Aspectos Práticos da Realização do Consenso FT-CUP em Redes Móveis Ad-Hoc *

Victor Franco Costa[1], **Fabíola Gonçalves Pereira Greve**[1]

[1]Programa de Pós-Graduação em Mecatrônica – Universidade Federal da Bahia (UFBA)
CEP 40.170-110 – Salvador – BA – Brasil

{vfcosta,fabiola}@dcc.ufba.br

Resumo. *O consenso é um problema fundamental no desenvolvimento de sistemas distribuídos confiáveis. As redes móveis ad-hoc (Manets) possuem alta dinamicidade, o que dificulta o conhecimento prévio dos participantes, fundamental para soluções clássicas do consenso. O problema do consenso tolerante a falhas com participantes desconhecidos passa a ser designado FT-CUP. Greve e Tixeuil propõem uma solução para o FT-CUP a partir da identificação de requisitos minimais de conectividade no grafo de conhecimento estabelecido pelos participantes e considerando-se os requisitos mínimos de sincronia para resolver o problema em presença de falhas. Este trabalho complementa Greve e Tixeuil e analisa os aspectos práticos da realização do FT-CUP em Manets. Ele propõe implementações para os algoritmos sugeridos e realiza diversas simulações em cenários variados. A partir dos resultados obtidos, chegou-se a um conjunto de parâmetros para os quais é possível a convergência do FT-CUP, definindo as características práticas de uma rede Manet onde o consenso pode ser resolvido.*

Abstract. *This work complements Greve and Tixeuil's FT-CUP (fault tolerant consensus with unknown participants), and analyzes the practical aspects of the execution of FT-CUP in mobile ad hoc networks (Manets). Through the aid of simulation experiments, it exhibits a set of parameters allowing FT-CUP to converge. The study conducted show that it is possible to attain consensus even in a network with little knowledge about nodes participating in the system.*

1. Introdução

Informalmente, o consenso tem o objetivo de fazer com que todos os processos corretos do sistema decidam por um valor único proposto pelos mesmos. Diversos problemas em sistemas distribuídos são redutíveis ou equivalentes ao consenso. Isso faz com que este seja fundamental no âmbito dos sistemas distribuídos, podendo ser usado como *middleware* de base para solucionar diversos problemas (como difusão confiável, *group membership*, eleição, etc.). Os detectores de falhas reunem as condições de sincronia necessárias e suficientes para a resolução do consenso em redes clássicas, onde o conjunto de participantes é conhecido. Informalmente, um detector de falhas é um conjunto de oráculos distribuídos que fornece dicas aos processos sobre quais deles estão falhos [Chandra and Toueg 1996].

*Este trabalho tem o apoio do CNPQ e FAPESB-Bahia.

Redes móveis *ad hoc* (Manets) são redes constituídas por nós móveis, cuja comunicação dá-se através de canais sem fio (*wireless*). Para que um nó possa se comunicar diretamente com outro, esse deve estar localizado no raio de transmissão do dispositivo de comunicação. Estas redes, ao contrário das redes do modelo clássico de sistemas distribuídos, não possuem infra-estrutura definida, justamente pelas propriedades de mobilidade que possuem. Estas características peculiares das redes Manets dificultam a resolução de problemas fundamentais em sistemas distribuídos, como o consenso e seus similares [Basile et al. 2003]. A mobilidade, a falta de estrutura da rede e as entradas e saídas deliberadas, fazem com que a dinamicidade seja alta, dificultando o conhecimento inicial sobre os participantes da rede. Numa Manet, no início da execução, um processo não conhece o conjunto de processos que compõem a rede, assim como desconhece a quantidade de processos existentes. Em soluções para o consenso clássico, como as descritas em [Chandra and Toueg 1996, Mostéfaoui and Raynal 1999], estas informações são essenciais, logo tais soluções não são adequadas ao modelo de redes Manets.

O CUP (*consensus with unknown participants*), definido por [Cavin et al. 2004], é uma variação ao problema do consenso, levando-se em consideração o desconhecimento da rede. A solução para esse problema contempla a definição de uma abstração, os *detectores de participação*. Estes são considerados oráculos distribuídos, associados a cada processo, que retornam um conhecimento parcial dos participantes existentes na rede. Diversas classes para estes detectores foram propostas. Cada classe estabelece um grafo de conectividade entre os participantes do sistema, definido a partir da relação de conhecimento estabelecida pelo detector. A classe minimal que possibilita resolver o CUP num contexto sem falhas de processos é o OSR (*One Sink Reducibility*) [Cavin et al. 2004].

Posteriormente, a proposta do CUP foi estendida para o FT-CUP, que é o CUP tolerante à falhas [Cavin et al. 2005]. Na solução apresentada são identificados os requisitos de sincronia necessários para a resolução do FT-CUP, considerando-se os requisitos mínimos de conectividade relativos ao detector de participação (os mesmos encontrados em [Cavin et al. 2004], ou seja o OSR). Tal requisito minimal de sincronia resume-se ao detector de falhas perfeito (classe $\mathcal{P}$), sendo que esse detector só pode ser implementado num ambiente síncrono [Larrea et al. 2004].

