Técnicas para Reduzir a Carga na Rede no Livro-Razão da *Bitcoin*

João Marçal, Miguel Matos, and Luís Rodrigues

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
{joao.marcal,miguel.marques.matos,ler}@tecnico.ulisboa.pt

Abstract. Os livros-razão distribuídos emergiram no contexto dos sistemas de gestão de criptomoedas mas têm-se relevado úteis para os mais diversos fins. Esta abstracção é tipicamente materializada agrupando sequências de operações em blocos que são, por sua vez, interligados numa cadeia (do inglês, *blockchain*). A troca de informação necessária para manter esta cadeia de forma descentralizada pode consumir recursos significativos na rede. Neste artigo estudamos os mecanismos de gestão da cadeia de blocos usada na criptomoeda *Bitcoin* e propomos um conjunto de alterações aos algoritmos usados para propagar as transacções com o objectivo de reduzir a carga na rede. Uma avaliação experimental dos mecanismos propostos mostra que é possível reduzir em cerca de 10.7% a quantidade de informação enviada na rede, e reduzir em 40.5% o número total de mensagens trocadas, sem qualquer efeito negativo observável na integração das transacções no livro-razão.

Keywords: Livros-razão, criptomoedas, disseminação de informação.

1 Introdução

A maioria das criptomodedas existentes, e em particular a *Bitcoin* (a criptomoeda mais usada correntemente), mantêm um livro-razão descentralizado que regista a sequência ordenada de todas as transacções executadas no sistema [9]. A manutenção de um livro-razão de forma descentralizada e aberta tem vindo a ser reconhecida como uma abstracção útil para várias aplicações, para além do uso original como meio de pagamento. Por exemplo, livros-razão distribuídos podem ser usados para registar todo o tipo de transacções, contratos, ou outros actos que tipicamente requerem o uso de notários.[1]

Um livro-razão distribuído é tipicamente mantido da seguinte forma. Em primeiro lugar, e por razões de eficiência, agrupam-se várias operações que são registadas em conjunto; estes grupos de transações são designadas por *blocos*. Posteriormente, os blocos são organizados numa lista ligada, também designada por *cadeia*. Na literatura anglo-saxónica, esta cadeira de blocos é simplesmente designada por *blockchain*. Um dos aspectos interessantes de criptomoedas como o

[1] Para uma lista de exemplos, consultar https://blockgeeks.com/guides/blockchain-applications/

Bitcoin consiste no facto dos nós participantes terem um mecanismo de filiação aberto e descentralizado. Isto é, nenhum nó no sistema necessita de conhecer todos os outros nós do sistema e qualquer nó pode juntar-se ou sair do sistema que o protocolo garante a coerência da cadeia gerada mesmo que alguns nós possam ter um comportamento racional ou bizantino.

Tipicamente, os sistemas de manutenção de cadeias de blocos funcionam da seguinte forma. Os nós do sistema, concorrentemente, recebem transacções que posteriormente distribuem pelos restantes nós. De forma também concorrente, os nós tentam criar o próximo bloco da cadeia, o que envolve a resolução de um puzzle criptográfico computacionalmente exigente. Assim que um nó gera um bloco, distribuí-o pela rede, o que leva ao cancelamento da geração de blocos concorrentes e dá inicio à criação de um novo bloco. Antes de aceitarem (e redistribuirem) um novo bloco, os nós da rede validam que o bloco é bem formado e constituído por transacções válidas.

Pela breve descrição acima, é fácil depreender que o processo de propagação de transacções é central na operação dos livro-razão distribuídos. Em primeiro lugar, as transacções necessitam de chegar aos nós que estão a criar blocos, de modo a poderem ser inseridas na cadeia. Em segundo lugar, necessitam também de ser conhecidas pelo restantes nós, pois são necessárias para validar a correcção dos blocos gerados. Na *Bitcoin*, a propagação de transacções baseia-se numa estratégia em que os nós anunciam periodicamente aos vizinhos quais as transacções que possuem. Estes, após receberem anúncios de transações em falta solicitam-nas aos primeiros. Este processo gera uma redundância de mensagens de anúncios. Apesar desta redundância ser necessária para tolerância a faltas, experimentalmente observámos que cada nó recebe um número excessivo de anúncios duplicados para cada transação no sistema.

