Resolução do Consenso em Redes Móveis Ad-Hoc a Partir de um Conjunto Dominante Conexo

Daniel Cason[1], **Fabíola Gonçalves Pereira Greve**[1]

[1]Departamento de Ciência da Computação – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brazil

Resumo. *O presente trabalho apresenta uma nova abordagem para a solução do consenso em redes móveis auto-organizáveis, baseada na construção de um conjunto dominante conexo de nós sobre a topologia da rede. Emprega-se abstrações para a detecção e o monitoramento de participantes a fim de manter as propriedades deste conjunto dinâmico na presença de falhas e mobilidade de processos. O resultado é uma estruturação lógica da rede baseada em seu estado real, que permite a realização de cálculos distribuídos consistentes, como a solução do consenso tolerante a falhas entre participantes desconhecidos.*

1. Introdução

O consenso é um problema fundamental no desenvolvimento de sistemas confiáveis, pois assegura que um conjunto de entidades concordem com uma determinada ação ao longo da execução distribuída, garantindo o progresso do sistema e a consistência do seu estado. Diversos trabalhos foram propostos para a resolução do consenso, direcionados aos mais variados modelos de comunicação e topologias de rede. Em sua grande totalidade, eles são voltados para contextos clássicos de redes estáticas, de topologias conhecidas. Mais recentemente, algumas propostas têm surgido para a sua resolução num contexto de redes dinâmicas, de topologias desconhecidas, onde destacam-se as redes móveis auto organizáveis (ou MANET) [Cavin et al. 2005, Greve and Tixeuil 2007].

As MANET são formadas por dispositivos (ou nós) que se deslocam pelo espaço e se comunicam através de canais sem fio. Cada nó se comunica com outros dispositivos situados dentro do raio de alcance de seu comunicador, chamados de vizinhos. Por serem alimentados por baterias, considera-se que a potência dos transmissores dos dispositivos é bastante limitada. Isto faz com que os nós da rede não se encontrem dentro de uma mesma área de alcance mútuo, caracterizando um ambiente de comunicação *multi-hop*. A ausência de uma infra-estrutura prévia faz com que dois dispositivos não vizinhos só possam se comunicar contando com a cooperação de nós intermediários, caracterizando um ambiente auto organizável. Nestas redes ocorrem entradas e saídas frequentes, além de falhas, e não há como prever o tempo em que as ações serão realizadas pelos processos e canais. Ou seja, o sistema como um todo é assíncrono.

No modelo assíncrono sujeito a falhas, o consenso não tem solução determinística [Fischer et al. 1985]. Para contornar tal impossibilidade, foram propostas algumas abstrações; dentre as mais importantes destacam-se os *detectores de falhas não confiáveis* [Chandra and Toueg 1996]. Estes são oráculos distribuídos que fornecem dicas aos processos sobre quais deles estão falhos. Os detectores estabelecem as condições mínimas de sincronia para a resolução do consenso. Grande parte dos trabalhos existentes para o consenso fundamentam-se nessa abstração [Greve 2005].

Em trabalhos mais recentes, alguns autores mostram que os detectores de falhas não são suficientes para resolver o consenso em redes dinâmicas [Cavin et al. 2005, Greve and Tixeuil 2007]. De fato, dado que em tais redes a cardinalidade e o conjunto global de nós é desconhecido, será necessário agregar ao sistema algum conhecimento sobre os seus nós. Esse conhecimento poderá ser fornecido pelos *detectores de participação* [Cavin et al. 2004]. Estes também são oráculos distribuídos que fornecem dicas aos processos sobre quais deles fazem parte da rede ou da computação distribuída.

O trabalho apresentado por Greve e Tixeuil [Greve and Tixeuil 2007] estabelece as condições teóricas mínimas de conhecimento (representadas pelo detector de participação da classe $K\text{-}OSR$) para que haja convergência do consenso, considerando-se o mínimo de sincronia possível (representado pelo detector de falhas $\diamond S$). Assim, para que haja consenso, a rede de conhecimento formada pelos participantes deve ser conexa e conter um único poço, i.e., uma componente poço formada por processos que se reconhecem mutuamente e que são conhecidos por todos os demais nós do sistema.

O presente trabalho apresenta uma nova abordagem para a solução do consenso em redes dinâmicas. Diferentemente da proposta de Greve e Tixeuil que, seguindo um estratégia *top-down*, impõe (ou determina) as propriedades da rede que permitem a resolução do consenso, apresentamos uma abordagem prática, do tipo *bottom-up*, que, a partir do estado real da rede determina uma rede lógica (*overlay*) onde o consenso poderá convergir de maneira segura.

A nossa abordagem fundamenta-se na construção e manutenção de um *conjunto dominante conexo* (ou CDS) de nós na rede [Dai and Wu 2003]. Este conjunto é obtido de maneira distribuída, através de trocas de mensagens entre vizinhos e a partir de abstrações conhecidas para a detecção de falhas e para a detecção de participantes. A rede virtual induzida, por sua vez, irá cumprir as condições teóricas mínimas de conectividade para a resolução do consenso definidas por [Greve and Tixeuil 2007]. Desta forma, utilizando-se de estratégias locais de auto organização em cima de um estado real, faz-se possível a obtenção de valores globais para a rede dinâmica, possibilitando a realização de uma computação distribuída confiável.

O artigo se organiza da seguinte maneira. A seção 2 apresenta o modelo do sistema e conceitos fundamentais. A seção 3 apresenta a estrutura da nova abordagem proposta. A seção 4 apresenta as bases de um protocolo de construção e manutencão de um CDS tolerante a falhas. A seção 5 traz um protocolo de consenso a partir do CDS e finalmente a seção 6 conclui o trabalho.

