Simulação3. Isto deve-se ao facto de nós com menor reputação terem menos armazenamento disponível o que leva a um menor número total de objectos escritos.

Na Fig. 5.12 está representado graficamente cada um dos patamares da tabela de reputação e incentivos e o número de nós que que está em cada patamar ao longo da simulação com os mecanismos propostos.

5.5 Discussão de resultados

Piatek et al. (2007) conclui que os incentivos desencorajam o comportamento de *free-riding* por parte dos nós das redes P2P. Com os resultados obtidos neste capitulo podemos comprovar que o uso do mecanismo proposto ajuda a reduzir os efeitos causados pelo *churn*. O que nos leva

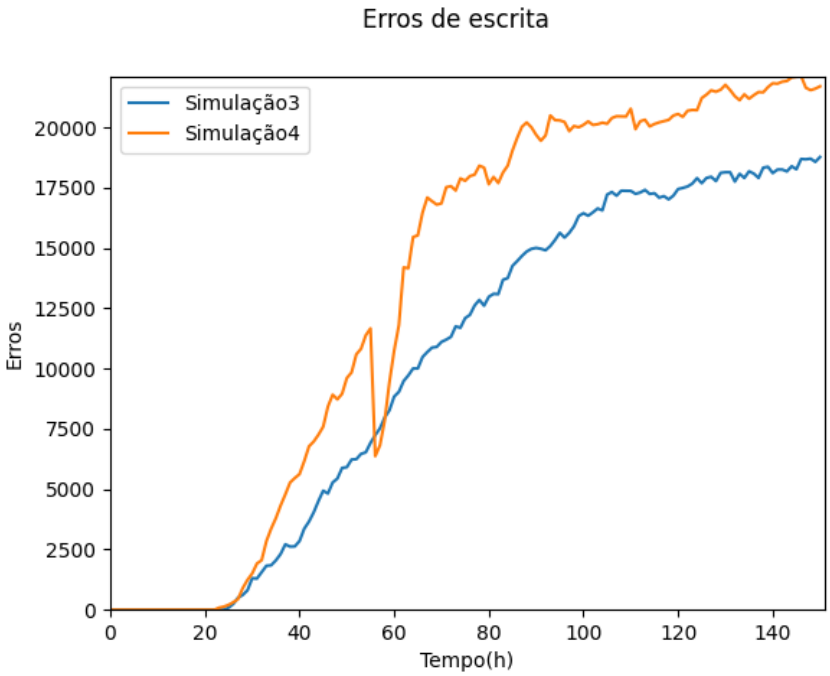


Figure 5.11: Erros de leitura ocorridos em ambos os sistemas no cenário 2

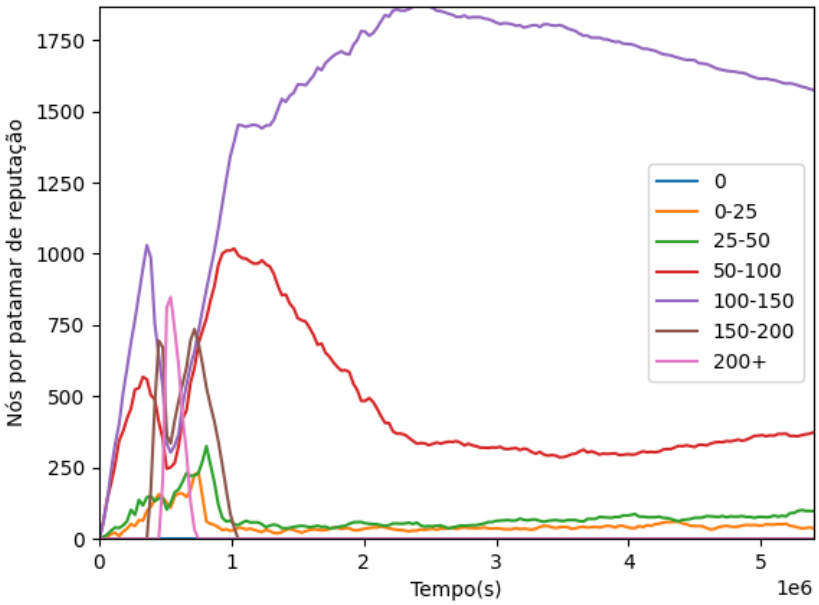


Figure 5.12: Distribuição dos nós pelos patamares de reputação/incentivos

a crer que o uso deste mecanismo leva a que os nós reduzam o seu *churn*. Em todas as simulações realizados foram usados os mesmos dados de churn, estes dados são dos estudos apresentados que observaram o comportamento de utilizadores de sistemas de ficheiros de redes P2P.

