Pesquisa Descentralizada: Proposta de otimização da solução não estruturada do IPFS

Rafael Sequeira, Pedro Ákos Costa, and João Leitão

NOVA LINCS & DI-FCT-UNL
{rm.sequeira,pah.costa}@campus.fct.unl.pt,jc.leitao@fct.unl.pt

Resumo As soluções centralizadas oferecidas pela computação na nuvem exigem que se deposite total confiança nos seus provedores e tornam-se assim também pontos de falha únicos. Os sistemas descentralizados, como o InterPlanetary File System (IPFS), têm ganho popularidade como alternativas ao uso de infraestrutura na nuvem que dominam a Internet atualmente. No entanto, as soluções descentralizadas apresentam ainda limitações que, frequentemente, constituem uma barreira à sua adoção. Em muitos sistemas, a pesquisa surge como um serviço primário. Este é o caso do IPFS, onde as pesquisas acontecem inicialmente recorrendo a um protocolo que opera sobre uma rede não estruturada, e no caso de insucesso, recorrem a uma tabela de dispersão distribuída (DHT, do inglês *distributed hash table*). Esta solução é simultaneamente dispendiosa do ponto de vista de comunicação entre nós e lenta.
Neste artigo, apresentamos uma alternativa ao mecanismo atualmente utilizado no IPFS, o protocolo Bitswap, cujo desenho foi inspirado pelo BitTorrent. A nossa alternativa pretende melhorar a eficiência e sucesso deste mecanismo de pesquisa descentralizada, e consequentemente melhorar o tempo de acesso do IPFS a conteúdos presentes na rede, reduzindo a necessidade de recorrer à DHT. A nossa solução introduz o uso de uma nova camada de indexação, construída sobre a rede não estruturada existente, sendo que exploramos diferentes alternativas de desenho, de forma a conseguirmos, por um lado reduzir o custo de mensagens durante a pesquisa, e por outro acelerar o download de blocos de ficheiros da rede. Os nossos resultados experimentais mostram que a solução proposta é eficaz e eficiente possibilitando uma redução significativa do recurso à DHT.

Keywords: IPFS · Sistemas entre pares · Rede não estruturada · Pesquisa descentralizada.

1 Introdução

Nos últimos anos, a Web3, que descentralizar a arquitetura da Internet [9], tem atraído atenção significativa pela sua promessa de democratizar o acesso a dados e computação na Internet. Este paradigma pretende evitar a crescente centralização da Internet onde grandes empresas controlam uma parte significativa da infraestrutura e das aplicações. A Web3 baseia-se no desenvolvimento e adoção de sistemas descentralizados, incluindo, naturalmente, sistemas entre pares.

Uma das necessidades fundamentais de sistemas descentralizados é a pesquisa, utilizada tanto pelo utilizador final quanto por outros serviços ou aplicações construídos sobre estes.

Soluções tradicionais para pesquisas eficientes em ambientes descentralizados passam pelo uso de redes estruturadas, onde os nós se organizam sobre uma topologia de acordo com regras estritas, formando uma Tabela de Dispersão Distribuída (DHT, do inglês *Distributed Hash Table*) [16,23,19]. As DHTs, apesar de facilitarem a pesquisa e obtenção de informações, são pouco robustas, podendo fragmentar-se com a entrada e saída repentina de uma quantidade elevada de nós, fenómeno conhecido como *churn* (em inglês) [14].

Uma alternativa às DHTs passa pela adoção de redes não estruturadas, onde os nós formam uma rede de topologia aleatório [17,4]. Nessas redes, a pesquisa é mais desafiante, exigindo (geralmente) que uma mensagem que descreve o recurso pesquisado (e.g., um ficheiro) seja enviada e difundida por um grande número de nós na rede para, de forma otimista, a pesquisa chegar a um nó que possua o recurso pesquisado. Com o aumento do poder computacional dos nós tornou-se viável que cada nó na rede mantenha um número maior de conexões abertas na rede, resultando num grau mais elevado neste grafo (i.e., cada nó mantém mais vizinhos). Este aumento permite a pesquisas chegarem de forma mais expedita a um maior número de nós, no entanto, com um custo de comunicação e processamento significativamente maior. Este fenómeno é evidente na análise do número de vizinhos dos nós no InterPlanetary File System (IPFS) [5,10], um sistema entre pares que se apresenta como um bloco de construção fundamental em sistemas e aplicações para a Web3.

O Bitswap [21], um protocolo inspirado pelo BitTorrent [18], é o mecanismo utilizado pelo IPFS para realizar pesquisas e obter conteúdo numa rede não estruturada. No Bitswap, as pesquisas são realizadas for difusão completa a um salto o que significa que apenas os vizinhos diretos do nó que inicia a pesquisa recebem essa pesquisa. Quando o protocolo é incapaz de encontrar recursos através deste passo de difusão, o IPFS recorre a uma pesquisa na sua DHT (Kademlia [16]). A estratégia do Bitswap de uso de difusão a um salto numa rede de elevado grau leva a um elevado número de mensagens e ao uso frequente da DHT, resultando em elevados custos em termos de mensagens e tempo, sem colher os benefícios completos desse custo.