2. Modelo de Sistema

A MANET é um grafo $G(\Pi, E)$ cujos vértices representam o conjunto de dispositivos (ou nós) $\Pi = \{v_1, v_2, \ldots v_n\}$. Dois nós v_i e v_j formam uma aresta $(v_i, v_j) \in E$ se forem capazes de se comunicar diretamente. Se v_i e v_j não são vizinhos, sua comunicação é garantida através de um caminho $v_i \rightsquigarrow v_j$ no grafo de comunicação.

Os canais locais são confiáveis, ou seja, entre vizinhos a troca de mensagens ocorre sem perdas (duplicação ou corrupção). Supõe-se que estas propriedades são garantidas pelas camadas inferiores de rede dos dispositivos, tal como em [Si and Li 2004]. O sistema é assíncrono para canais e processos, logo, não existem limites de tempo definidos

nem para a entrega das mensagens, nem para as ações realizadas pelos processos. Apenas para facilitar a apresentação dos conceitos, define-se um relógio global $T \subseteq \mathbb{N}$. Entretanto, os processos não têm acesso a T.

A rede é dinâmica, havendo a entrada e saída de participantes, que podem ser corretos ou falhos. Processos falham por parada, deixando de participar da rede de forma definitiva. A ocorrência de falhas e a mobilidade não desconectam a rede ou, se o fizerem, a rede se reconecta em tempo finito, possivelmente curto. Assim, supõem-se que o grafo de comunicação da rede é conexo, possibilitando a comunicação entre quaisquer pares de nós corretos.

2.1. O Problema do Consenso

O consenso é definido como se segue: dado um conjunto de processos $\Pi = \{v_1, v_2, \ldots v_n\}$, cada processo v_i propõe um valor x_i invocando a primitiva $propose(x_i)$. Após um número finito de interações, todos os processos decidem por um valor proposto utilizando a primitiva $decide(x)$, $x \in \{x_1, x_2, ..., x_n\}$. Formalmente e supondo a existência de falhas, o *consenso clássico uniforme* é definido através de suas propriedades [Chandra and Toueg 1996]:

Terminação : Todo processo correto decide, após algum tempo, por um valor;
Validade : Se um processo decide um valor x, então x foi proposto por algum processo;
Acordo : Dois processos (corretos ou não) não decidem diferentemente.

Nas MANET, tal como nas redes dinâmicas e auto-organizáveis, o conjunto de processos participantes não é previamente conhecido e não é fixo. Para tais ambientes definiu-se uma nova variante do consenso, o FT-CUP (de *Fault Tolerant Consensus with Unknown Participants*)[Cavin et al. 2004, Greve and Tixeuil 2007]. O FT-CUP tem especificação idêntica à do consenso clássico, com a suposição extra de que nem a quantidade (n) nem o conjunto dos processos (Π) é inicialmente conhecido.

2.2. Detecção de Falhas

O consenso não possui solução determinística no modelo assíncrono, na presença de (mesmo que somente) uma falha de processo [Fischer et al. 1985]. Isto se deve à impossibilidade de se distinguir entre um processo lento e um processo falho, o que pode levar a um bloqueio do protocolo ou à violação da propriedade de acordo.

No intuito de possibilitar o emprego de algoritmos assíncronos em quaisquer modelos de sincronia, [Chandra and Toueg 1996] propuseram a abstração de detectores de falhas não-confiáveis. Eles são oráculos distribuídos, associados a cada processo, responsáveis por monitorar um grupo de processos e arbitrar sobre seu estado de maneira não confiável. Assim, podem suspeitar da falha de um processo correto (falso positivo), ou inversamente, determinar que um processo falho é correto (falso negativo).

Os detectores de falhas foram sugeridos para o modelo clássico de computação, onde o conjunto de nós participantes é conhecido e é possível se comunicar diretamente com todos eles, ou seja, considera-se que grafo de comunicação é completo. Este não é o caso das MANET. Num contexto de redes dinâmicas, detectores mais adequados à comunicação *multi-hop* vêm sendo propostos [Hutle and Widder 2004, Sridhar 2006]. Dentre estes destacam-se os detectores de falhas locais. O detector de falha local da classe $\Diamond P_\ell$ tem as seguintes propriedades [Hutle and Widder 2004]:

Completude Local Forte : existe um instante de tempo, após o qual todo processo p falho é permanentemente suspeito por qualquer processo vizinho q;

Precisão Local Finalmente Forte : existe um instante de tempo, após o qual processos corretos não serão supeitos por nenhum processo em suas vizinhanças.

2.3. Detecção de Participantes

As MANET são ambientes de rede caracterizados pela ausência de infra estrutura prévia de comunicação. Assim, supõe-se que os dispositivos não possuem uma pré-programação e nem ao menos uma entidade central à qual poderiam recorrer para obter informações sobre a rede. Isto faz com que os nós tenham que descobrir uns aos outros e elaborar uma estrutura de rede que permita a comunicação.

No intuito de abstrair os processos das características da rede e do meio subjacente e assim, possibilitar a construção de soluções genéricas, foi proposto o conceito de detectores de participantes [Cavin et al. 2004]. Estes são oráculos distribuídos associados a cada processo, responsáveis por retornar um conjunto de processos "conhecidos" na rede (ou seja, detectados). Seja $v.PD$ o conjunto de nós fornecidos ao processo v pelo detector de participantes. Tem-se as seguintes propriedades [Cavin et al. 2004]:

Monotonicidade: a lista de processos retornada pelo detector é monotônica crescente, e formada por: $\forall v \in \Pi, \forall t', t \in T, t' \geq t : v.PD(t) \subseteq v.PD(t')$;

Corretude: o detector de participantes não comete erros, retornando somente processos que pertecem à rede: $\forall v \in \Pi, \forall t \in T : v.PD(t) \subseteq \Pi$.

