Construção Observável de um Quorum de Nós Não-Sybil na Vizinhança Rádio de uma Rede Ad Hoc Sem Fios *

Diogo Mónica
INESC-ID/IST
diogo.monica@gsd.inesc-id.pt

João Leitão
INESC-ID/IST
jleitao@gsd.inesc-id.pt

Luís Rodrigues
INESC-ID/IST
ler@ist.utl.pt

Carlos Ribeiro
INESC-ID/IST
carlos.ribeiro@ist.utl.pt

Abstract

O ataque Sybil é uma ameaça importante à operação segura e confiável das redes ad hoc sem fios. Propomos um algoritmo para a construção de um quorum parcialmente consistente de nós não sybil, na vizinhança local de uma rede sem fios. O algoritmo é baseado na combinação de diferentes tipos de testes de recursos, de forma a garantir não só a detecção e posterior exclusão das identidades sybil, e também garantir a eficiência na construção do quorum. O algoritmo garante que todos os nós correctos no sistema possuem um quorum válido.

Abstract *The sybil attack is a relevant treat to the secure operation of wireless ad hoc networks. We propose a novel algorithm to construct a partially consistent sybil-free quorum on a wireless one-hop neighborhood. The algorithm is based on the combination of distinct resource tests that ensure the detection and removal of sybil identities, and the efficiency of the quorum construction. Our algorithm ensures that all correct participants in the system own a valid quorum.*

1 Introdução

Os quorums têm vindo a ser utilizados como um mecanismo para aumentar a disponibilidade e eficiência de vários serviços em sistemas distribuídos [9, 10]. Tipicamente nestes sistemas, existe um conjunto de servidores que disponibiliza o mesmo serviço de forma replicada a um conjunto de clientes. De forma a garantir a operação correcta destes sistemas, qualquer operação deve ser realizada sobre uma fracção de réplicas, de forma a que todas as operações se interceptem num número suficiente de réplicas (tipicamente, o necessário para mascarar a existência de réplicas incorrectas). Em redes sem fios, a possibilidade de um nó malicioso usar múltiplas identidades torna a construção e utilização de um quorum uma tarefa não trivial.

A utilização de múltiplas identidades por um nó malicioso é designada na literatura por "Ataque Sybil". Este ataque consegue comprometer a integridade dos sistemas baseados em quorums, assim como de um conjunto de protocolos distribuídos; e.g. armazenamento, encaminhamento, agregação de dados, votação, detecção de intrusões e partilha de recursos [4, 13]. Embora tenham sido propostas várias soluções para impedir ou mitigar este ataque [16, 17, 13], a maior parte delas requer a existência de uma entidade centralizada confiável ou a

*Este trabalho foi parcialmente suportado pela FCT com a bolsa PTDC/EIA/65588/2006 e pelo LEMe pelo projecto WiMesh.

existência de segredos pré-partilhados, sendo sua utilização impraticável em redes ad hoc expontâneas, e.g. sem pré-configuração [14].

Uma das técnicas que pode ser usada para combater o ataque Sybil em redes ad hoc passa pela utilização de testes aos recursos. Estas soluções funcionam partindo da premissa de que é possível estabelecer um limite à quantidade de recursos disponível em cada nó. A verificação desta premissa num conjunto de identidades permite verificar se existem no sistema mais identidades do que nós.

Este artigo propõe um algoritmo de criação de um quorum de tamanho q, constituído por identidades não sybil, o Quorum Não-Sybil (QNS), numa vizinhança rádio de uma rede ad hoc sem fios (i.e. uma rede com um único salto). No final da execução do algoritmo, todos os nós correctos possuem um quorum constituído por uma maioria de identidades utilizadas por nós correctos, e que se intercepta globalmente em $q - f$ destas identidades. O tamanho do quorum final é um parâmetro do algoritmo. Por exemplo, se existirem f nós bizantinos na vizinhança de um nó, poderá ser necessário ter um quorum de tamanho q igual a $3f + 1$, de forma a que este seja tolerante a falhas bizantinas. A existência de um quorum permite a delegação, num conjunto de nós correctos, de decisões críticas relativamente à operação da rede. O algoritmo QNS utiliza uma combinação de diferentes tipos de testes de recurso, de forma a tonar a sua execução mais eficiente e robusta.

O resto do artigo está organizado da seguinte forma. Na Secção 2 é introduzido o modelo e enunciado o problema a resolver. A Secção 3 descreve a nossa aproximação ao problema. Na Secção 4 abordamos a correcção da solução apresentada. Finalmente, na Secção 5 concluímos o artigo, fornecendo algumas direcções para trabalho futuro.

2 Modelo de Sistema e Enunciado do Problema

2.1 Modelo de sistema

Neste artigo consideramos uma rede sem fios síncrona, ponto-para-multiponto, composta por N nós. Adicionalmente, fazemos as seguintes hipóteses sobre o funcionamento do sistema.