Neste artigo propomos alterações ao processo de propagação das transacções na rede *Bitcoin* que permitem aumentar a sua eficiência. As nossas alterações exploram assimetrias que actualmente existem neste tipo de redes. De facto, numa rede *Bitcoin*, apenas uma fracção reduzida de nós, cerca de 20%, gasta recursos para criar novos blocos (este nós, são designados por *mineiros*); a grande maioria dos nós limita-se a propagar informação. A nossa estratégia consiste em enviesar a propagação das transações para que estas cheguem rapidamente aos mineiros ao mesmo tempo que se reduz a velocidade de propagação, e o número de duplicados, entre os restantes nós. Esta estratégia explora o facto dos requisitos de latência para a propagação de transações entre nós não mineiros serem relaxados, e também o facto do protocolo *Bitcoin* já possuir mecanismos que permitem propagar as transacções de forma mais eficiente após estas serem inseridas em blocos. Uma avaliação experimental dos mecanismos propostos mostra que é possível reduzir em cerca de 10.7% a quantidade de informação enviada na rede, e reduzir em 40.5% o número total de mensagens trocadas, sem qualquer efeito negativo observável na inserção das transacções no livro-razão.

2 O Livro-Razão da Bitcoin

O sistema Bitcoin [9] foi criado em 2008 com o objetivo de disponibilizar um meio seguro e anónimo onde duas entidades possam realizar transações entre si sem terem de confiar em terceiros. Para este propósito o sistema possuí uma criptomoeda que pode ser trocada entre várias entidades envolvidas numa transação. Todas as transacções são agrupadas em blocos e registadas num livro-razão, mantido de forma descentralizada. Para além de manter um registo das transacções, o livro-razão permite também seriar todas as transacções e, desta forma, evitar que um utilizador use as mesmas moedas em mais que uma transacção (fenómeno designado na literatura anglo-saxónica por *double spending*). O livro-razão da Bitcoin é construído ligando cada bloco ao seu predecessor formando uma corrente infinita de blocos conhecida como a *blockchain*.

O protocolo para manter o livro-razão na Bitcoin é relativamente complexo, e inclui várias funcionalidades que se complementam. Em primeiro lugar, a Bitcoin corre algoritmos de filiação, que tentam assegurar que cada nó mantêm ligações com outros nós do sistema escolhidos de forma aleatória. Estas relações de vizinhança criam uma rede sobreposta que é usada para disseminar informação na rede, nomeadamente: informação sobre as transacções submetidas pelos clientes e informação sobre os novos blocos da cadeia.

Como referimos anteriormente, os novos blocos da cadeia são gerados de forma concorrente por nós designados por mineiros. Cada mineiro recolhe um conjunto de transacções que conhece, que agrupa num bloco. Para o bloco ser válido, para além de todas as transações que inclui necessitarem de ser válidas, tem de conter a resposta a um puzzle criptográfico computacionalmente exigente que depende, entre outros, do bloco anterior e das transacções incluídas no novo bloco. Isto tem duas vantagens. Em primeiro lugar, desencoraja a criação de blocos incluindo transacções inválidas, uma vez que a única forma de um mineiro ser compensado pela energia gasta na resolução do puzzle é através da aceitação do seu bloco na cadeia. Para além disso, a dificuldade do puzzle reduz a probabilidade de dois mineiros gerarem blocos simultaneamente, o que limita a ocorrência de bifurcações na cadeia. Note-se que, uma vez que as transacções demoram a ser propagadas pela rede, é provável que quando dois mineiros tentam concorrentemente criar um novo bloco, não usem exactamente o mesmo conjunto de transacções e, portanto, não tenham de resolver o mesmo puzzle. Assim que um novo bloco é gerado, este é propagado na rede. Para evitar a propagação de blocos mal formados, cada nó valida o bloco antes de o propagar. Para verificar a correcção de um bloco, um nó deve possuir a seguinte informação: registo dos blocos anteriores (uma vez que cada bloco, contém uma referência para o bloco anterior) e ter informação sobre as transacções que estão incluídas nesse bloco, de forma a validar a sua autenticidade. Se um nó não possuir esta informação, tem que a solicitar primeiro, esperar que esta esteja disponível localmente, e só depois pode dar continuidade à propagação dos blocos.