2.4. Conjuntos Dominantes Conexos

Definição 2.1. *Um conjunto $V' \subset V$ é um **conjunto dominante (DS)** de um grafo $G(V, E)$ se todo nó $v \in V - V'$ é dominado por algum nó de V', ou seja:*

$$\forall v \in V - V' \Rightarrow \exists u \in V' : (u, v) \in E.$$

Definição 2.2. *Um conjunto $V' \subset V$ é um **conjunto dominante conexo (CDS)** de um grafo $G(V, E)$ se V' é um conjunto dominante de G e o subgrafo de G induzido por V' é conexo.*

Para qualquer grafo, existe (ao menos) um CDS mínimo (ou MCDS) que é minimal, ou seja, tem cardinalidade mínima. A obtenção de um MCDS é um caso especial do problema de cobertura de vértices (*set-cover problem*), que é NP-completo [Karp 1972]. Existem, porém, algoritmos que obtêm soluções que aproximam o mínimo, segundo uma taxa de aproximação β.

O MCDS tem sido empregado em MANET com objetivo de se obter um conjunto mínimo e conexo de nós dominantes para a rede. Um dos primeiros algoritmos com tal intuito foi proposto por [Das and Bharghavan 1997], que empregava o algoritmo de [Guha 1996] para realizar a escolha dos nós que farão parte do CDS, com β bastante reduzido. Outra linha de abordagem é proposta por [Wu and Li 1999], tendo como prerrogativa principal a realização desta escolha de forma distribuída e simples, com base apenas no conhecimento local do nó e em um número constante de passos de comunicação com vizinhos. Trabalhos posteriores nesta linha estabeleceram a relação de dominância entre nós [Wu 2002] e regras mais eficientes de marcação de nós [Dai and Wu 2003, Wu and Dai 2004].

3. Resolução do Consenso a Partir de um Conjunto Dominante Conexo

As soluções para o consenso clássico prescindem do conhecimento dos participantes na rede. Assim, o modelo clássico considera um conhecimento à priori tanto de Π quanto de n. Adota-se um modelo de computação caracterizado tanto por um grafo de comunicação, quanto por um grafo de conhecimento completo. O consenso para redes dinâmicas deve considerar a hipótese de participantes desconhecidos. Tal característica reflete no grafo de comunicação e de conhecimento, que deixam de ser completos.

O trabalho de Greve e Tixeuil [Greve and Tixeuil 2007] mostra que, para que se possa atingir o consenso numa rede dinâmica, existe uma solução de compromisso entre as condições de sincronia, representadas pelos detectores de falhas [Chandra and Toueg 1996], e as relações de conhecimento no sistema, representadas pelos detectores de participação [Cavin et al. 2004]. Para se resolver o consenso com o detector de falhas $\diamond S$ será preciso enriquecer o sistema com um detector de participantes do tipo *K-OSR* (*k-One Sink Redutibility*). Nesse caso, espera-se que o grafo de conhecimento formado pelo detector *K-OSR* tenha somente uma única componente poço na rede. Na prática, isso significa que existe um conjunto de nós da rede que é conhecido por todos os demais. Além disso, para todo processo participante, existe pelo menos um caminho de nós corretos no grafo de conhecimento que o conecta a cada um dos nós do poço. Durante a execução do protocolo, um desses caminhos continuará existindo, mesmo na ocorrência de falhas de processo ou em caso de mobilidade.

O protocolo de consenso FT-CUP de Greve e Tixeuil não determina como os detectores de participação da classe *K-OSR* seriam implementados num sistema real. De fato, implementar um detector de participação de uma determinada classe é um problema em aberto. Ocorre que se a topologia da rede não é a esperada, ou seja, o grafo formado pela rede não satisfaz os requisitos da classe *K-OSR*, a corretude do consenso estará comprometida. Na prática, se a rede contém mais de uma componente poço, será possível decidir mais de um valor e a propriedade de acordo do consenso estaria comprometida [Greve and Tixeuil 2007].

O presente trabalho apresenta uma abordagem prática para a resolução do consenso em redes dinâmicas. Diferentemente da proposta de Greve e Tixeuil, que identifica condições teóricas e propõe um protocolo de consenso no estilo *top-down*, em que a topologia da rede é, de certa maneira, imposta para satisfazer as condições exigidas, apresentamos uma abordagem prática, do tipo *bottom-up*, que, a partir do estado real do sistema determina uma rede lógica onde o consenso poderá convergir de maneira segura.

A abordagem adotada irá identificar esta rede lógica a partir do estabelecimento de um *conjunto dominante conexo* na rede de comunicação. O CDS já vem sendo empregado como solução para proporcionar roteamento e *broadcast* eficientes e tolerantes a falhas em MANET [Das and Bharghavan 1997, Wu and Li 1999, Dai and Wu 2003, Wu and Dai 2004, Dai and Wu 2005]. Neste trabalho, advogamos o seu uso para a resolução do problema do consenso. Na rede lógica estabelecida, os nós que fazem parte do CDS conhecem-se mutuamente e estão conectados. Além disso, todos os demais nós são vizinhos de ao menos um nó pertencente ao CDS. Tais propriedades garantem a convergência do consenso de maneira segura, pois coincidem com as condições teóricas de conectividade estabelecidas por [Greve and Tixeuil 2007] para a sua resolução.