Para mitigar estas limitações, neste artigo apresentamos uma proposta de otimização. Esta solução introduz uma camada de indexação construída sobre a rede não estruturada existente. Recorrendo à informação partilhada nela, são introduzidos dois mecanismos que permitem realizar uma pesquisa informada e removem a limitação de só ser possível encontrar recursos apenas a um salto de distância. Recorremos a um protótipo da nossa solução implementado sobre o IPFS para demonstrar os benefícios e custo da nossa solução que levam a uma redução significativa da dependência da DHT.

Este artigo têm a seguinte estrutura: na secção 2 descrevemos o funcionamento do IPFS e do Bitswap, focando-nos na integração dos dois e na pesquisa do segundo. Na secção 3 apresentamos a nossa solução e detalhamos o seu funci-

onamento. Na secção 4 descrevemos a nossa avaliação experimental e discutimos os resultados experimentais obtidos. Na secção 5 apresentamos e discutimos trabalho relacionado. A secção 6 concluí o artigo.

2 Preliminares

IPFS O InterPlanetary File System (IPFS) [5] é um sistema de ficheiros entre pares que permite aos seus utilizadores partilhar e aceder a conteúdo. A ambição deste sistema é ser a fundação para a distribuição de páginas na Internet, formando uma arquitetura descentralizada e resistente à censura.

No IPFS o conteúdo é dividido em blocos que têm um identificador único (CID) derivado através de uma função de síntese sobre o seu conteúdo. Os blocos são organizados em grafos acíclicos dirigidos (DAG, do inglês *directed acyclic graph*). Algumas das propriedades e funcionalidades do IPFS derivam diretamente desta abordagem, por exemplo, o conteúdo é imutável, pois qualquer alteração aos dados gera um CID diferente, a propagação desta mudança no DAG e leva a um CID diferente para a raiz (i.e., o CID do ficheiro). Os nós do IPFS (identificados na rede por um PeerID) participam em duas redes sobrepostas com arquiteturas diferentes, mas conexões partilhadas.

A primeira é estruturada e implementa uma DHT baseada no Kademlia [16]. Nela, os nós (acessíveis através da Internet) publicam, propagam e guardam apontadores tanto para conteúdo na rede como para endereços de outros nós. Naturalmente, a motivação para a existência destes é suportar a pesquisa de conteúdos. Para o fazerem, a DHT é percorrida uma primeira vez para obter o PeerID de um provedor do CID desejado. Depois, caso o endereço não esteja em memória transitória (*cache* em inglês), é percorrida uma segunda vez para o obter essa informação permitindo o contacto direto com esse nó. A segunda rede, o foco deste trabalho, é não estruturada e suporta a operação do Bitswap.

Bitswap O Bitswap [21] é utilizado pelos nós no IPFS para realizar tanto pesquisas otimistas por inundação a um salto como para transferir o conteúdo encontrado. Devido à forma como o conteúdo é organizado, uma pesquisa é composta por um conjunto de várias (sub-)pesquisas feitas de forma recursiva: primeiro é necessário obter o conteúdo da raiz do DAG que descreve um ficheiro, e depois, dos vários blocos que constituem o ficheiro.

Para otimizar este processo, e partindo do princípio que um nó que contém parte do conteúdo poderá conter o resto, o Bitswap introduz o conceito de sessões, estas que servem como agregadores de todos os vizinhos potencialmente relevantes para uma pesquisa. Todas as pesquisas começam com a criação de uma sessão. As sessões são inicialmente populadas através de uma pesquisa por difusão a um salto. Os nós enviam uma mensagem WANT-HAVE contendo o CID da raiz para todos os vizinhos. A esta, os vizinhos respondem com uma mensagem HAVE, que indica que possuem o conteúdo, ou por oposição com um DONT-HAVE. No entanto, alguns vizinhos (os que usam versões mais recentes

do IPFS) abstêm-se de enviar a última como resposta, sendo os DONT-HAVEs assumidos pelo emissor da pesquisa.

Todos os vizinhos que respondem positivamente à primeira mensagem são adicionados à sessão. Dentro da sessão, um destes recebe uma mensagem WANT-BLOCK que indica que deve transferir o conteúdo para o emissor. Quando o conteúdo é recebido, numa mensagem BLOCK, todos os nós que receberam um WANT-HAVE ou WANT-BLOCK recebem um CANCEL, que indica que o nó que iniciou a pesquisa já obteve esse recurso. O processo volta a repetir-se para os restantes blocos dentro da sessão.

Note-se que numa sessão só volta a contactar vizinhos fora da sessão caso ocorra uma das seguintes três situações:

1. Não existe nenhum nó na sessão capaz de responder positivamente a um WANT, o que pode ser causado por todos os elementos da sessão terem respondido com um DONT-HAVE ou terem sido removidos da sessão após responderem com um número consecutivo (16 por defeito) de DONT-HAVEs.
2. Nenhum bloco é recebido durante um dado intervalo de tempo. Este é inicialmente igual a 1 segundo (por defeito), mas é recalculado com base na média das latências das respostas anteriores.
3. Um limite temporal, igual a 1 minuto por defeito, é ultrapassado.

Nestas situações, a sessão pode recorrer à DHT para pesquisar um CID ainda não encontrado, sendo um nó que o contenha adicionado à sessão, partindo mais uma vez do princípio que um nó que serve um bloco de um ficheiro poderá ter os restantes. Na terceira situação esta pesquisa acontece sempre. O Bitswap funciona por defeito como uma cache, o que quer dizer que qualquer bloco recebido por um nó passa a ser servido por esse nó através do Bitswap.