O uso do mecanismo de compressão de dados é benéfico para o sistema dado que com compressão de dado é possível reduzir o tamanho destes na rede e assim sejam oferecidos mais incentivos aos utilizadores o que poderia motivar mais ainda estes a reduzir o *churn*.

Os patamares de reputação são atribuídos aos nós de acordo com a sua reputação que obtém. Os nós para reduzirem o seu churn e consequentemente subirem nos patamares e obterem melhores recompensas devem permanecer mais tempo conectados à rede e evitar sairem desta.

6 Conclusão

As redes P2P são seriamente afectadas pelo *churn* e os SA P2P não são excepção. Nestes sistemas, devido ao *churn*, a saída permanente ou temporária dos nós da rede leva a que exista perda de dados ou que exista uma indisponibilidade temporária dos dados.

Nas rede P2P existe um comportamento denominado de *free-riding* que é quando os utilizadores consumem recursos do sistema sem contribuir para este. Vimos que podemos mitigar este comportamento oferecendo incentivos a quem possuir um melhor comportamento. Para isso os utilizadores procuram ter uma melhor comportamento e assim diminuir o seu *churn*, mitigando assim o comportamento de *free-riding*.

Neste trabalho foi desenhado um protocolo que visa mitigar o *churn* e os efeitos causados por este. Este protocolo pretende através da oferta de incentivos mitigar o *churn*. Para isso os nós pertencentes à rede são avaliados consoante a sua ligação à rede, ou seja, quanto tempo passam conectados à rede. Deste modo é possível saber quais os nós que possuem melhor comportamento. De seguida são oferecidas melhores recompensas consoante melhor for a sua reputação. Para um melhor funcionamento do protocolo proposto é recomendado o uso de um sistema de compressão de dados de forma a que se aproveite o armazenamento disponível na rede mais eficientemente.

O protocolo foi avaliado através de simulações. Para a realização das simulações foi utilizado um simulador P2P. De forma a avaliar o protocolo, este foi implementado num SA que seguiu a estrutura do Chord proposto por Stoica et al. (2001), ou seja, toda a organização interna dos nós, disseminação das mensagens, entrada de nós na rede, distribuição dos dados pela rede está de acordo com o Chord.

6.1 Trabalho futuro

Como trabalho futuro, o protocolo pode ser melhorado de forma a avaliar em função do comportamento dos utilizadores conseguir variar a tabela de reputação/beneficio de forma a ajustar as recompensas com o comportamento geral da rede, de forma a que seja possível oferecer uma

boa qualidade de serviço aos utilizadores e oferecer ainda melhores recompensas para motivar ainda mais os utilizadores a melhor o seu comportamento. Outra funcionalidade interessante que pode ser desenvolvida para o protocolo é permitir que através da reputação dos nós, o protocolo possa estimar quantas cópias de um objecto devem ser criadas em função da reputação dos nós que armazenam cópias do objecto, de forma a que esse objecto não seja perdido permanentemente ou esteja indisponível durante um intervalo de tempo devido aos nós que armazenam as cópias de um determinado objecto saíram da rede (permanentemente ou temporariamente). Deste modo é possível libertar ainda mais armazenamento para a rede e assim aumentar as recompensas oferecidas aos utilizadores, continuando a oferecer uma boa qualidade de serviço.

Além disso, uma versão futura do protocolo deve permitir que o protocolo suporte falhas bizantinas dos nós pertencentes à rede deixando assim de suportar exclusivamente as falhas do tipo *crash*.

Bibliography

Agrawal, N., Bolosky, W. J., Douceur, J. R., and Lorch, J. R. (2007). A five-year study of file-system metadata. *ACM Transactions on Storage (TOS)*, 3(3).