Na solução de consenso proposta, deve-se, inicialmente determinar um CDS entre os nós do sistema. Note-se que este conjunto é calculado a partir do grafo de comunicação da rede. Os nós marcados pertencem ao CDS e terão papel ativo no consenso, já os nós fora do CDS não estão marcados e terão papel passivo. Assim, o consenso propriamente dito será realizado apenas pelos nós ativos, que decidem por um valor único e o repassam para os demais. Os nós passivos não participam do consenso e apenas aguardam a decisão proveniente de um nó vizinho ativo.

É importante notar que, na solução proposta por [Greve and Tixeuil 2007], o conjunto de nós que resolvem o consenso propriamente dito fazem parte da componente poço. Além disso, supõem-se que dois nós do grafo de conhecimento podem se comunicar de maneira confiável, visto a existência de uma infra-estrutura subjacente de roteamento confiável. No nosso trabalho, o conjunto de nós que irão executar o consenso é calculado no grafo de comunicação da rede, onde dispositivos são vizinhos enquanto estiverem no alcance mútuo de seus comunicadores. Por conta disto, enquanto o trabalho de [Greve and Tixeuil 2007] supõe uma conectividade lógica suficiente, o presente trabalho precisa construir e manter uma estruturação (rede *overlay*) dentro da rede física que atenda às condições teóricas definidas. Assim, as mudanças na topologia da rede e as falhas devem ser tratadas a fim de possibilitar a comunicação e o conhecimento mútuo dentre os processos ativos. Para tanto, lança-se mão de abstrações de *detecção local de falhas* e *detecção de participantes* para construir e manter o CDS, que serve como base para a obtenção do conjunto de processos ativos e a consequente solução do consenso.

O protocolo de consenso FT-CUP aqui proposto tem a seguinte estrutura. Inicialmente, faz-se uso do protocolo FT-CDS (*Fault-Tolerant Connecting Dominating Set Protocol*), que será responsável pela construção e manutenção do CDS na MANET. Este protocolo ficará executando de maneira contínua, pois a topologia da rede de comunicação modifica-se, devido a entradas, saídas, falhas e movimentação na rede. Para a realizacão do consenso, será executado inicialmente o protocolo Collect, responsável por determinar os processos ativos da rede, a partir do CDS determinado. Finalmente, dado que o conjunto de nós ativos foi estabelecido, reduz-se o problema ao consenso clássico com participantes conhecidos, e pode-se executar qualquer Consenso_Uniforme anteriormente proposto na literatura [Greve 2005]. No resto do trabalho, tais protocolos serão detalhados.

4. FT-CDS: Um Protocolo de CDS Tolerante a Falhas numa MANET

Numa MANET, o cálculo de um conjunto dominante conexo mínimo foi inicialmente proposto por [Das and Bharghavan 1997]. Neste trabalho, os nós pertencentes ao MCDS formam um *backbone virtual* dinâmico da rede, sendo responsáveis por manter cópias locais da sua topologia. Com esta informação, estes nós são capazes de calcular e manter as tabelas de roteamento, permitindo otimizar e prover tolerância a falhas ao roteamento de mensagens a um ou mais destinatários dentro de uma MANET. Trabalhos posteriores empregam e ampliam este conceito, focando-se tanto na construção do CDS como em suas aplicações. Dentre os primeiros, destaca-se a abordagem de [Wu and Li 1999], cuja prerrogativa principal é a realização do cálculo de forma simples e distribuída, com base apenas no conhecimento local do nó e num número constante de passos de comunicação com vizinhos. Esta proposta inicial foi aperfeiçoada, com a sua aplicação

para grafos direcionados [Wu 2002] e com a determinação de regras genéricas de marcação [Dai and Wu 2003, Wu and Dai 2004].

Apesar de interessantes, visto a sua eficiência e extensibilidade, as abordagem distribuídas de cálculo de CDS não podem ser implementadas diretamente no modelo de rede adotado nesse artigo. Nestes trabalhos [Wu 2002, Dai and Wu 2003, Wu and Dai 2004] considera-se que um nó conhece todos os seus vizinhos, e além disso, é capaz de detectar imediatamente o seu afastamento ou falha. Ao mesmo tempo, nenhum mecanismo efetivo de tolerância a falhas é delineado, havendo somente a análise do efeito da remoção de vértices e arestas do grafo G de comunicação na composição do CDS.

A fim de construir um CDS tolerante a falhas dentro do modelo de rede aqui estabelecido, iremos fazer uso de abstrações já conhecidas, a saber detecção de falhas e detecção de participantes. Inicialmente é sugerida uma implementação de um oráculo de detecção de dispositivos participantes, que identifica um grafo de conhecimento $\mathcal{G}$ estabelecido a partir das relações de vizinhança física entre os nós. Em seguida, adota-se um algoritmo, baseado em [Dai and Wu 2003], para a construção de um CDS para o grafo $\mathcal{G}$. Finalmente, são feitas adequações ao CDS com o intuito de realizar a sua manutenção a partir das informações de falhas de vizinhos provenientes do detector de falhas e de informações de conhecimento de novos vizinhos, provenientes do detector de participantes.

4.1. Detecção de Vizinhos

Assim como nas redes celulares, considera-se que a camada física de rede emita mensagens de controle periódicas e que ela seja capaz de monitorar o meio físico, identificando comunicações entre nós e mensagens de controle. A fim de sistematizar este processo, realiza-se o seguinte procedimento. Um nó participante v difunde mensagens de descoberta $discovery(v.id)$, contendo o seu identificador único $v.id$. Um nó u, ao receber tal mensagem, responderá ao seu remetente com um $ack(u.id)$, enviando também o seu identificador $u.id$. Estabelece-se, então, uma relação de dominância em que o nó v domina o nó u se este último responde ao $discovery()$ do primeiro.