3 Otimização da Pesquisa no Bitswap

A nossa proposta começa com a introdução de uma nova camada de indexação, esta tem como objetivo principal permitir aos nós informar e receber informação sobre os conteúdos disponíveis nos nós à sua volta, para permitir melhores decisões durante a propagação de pesquisas (quer suas, quer durante o processamento de pesquisas originadas pelos seus vizinhos). Esta informação é agregada a dois níveis, o primeiro incide apenas sobre a sua vizinhança direta e o segundo captura informação sobre nós a dois saltos (i.e., *hops*) de distância na rede sobreposta não estruturada.

Para capitalizar nesta informação, introduzimos um método simples de pesquisa informada. Antes propagar uma pesquisa, os nós consultam o seu indice local de forma a limitar o envio da pesquisa a nós que possam ter os conteúdos procurados. Partindo da mesma motivação, também adicionamos um novo tipo de mensagem que permite a um nó usar o seu conhecimento local para auxiliar um vizinho a encontrar possíveis provedores para o conteúdo descrito na pesquisa, aumentando efetivamente o horizonte da pesquisa sem aumentar o escopo de comunicação de um protocolo como o Bitswap.

3.1 Camada de Indexação

Para implementar a base da nossa solução, cada nó participa numa nova camada construída sobre a rede não estruturada do IPFS. Cada par mantém uma vista parcial [12] assimétrica de tamanho configurável, com a qual partilha periodicamente um índice e um *bloom filter* [3] que agrega todos os índices conhecidos dos seus vizinhos diretos (ao qual chamaremos meta-índice). Note-se que esta informação não é enviada periodicamente, e apenas atualização são enviadas quando necessário. Chamaremos aos membros desta vista de um dado nó de *vizinhos próximos* daqui em diante.

A vista é construída adicionando novos vizinhos à mesta até estar cheia. É gerida de forma reativa [12,13], o que significa que vizinhos próximos só são removidos devido a eventos externos (i.e., o vizinho abandona o sistema ou a conexão é terminada/falha). Porém, para manter a vista o mais estável possível, vizinhos próximos têm as suas conexões protegidas utilizando mecanismos oferecidos pelo IPFS.

É importante notar que um dos objetivos desta camada é ter um custo baixo (em número de mensagens trocadas) tanto de construção como de manutenção. Se o custo for demasiado alto, os benefícios obtidos através da informação partilhada podem não ser suficientes para compensar o seu custo e, em última análise, ameaçar a viabilidade da solução na totalidade.

O índice enviado por um nó contém o conjunto de CIDs dos blocos que o nó consegue servir. Apenas atualizações são enviadas para vizinhos próximos que já receberam o índice integral no passado (note-se que o índice pode ser enviado em segmentos e atualizado de forma fina). O meta-índice, por sua vez, é um agregado de todos os índices que um dado nó recebeu, sob a forma de um *bloom filter*. Estes têm um tamanho dinâmico, sendo reconstruídos caso a quantidade de entradas seja superior à ou muito inferior à sua capacidade (configurada de forma a garantir uma probabilidade de falsos positivos máxima de 1%).

Apesar da vista ser assimétrica, o que significa que um vizinho próximo v de um nó n poderá não ter n como vizinho próximo, todos os pares na nossa solução aceitam informação (o índice e/ou o meta-índice) de qualquer vizinho e guardam-na enquanto esta relação de vizinhança se mantiver. Intuitivamente, isto acontece porque, como os índices são partilhados sem coordenação prévia, nomeadamente, quando um nó recebe um índice, o custo já foi pago pelo nó que o enviou.

3.2 Partilha de Fonte

Recorrendo aos índices recebidos, os nós passam a saber (com alguma confiança) que um dos seus vizinhos possui um dado recurso identificado por um CID. O mecanismo de partilha de fonte usa esta informação para aumentar a cooperação entre os vizinhos e remover a limitação do horizonte de pesquisa no Bitswap de um salto, mas, ao contrário de outras alternativas clássicas [4], isto é feito sem a necessidade de mais passos de difusão da pesquisa. Para isso, introduzimos uma

nova mensagem, à qual chamamos SOURCE, que informa um nó que faz uma pesquisa sobre possíveis provedores para o conteúdo pesquisado.

Uma mensagem SOURCE é enviada como resposta a uma mensagem WANT quando um nó recebe uma pesquisa que não consegue satisfazer localmente, mas o CID está contido em algum dos índices que recebeu de um vizinho. A mensagem carrega o CID pesquisado e um conjunto dos PeerIDs conhecidos localmente que contenham esse CID no seu índice.

Por sua vez, ao receber uma mensagem deste tipo, o nó que iniciou a pesquisa notifica a sessão do Bitswap. Do ponto de vista da sessão, essa informação é equivalente a receber uma mensagem HAVE dessa fonte. Porém, existem duas situações onde este processo não é tão direto. A primeira ocorre quando uma mensagem com a mesma fonte já foi recebida, para evitar que a sessão envie múltiplas mensagens para o mesmo nó, foi implementado um mecanismo de de-duplicação que mantêm estado temporário sobre as fontes previamente notificadas. A segunda é quando o nó que iniciou a pesquisa não tem uma conexão aberta com a fonte; nesse caso é aberta uma nova conexão antes de notificar a sessão.