Nós Correctos e Nós Bizantinos Um nó pode ser correcto ou bizantino (não-correcto). Um nó bizantino pode exibir um comportamento não previsível, como por exemplo o envio de mensagens arbitrárias. Os nós bizantinos podem também estar em conluio entre si. No máximo existem $f < N$ nós bizantinos no sistema.

Sincronismo Assumimos que o sistema é síncrono e que o tempo é modelado como uma sequência de passos independentes. Em cada passo, todos os nós executam uma quantidade limitada de computação, e podem enviar ou receber uma mensagem de/para a rede. Em cada passo pode ser enviada no máximo uma mensagem em cada um dos canais de comunicação disponíveis. Se num determinado passo mais do que dois nós enviarem uma mensagem no mesmo canal rádio, assumimos que ocorre uma colisão.

Recursos Limitados Assumimos também que todos os nós, quer sejam correctos ou bizantinos, possuem apenas um dispositivo de rádio e uma capacidade de computação limitada. Os dispositivos de rádio podem operar em qualquer um de K canais disponíveis, não podendo no entanto, comunicar em mais do que um canal simultaneamente. Finalmente, limitamos também o número de colisões intencionais ci, que um nó bizantino pode provocar numa sequência de passos de dimensão P. Neste cenário assumimos que nós correctos têm oportunidade não nula de comunicar se a condição $ci \cdot f < P$ for verdadeira (os nós correctos podem, no entanto, gerar também colisões não intencionais)[1].

Comunicação Os nós comunicam através do envio de mensagens numa reda partilhada, sem fios, numa única vizinhança rádio. Assumimos que todas as mensagens são enviadas em modo de difusão, num canal único, e recebidos por todos os outros nós que se encontrem na vizinhança rádio a escutar esse canal. Salvo informação explícita em contrário, toda a comunicação entre os nós é realizada através de um canal único. Como se verá, os restantes canais serão utilizados para realizar testes de recurso rádio. Assumimos também que os canais de

[1] Note-se que esta hipótese baseia-se numa outra hipótese mais genérica, de que os nós têm limites à capacidade de transmissão

transmissão são confiáveis: quando não há colisão, as mensagens são entregues sem perdas ou corrupção dos dados. Adicionalmente, assumimos que sempre que uma colisão ocorre no meio sem fios, todos os nós, incluindo os emissores da mensagem, são capazes de detectá-la. Este objectivo pode ser atingido através da utilização de um mecanismo de reforço de colisão: Após a detecção de uma colisão, os nós correctos transmitem por tempo suficiente para garantir que todos os nós que transmitiram as mensagens que causaram a colisão estão cientes de que esta ocorreu [7]. Quando ocorre uma colisão, nenhum nó é capaz de receber uma mensagem. Finalmente, os recursos limitados dos nós impedem que o meio de comunicação seja alvo de ataques de negação de serviço, como por exemplo, através da transmissão contínua de sinal para o meio.

Nós e identidades Todas as mensagens enviadas por um nó estão associadas a uma *identidade*. O receptor de uma mensagem consegue sempre associar uma mensagem recebida com a sua identidade de origem, excluindo-se a possibilidade de falsificação da origem das mensagens. Esta propriedade pode ser garantida através da utilização de criptografia de chave pública, e fazendo com que cada nó assine digitalmente todas as mensagens enviadas com a chave associada à identidade. Note-se que a utilização de criptografia de chave pública não implica a utilização de um PKI: pares de chaves assimétricas pública-privada podem ser geradas aleatoriamente, sem necessitar da intervenção de terceiros, e utilizadas como identidades.

Um nó pode usar uma ou mais identidades para comunicação. Representamos como *ids(n_i)* o conjunto de identidades utilizadas por um nó n_i. De forma inversa, representamos como *usa(id)* o conjunto de nós que utilizam uma certa identidade *id*. Um nó correcto utiliza sempre uma única identidade que não é usada por mais nenhum outro nó na rede (correcto ou bizantino). De forma inversa, os nós bizantinos podem tentar utilizar mais do que uma identidade para comunicação. Estas múltiplas identidades são chamadas identidades sybil. Assim, para identidades utilizadas por nós correctos, temos que $\{id_i\} = ids(n_i)$ e $\{n_i\} = usa(id_i)$. Não existe limite para o número de identidades $|ids(b)|$ que um nó bizantino b pode utilizar. Adicionalmente, e uma vez que os nós bizantinos podem estar em conluio, a mesma identidade pode ser utilizada por múltiplos nós bizantinos. Consequentemente, se *bid* é uma identidade utilizada por um nó bizantino, temos $1 \leq |usa(bid)| \leq f$.