O algoritmo 1 apresenta a implementação deste detector de participantes (vizinhos), que atua como um oráculo de execução constante e que satisfaz as propriedades definidas na seção 2.3. O nó v mantém duas listas de vizinhos $v.PD.Abs$ e $v.PD.Dom$. Os vizinhos absorventes $v.PD.Abs$ são os nós que responderam mensagens de $discovery()$, sendo formada de nós dominados por v (linha 11). Já os vizinhos dominantes $v.PD.Dom$ é o conjunto de nós dos quais se recebeu uma mensagem de $discovery()$, mensagens estas que foram respondidas (linha 8). A detecção de um novo nó leva ao envio de uma mensagem de $discovery()$ (linha 5) que, se confirmada por um ack faz com que o nó seja adicionando ao conjunto absorvente de nós.

Pela construção do algoritmo 1 é possível afirmar, trivialmente, que um nó dominado por v sabe que v é seu dominante. Assim: $v \in u.PD.Abs \Rightarrow u \in v.PD.Dom$.

Ao mesmo tempo, dado que canais são confiáveis, os ack's chegam em algum momento ao seu destino, o que determina que, supondo u, v corretos, tem-se: $v \in u.PD.Abs \Leftrightarrow \Diamond u \in v.PD.Dom$. O símbolo $\Diamond$ estabelece que a propriedade é válida a partir de algum instante de tempo, finito porém desconhecido.

Algorithm 1 Detector de Vizinhos para o processo v

Outputs:

1: $v.PD.Abs$: conjunto de processos {conjunto absorvente}
2: $v.PD.Dom$: conjunto de processos {conjunto dominante}
3: $v.PD.set$: conjunto de processos {nós detectados}

4: **upon detection of** node u
5: SEND $discovery(v.id)$

6: **upon reception of** $discovery(u.id)$ from u
7: SEND $ack(v.id)$ to u
8: add u to $v.PD.Dom$
9: add u to $v.PD.set$

10: **upon reception of** $ack(u.id)$ from u
11: add u to $v.PD.Abs$
12: add u to $v.PD.set$

Tolerância a Falhas. O detector de participantes retorna para v um conjunto $v.PD.set$ de processos vizinhos, com os quais ele pode se comunicar, ou seja, um subconjunto do conjunto total Π. Visto a mobilidade da rede e a presença de falhas de processo, é necessário monitorar os nós detectados para que a lista de processos retornados seja coerente com o grafo de comunicação a todo instante da computação. Assim, será necessário suprimir da lista obtida nós falhos ou nós não mais diretamente comunicáveis (por conta da mobilidade).

Para tal atualização, é empregado um detector de falhas da classe $\diamond P_\ell$ para monitorar os dispositivos do conjunto de processos $v.PD.set$. O detector retorna, sempre que consultado, um conjunto $Suspect(v) \subseteq v.PD.set$ de processos suspeitos de falha. É interessante observar que, visto suas propriedades, um detector da classe $\diamond P_\ell$ que monitore os nós de $v.PD.set$ pode retornar um conjunto $Suspect(v) = v.PD.set$, ou seja, suspeitar de todos os nós detectados. Supondo, porém, que haja algum vizinho correto, pela propriedade de *precisão forte*, ele não será suspeitado após um intervalo de tempo (finito mas desconhecido), permitindo a terminação dos algoritmos que empreguem o detector de falhas.

Define-se, então, para um instante de tempo qualquer, o conjunto de vizinhos efetivos de um nó v:

$$v.Neigh = v.PD.set \backslash Suspect(v) \tag{4.1}$$

$v.Neigh$ é dito efetivo dado que corresponde a um subconjunto de dispositivos, no alcance de transmissão de v, com os quais há uma troca efetiva, constante e funcional de mensagens por parte das camadas inferiores, o que os elege como processos participantes vizinhos, com os quais é possível realizar algum processamento.

Para que um nó faça parte da rede, é desejável que ele possa ser detectado por uma quantidade razoável de dispositivos ao seu redor. Assim, adota-se a constante k do

sistema para definir tal quantidade. Evidentemente, k será maior ou igual ao grau mínimo da rede e seu valor depende da aplicação subjacente. Define-se, então, o conjunto de nós participantes π como:

$$\pi = \{u \mid (u \in \Pi) \wedge (\mid u.PD.Dom \mid \geq k)\} \tag{4.2}$$

Assim, enquanto um nó não recebe k mensagens de *discovery*() ele não participa de nenhum processamento e só troca mensagens de controle. Tal estratégia assemelha-se a de um dispositivo que ingressa em uma rede celular, e ainda não foi incorporado por uma estação base.