3.3 Pesquisa Informada

Com a introdução da camada de indexação, os nós passam a conseguir consultar o índice dos conteúdos disponíveis nos seus vizinhos próximos.

Algorithm 1 Pesquisa informada com consulta ao índice e meta-índice

```
1: function PesquisaInformada(CID)
2:     if ExistemEntradasNosÍndices(CID) then
3:         vizinhos ← ObterEntradas(CID)
4:         EnviaWantBlock(CID, vizinhos[0])
5:         for cada vizinho em vizinhos[1..n] do
6:             EnviaWantHave(CID, vizinho)
7:         end for
8:     else if ExistemEntradasNosMetaÍndices(CID) then
9:         vizinhos ← ObterMetaEntradas(CID)
10:        for cada vizinhos em vizinhos do
11:            EnviaWantHave(CID, vizinho)
12:        end for
13:    else
14:        PesquisaPorInundação(CID)
15:    end if
16: end function
```

O Algoritmo 1 apresenta de forma simplificada o funcionamento da pesquisa informada. Antes de fazer uma pesquisa por inundação a um salto, os índices são consultados (Alg. 1, linha 2). Se houver pelo menos uma entrada, o bloco é pedido diretamente a um vizinho com uma mensagem WANT-BLOCK (Alg. 1,

linha 4). Optamos por enviar de forma otimista um WANT-BLOCK ao invés de um WANT-HAVE (como é habitual no Bitswap) pois, por estar no índice, é provável que este nó ainda tenha o bloco e, desta forma, podem ser poupadas mensagens e um *round trip time* (RTT). No entanto, o leitor deve notar que o índice local pode estar desatualizado, por isso enviamos também mensagens WANT-HAVE para todas as correspondências restantes (Alg. 1, linha 5).

Caso não existam entradas no índice, recorremos ao meta-índice. Para o fazer o processo de consulta é semelhante (Alg 1, linha 4). É importante lembrar que, como os meta-índices incluem informação a dois saltos, uma correspondência indica (geralmente) que o respetivo par conseguirá responder com uma fonte (como descrito na Secção 3.2). Assim, todos vizinhos nesta situação recebem um WANT-HAVE ao contrário do que acontece na consulta direta do índice.

Por fim, se não houver entradas em nenhum índice, é feita a pesquisa por inundação a um salto.

4 Resultados Experimentais

4.1 Configuração

Para validar a nossa solução e medir o seu desempenho emulámos uma rede distribuída com 500 nós. Cada participante foi hospedado num contentor Docker, onde uma instância do IPFS foi executada durante cada experiência. Estes contentores foram distribuídos de forma uniforme por duas máquinas do *Cluster-DI* recorrendo ao *docker swarm*. Ambas as máquinas têm as seguintes especificações: dois processadores Intel Xeon Gold 6346, 128 GiB de memória e uma ligação *bond* de 20Gbps (que recorrer a duas ligações físicas de 10Gbps cada uma). Para melhor capturar uma rede real, foi introduzida latência artificial utilizando o Linux Traffic Control (Linux TC) [2], estabelecendo latências entre contentores diferentes entre 75 e 225 milisegundos. A configuração da latência foi mantida em todas as experiências.

As configurações do IPFS foram ajustadas para se adequarem à dimensão de 500 nós do nosso ambiente experimental. Primeiro, para garantir que todos os nós contribuíssem para a DHT, configurámo-los no modo *dhtserver*. Em segundo lugar, ao nível do transporte, todos os protocolos, exceto TCP e QUIC, foram desativados. Por fim, devido ao tamanho reduzido da rede, os parâmetros do gestor de conexões *LowWater* e *HighWater* foram ajustados para 16 e 48, respetivamente. No entanto, é importante referir que o gestor de conexões apenas remove conexões inativas. Adicionalmente, porque os vizinhos da DHT e do Bitswap são partilhados (inclusive na nossa solução), o *BucketSize* da DHT foi diminuído de 20 para 10. Além disso, o intervalo de tempo até ser feita a segunda inundação (e uma possível pesquisa na DHT) apenas na nossa solução foi ajustado de 1 para 10 segundos, pois, com a existência de novas mensagens, as sessões podem por vezes receber as respostas às primeiras mensagens mais tarde do que no Bitswap (especialmente se forem abertas novas conexões).

As nossas experiências começam com a cópia para cada contentor de um binário do IPFS (modificado no caso da nossa solução), um conjunto de ficheiros

que o nó é responsável por servir e um conjunto de CIDs que o nó deve pesquisar. É importante referir que estes conjuntos nunca se intercetam. Depois disso, os nós são iniciados sendo que esperamos 10 minutos para este processo terminar e a rede poder estabilizar.

Após esse período, os nós começam a pesquisar iterativamente por cada entrada. No entanto, para evitar a sincronização de todas as pesquisas, é introduzido um intervalo aleatório antes da primeira e entre as pesquisas, variando entre 1 e 40 segundos e 1 e 20 segundos, respetivamente. Todas as pesquisas estão limitadas a 60 segundos; se não for possível obter uma resposta ao fim deste tempo, a pesquisa é cancelada.