2.2 Enunciado do Problema

O nosso objectivo é retornar um quorum de identidades QNS_i, em cada nó correcto n_i, cujo tamanho alvo (e máximo) é dado pelo parâmetro q. As identidades em cada QNS_i podem pertencer a nós correctos ou bizantinos. Repare-se que, uma vez que os nós bizantinos podem deixar de operar em qualquer momento, o quorum final resultante pode conter menos do que q identidades, contendo no mínimo $q - f$. Assim, a intercepção do quorum retornado em cada nó correcto, contem pelo menos $q - f$ identidades pertencentes a nós correctos. Mais precisamente, o problema pode ser definido através das seguintes propriedades:

Quorum Parcialmente Consistente e sem Sybils Existe um conjunto de identidades de nós correctos (QNS), comum a todos os QNS_i, tal que $q \geq |QNS| \geq q - f$, com uma probabilidade arbitrariamente perto de 1. Defina-se $\Gamma()$ como:

$$\Gamma(n_i) = \begin{cases} 1, \text{ if } n_i \text{ é um nó correcto} \\ 0, \text{ caso contrário} \end{cases} .$$

Seja $QNS = QNS_1 \cap QNS_2 \cap ... \cap QNS_{N-f}$ o conjunto de identidades comuns a todos os QNS_i dos nós. Então,

$$\sum_{n_i}^{usa(QNS)} \Gamma(n_i) \geq q - f.$$

Terminação Probabilista Todos os nós correctos retornam o quorum com uma probabilidade arbitrariamente próxima de 1, num número finito de passos.

algorithm QNS **is**
$C \longleftarrow \emptyset$; //conjunto de identidades candidatas ao quorum
nonce $\longleftarrow$ null; // string, nonce para o TRC
nonce $\longleftarrow$ nonceGeneration(n); // Primeira fase
$C \longleftarrow$ candidateSelection (nonce); // Segunda fase
$C \longleftarrow$ quorumValidation (C); // Terceira fase
return C;

Figura 1. Esqueleto do algoritmo de construção do QNS (executado em cada nó i).

O tamanho alvo do conjunto QNS (q), é um parâmetro do nosso algoritmo. Repare-se que não é possível impedir que os nós bizantinos façam parte do quorum. Caso um nó bizantino se comporte de forma correcta, não é possível distingui-lo de um nó correcto, e consequentemente, pode vir a fazer parte do quorum final. Assim, o parâmetro q tem de ser escolhido de acordo com o tipo de aplicações para o qual o quorum será utilizado. Tipicamente, q terá um valor que assegure que o número de nós bizantinos não consegue influenciar a aplicação para a qual o quorum está a ser usado (e.g.: $q = 3f + 1$ [8]).

3 Construção do Quorum não-Sybil

O algoritmo de construção do quorum não-Sybil (QNS) utiliza três fases distintas, a saber: fase de geração do *nonce*; fase de selecção de candidatos; e a fase de validação do quorum. O esqueleto do algoritmo encontra-se representado na Figura 1. Os parâmetros do algoritmo serão descritos mais à frente.

No centro do algoritmo está a fase de selecção de candidatos (fase 2). O objectivo desta fase é encontrar um conjunto adequado de identidades, de tamanho σ, para formar o quorum não-sybil. Este conjunto é obtido recorrendo a um *teste de recursos computacional* (TRC). Para cada identidade que pretende propor, um nó necessita de resolver um problema que lhe consumirá tempo de processador. Atendendo aos recursos limitados que cada nó possui, este teste limita o número de identidades que cada nó pode propor. De modo a evitar a utilização de soluções calculadas previamente, o problema possui como parâmetro um nonce desconhecido até ao momento da sua utilização. Este nonce é obtido através da combinação da contribuição de n identidades distintas (n é outro dos parâmetros do algoritmo QNS). Este é o propósito da primeira fase do algoritmo. Como se verá, é extremamente difícil recorrendo apenas a um TRC assegurar que não existem entidades Sybil na lista de candidatos. Por este motivo, o algoritmo possui uma terceira fase, que recorre a um *teste de recursos rádio* (TRR), e que possui por objectivo eliminar identidades sybil do conjunto de candidatos e provar a todos os restantes elementos da rede que as identidades do quorum são, de facto, usadas por, pelo menos, um igual número de nós distintos.

3.1 Geração do Nonce

O objectivo da fase de geração do nonce é a criação de uma sequência de dígitos aleatória, que depende das contribuições de n identidades distintas. Nesta fase, o algoritmo itera através de uma série de passos de comunicação, onde em cada passo um nó ou tenta contribuir para o nonce, ou apenas escuta outras contribuições. Quando um ou mais nós tentam contribuir para o nonce no mesmo passo, é gerada uma colisão. Como foi descrito previamente, no caso da existência de uma colisão, todos os nós (incluindo os emissores da mensagem que colidiu), detectam a colisão e descartam as contribuições, garantido a coerência do nonce em todos os nós correctos.