Se um mineiro recebe o n-ésimo da cadeia enquanto ele próprio o está a tentar gerar, o processo de criar um novo bloco é abortado. Este mecanismo faz com que a probabilidade de se gerarem duas versões distintas, criando uma

bifurcação, seja muito pequena. Nos casos em que uma bifurcação acontece, o desempate é feito com base no comprimento de cada ramo. Quando um nó se apercebe que está a trabalhar sobre uma bifurcação que não é a mais longa, descarta essa bifurcação e junta-se ao ramo (que considera ser o) principal.

Os algoritmos usados na Bitcoin para propagar blocos e transações têm evoluído com o tempo, e incluem um conjunto de mecanismos que pretendem evitar o desperdício de largura de banda na rede. Descrevemos de seguida sucintamente o protocolo. Como referido anteriormente, as transações são disseminadas através de mensagens de anúncio *Inv* que contêm um conjunto de transações conhecidas pelo emissor. Ao receber uma mensagem de *Inv*, o receptor determina quais as transações que desconhece e pede-as ao emissor, enviando-lhe uma mensagem de *GetData* que, por sua vez, responde com uma mensagem *TX* por cada transação pedida. Os blocos são disseminados através de duas estratégias distintas. A primeira estratégia, e também a mais antiga, consiste em anunciar blocos através de mensagens *Headers*. Caso o bloco seja desconhecido, ou algumas das transações que o compõem, esse bloco é pedido através da mensagem *GetData*, sendo o bloco completo enviado através de uma mensagem *Block* que contem toda a informação necessária para validar o bloco. A segunda estratégia consiste em enviar somente um sumário do bloco, através de uma mensagem de *CmpctBlock* (bloco compacto). Caso o bloco ainda não seja conhecido, o nó valida o bloco, pedindo ao emissor as transações desconhecidas, se alguma, através de uma mensagem *GetBlockTX*. A primeira estratégia evita a troca adicional de mensagens mas, no entanto, resulta em mais redundância pois como as transações são disseminadas concorrentemente com os blocos é provável que a maior parte das transações de um bloco já sejam conhecidas por um nó.

Adicionalmente, os nós mantêm, para cada vizinho, uma fila de saída que contêm as mensagens a ser enviadas no futuro. Quando uma nova mensagem de *TX* é recebida, esta é colocada na fila de todos os vizinhos. À medida que se recebem anúncios e blocos de vizinhos estas filas são atualizadas para evitar enviar de volta transações que os vizinhos já conhecem. Periodicamente, estas filas são percorridas e é enviado aos respetivos nós um *Inv* que contêm as transações conhecidas localmente, permitindo que o destinatário peça as mensagens desconhecidas conforme indicado acima.

Embora já existam algumas preocupações em reduzir o número total de mensagens trocadas, bem como a quantidade de informação, este mecanismo ainda possuí várias ineficiências que discutimos na secção seguinte.

3 Melhorando a Propagação das Transacções

Nesta secção propomos um conjunto de alterações ao mecanismo de propagação de transacções na Bitcoin com o objectivo o tornar mais eficiente, nomeadamente, de reduzir o número de anúncios redundantes que cada nó da rede recebe. A nossa proposta é fundamentada nas seguintes observações:

- Actualmente, os nós recebem em média 3.33 anúncios para cada transacção (quando bastaria receberem no máximo um único para garantir a recepção da transacção).
- A rede possui dois mecanismos complementares para propagar transacções: a troca de anúncios (usada antes da transacção estar inserida num bloco) e a troca de blocos (que implicitamente anunciam as transacções que lhe estão associadas).
- Por razões históricas, o segundo mecanismo é mais eficiente, pois todas as transações em falta num nó são enviadas numa única mensagem (enquanto que o mecanismo de propagação epidémica inicial, obriga a troca de uma mensagem individual para cada transacção).
- A rede Bitcoin não tem requisitos estritos de latência para a propagação das transações, uma vez que a taxa de geração de blocos é significativamente mais lenta que o processo de propagação epidémico (em média, é gerado um novo bloco a cada 10 minutos).
- Os mineiros são uma fracção reduzida do número total de nós. Apesar de ser fundamental que as transacções sejam conhecidas pelo mineiros, o protocolo de disseminação não distingue os mineiros dos restantes nós.
- No protocolo epidémico actualmente usado na Bitcoin, os anúncios são escalonados para serem propagados para todos os vizinhos que o nó conhece (cerca de 125). Este valor é substancialmente maior do que o valor teórico dos protocolos de disseminação epidêmicos que sugere que, mesmo na presença de faltas, basta propagar informação para um número de vizinhos que é aproximadamente logarítmico com o tamanho da rede [6]. Para o tamanho atual da rede, que possuí cerca de 10 000 nós bastaria portanto propagar para $ln(10\ 000) \approx 10$ vizinhos.

O principal desafio é reduzir a quantidade de anúncios redundantes trocada na rede e, ao mesmo tempo, assegurar que as transações chegam aos nós mineiros. A intuição da abordagem proposta é enviesar o processo de propagação na direcção dos mineiros mais produtivos na rede sem, no entanto, deixar de assegurar que a informação é também propagada por caminhos alternativos para os restantes nós e mineiros. Isto é concretizado através de duas modificações ao protocolo. A primeira consiste em adicionar um novo campo às mensagens que propagam os blocos que indica o número de saltos que o bloco dá na rede. Isto permite que cada nó fique a saber a que distância está o mineiro que gerou esse bloco. Esta informação é codificada num inteiro e portanto tem um impacto negligenciável no tamanho total do bloco que ronda 1 MB. A segunda modificação consiste em identificar localmente, com o passar do tempo, qual o caminho para os mineiros mais próximos e enviar a propagação de anúncios preferencialmente por estes caminhos. Note-se que a partir do momento em que uma transacção é incluída num bloco, o processo de propagação por troca epidémica de anúncios deixa de ser relevante.

3.1 Seriação dos Vizinhos

Como foi referido acima, a estratégia proposta para poupar largura de banda pressupõe que cada nó descobre quais os melhores caminhos para chegar aos mineiros mais próximos. Isto é conseguido seriando o vizinhos por um critério de proximidade aos mineiros. Os vizinhos mais perto dos mineiros ficam listados no topo da tabela e os mais afastados são seriados no fundo da tabela.

Para obter esta classificação, cada nó mantém, para cada vizinho, três variáveis: n, o número total de blocos que recebeu desse vizinho; k a distância acumulada desses blocos até aos mineiros que o produziram; e finalmente a, o número total de blocos recebidos. Para permitir actualizar k, associamos ao processo de propagação de um novo bloco, um contador que regista quantos saltos o bloco deu na rede Bitcoin (este campo é incrementado sempre que um nó reencaminha o bloco). Com base nestas variáveis, a classificação dos vizinhos num determinado intervalo de tempo T é feita de acordo com a seguinte fórmula (em que os valores de k, a e n são os acumulados no período T):

$$\text{class}^T = (\frac{k^T}{n^T} + a^T - n^T)$$

Esta caracterização evita que nós com baixa capacidade de mineração que geram blocos muito esporadicamente, fiquem indefinidamente com uma boa classificação. Desta forma, balanceamos não só os vizinhos que têm melhor rácio distancia aos mineiros/número de blocos providenciados, mas também conseguimos ter em conta o número total de blocos que esse vizinho nos forneceu relativamente ao número total de blocos recebidos.

Uma vez que a classificação de cada um dos vizinhos se vai alterando ao longo do tempo, o valor usado para seriar os vizinhos é uma média deslizante da classificação instantânea acima referida, dada por:

$$\text{class}^t = (1 - \alpha) \cdot \text{class}^{t-1} + \alpha \cdot \text{class}^T$$

O factor α existe para evitar que vizinhos de mineiros que já tenham sido extremamente activos no passado mas que no tempo actual já não o são, permanecem para sempre no topo da tabela. Nas nossas experiências usamos $\alpha = 0.3$ e T é configurado como sendo um intervalo de 4 horas.