O conjunto de ficheiros hospedado por cada nó e a lista das pesquisas que este realiza são pré-gerados com base numa distribuição da popularidade dos ficheiros. Esta popularidade determina o número de réplicas de cada ficheiro (ou seja, quantos nós servem cada ficheiro) e a frequência com que cada ficheiro é pesquisado. No entanto, existe uma diferença entre os dois: enquanto o número de réplicas está diretamente relacionado com a popularidade e o número de nós no sistema, as pesquisas foram limitadas a 4000 por experiência. Assim, utilizamos um número de pesquisas proporcional à popularidade de cada ficheiro. No total existem 3000 ficheiros distribuídos entre os 500 nós em cada experiência.

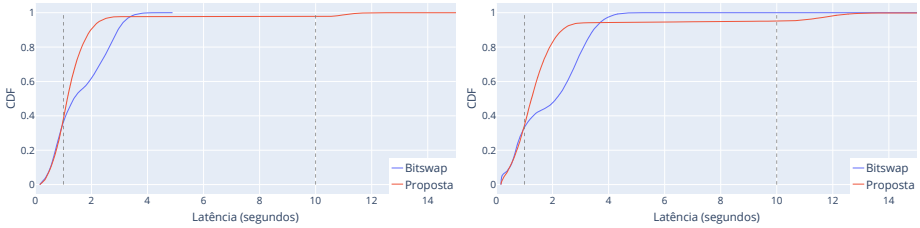
Nas nossas experiências estudamos o efeito de duas distribuições para a popularidade dos ficheiros: uma distribuição uniforme onde todos os ficheiros têm $0,01$ uma popularidade e uma distribuição zipfiana com $\alpha = 0.82$. Enquanto a primeira representa um cenário ideal, a segunda é mais representativa da realidade no IPFS [7]. Os nossos resultados agregam 5 repetições de cada experiência, cada uma utiliza uma distribuição de popularidade diferente.

4.2 Resultados e Análise

Das experiências previamente descritas, os nossos resultados focam-se em três métricas: a taxa de sucesso, a latência até ser obtido o primeiro byte e o número de mensagens ao longo da experiência. Através destas métricas, analisamos o desempenho do Bitswap em comparação com a nossa solução, em dois cenários distintos. No primeiro, os nós podem ser auxiliados pela DHT para realizar pesquisas caso seja necessário. Devido a isso, é difícil analisar a taxa de sucesso das pesquisas ao nível do Bitswap. Tendo isso em consideração, no segundo cenário a DHT é desativada durante as experiências para ambas as alternativas.

Com auxílio da DHT Neste cenário, a taxa de sucesso de todas as alternativas e em ambas as distribuições de popularidade é muito virtualmente de 100% (sendo que as variações de devem a ruído experimental). Sendo por isso todas as soluções equivalentes. No entanto, as restantes métricas variam significativamente.

A Figura 1 apresenta a função de distribuição acumulada (CDF do inglês *cumulative distribution function*) das latências das pesquisas em ambas as distribuições. Ambos os gráficos capturam a CDF até aos 15 segundos apenas para facilitar a sua leitura.



(a) Ditribuição uniforme (b) Distribuição zipfiana

Figura 1: CDF da latência das pesquisas

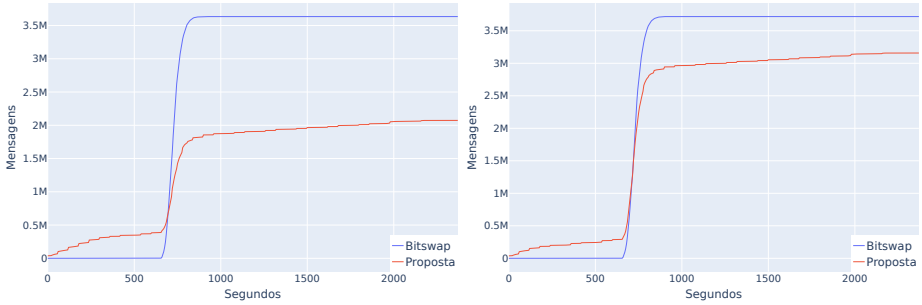
Como é possível ver, ambas as soluções (nas duas distribuições) são bastante semelhantes nas pesquisas que têm menos de $0,95$ segundos de latência (aproximadamente 32% das pesquisas). Porém, a tendência muda após este ponto. O declive do Bitswap diminui significativamente e depois, aos $1,8$ segundos, volta a aumentar, só voltando a mudar após 95% das pesquisas terminarem, o que acontece por volta dos $3,17$ segundos na distribuição uniforme e $3,75$ segundos na distribuição zipfiana (respetivamente Figuras 1a e 1b). Em comparação, a nossa proposta apesar de ter uma cauda mais longa para as pesquisas que demoram mais tempo, consegue resolver mais de 50% de todas as pesquisas mais rápido do que o Bitswap. A longa é bastante mais acentuada na distribuição zipfiana, o que é expectável considerando que ficheiros menos populares têm muito menor probabilidade de serem localizados na rede não estruturada.