O procedimento da geração do nonce é simples. São recolhidas contribuições dos participantes e colocadas num conjunto, pela ordem pela qual foram recebidas. Este procedimento termina assim que o conjunto atinge um tamanho alvo n. O algoritmo encontra-se representado na Figura 2. Inicialmente, o conjunto de contribuições para

```
algorithm nonceGeneration (n) is
    nonceSet ⟵ ∅
    while (|nonceSet| < n) do
        if id_i ∉ nonceSet and rand() < p_t then
            rndvalue ⟵ rand();
            if broadcast.send ([id_i, rndvalue]) ≠ collision then
                nonce ⟵ nonce ∪ [id_i, rndvalue];
        else
            msg ⟵ broadcast.receive();
            if msg ≠ null and msg ≠collision then
                if source(msg) ∉ ids(nonce) then
                    nonce ⟵ nonce ∪ msg;
    return hash(values(nonce));
```

Figura 2. Geração do nonce (executado em cada nó i).

o nonce é vazio. Em cada passo, os nós que ainda não conseguiram adicionar a sua contribuição, podem tentar difundi-la através do envio de uma mensagem que contem o identificador do nó, e um valor aleatório. Para evitar colisões desnecessárias, os nós apenas transmitem com uma probabilidade p_t. Caso não seja gerada nenhuma colisão devido a esta transmissão, todos os nós adicionam a contribuição do emissor ao nonce.

Os nós que, num determinado passo, não tentarem adicionar o seu identificador, vão escutar o meio de forma a receberem contribuições dos outros nós. Se um nó recebe uma contribuição de qualquer outro participante, adiciona-a ao conjunto de contribuições. Este passo não gera incoerências, uma vez que assumimos que no evento de uma colisão todos os nós recebem a mensagem correctamente. É possível que um nó que esteja à escuta não receba nenhuma contribuição. Isto pode acontecer ou porque nenhum nó decidiu difundir a sua identidade nesse passo, ou devido à existência de uma colisão.

Este mecanismo é suficiente para garantir que, desde que haja mais do que n nós no sistema, todos os participantes correctos vão convergir para o mesmo conjunto de geração do nonce. O facto dos nós possuírem recursos limitados, assegura que se n for suficientemente grande, pelo menos um nó correcto consegue contribuir para este conjunto. No final, todos os nós retornam o mesmo valor de forma determinista, recorrendo à utilização de uma função de síntese criptográfica (e.g., a SHA-1 [6])).

O valor de n deve garantir que pelo menos um nó correcto consegue participar na geração do nonce. Assumindo que um nó malicioso pode transmitir ci mensagens em P passos, n deve respeitar: $ci \cdot f < n \leq P$. Note-se que com esta condição este mecanismo termina, de forma correcta, num número finito de passos, visto que eventualmente n nós correctos terão a oportunidade de transmitir.

3.2 Selecção de Candidatos

O objectivo da fase de selecção de candidatos é obter um conjunto de σ identidades que são candidatos para formar o quorum de q identidades não-sybil. Para evitar que os nós bizantinos proponham múltiplas identidades sybil para este conjunto de candidatos, esta fase recorre à utilização de um teste de recurso computacional (TRC). A ideia passa essencialmente por fazer com que os nós sejam obrigados a resolver um puzzle criptográfico antes de conseguirem propor uma identidade. Este puzzle recebe como parâmetros o nonce gerado na fase anterior, e a identidade a ser proposta. Assim, esta fase de selecção de candidatos consiste em: receber e validar as propostas que são enviadas assim que outros nós resolvem o puzzle criptográfico; e resolver o puzzle criptográfico, para tentar fazer parte dos candidatos.

```
algorithm candidateSelection(nonce,q) is
    C ⟵ ∅;  done ⟵ 0;  t ⟵ 0;
    σ ⟵ ; // Número de propostas necessárias
    Task 1: // Resolução do crypto-puzzle
        a_i ⟵ trc.solve(nonce, id_i);
        t ⟵ 1;
    Task 2: // Verificação da resposta
        while |C| ≤ σ | timeout do
            if t = 1 and rand() < p_t and done = 0 do
                if broadcast.send(id_i,a_i) ≠ collision do
                    C ⟵ C ∪ {id_i};
                    done ⟵ 1;
            else
                m ⟵ broadcast.receive();
                if m = (PROPOSAL) do
                    if trc.verify(nonce,m.id, m.a) do
                        C ⟵ C ∪ {id};
        return C;
```

Figura 3. Fase de selecção de candidatos (executado em cada nó i).

Este procedimento assume que o tempo médio para efectuar o teste de recurso computacional (i.e., para resolver o puzzle criptográfico), é muito superior ao tempo necessário para disseminar (e validar) todas as propostas dos σ nós. Desta forma, no momento em que um nó bizantino consegue acabar o cálculo do puzzle criptográfico para uma segunda identidade (sybil), já todos os nós correctos foram capazes de difundir as suas propostas (legítimas).

Repare-se que caso um nó bizantino conheça o nonce *a priori*, é capaz de pré-calcular tantas soluções quantas necessário, de forma a conseguir propor um qualquer número arbitrariamente grande de identidades sybil ao sistema. Daqui a importância da fase de geração do nonce.