A relação de dominância estabelecida entre os nós, baseada nas abstrações de detecção e monitoramento de nós vizinhos, permite a construção de um grafo de conhecimento $\mathcal{G}(\pi, \Re)$ da rede. Formalmente, $\Re$ é definida por:

$$\Re = \{(u, v) \mid (v \in u.PD.Abs) \wedge (v \notin Suspect(u))\} \tag{4.3}$$

Em particular, a relação $\Re$ é também de comunicação, visto a forma como é construída e mantida a lista de vizinhos $PD.Abs$. Assim, se $G(\Pi, E)$ é o grafo de comunicação da rede e $\mathcal{G}(\pi, \Re)$ é o grafo de conhecimento, é válido afirmar:

$$(u, v) \in \Re \Rightarrow (u, v) \in E \tag{4.4}$$

4.2. Determinação e Manutenção do CDS Tolerante a Falhas

Dentre os algoritmos propostos na literatura para o cálculo do CDS em MANET, destaca-se o de [Dai and Wu 2003]. Ele é formado por dois procedimentos. O primeiro procedimento realiza uma troca de mensagens entre nós vizinhos a fim de obter um conhecimento em dois níveis (2-*hop*) da rede. Emprega-se, então, a propriedade de *marcação básica*, definida por [Wu and Li 1999, Wu 2002], para determinar a marcação inicial do nó. Os nós marcados nesta primeira fase iniciam um segundo procedimento, no qual identifica-se os vizinhos marcados e aplica-se a *regra k*. Ela faz com que uma parte dos nós marcados inicialmente se desmarque, com intuito de eliminar redundâncias na cobertura de nós [Dai and Wu 2003]. O conjunto de nós que permanecem marcados após esta etapa forma um CDS da rede.

No nosso trabalho, calcula-se e mantém-se um CDS da rede representada pelo grafo de conhecimento $\mathcal{G}$. Para tal, emprega-se os procedimentos *marcação básica* e *regra k* (definidos por [Dai and Wu 2003]) sobre a relação $\Re$ de dominância. O conhecimento da vizinhança é obtido através do monitoramento dos nós detectados pelo detector de participantes (execução do algoritmo 1) e pelas suspeitas fornecidas pelo detector de falhas $\diamond P_\ell$ (representadas pelo conjunto $Suspect(v)$).

As mudanças na topologia da rede, devido à detecção de novos vizinhos ou perdas de vizinhança, em função de falhas ou mobilidade, levam à atualização do CDS. Visto a não confiabilidade dos oráculos empregados, preferiu-se adotar uma estratégia conservadora, que sacrifica a obtenção de um CDS com cardinalidade mínima para garantir um CDS em que as propriedades de conectividade e dominância entre os nós sejam mantidas.

Outra propriedade de interesse é a estabilidade do CDS, de forma que nós marcados não sejam desmarcados facilmente. Como a marcação emprega o conceito de prioridade de nós, é interessante que ela seja associada à participação do nó na computação

realizada. Neste trabalho a prioridade de um nó v é dada por $v.Prior = (v.state, v.id)$, onde $v.state$ representa a forma de participação do nó no protocolo de consenso, se ativa ou passiva. Define-se que $Prior(ativo, u.id) > Prior(passivo, v.id)$ para quaisquer nós u, v, garantindo que nós ativos não sejam removidos do CDS pelo emprego da *regra k*.

O algoritmo completo para a manutenção do CDS encontra-se em [Daniel Cason 2009]. Por questões de espaço, ele não poderá ser apresentado neste artigo. O algoritmo retorna, sempre que consultado, as seguintes informações:

- $v.marked \in \{True, False\}$, com a situação do nó v perante o CDS. Se $v.marked = True$ então $v \in$ CDS.
- $v.Dominants$. Contém o conjunto de nós vizinhos dominantes de v. Ou seja, nós vizinhos de v em que $marked = True$.

5. Protocolo de Consenso FT-CUP a partir do FT-CDS

Com a determinação e manutenção de um CDS na rede é possível estabelecer a seguinte propriedade:

Propriedade 5.1. *No início de uma execução do protocolo do consenso com participantes desconhecidos, o conjunto* P *de processos que vão resolver uma instância do consenso uniforme é formado pelos nós pertencentes ao CDS, ou seja, por todo v, tal que* $v.marked = True$.

O algoritmo 2 apresenta o *Consensus()* e é executado por todo nó que queira iniciar uma instância do protocolo. Na *Task 1* do algoritmo, o nó v estabelece seu estado no cálculo do consenso: nós do CDS são ativos e os demais, passivos (linhas 4 e 5). Envia-se, então, a todos os vizinhos conhecidos uma mensagem $start(v.marked, v.Dominants)$ (linha 8), contendo o estado do nó perante o CDS (marcado ou não) e uma lista de vizinhos conhecidos pertencentes ao CDS. Pelas propriedades de construção do CDS (definição 2.2), todo nó da rede pertence ao CDS ou é vizinho de algum nó que dele pertença. Ao mesmo tempo, o CDS é conexo, o que faz com que todo nó participante tenha ao menos um vizinho dominante. Ou seja, $v.Dominants$ não é vazio.

Ao receber uma mensagem de *start*, nós que ainda não estão participando do protocolo também iniciam a execução do *Consensus()*. Todos nós aguardam a recepção de mensagens *start* de seus vizinhos, menos daqueles suspeitos pelo detector de falhas (linha 9). Estas mensagens, recebidas através da linha 15, permitem ao nó determinar um conjunto inicial de nós vizinhos ativos (conjunto $nodes$, na linha 18), assim como os seus dominantes (conjuntos $known(u)$, da linha 17) que serão empregados no procedimento *Collect()* do algoritmo 3.

Munidos deste conhecimento inicial do conjunto de nós ativos, um nó v ativo passa à execução da rotina *Collect()*, descrita na seção seguinte. Esta é responsável por fazer com que o nó conheça os demais processos ativos, representados pelo conjunto P (linha 11), e com eles resolva uma instância do consenso uniforme (linha 12).

Já os nós não marcados aguardam a recepção do valor decidido, repassado a eles por seus nós dominantes (linha 14), a menos que, por mudanças topológicas na rede, eles sejam incorporados ao CDS. Neste caso (linhas 21 e 22), estes nós de estado passivo, passam a executar também a rotina *Collect()*, garantindo que, mesmo na presença de falhas, ela será executada por um conjunto conexo de nós corretos.