Estes resultados são influenciados pelos limites temporais impostos na pesquisa, assinalados nas figuras com as linhas verticais cinzentas. No Bitswap, ao fim de um segundo (primeira linha), é feita uma nova pesquisa por inundação e a DHT é usada. Isto corresponde à primeira mudança no declive das suas latências. No nosso protocolo, o mesmo procedimento só acontece ao fim de 10 segundos (segunda linha), significando pesquisas com latências inferiores a este valor não usam a DHT nem voltam a inundar a rede. Isto é evidenciado nos resultados com o aumento do declive na nossa solução após os 10 segundos.

Note-se no entanto que na distribuição uniforme, $90,44\%$ das pesquisas do nosso protocolo terminam em menos de dois segundos, enquanto o Bitswap nesse intervalo de tempo apenas resolve $61,88\%$ das pesquisas. Na distribuição zipfiana, onde o mesmo é mais pronunciado e as pesquisas são mais desafiantes em geral, estes valores são de $82,85\%$ e $47,58\%$, respetivamente.

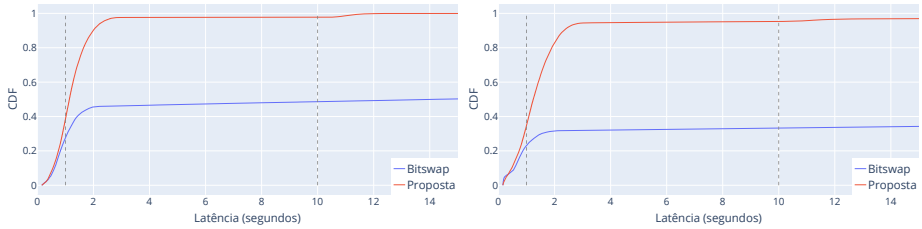
A Figura 2 mostra o número de mensagens enviadas (de forma comlativa) por cada protocolo ao longo das experiências. Esta métrica é importante pois revela o custo das pesquisas (inclusive o custo de partilhar os índices no caso da nossa solução). É também importante lembrar que apenas mensagens ao nível deste protocolo foram medidas (i.e., não medimos as mensagens trocadas pela DHT por exemplo).

Como é possível ver, em ambas as distribuições, o Bitswap não envia mensagens até as pesquisas começarem, o que acontece por volta dos 660 segundos. No entanto, isto não acontece no nosso protocolo, pois são enviadas mensagens



(a) Ditribuição uniforme (b) Distribuição zipfiana

Figura 2: Número de mensagens cumulativo enviadas durante as experiências



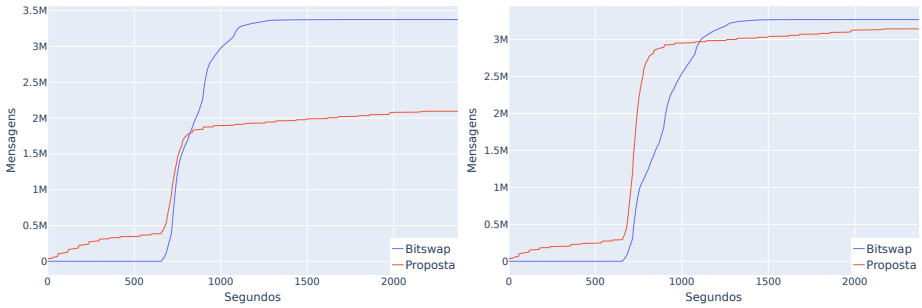
(a) Ditribuição uniforme (b) Distribuição zipfiana

Figura 3: CDF da latência das pesquisas

necessárias à construção e atualização dos índices. Não obstante, durante a fase de pesquisa, um elevado número de mensagens é enviado por ambos os protocolos, porém, o nosso protocolo envia um número significativamente menor. Esta diferença varia com a distribuição: na uniforme (Figura 2a) a diferença é de sensivelmente metade aos 1000 segundos e na zipfiana (Figura 2b) a diferença é menor reduzida. Estes resultados mostram que apesar do custo de manutenção dos indicies, em sistemas de uso regular, a nossa solução apresenta um custo de comunicação inferior.

Sem auxílio da DHT Ao contrário das experiências anteriores, neste conjunto de experiências existem diferenças significativas na taxa de sucesso. O Bitswap alcança sucesso em apenas $63,47\%$ das pesquisas na distribuição uniforme. Na distribuição zipfiana, este valor decaí para $42,12\%$ o que suporta a observação anterior da dependência da DHT por parte do Bitswap. Em contraste, o nosso protocolo teve uma taxa de sucesso de $99,98\%$ das pesquisas na distribuição uniforme e $99,37\%$ na distribuição zipfiana, o que mostra que a nossa solução reduz a dependência da DHT de forma muito significativa.

Na Figura 3 é apresentada a CDF da latência observada nas pesquisas sem DHT. Como é possível ver, os resultados do nosso protocolo são muito semelhantes aos apresentados na Figura 1. No entanto, há uma exceção: na distribuição zipfiana (apresentada na Figura 3b), o crescimento após os 10 segundos (assinalado pela segunda linha e atribuído apenas à segunda pesquisa por inundação)



(a) Ditribuição uniforme (b) Distribuição zipfiana

Figura 4: Número de mensagens cumulativo enviadas durante as experiências

foi menor. Os resultados mostram que não só a nossa alternativa consegue ter sucesso em mais pesquisas, mas também que sistematicamente a latência destas é igual ou inferior quando comparado com o Bitswap.