Cada vez que um nó recebe um bloco de um vizinho, a classificação é atualizada conforme descrito no Algoritmo 1.

3.2 Propagação Enviesada

Dado que o nosso objectivo é fazer com que as transações que conhecemos cheguem rapidamente aos mineiros podemos utilizar o mecanismo descrito na secção anterior para atingir este objectivo. Assim se estivéssemos perante uma rede onde todos os nós cumpriam com o protocolo e os caminhos fossem resilientes o suficiente poderíamos apenas enviar as transações que recebíamos para o nosso vizinho que tivesse menor valor de classificação e inevitavelmente essas transações seriam adicionadas a um bloco.

Algorithm 1 Computação dos m melhores vizinhos

```
1: function ACTUALIZAR_ESTATISTICAS_NO(no_a_actualizar, saltos_bloco)
2:     nova_class ← recalcular_classificacao(no_a_actualizar, saltos_bloco)
3:     pior_class ← −1
4:     pior_no ← None
5:     for no in m do
6:         if nova_class =< no.class() and pior_class < no.class() then
7:             pior_class ← no.class()
8:             pior_no ← no
9:         end if
10:    end for
11:    if pior_class! = −1 then
12:        m.remove(pior_no)
13:        m.append(no_a_actualizar)
14:    end if
15: end function
```

No entanto, para além das falhas por paragem, temos de ter em conta que os nós podem manipular o k enviado aos seus vizinhos, colocando-o a zero. Deste modo seriam erradamente priorizados e poderiam ficar em vantagem de duas maneiras: conhecer as transações mais cedo, e não propagar as transações recebidas. Para evitar este problema, além de disseminarmos as transações para os vizinhos com mais prioridade, enviamos também para um conjunto aleatório de vizinhos, conforme descrito no Algoritmo 2. Desta forma asseguramos que as transações continuam a ser disseminadas pelo resto da rede mesmo que tenhamos como vizinho um nó malicioso.

O valor de *pi* (Push inicial) indica se quando uma transação é gerada deve ser enviada apenas para m e a ou para todos os vizinhos. Desta forma os valores de *tamanho_m*, *tamanho_a* e *pi* podem depois ser alterados para obtermos diferentes padrões de propagação. Na Secção 4.1 discutimos as diferentes configurações testadas.

4 Avaliação

Para avaliar a abordagem proposta, construímos um simulador de eventos que modela a propagação de transações na rede Bitcoin (este simulador foi desenvolvido em Python). Decidimos implementar o nosso próprio simulador pois todos os outros que encontrámos não implementavam a versão mais actual do protocolo Bitcoin[2].

4.1 Calibração do Simulador

De modo a calibrar o simulador para ser o mais fiel possível ao protocolo original, extendemos o *Bitcoin Core*, o cliente Bitcoin mais usado, para recolher métricas

[2] Entre outros https://github.com/shadow/shadow-plugin-bitcoin e https://github.com/arthurgervais/Bitcoin-Simulator

Algorithm 2 Computação dos nós para enviar anúncios de transações

```
1: function NOS_PARA_ENVIAR(transacao)
2:     if m is not empty or (pi == True and transacao.criador() == mim) then
3:         return todos_vizinhos
4:     end if
5:     total ← tamanho_m + tamanho_a
6:     melhores_nos ← m
7:     if tamanho(todos_vizinhos) < total then
8:         total ← tamanho(todos_vizinhos) − tamanho_m
9:     else
10:        total ← total − tamanho_m
11:    end if
12:    if total > 0 then
13:        nos_aleatorios ← escolha_aleatoria(todos_vizinhos, tamanho_a)
14:    end if
15:    return melhores_nos + nos_aleatorios
16: end function
```