Algorithm 2 Consenso FT-CUP a partir de FT-CDS

Inputs:

1: $v.Neigh$: conjunto de nós {Vizinhos efetivos, ou seja, $v.PD.set \backslash Suspect(v)$}
2: $v.Dominants$: conjunto de nós {Vizinhos dominantes}
3: $v.marked$: booleano {Situação do nó perante o CDS}

Procedure Consensus():

**** Task 1 ****

4: **if** $v.marked = True$ **then**
5: $v.state \leftarrow ativo$
6: **else**
7: $v.state \leftarrow passivo$
8: SEND $start(v.marked, v.Dominants)$ to all $u \in N$
9: **wait until** *start* received from all $u \in N \backslash Suspect(v)$
10: **if** $v.state = ativo$ **then**
11: $P \leftarrow$ Collect()
12: **call** Uniform_Consensus (P)
13: **else**
14: **wait until** $decide(x)$ received from some $u \in v.Dominants$

**** Task 2 ****

15: **upon reception of** $start(u.marked, u.Dominants)$ from u
16: **if** $u.marked$ **then**
17: $known(u) \leftarrow u.Dominants$
18: add u to $nodes$
19: **if** *start* not already sent to u **then**
20: SEND $start(v.marked, v.Dominants)$ to u

21: **When** $v.marked = True$
22: **call** Collect()

5.1. Obtenção dos Nós Ativos

A determinação dos processos ativos da rede é obtida pelo algoritmo 3, através de *Collect()*. Ele realiza uma busca dentro do CDS por nós cujo estado seja ativo, através de rodadas iterativas de troca de conhecimento entre nós vizinhos. O conhecimento inicial dos nós ativos (conjuntos *nodes* e *known*) é obtido na inicialização do consenso (*Task 2* do algoritmo 2).

O princípio de seu funcionamento é que a incorporação de um processo u ativo permite a obtenção de sua lista de vizinhos dominantes $u.Dominants$. Esta lista foi enviada por u a todos os seus vizinhos no início do consenso (linha 8 do algoritmo 2), o que garante que, mesmo na falha deste processo, todos seus vizinhos possuem informações sobre sua vizinhança ativa. Como a falha de u é suspeitada por seus vizinhos, alguns deles serão incorporados ao CDS e poderão responder em nome do nó ativo, porém falho, u.

O procedimento mantém uma variável *nodes* que é o conjunto de nós ativos incor-

Algorithm 3 Obtenção do Conjunto de Participantes para o Consenso

Variables:

1: $nodes$: conjunto de nós {Vizinhos ativos de v}
2: $\forall u \in v.Dominants$: $known(u)$ {Nós conhecidos ainda não incorporados}

Procedure Collect():

**** Task 1 ****

3: **repeat**
4: SEND $request(known(u) \backslash nodes)$ to all $u \in v.Dominants$
5: **wait until** $response$ received from $u \in v.Dominants - Suspect(v)$
6: **for all** $asked(u)$ **do**
7: $Y \leftarrow \{(w, w.Dominants) / w \in asked(u) \cap nodes\}$
8: free $asked(u)$
9: SEND $response(Y)$ to u
10: **until** $\bigcup_{u \in nodes} known(u) \subseteq nodes$
11: **return** $nodes$

**** Task 2 ****

12: **upon reception of** $request(list)$ from u
13: **if** $list \subseteq nodes$ **then**
14: $Y \leftarrow \{(w, w.Dominants) / w \in list\}$
15: SEND $response(Y)$ to u
16: **else**
17: $asked(u) \leftarrow list$

18: **upon reception of** $response(list)$ from u
19: **for all** $(w, w.Dominants) \in list$ **do**
20: add w to $nodes$
21: add $w.Dominants$ to $known(u)$

porados. Para cada vizinho no CDS $u \in v.Dominants$ é mantida uma lista de nós ativos dele recebida $known(u)$. A cada rodada (*Task 1* do algoritmo 3) solicita-se a u informações sobre os nós que ele alega conhecer $known(u)$ a menos daqueles já incorporados por v, armazenados em $nodes$ (linha 4). A rodada é finalizada quando obtém-se respostas de todos os nós corretos (linha 5). O nó, então, envia a seus vizinhos os nós por eles solicitados $asked(u)$ incorporados naquela rodada.

Paralelamente, a *Task 2* lida com as mensagens recebidas. O recebimento de informações de algum nó vizinho u permite a incorporação de novos nós ativos (linha 20) e a atualização da lista de nós conhecidos por u, $known(u)$ (linha 21). Solicitações de informações por parte dos vizinhos são tratadas, através das listas $asked(u)$.

A terminação da *Task 1* em v ocorre quando nenhum conjunto $known(u)$ possuir nós ainda não incorporados e, assim, adicionados a $nodes$. Isto somente ocorre quando $nodes = P$ (definido na propriedade 5.1), pela construção iterativa do algoritmo. Já a *Task*

2 não se encerra imediatamente, visto que as informações que o nó *v* possui devem ser repassadas a seus vizinhos que a solicitem, possibilitando a terminação de sua busca.