Por fim, a Figura 4 mostra o número de mensagens enviadas por cada alternativa de forma cumulativa. O comportamento da nossa solução é semelhante ao observado com a DHT disponível. Em contraste, o Bitswap apresenta resultados significativamente diferentes. Primeiro, o período de pesquisa foi bastante maior. Isso ocorre porque o protocolo itera sequencialmente pelas entradas na sua lista de pesquisa, significando que uma nova pesquisa só começa após a anterior terminar. Em segundo lugar, o Bitswap parece enviar ligeiramente menos mensagens. Uma hipótese para explicar este último ponto é que, como as pesquisas demoram mais, o gestor de conexões (que funciona periodicamente) consegue eliminar mais vizinhos ativos e, consequentemente, o passo da inundação envia menos mensagens. Deve no entanto notar-se que visto as alternativas não terem taxas de sucesso semelhantes, estes resultados não são diretamente comparáveis (i.e., seria fácil enviar zero mensagens para uma taxa de sucesso de 0%).

5 Trabalho Relacionado

Historicamente, os primeiros métodos de pesquisa em redes não estruturadas adotaram abordagens não informadas, onde as pesquisas são cegamente disseminadas na rede. Dentro destas, caracterizadas pela sua simplicidade, existe uma grande diversidade de algoritmos, sendo uma das mais populares a pesquisa por inundação [4]. Nestas soluções as consultas são enviadas para todos os vizinhos, que verificam se conseguem resolver a pesquisa e a propagam para os seus vizinhos. Isto acontece até que um limite, geralmente definido em número de saltos na rede, seja atingido. Esta foi a abordagem adotada pela primeira iteração Gnutella [1], um sistema de larga escala entre pares que foi alvo de diversos estudos por parte da comunidade cientifica [22,20], mas também pelo Bitswap, porém, neste está limitado a um salto.

Estas abordagens têm algumas limitações, sendo a mais relevante a geração de uma grande quantidade de mensagens, que pode sobrecarregar os nós

na rede, reduzindo assim a escalabilidade do sistema [15]. Devido a isso, várias alternativas foram propostas com o objetivo de reduzir o número de mensagens, nomeadamente métodos de pesquisa informada. Estes tentam utilizar informação obtida ou trocada entre os nós previamente para guiar as pesquisas para os nós com maior probabilidade de possuir conteúdos uteis. Esta informação pode ser obtida de maneira reativa (apenas observando a rede) o que não aumenta o número de mensagens trocadas [24,11,26]. Alternativamente, a informação pode ser obtida de forma proativa, onde os nós trocam mensagens entre si para construir e manter atualizado o seu conhecimento sobre o que os rodeia. Esta abordagem, embora mais cara, permite lidar melhor com redes dinâmicas. Dentro desta categoria, o Gia [6] apresenta várias melhorias em relação ao Gnutella, incluindo a introdução da partilha de índices com os vizinhos diretos, embora divirja da nossa solução ao adotar o uso de uma topologia enviesada e a introdução de um mecanismo baseado em *tokens*.

Uma outra proposta recorre a Routing Indices [8], utilizados para guiar as pesquisas na "direção" certa (nós com maior probabilidade de poder satisfazer uma pesquisa). À semelhança do nosso meta-índice, esta solução permite manter o tamanho dos índices proporcional ao número de vizinhos, enquanto a pesquisa beneficia de informação de um horizonte maior que este. No entanto, os indices são orientados a características dos ficheiros. Por exemplo, a entrada de um nó pode indicar que ele tem vários documentos sobre um tema. O sistema Quasar [25] propõe uma solução semelhante à anterior, utilizando tal como nós *bloom filters*, para codificar os indices. Ambas as alternativas não são aplicáveis ao IPFS devido à pesquisa baseada em CIDs sem semântica.

6 Conclusão

Neste artigo, explorámos a pesquisa realizada pelo Bitswap no contexto do sistema IPFS e apresentámos uma proposta para a sua otimização. A nossa solução consiste na introdução de uma nova camada de indexação, dois tipos de índices colaborativos e localizados geridos por essa camada, e dois mecanismos que os utilizam para aumentar o alcance das pesquisas e reduzir o número de mensagens enviadas durante o processo.

A nossa avaliação experimental, realizada através da emulação de uma rede semelhante à do IPFS, valida a eficácia da nossa solução e os seus benefícios. Os resultados demonstram melhorias significativas na latência, no número de mensagens enviadas durante a pesquisa e na taxa de sucesso na ausência da DHT, comprovando o potencial da nossa abordagem, em particular, por tornar o sistema significativamente independente da DHT.

Agradecimentos: O trabalho apresentado neste artigo foi suportado pela Fundação para a Ciência e Tecnologia através do laboratório NOVA LINCS (UIDP/04516/2020: DOI 10.54499/UIDP/04516/2020) e pelo projeto Horizon Europe TaRDIS financiado pela Comissão Europeia (Grant Agreement number 101093006).