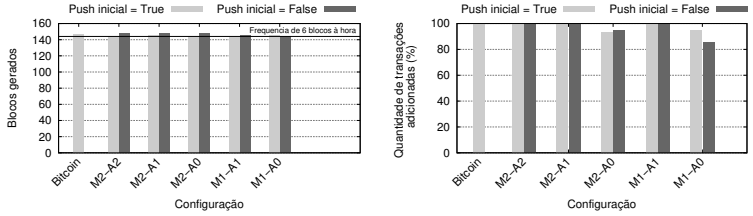
relativamente às mensagens trocadas pelos clientes. As métricas recolhidas são as seguintes: i) anúncios de transações; ii) transações recebidas; iii) transações presentes em blocos compactos que o nó não possuía e tinha de pedir de forma a poder reconstruir o bloco. Corremos duas instâncias deste cliente, em localizações físicas distintas durante um mês e usámos as métricas recolhidas para calibrar e validar o simulador. Além disso usámos a informação publicamente disponível em https://blockchain.info/ para determinar o número de transações criadas por dia, a distribuição de blocos gerados por mineiro, e o tamanho médio de uma transação. Munidos destas métricas, implementámos o protocolo original no simulador, bem como as nossas modificações. Calibrámos experimentalmente o simulador de modo a que os resultados observados no Bitcoin fossem equivalentes às observações efetuadas no cliente real. O modelo de rede que usámos para as simulações foi composto apenas por nós que seguem o protocolo devidamente.

Devido às simulações serem computacionalmente dispendiosas e demorarem muito tempo a correr (na ordem dos dias) para redes grandes, fizemos testes preliminares para verificarmos se podíamos diminuir proporcionalmente o tamanho da rede e respectivas propriedades sem comprometer os resultados obtidos. Para este efeito, corremos o protocolo original com 6000 nós e com 625 nós e comparámos várias métricas, que discutiremos mais adiante. Os resultados obtidos foram equivalentes para ambos os tamanhos de rede pelo que, no resto desta secção consideramos uma rede de 625 nós. Este escalonamento proporcional entre 6000 nós (tamanho de rede registado quando começamos as experiências) e 625 nós permitiu-nos explorar mais rapidamente o espaço de soluções e correr várias instâncias de cada teste. Os resultados apresentados são a média de 3 testes independentes e correspondem a 24 horas de tempo real.

4.2 Impacto da Propagação Enviesada

Começámos por explorar o espaço de soluções à procura de compromissos que nos permitam reduzir o consumo de recursos de rede sem ter um impacto negativo nas propriedades dos sistema. Para ser mais fácil denominar as diferentes experiências realizadas usamos a seguinte notação: *Mn*, em que n especifica o valor da variável *tamanho_m* ; e *An*, em que n especifica o tamanho da variável *tamanho_a* conforme especificado no Algoritmo 2.

Inicialmente, testámos as combinações de $n = 1, 2, 3, 4$ quer para M quer para A. Após este conjunto preliminar de experiências, observámos que os resultados para $n = 3, 4$ eram praticamente indistinguíveis de $n = 1, 2$, com a excepção do número de duplicados que permaneceu perto do obtido na Bitcoin original. Estes resultados suportam também as medições no cliente real, onde observámos um número médio de duplicados de 3.33. Deste modo, para o resto das experiências efectuadas, consideramos somente as combinações: M2_A2; M2_A1; M2_A0; M1_A1; M1_A0. Adicionalmente, para cada configuração variámos o valor da variável *pi*. Nos dados apresentados abaixo descartámos também a primeira e a última hora de simulação de forma a podermos estudar o sistema em estado estável.



(a) Quantidade de blocos criados.

(b) Percentagem de transações adicionadas aos blocos.

Fig. 1: Blocos criados e percentagem de transações adicionadas aos blocos nos diferentes cenários considerados.

A Figura 1a apresenta, para cada configurações a quantidade de blocos que foi gerada ao longo da experiência, enquanto que a Figura 1b mostra a percentagem de transações que foram adicionadas aos blocos. Para o cenário Bitcoin obtemos o número esperado de blocos num dia (≈ 144) bem como o número de transações incluídas nos blocos ($\approx 100\%$). Todas as configurações produzem aproximadamente o mesmo número de blocos, contudo para as configurações M2_A0 e M1_A0 nem todas as transações são incluídas nos blocos. Na prática isto quer dizer que nem todas as transações chegam a todos os mineiros, e ilustra a importância de manter a aleatoriedade da disseminação.