Greve e Tixeuil apresentam uma proposta alternativa para solucionar o FT-CUP [Greve and Tixeuil 2007]. Nesta, os autores consideram os requisitos minimais de sincronia, já identificados para a resolução do consenso no modelo clássico, e buscam os requisitos mínimos para a conectividade do grafo de conhecimento que possibilitam resolver o problema. Como demonstrado em [Chandra and Toueg 1996], o requisito mínimo de sincronia resume-se ao detector de falhas da classe $\diamondsuit\mathcal{S}$. Esses detectores possuem propriedades não-confiáveis e podem ser implementados em sistemas dinâmicos, como Manets, redes de sensores e P2P (*peer-to-peer*). Por isso, esta solução é passível de implementação num modelo assíncrono, estendido com requisitos temporais mais fracos. Além disso, ela resolve também a versão uniforme para o consenso. Nesta versão, todos os processos que terminam, decidem pelo mesmo valor, sejam estes corretos ou não.

Este trabalho destina-se a avaliar, em termos práticos, o desempenho da solução proposta em [Greve and Tixeuil 2007] para resolver o FT-CUP em redes móveis *ad-hoc*. Um aspecto crítico da abordagem seguida por todos os algoritmos para resolução do CUP e FT-CUP é o fato de que, se os detectores de participação não satisfazem as propriedades

identificadas, a corretude do consenso não pode ser garantida. Logo, uma questão fundamental, em qualquer uma das soluções, é a implementação dos detectores de participação. Ocorre que não existem trabalhos que propõem implementações que garantam as propriedades para os detectores de participação. Como nosso intuito, neste artigo, é avaliar as possibilidades de convergência do consenso, mesmo diante de uma rede Manet com uma topologia arbitrária de conhecimento, são propostas implementações simples de detectores de participação, que originam grafos de conhecimento arbitrários e que, não necessariamente, satisfazem as propriedades estabelecidas para a resolução do consenso.

Com base nas implementações dos detectores, foram realizadas simulações, variando-se tanto os parâmetros da Manet, quanto os do FT-CUP, estes relativos à conectividade do grafo e ao número suportado de falhas de nós. Busca-se, através dessa análise, avaliar o desempenho dos protocolos envolvidos no que diz respeito à sua latência e ao grau de convergência para uma decisão. A partir dos resultados obtidos, foi possível identificar um conjunto de valores que tornam possível a convergência do consenso. Desta forma, conseguiu-se identificar as características práticas da rede, onde o consenso pode ser executado corretamente, mesmo considerando-se grafos arbitrários de conhecimento.

Este artigo está organizado da seguinte forma. A seção 2 descreve o consenso para redes desconhecidas e abstrações para a sua resolução. A seção 3 apresenta aspectos relacionados à implementações para as simulações, com seus resultados mostradas na seção 4. Por fim, apresentam-se as conclusões na seção 5.

2. O Problema do Consenso e Abstrações para sua Resolução

Consideramos um sistema distribuído formado por um conjunto finito Π de $n > 1$ processos, que se comunicam através de canais confiáveis (sem perdas de mensagens) e de forma assíncrona (sem que sejam feitas hipóteses temporais sobre as ações dos processos e canais). Considera-se que $f < n$ processos podem falhar por parada (*crash*), através de um colapso brusco ou saída deliberada (*switched off*). Considera-se ainda a existência de um protocolo de roteamento confiável tal que, se o processo p conhece o processo q, então p é capaz de enviar mensagens para q de forma confiável.

2.1. Consenso Clássico e Detectores de Falhas

Consenso Clássico. A definição formal do consenso clássico consiste nas seguintes propriedades [Chandra and Toueg 1996]:

- Terminação. Todo processo correto decide por um valor;
- Validade. Se um processo decide por um valor, ele foi proposto por algum processo;
- Acordo. Todos os processos corretos decidem pelo mesmo valor.

Em [Fischer et al. 1985] foi provado que o consenso não pode ser resolvido em ambientes assíncronos na presença de falhas. Isso fez com que surgissem soluções para o consenso em ambientes estendidos com requisitos temporais. Nessa linha, uma das propostas mais significativas é o uso da abstração de detectores de falhas não-confiáveis.

Detector de Falhas. O detector de falhas (FD) é uma abstração de sincronia que funciona como um oráculo associado aos processos distribuídos, informando-os sobre as suspeitas de falhas ocorridas nos componentes dos sistemas [Chandra and Toueg 1996]. De certa forma, os detectores de falhas foram criados para abstrair os requisitos temporais necessários para solucionar problemas de acordo, como o consenso e a difusão atômica,

permitindo que algoritmos sejam propostos para ambientes assíncronos. O detector de falhas é definido pelas propriedades de completude e exatidão. A completude garante que todos processos que falham serão suspeitos. A exatidão faz restrições em relação aos erros do detector. A combinação destas propriedades define as classes dos detectores. Neste trabalho são usadas duas classes de detectores [Chandra and Toueg 1996]:

• Classe $\diamondsuit\mathcal{S}$. Possui as propriedades de *completude forte*, onde todo processo falho será suspeito por todos os processos corretos e a *exatidão fraca após um tempo*, que assegura que, em algum tempo futuro, algum processo correto não será suspeito por nenhum processo correto;

• Classe $\mathcal{P}$. Possui a mesma propriedade de *completude forte* e a *exatidão forte perpetual*, que assegura que um processo nunca será suspeito antes que falhe.