A Figura 3 apresenta o pseudo-código para a fase de selecção de candidatos. O algoritmo usa duas tarefas. Uma das tarefas é utilizada para resolver o puzzle criptográfico, e retornar a solução numa variável t. A outra tarefa é utilizada para receber as respectivas propostas dos outros participantes ou, caso a solução do puzzle já tenha sido calculada, para propor a sua identidade. Uma mensagem de PROPOSAL, inclui a identidade que está a ser proposta e a respectiva solução para o puzzle criptográfico. Quando os nós recebem uma proposta, validam a solução do puzzle. Caso seja correcta, todos os nós adicionam a identidade correspondente ao conjunto de candidatos.

Idealmente, um TRC perfeito não permitiria que um participante resolvesse um segundo puzzle a tempo de propor mais que uma identidade para a lista de candidatos. No entanto, a natureza destes testes faz com que o tempo de resolução do puzzle possua uma distribuição aleatória, existindo uma probabilidade não-zero de um nó bizantino propor mais que uma identidade. No entanto, assumindo que existe um limite $s > 0$ ao número total de identidades sybil que os nós bizantinos conseguem propor no TRC, podemos obter o número de respostas necessárias σ, de forma a conseguir garantir que são propostas pelo menos $q - f$ identidades correctas no pior caso, $\sigma \geq q + s$. O limite s é possível de demonstrar, podendo o leitor interessado encontrar mais detalhes em [12].

3.2.1 Teste de Recurso Computacional

Existem vários formas de concretizar um TRC. A utilização de TRCs foi inicialmente proposta em [5] como um método de defesa contra e-mail não desejado ("spam"), e mais tarde usada como um método de defesa contra ataques de negação de serviço em [1, 3]. Em [2], os autores usam um TRC para verificarem se os participantes de um sistema possuem a quantidade expectável de poder de computação. Este teste, intitulado crypto-puzzle, consiste em forçar as identidades participantes do sistema a resolver um problema criptográfico, solúvel apenas por cálculo de força-bruta, num certo limite de tempo. Desta forma, um nó com restrições de poder computacional, tem um limite ao número de puzzles criptográficos que consegue resolver num certo período de tempo, garantido assim um limite superior no número de identidades sybil que consegue apresentar ao sistema. No nosso algoritmo, assumimos que o teste de recurso computacional tem o seguinte interface: Um nó tem de invocar *trc.solve(nonce,id)* para resolver o puzzle criptográfico, usando um nonce e uma identidade como parâmetros de entrada; a invocação retorna a resposta ao puzzle. Existe também um método *trc.verify(nonce, id,* t_{id}*)* que permite qualquer nó validar uma resposta ao puzzle criptográfico. Assumimos que uma resposta pode ser verificada num único passo.

3.2.2 Exemplo de Puzzle Criptográfico

Embora o nosso algoritmo não esteja limitado a um puzzle criptográfico específico, damos o exemplo de um puzzle criptográfico que cumpre os nossos requisitos. Dada uma função de síntese criptográfica $\mathcal{H}$, e um operador $\mathbf{left}_w(str)$, que retorna uma sub-sequência composta pelos w dígitos mais à esquerda da sequência str, o puzzle criptográfico pode ser definido da seguinte forma (t_{id} é a solução para o puzzle criptográfico, w um parâmetro estático de configuração do puzzle, que controla a *dificuldade*):

$$trc.solve(nonce, id) = \{t \in \Re : left_w(\mathcal{H}(nonce||id||t)) = 0^w\}$$

$$trc.verify(nonce, id, t) = \begin{cases} true, \ \mathbf{left}_w(\mathcal{H}(nonce||id||t)) = 0^w \\ false, \ \mathbf{left}_w(\mathcal{H}(nonce||id||t)) \neq 0^w \end{cases}$$

De forma a resolver o puzzle criptográfico, o nó tem de encontrar a sequência de dígitos t, tal que o valor resultante da aplicação da função de síntese criptográfica à concatenação do nonce, a identidade e t, é uma sequência de dígitos onde os w dígitos mais à esquerda são zero.

O algoritmo conhecido mais rápido para a resolução de uma colisão parcial numa função de síntese criptográfica é o calculo através de força bruta [15]. Note-se que o puzzle não é interactivo, uma vez que o nonce torna-se globalmente conhecido na fase anterior do nosso protocolo, e id é a identidade para a qual o nó se encontra a calcular a resposta. Esta não-interactividade é importante, porque faz com que o puzzle criptográfico seja globalmente confiado, e não esteja associado um par de nós específicos.

A verificação da validade da resposta ao puzzle pode ser facilmente obtida através do calculo de uma operação de síntese criptográfica da resposta transmitida por um qualquer nó, e da verificação de existência de zeros nos primeiros w dígitos da sequência resultante. Além disso, este puzzle, em particular, é publicamente verificável, uma vez que pode ser verificado de forma independente por qualquer participante do sistema, sem necessidade de possuir um segredo.