Teorema 5.1. *Todo nó ativo correto obtém com a rotina* Collect() *um mesmo conjunto* P *de processos ativos participantes, corretos ou não, com os quais resolver o consenso.*

Demonstração. Seja um nó qualquer *v* correto e ativo que executa o *Collect()*. Entre *v* e qualquer outro nó *u* ativo existe, a qualquer momento do processamento, mesmo na ocorrência de falhas, um caminho dentro do CDS que os conecta. Seja $P(v,u) = v, w_1, w_2, \ldots, w_h, u$ um caminho mínimo entre *v* e *u*, no qual os nós w_i intermediários são corretos. Todos os nós w_i são marcados e, pelas linhas 10 ou 21 do algoritmo 2, executam também o *Collect()*. Na inicialização do consenso, cada nó w_i adiciona (caso existam) w_{i+1} ao seu conjunto *nodes*, assim como w_{i+2} a $know(w_{i+1})$, segundo as linhas 17 e 18 do algoritmo 2 (*Consensus()*). Em particular, $w_1 \in v.nodes$, $w_2 \in v.known(w_1)$ e $u \in w_h.nodes$. Além disso, $u.Dominants$ é conhecido por w_h.
Considere a *i*-ésima iteração da *Task 1* do algoritmo 3 (*Collect()*) em *v*. No seu início, $w_i \in nodes$ e $w_{i+1} \in known(w_i)$; na linha 2, solicita-se ao nó $w_1 \in v.Dominants$ informações sobre w_{i+1}. A resposta chega após algum tempo (linha 16) e o nó w_{i+1} será adicionado a *nodes* (linha 18). Assim, ao final da *h*-ésima iteração, *v* adiciona *u* à *nodes*, visto que $u \in known(w_h)$ no início da iteração. □

6. Conclusão

Este trabalho apresentou uma nova abordagem para a resolução do consenso em redes móveis auto-organizáveis. Diferentemente de propostas anteriores, ele segue uma abordagem prática, que, a partir do estado real determina uma estruturação lógica da rede. Esta estruturação estabelece um *conjunto dominante conexo* na rede de comunicação e possui propriedades que garantem a convergência do consenso de maneira segura, pois coincidem com as condições teóricas de conectividade estabelecidas anteriormente para a resolução do problema em redes desconhecidas [Greve and Tixeuil 2007]. São apresentados três protocolos. O FT-CDS permite o cálculo e a manutenção do CDS na rede, o *Collect()* determina um conjunto de processos *P* da rede aptos a resolverem uma instância do consenso e o *Consensus()*, responsável pelo consenso propriamente dito. Assim, através de mecanismos assíncronos e locais, abstrai-se as prerrogativas temporais e de conhecimento da rede, permitindo a auto organização dos processos e a consequente realização de computações distribuídas confiáveis no ambiente MANET.

Referências

Cavin, D., Sasson, Y., and Schiper, A. (2004). Consensus with Unknown Participants or Fundamental Self-Organization. *Lecture Notes in Computer Science*, pages 135–148.

Cavin, D., Sasson, Y., and Schiper, A. (2005). Reaching Agreement with Unknown Participants in Mobile Self-organized Networks in Spite of Process Crashes. Technical report, Research Report IC/2005/026, EPFL, 2005.

Chandra, T. and Toueg, S. (1996). Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the Association for Computing Machinery*, 43(2):225–267.

Dai, F. and Wu, J. (2003). Distributed Dominant Pruning in Ad Hoc Networks. In *IEEE International Conference on Communications, 2003. ICC'03*, volume 1, pages 353–357.

Dai, F. and Wu, J. (2005). On constructing k-connected k-dominating set in wireless networks. In *19th IEEE International Parallel and Distributed Processing Symposium, 2005. Proceedings*, page 81a.

Daniel Cason, F. G. P. G. (2009). Desenvolvimento de aplicações confiáveis em MANET a partir de um conjunto dominante conexo. Trabalho de conclusão de curso, Universidade Federal da Bahia. Disponível em `http://disciplinas.dcc.ufba.br/pub/MATA67/TrabalhosSemestre20091/Monografia_Daniel_Cason_20091.pdf`.

Das, B. and Bharghavan, V. (1997). Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets. In *1997 IEEE International Conference on Communications, 1997. ICC 97 Montreal,'Towards the Knowledge Millennium'*, volume 1, pages 376–380.

Fischer, M., Lynch, N., and Paterson, M. (1985). Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM (JACM)*, 32(2):374–382.

Greve, F. (2005). Protocolos fundamentais para o desenvolvimento de aplicações robustas. In *Minicursos SBRC 2005: Brazilian Symposium on Computer Networks*, pages 330–398, Fortaleza, CE, Brazil.

Greve, F. and Tixeuil, S. (2007). Knowledge connectivity vs. synchrony requirements for fault-tolerant agreement in unknown networks. In *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 82–91. IEEE Computer Society Washington, DC, USA.

Guha, S. (1996). Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387.

Hutle, M. and Widder, J. (2004). Time free self-stabilizing local failure detection. Research Report 33, TU Wien, Technische Universität Wien.

Karp, R. (1972). Reducibility among Combinatorial Problems. *Complexity of computer computations: proceedings*, page 85.

Si, W. and Li, C. (2004). Rmac: A reliable multicast mac protocol for wireless ad hoc networks. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing*, pages 494–501, Washington, DC, USA. IEEE Computer Society.

Sridhar, N. (2006). Decentralized local failure detection in dynamic distributed systems. In *The 25th IEEE Symposium on Reliable Distributed Systems*, pages 143–154.

Wu, J. (2002). Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):866–881.

Wu, J. and Dai, F. (2004). A generic distributed broadcast scheme in ad hoc wireless networks. *IEEE Transactions on Computers*, 53(10):1343–1354.

Wu, J. and Li, H. (1999). On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIALM '99: Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pages 7–14, New York, NY, USA. ACM.