Na Figura 2 estudamos o tempo médio desde que uma transação é criada até ser incluída num bloco. As linhas horizontais denotam o tempo médio para uma transação ser incluída num bloco no Bitcoin. Mais uma vez, a aleatoriedade ajuda não só a a que as transações cheguem a todos os nós como também diminui

consideravelmente o tempo necessário para as transações serem incluídas nos blocos.

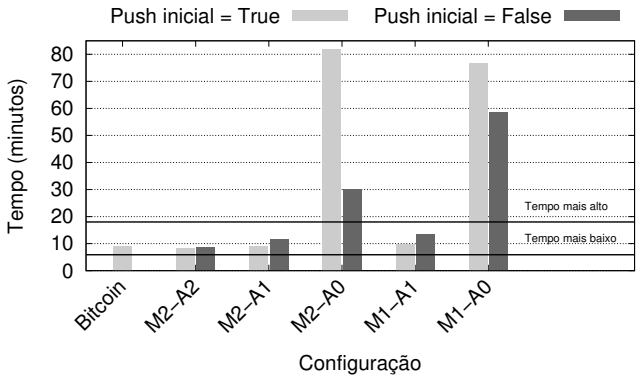


Fig. 2: Tempo médio que demora desde que uma transação é criada até ser incluída num bloco.

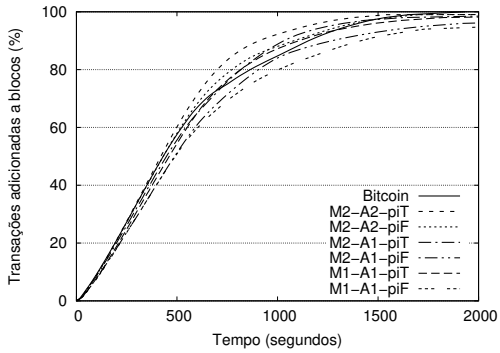


Fig. 3: Função de distribuição acumulada do tempo necessário para uma transação ser incluída num bloco.

A Figura 3 apresenta a função de distribuição acumulada do tempo necessário para incluir transações num bloco, sendo portanto uma perspectiva diferente da Figura 2. Curiosamente, enviar a transação a primeira vez para todos os vizinhos ($pi{=}T$) tem um impacto residual no tempo que as transações demoram a ser incluídas no bloco. Isto deve-se ao facto de a taxa de criação de blocos ser bastante reduzida face ao tempo de propagação fim-a-fim da rede.

Analisado o impacto nos efeitos observáveis da rede, nomeadamente o tempo que uma transação demora a ser incluída num bloco e a quantidade de transações incluídas, focamo-nos agora no ganhos, em termos de mensagens poupadas e redução da quantidade de informação transmitida.

A Figura 4 mostra o número total de mensagens que foram enviadas em percentagem das enviadas na rede Bitcoin. Como esperado, as configurações com

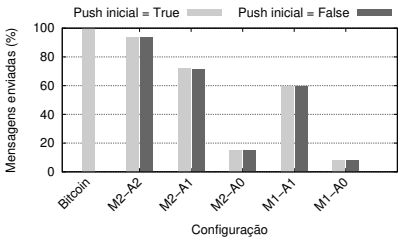


Fig. 4: Numero total de mensagens enviadas.

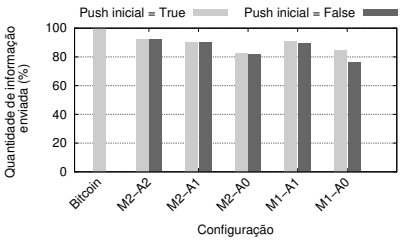


Fig. 5: Quantidade de informação enviada.

mais poupanças são as que não enviam transações para nós aleatórios, pois é provável que esses nós não tenham a transação e logo irão pedi-la ao emissor. No entanto, como vimos anteriormente, estas configurações não são viáveis pois nem todas as transações são incluídas e, as que são, tendem a demorar bastante tempo. A Figura 5 mostra a quantidade de informação total enviada em percentagem que, como esperado, segue uma tendência semelhante à Figura 4. Podemos observar que a poupança em quantidade de informação enviada não é tão significativa como a das mensagens, pois as mensagens de anúncio são pequenas. No entanto, na prática, processar essas mensagens desnecessariamente impõe um custo nos nós.