3.3 Validação do Quorum

A fase de validação do quorum é capaz de detectar identidades sybil num conjunto de identidades C, através de um teste de recurso rádio. Este teste é baseado na hipótese que nenhum nó tem a capacidade de transmitir simultaneamente em dois canais distintos. Ao obrigar os utilizadores de cada identidade a transmitir num canal diferente ao mesmo tempo, impede-se um nó bizantino de defender mais do que uma identidade. Infelizmente, como os nós que pretendem validar essa identidade também não podem ouvir em mais do que um canal ao mesmo

tempo, o teste tem que ser repetido várias vezes para que com elevada probabilidade se detecte um nó que pretenda defender mais do que uma identidade. Por outro lado se o número de nós a ser testado for superior ao número de canais disponíveis é necessário testar todas as combinações de nós, pelo que o teste tem que ser repetido várias vezes.

O método "trr.lenght (C)" retorna o número de passos necessários para testar $|C|$ identidades com K canais rádio, para que com elevada probabilidade se detecte as identidades sybil. O método "trr.schedule(C)" retorna uma lista de identidades que são supostas transmitir em cada passo do teste. Assumimos que os canais são atribuídos às identidades do 1 até ao canal h, em ordem crescente.

Esta fase do algoritmo é representada na Figura 4, e executa-se durante um período fixo de passos, definidos pela função "trr.lenght()". Em cada passo, o nó verifica se é suposto transmitir. No caso afirmativo, transmite uma mensagem de VALIDATE no canal especificado pelo agendamento do teste de recurso rádio. Caso contrário, escuta em um dos canais de rádio disponíveis, aleatoriamente. Note-se que os nós que não pertencem ao conjunto de candidatos, escutam sempre, em todos os passos. Caso seja detectado silêncio no canal (não houver transmissão), a identidade que estava a agendada para transmitir nesse canal é adicionada à lista de exclusão.

Esta fase termina com o retorno de no máximo q identidades da seguinte forma. Todos os nós ordenam, por ordem lexicográfica, as identidades validadas pelo teste de recurso de rádio. Este conjunto é concatenado com uma ordenação similar das identidades excluídas pelo teste. Do conjunto resultante cada nó selecciona as primeira q identidades, e esse conjunto é considerado como o quorum final QNS.

Note-se que todos os nós retornam um conjunto com no mínimo $q - f$ identidades ainda que que cada nó possa retornar um conjunto diferente, uma vez que nós bizantinos podem defender diferentes identidades Sybil em fases diferentes do TRR. As propriedades do teste de recurso de rádio, e a ordenação dos conjuntos, garantem que o quorum retornado em cada nó correcto respeita a propriedade de "Quorum Parcialmente Consistente e sem Sybils" (Secção 2.2). Iremos discutir esta propriedade na Secção 4.

3.3.1 Teste de Recurso Rádio

O teste de recurso rádio (TRR) é um tipo particular de um teste de recursos. Um TRR assume que cada um dos nós tem acesso a apenas um único dispositivo de rádio, e utiliza limitações conhecidas destes dispositivos. Por exemplo, se os dispositivos de rádio forem incapazes de transmitir simultaneamente em frequências diferentes, este facto pode ser explorado. Assim como a maior parte das soluções de teste de recursos, os TRR têm o potencial de suportar protocolos que não necessitem de pré-configuração ou segredos pré-partilhados. Os TRRs têm ainda uma vantagem adicional comparativamente com os outros tipos de testes de recursos; tanto quanto sabemos, são o único tipo de testes de recursos que permite que identidades não participantes num teste específico consigam avaliar, e validar os resultados do teste, não necessitando para isso de confiança em nenhum dos participantes.

Em [11], analisámos a complexidade de vários TRRs diferentes. Mostrou-se que o custo de correr sistematicamente TRRs de forma a detectar identidades sybil, numa população com muitas identidades, é proibitivamente caro em termos de mensagens, limitando a escalabilidade das soluções que o façam. Esta observação motiva a necessidade de explorar métodos alternativos de aproveitar as vantagens dos TRRs, evitando o custo de testar cada uma das identidades que participam na rede, permitindo melhorar o desempenho destas soluções.

3.3.2 TRR Utilizado

O algoritmo apresentado na Figura 4 pressupõe que são testadas todas as combinações das $\mathcal{C}$ identidades, K a K. Para além disso, como referimos, para cada combinação, é necessário que os nós transmitam passos suficientes para que a a probabilidade de detecção de identidades Sybil seja tão grande quanto necessário. Por razões de espaço, é impossível reproduzir aqui os cálculos que permitem mostrar o valor de *trr.lenght($\mathcal{C}$)* em função da probabilidade de detecção alvo. Estes resultados, assim como uma descrição do escalonamento dos testes, podem ser encontrados em [11].

algorithm quorumValidation ($\mathcal{C}$) **is**
excluded $\longleftarrow \emptyset$;
$\mathcal{I} \longleftarrow$ trr.schedule($\mathcal{C}$);
for ($j = 0$ to trr.length ($\mathcal{C}$)) **do**
if $id_i \in \mathcal{I}[j]$ **then**
broadcast.send ($\mathcal{I}[j]$.indexOf(id_i), VALIDATE);
else
channel $\longleftarrow$ rand(1,K);
if broadcast.receive (channel) = `null` **then**
excluded $\longleftarrow \cup \{\mathcal{I}[j][channel]\}$;
return [sort($\mathcal{C} \setminus$ excluded)];

Figura 4. Procedimento de Validação do Quorum (executado em cada nó i).