Referências

1. Gnutella - stable - 0.4, \url{http://rfc-gnutella.sourceforge.net/developer/stable/index.html#t3-2-3}
2. Tc(8) — linux manual page, https://man7.org/linux/man-pages/man8/tc.8.html
3. Almeida, P.S., Baquero, C., Preguiça, N., Hutchison, D.: Scalable bloom filters. Information Processing Letters **101**(6), 255–261 (2007)
4. Barjini, H., Othman, M., Ibrahim, H., Udzir, N.I.: Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured p2p networks. Peer-to-Peer Networking and Applications **5**, 1–13 (3 2012). https://doi.org/10.1007/S12083-011-0101-Y/FIGURES/13, https://link.springer.com/article/10.1007/s12083-011-0101-y
5. Benet, J.: Ipfs - content addressed, versioned, p2p file system (7 2014), https://arxiv.org/abs/1407.3561v1
6. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making gnutella-like p2p systems scalable. Computer Communication Review **33**, 407–418 (2003). https://doi.org/10.1145/863955.864000
7. Ákos Costa, P., Leitão, J., Psaras, Y.: Studying the workload of a fully decentralized web3 system: Ipfs. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **13909 LNCS**, 20–36 (12 2022). https://doi.org/10.1007/978-3-031-35260-7_2, https://arxiv.org/abs/2212.07375v1
8. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. Proceedings - International Conference on Distributed Computing Systems pp. 23–32 (2002). https://doi.org/10.1109/ICDCS.2002.1022239
9. Guan, C., Ding, D., Guo, J.: Web3. 0: A review and research agenda. In: 2022 RIVF international conference on computing and communication technologies (RIVF). pp. 653–658. IEEE (2022)
10. Henningsen, S., Florian, M., Rust, S., Scheuermann, B.: Mapping the interplanetary filesystem. In: 2020 IFIP Networking Conference (Networking). pp. 289–297. IEEE (2020)
11. Kalogeraki, V., Gunopulos, D., Zeinalipour-Yazti, D.: A local search mechanism for peer-to-peer networks. International Conference on Information and Knowledge Management, Proceedings pp. 300–307 (2002). https://doi.org/10.1145/584792.584842, https://dl.acm.org/doi/10.1145/584792.584842
12. Leitão, J.: Gossip-Based Broadcast Protocols. Master's thesis, Faculdade de Ciências da Universidade de Lisboa (May 2007)
13. Leitao, J., Pereira, J., Rodrigues, L.: Hyparview: A membership protocol for reliable gossip-based broadcast. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07). pp. 419–429. IEEE (2007)
14. Leitão, J., Rodrigues, L.: Overnesia: a resilient overlay network for virtual superpeers. In: 2014 IEEE 33rd International Symposium on Reliable Distributed Systems. pp. 281–290. IEEE (2014)
15. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks pp. 84–95 (6 2002). https://doi.org/10.1145/514191.514206, https://dl.acm.org/doi/10.1145/514191.514206

16. Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **2429**, 53–65 (2002). https://doi.org/10.1007/3-540-45748-8_5, https://link.springer.com/chapter/10.1007/3-540-45748-8_5
17. MehrUnNisa, Ashraf, F., Naseer, A., Iqbal, S.: Comparative analysis of unstructured p2p file sharing networks. PervasiveHealth: Pervasive Computing Technologies for Healthcare pp. 148–153 (4 2019). https://doi.org/10.1145/3325917.3325952
18. Pouwelse, J., Garbacki, P., Epema, D., Sips, H.: The bittorrent p2p file-sharing system: Measurements and analysis. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **3640 LNCS**, 205–216 (2005). https://doi.org/10.1007/11558989_19, https://link.springer.com/chapter/10.1007/11558989_19
19. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '01 (2001). https://doi.org/10.1145/383059
20. Ripeanu, M.: Peer-to-peer architecture case study: Gnutella network. Proceedings - 1st International Conference on Peer-to-Peer Computing, P2P 2001 pp. 99–100 (2001). https://doi.org/10.1109/P2P.2001.990433
21. Rocha, A.D.L., Dias, D., Psaras, Y.: Accelerating content routing with bitswap: A multi-path file transfer protocol in ipfs and filecoin
22. Saroiu, S., Gummadi, P.K., Gribble, S.D.: Measurement study of peer-to-peer file sharing systems. https://doi.org/10.1117/12.449977 **4673**, 156–170 (12 2001). https://doi.org/10.1117/12.449977, https://www.spiedigitallibrary.org/conference-proceedings-of-spie/4673/0000/Measurement-study-of-peer-to-peer-file-sharing-systems/10.1117/12.449977.fullhttps://www.spiedigitallibrary.org/conference-proceedings-of-spie/4673/0000/Measurement-study-of-peer-to-peer-file-sharing-systems/10.1117/12.449977.short
23. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Ý, H.B.: Chord: A scalable peer-to-peer lookup service for internet applications (2001). https://doi.org/10.1145/964723, http://pdos.lcs.mit.edu/chord/
24. Tsoumakos, D., Roussopoulos, N.: Adaptive probabilistic search for peer-to-peer networks. Proceedings - 3rd International Conference on Peer-to-Peer Computing, P2P 2003 pp. 102–109 (2003). https://doi.org/10.1109/PTP.2003.1231509
25. Wong, B., Guha, S.: Quasar: a probabilistic publish-subscribe system for social networks. In: Proceedings of the 7th international conference on Peer-to-peer systems. pp. 2–2 (2008)
26. Yang, B., Garcia-Molina, H.: Improving search in peer-to-peer networks. Proceedings - International Conference on Distributed Computing Systems pp. 5–14 (2002). https://doi.org/10.1109/ICDCS.2002.1022237