Analisando estes resultados, é possível concluir que a configuração mais atrativa é M1_A1 com $pi{=}F$ pois obtém bons ganhos (redução da quantidade de mensagens em 40.5% e redução total da quantidade de informação enviada em 10.7%) ao mesmo tempo que preserva as propriedades da Bitcoin original.

5 Trabalho Relacionado

Na Bitcoin a disseminação da informação é um dos mecanismos mais importantes para garantir o bom funcionamento da rede. Vários estudos têm-se focado em analisar a disseminação de informação, e a medida em que os problemas na implementação atual podem levar a bifurcações [4, 3, 8]. Outros trabalhos, focam-se em como explorar as vulnerabilidades existentes nos mecanismos de disseminação de modo a beneficiar o atacante, ou prejudicar a vítima [1] [10] [7]. Por exemplo atrasar a disseminação de informação pode conduzir a que mineiros adversários não tenham o bloco mais recente e logo estejam em desvantagem.

Os mecanismos de disseminação já sofreram diversas modificações desde a sua criação [9]. Parte destas modificações estão reflectidas em vários *Bitcoin Improvement Proposals* [2, 5] no entanto estes não refletem a implementação atual do cliente de referência que tem modificações adicionais que só conseguimos descobrir após uma análise aprofundada do código fonte. Tanto quanto conseguimos apurar até à data da escrita deste artigo, este é o primeiro trabalho com uma abordagem sistemática focada em melhorar o desempenho e robustez da disseminação da informação.

6 Conclusão e Trabalho Futuro

Apesar do protocolo de disseminação da Bitcoin ter sofrido várias iterações e melhorias desde que o sistema foi originalmente introduzido, ainda existem hoje em dia várias ineficiências. Dado o tamanho da rede, e a cada vez maior adoção deste tipo de sistemas, torna-se imperioso construir sistemas cada vez mais eficientes, sem comprometer a sua correção e robustez. Neste artigo propusemos várias melhorias ao algoritmo de disseminação que obtêm reduções da largura de banda na ordem dos 10.7% e uma redução no número de mensagens trocadas na ordem dos 40.5%.

Como trabalho futuro, tencionamos estudar mais aprofundadamente o espaço de soluções à procura de combinações que permitam melhorar os resultados obtidos, bem como enriquecer o modelo de faltas dos nós maliciosos. Adicionalmente, pretendemos estudar em que medida o sistema de filiação pode ser enriquecido com a informação recolhida de modo a permitir construir soluções mais eficientes.

Agradecimentos Este trabalho é parcialmente financiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER) através do Programa Operacional Regional de Lisboa e por Fundos Nacionais através da FCT - Fundação para a Ciência e a Tecnologia no âmbito do projeto LISBOA-01-0145-FEDER-031456 e UID/CEC/50021/2013.

References

1. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking bitcoin: Routing attacks on cryptocurrencies. arXiv preprint arXiv:1605.07524 (2016)
2. Corallo, M.: Compact block relay (2016), https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki
3. Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E.G., et al.: On scaling decentralized blockchains. In: International Conference on Financial Cryptography and Data Security. pp. 106–125. Springer (2016)
4. Decker, C., Wattenhofer, R.: Information propagation in the bitcoin network. In: Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on. pp. 1–10. IEEE (2013)
5. Developers, B.: Bitcoin improvement proposal (2018), https://github.com/bitcoin/bips
6. Eugster, P.T., Guerraoui, R., Kermarrec, A.M., Massoulie, L.: Epidemic information dissemination in distributed systems. Computer **37**(5), 60–67 (May 2004). https://doi.org/10.1109/MC.2004.1297243
7. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: International conference on financial cryptography and data security. pp. 436–454. Springer (2014)
8. Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., Bhattacharjee, B.: Discovering bitcoin?s public topology and influential nodes. et al. (2015)
9. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
10. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 515–532. Springer (2016)