4 Correcção do Algoritmo

Nesta secção apresenta-se um rascunho das provas da correcção do protocolo, de acordo com as propriedades defendidas anteriormente.

Lema 1: O nonce é gerado com a participação de um nó correcto, com uma probabilidade arbitrariamente próxima de 1.
Esboço da prova: Esta propriedade é garantida pela limitação dos nós bizantinos em efectuarem transmissões, e pelo valor n de contribuições necessárias para gerar o nonce. □

Lema 2: Assumindo a existência de um nonce correcto, o conjunto de candidatos resultante do teste de recurso computacional contêm no mínimo $q - f$ identidades pertencentes a nós correctos, com uma probabilidade arbitrariamente próxima de 1.
Esboço da prova: Uma vez que o algoritmo de selecção de candidatos termina quando forem atingidas as $\sigma = q+s$ identidades, e uma vez que com uma probabilidade arbitrariamente alta, os nós bizantinos apenas conseguem propor $f + s$ identidades, então as $(q - f)$ identidades restantes pertencem necessariamente a nós correctos. □

Lema 3: Nenhum identificador pertencente a um nó correcto é eliminado pelo teste de recurso de rádio por outro nó correcto.
Esboço da prova: Assuma-se o contrário, e que o identificador de um nó correcto é eliminado por um nó correcto devido ao teste de recurso de rádio. Nesse caso, o nó que propôs essa identidade ou omite um passo de comunicação devido a uma falha, ou usa esse passo para defender uma outra identidade. Em qualquer dos casos esse nó não é correcto - uma contradição. □

Lema 4: O teste de recurso de rádio, com uma probabilidade arbitrariamente próxima de 1, limita nos quorums de todos os nós correctos o número de identidades propostas por nós bizantinos a um máximo de f, para valores de f inferiores a K.
Esboço da prova: Esta é uma propriedade do teste de recurso de rádio que deriva directamente das limitações dos rádios e do desenho deste teste. Um conjunto de nós f apenas pode transmitir simultaneamente em f canais de rádio distintos, indicando que simultaneamente apenas podem defender f identidades. A melhor estratégia

para um nó nestas condições é defender sempre a mesma identidade, o que resulta na presença de, no máximo, f identidades pertencentes a nós bizantinos nos quorum. □

Como descrito na Secção 2, pretendemos que o nosso protocolo possua as propriedades de "Quorum Parcialmente Consistente e sem Sybils" e "Terminação Probabilista". De forma a provar a propriedade "Quorum Parcialmente Consistente e sem Sybils", provamos primeiro os seguintes Lemmas:

Lemma 5 (Sem Sybils): Com uma probabilidade arbitrariamente perto de 1, não há identidades sybil no QNS_i de uma identidade correcta n_i. Mais precisamente, seja:
$U_i = \bigcup_{id \in QNS_i} usa(id)$. Temos que $|U_i| \geq |QNS_i|$.
Esboço da prova: Esta propriedade deriva directamente do Lema 4. Assuma-se o contrário, e que existe uma identidade sybil no QNS_i de um nó correcto n_i. Uma vez que na última fase do QNS todas as identidades participam em testes de recurso rádio, a existência de uma identidade sybil implicaria que o TRR não eliminou todas as identidades sybil do conjunto de candidatos. No entanto, e de acordo com o Lemma 4, o resultado do TRR não contêm identidades sybil, com uma probabilidade arbitrariamente perto de 1, o que contradiz a hipótese. □

Lemma 6 (Do no harm): Seja Q_i o resultado do TRR de um nó correcto n_i. Caso haja uma identidade correcta A em Q_i, então, a identidade A também está presente em todos os resultados da fase de validação de todos os outros nós correctos. Mais precisamente, se $A \in Q_i$ para o nó n_i, então $A \in Q_i \; \forall_{n_i}$.
Esboço da prova: Este Lemma deriva directamente do Lemma 3, e do facto de a fase de selecção de candidatos retornar o mesmo conjunto de identidade $\mathcal{C}$ em todos os nós. Assuma-se o contrário, e que existe uma identidade A de um nó correcto que está presente no resultado da fase de validação do quorum Q_i de um nó correcto n_i, mas não no resultado Q_j de um outro nó correcto n_j. No entanto, e tendo em conta que o conjunto de entrada da fase de validação é o mesmo em todos os nós ($\mathcal{C}$), e que o TRR não elimina identidades de nós correctos (Lemma 3), se $A \in Q_i$ então $A \in Q_j$, o que contradiz a hipótese. □

Com estes dois Lemmas conseguimos agora provar a propriedade "Quorum Parcialmente Consistente e sem Sybils" do QNS.