Anderson, D. P. and Fedak, G. (2006). The computational and storage potential of volunteer computing. In *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, volume 1, pages 73–80. IEEE.

Belaramani, N. M., Dahlin, M., Gao, L., Nayate, A., Venkataramani, A., Yalagandula, P., and Zheng, J. (2006). PRACTI Replication. In *NSDI*, volume 6, pages 5–5.

Buragohain, C., Agrawal, D., and Suri, S. (2003). A game theoretic framework for incentives in P2P systems. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, pages 48–56. IEEE.

Burrows, M. and Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm.

Christen, P. (2011). A survey of indexing techniques for scalable record linkage and deduplication. *IEEE transactions on knowledge and data engineering*, 24(9):1537–1555.

Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. (2001). Freenet: A distributed anonymous information storage and retrieval system. In *Designing privacy enhancing technologies*, pages 46–66. Springer.

Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., and Stoica, I. (2001). Wide-area cooperative storage with CFS. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 202–215. ACM.

Gantz, J. and Reinsel, D. (2012). The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the future*, 2007(2012):1–16.

Haddi, F. L. and Benchaïba, M. (2015). A survey of incentive mechanisms in static and mobile P2P systems. *Journal of Network and Computer Applications*, 58:108–118.

Hasan, R., Anwar, Z., Yurcik, W., Brumbaugh, L., and Campbell, R. (2005). A survey of Peer-to-Peer storage techniques for distributed file systems. In *International Conference on*

Information Technology: Coding and Computing (ITCC'05)-Volume II, volume 2, pages 205–213. IEEE.

Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. (2003). The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM.

Kant, K., Iyer, R., and Tewari, V. (2002). A framework for classifying Peer-to-Peer technologies. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, pages 368–368. IEEE.

Kodituwakku, S. and Amarasinghe, U. (2010). Comparison of lossless data compression algorithms for text data. *Indian journal of computer science and engineering*, 1(4):416–425.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., et al. (2000). Oceanstore: An architecture for global-scale persistent storage. In *ACM SIGARCH Computer Architecture News*, volume 28, pages 190–201. ACM.

Muthitacharoen, A., Morris, R., Gil, T. M., and Chen, B. (2002). IVY: A Read/Write Peer-to-Peer file system. *ACM SIGOPS Operating Systems Review*, 36(SI):31–44.

Piatek, M., Isdal, T., Anderson, T., Krishnamurthy, A., and Venkataramani, A. (2007). Do incentives build robustness in bittorrent. In *Proc. of NSDI*, volume 7.

Rowstron, A. and Druschel, P. (2001a). Pastry: Scalable, decentralized object location, and routing for large-scale Peer-to-Peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 329–350. Springer.

Rowstron, A. and Druschel, P. (2001b). Storage management and caching in PAST, a large-scale, persistent Peer-to-Peer storage utility. In *ACM SIGOPS Operating Systems Review*, volume 35, pages 188–201. ACM.

Schollmeier, R. (2001). A definition of Peer-to-Peer networking for the classification of peer-to-peer architectures and applications. In *Proceedings First International Conference on Peer-to-Peer Computing*, pages 101–102. IEEE.

Sequeira, I. (2019). Corten: Large scale distributed algorithms simulator. Instituto Superior Técnico, Universidade de Lisboa.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable Peer-to-Peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160.

Strauss, J., Paluska, J. M., Lesniewski-Laas, C., Ford, B., Morris, R. T., and Kaashoek, M. F. (2011). EYO: Device-transparent personal storage. In *USENIX Annual Technical Conference*.

Stribling, J., Sovran, Y., Zhang, I., Pretzer, X., Li, J., Kaashoek, M. F., and Morris, R. T. (2009). Flexible, wide-area storage for distributed systems with WheelFS. In *NSDI*, volume 9, pages 43–58.

Stutzbach, D. and Rejaie, R. (2006). Understanding churn in Peer-to-Peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202. ACM.

Surati, S., Jinwala, D. C., and Garg, S. (2017). A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator. *Engineering Science and Technology, an International Journal*, 20(2):705–720.