Teorema 1 (Quorum Parcialmente Consistente e sem Sybils) Existe um conjunto de identidades de nós correctos (QNS), comum a todos os QNS_i, tal que $q \geq |QNS| \geq q - f$, com uma probabilidade arbitrariamente perto de 1, tal como definido na Secção 2.2.
Esboço da prova: Este Teorema deriva dos Lemmas 2, 3, 5 e 6. De acordo com o Lemma 2, o resultado da fase de selecção de candidatos ($\mathcal{C}$), contem pelo menos $q - f$ identidades correctas. Considere-se ainda que: i) o conjunto $\mathcal{C}$ é o mesmo para cada nó; ii) o TRR não elimina identidades de nós correctos (Lemma 3), e todas as identidades correctas estão presentes no resultado de cada nó correcto (Lemma 6); iii) existem no máximo f identidades de nós bizantinos em cada QNS_i de cada nó correcto (Lemma 5). Assim, e se ordenarmos por ordem alfabética o resultado do algoritmo QNS, e limitarmos o conjunto às primeiras q identidades, temos a garantia que: o conjunto resultante é composto por, no máximo, f identidades de nós bizantinos e, no mínimo, pelas primeiras $q - f$ identidades de nós correctos, do conjunto ordenado do NSQ_i. □

Teorema 2 (Terminação Probabilista): Todos os nós correctos retornam o quorum com uma probabilidade arbitrariamente próxima de 1, num número finito de passos.
Esboço da prova: Para cada uma das fases do protocolo, existe um número finito de passos que assegura as suas propriedades com elevada probabilidade (como descrito na Secção 3). O número total de passos necessário para terminar o protocolo, é a soma dos passos consumidos em cada fase. □

5 Conclusões e Trabalho Futuro

Neste artigo foi proposto um algoritmo para construir um quorum parcialmente consistente composto apenas por identidades não-sybil numa rede sem fios de vizinhança local. O algoritmo é baseado no uso de testes de recursos, não só para detectar (e excluir) identidades sybil, mas também para optimizar o processo de construção do quorum. Foram também enunciadas e verificadas algumas propriedades que o algoritmo possui. Como trabalho futuro, fica a expansão da solução para redes com vários saltos.

Referências

[1] J. B. A. Juels. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *NDSS '09: Proceedings of NDSS '99 (Networks and Distributed Security Systems)*, pages 151–165, 1999.

[2] J. Aspnes, C. Jackson, and A. Krishnamurthy. Exposing computationally-challenged Byzantine impostors. Technical Report YALEU/DCS/TR-1332, Yale University Department of Computer Science, July 2005.

[3] A. Back. Hashcash - a denial of service counter-measure. Technical report, 2002.

[4] J. R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[5] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO '92: Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 139–147, London, UK, 1993. Springer-Verlag.

[6] D. E. Eastlake and P. E. Jones. Us secure hash algorithm 1 (sha1). `http://www.ietf.org/rfc/rfc3174.txt?number=3174`.

[7] C. L. Fullmer and J. Garcia-Luna-Aceves. Fama-pj: A channel access protocol for wireless lans. In *Proc. ACM Mobile Computing and Networking '95*, pages 76–85. ACM, 1995.

[8] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4:382–401, 1982.

[9] D. Malkhi and M. Reiter. Byzantine quorum system. *Distributed Computing*, 11:569–578, 1998.

[10] D. Malkhi, M. Reiter, and A. Wool. The load and availability of byzantine quorum systems. In *SIAM Journal of Computing*, pages 249–257, 1997.

[11] D. Mónica, J. Leitão, L. Rodrigues, and C. Ribeiro. On the use of radio resource tests in wireless ad hoc networks. In *Proceedings of the 3rd Workshop on Recent Advances on Intrusion-Tolerant Systems (WRAITS)*, pages F21–F26, Estoril, Portugal, June 2009.

[12] D. Mónica, J. Leitão, L. Rodrigues, and C. Ribeiro. Observable non-sybil quorum construction in one-hop wireless ad hoc networks. Technical Report 34, INESC-ID, June 2009.

[13] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 259–268, 2004.

[14] A. A. Pirzada and C. McDonald. Establishing trust in pure ad-hoc networks. In *ACSC '04: Proceedings of the 27th Australasian conference on Computer science*, pages 47–54, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[15] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13:361–396, 2000.

[16] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao. Sybillimit: A near-optimal social network defense against sybil attacks. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 3–17, May 2008.

[17] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. Sybilguard: defending against sybil attacks via social networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):267–278, 2006.