O detector minimal para resolver o consenso clássico em ambientes sujeitos à falhas e com uma maioria de processos corretos ($f < n/2$) é o $\diamondsuit\mathcal{S}$ [Chandra et al. 1996]. Uma variação do consenso clássico é o *consenso uniforme*, no qual a propriedade de acordo é válida tanto para os processos corretos como para processos falhos. Este também pode ser solucionado com o uso do $\diamondsuit\mathcal{S}$. Foi provado em [Larrea et al. 2004] que os detectores perpetuais, inclusive o detector perfeito ($\mathcal{P}$), não podem ser implementados em ambientes parcialmente síncronos.

2.2. Consenso com Participantes Desconhecidos e Detectores de Participação

CUP. Em [Cavin et al. 2004] é introduzido o problema de consenso em redes auto-organizáveis, onde os participantes da rede não são conhecidos inicialmente, problema denominado CUP (*consensus with unknown participants*). As propriedades para o CUP são as mesmas do consenso clássico, porém no CUP os processos não conhecem o Π, conjunto dos processos que compõem a rede. Além disso, não são consideradas falhas dos processos.

Detectores de Participação. Os detectores de participação (PD) surgem para abstrair requisitos de conectividade no problema do consenso. Esses detectores funcionam como oráculos pertencentes a cada processo e retornam um subconjunto aproximado de Π. Seja PD_p o detector de participação associado ao processo p. Quando consultado por p, PD_p retorna um subconjunto de processos de Π com os quais ele pode colaborar. A informação retornada por todos os PDs associados aos processos enriquece o sistema com um grafo de conectividade por conhecimento. Este grafo é orientado, pois a relação de conhecimento não é necessariamente bidirecional. Assim, se $q \in PD_p$, então não necessariamente $p \in PD_q$.

Considere o grafo não direcionado $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, onde $\mathcal{V} = \Pi$ e a aresta $(p, q) \in \mathcal{E}$, se $q \in PD_p$ ou $p \in PD_q$. Para o grafo direcionado $\mathcal{G}_{di} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \Pi$ e a aresta $(p, q) \in \mathcal{E}$ se $q \in PD_p$. [Cavin et al. 2004] definem quatro classes de detectores a partir desses grafos:

• Conexo (CO). O grafo $\mathcal{G}$ é conexo;

• Fortemente conexo (SCO). O grafo $\mathcal{G}_{di}$ é fortemente conexo;

• Completo (FCO). O grafo $\mathcal{G}_{di}$ é completo, ou seja, para todo $p, q \in \Pi$ existe a aresta $(p, q) \in \mathcal{E}$;

• Redutível a Único Poço (OSR). O grafo $\mathcal{G}$ é conexo e a redução de $\mathcal{G}_{di}$ para suas

componentes fortemente conexas possui apenas um poço (*sink*)[1].

De acordo com o grau de conectividade do grafo estabelecido pelo detector, o consenso pode ou não ser resolvido. Sendo assim, para resolver o problema do CUP foi provado que i) o detector $PD \in CO$ é necessário, mas não é suficiente para resolver o consenso, ii) $PD \in SCO$ e $PD \in FCO$ são equivalentes e suficientes, mas não necessários e iii) $PD \in OSR$ é suficiente e necessário, sendo portanto o detector minimal para resolver o CUP [Cavin et al. 2004].

FT-CUP. O trabalho de [Cavin et al. 2005] é uma evolução do trabalho proposto em [Cavin et al. 2004]. Os autores abordam o mesmo problema do CUP, complementando-o com a possibilidade de ocorrer falhas de processos. Esse problema recebe o nome de FT-CUP (*fault tolerant consensus with unknown participants*). Para resolver o consenso, consideram as condições de conectividade minimais identificadas em [Cavin et al. 2004], utilizando o $PD \in OSR$, e provam que o detector de falhas minimal para resolver o FT-CUP, com o $PD \in OSR$, é o detector perpetual perfeito ($FD \in$ classe $\mathcal{P}$). Como o detector perfeito não é implementável no ambiente parcialmente síncrono, a implementação da solução proposta em [Cavin et al. 2005] não pode ser garantida no modelo adotado para redes Manets, onde condições de sincronia mais fortes não são adequadas. Além disso, outra desvantagem desta abordagem é que, mesmo considerando detectores perfeitos, o consenso uniforme não pode ser resolvido quando $PD \in OSR$.

2.3. FT-CUP com Requisitos Minimais de Sincronia

Greve e Tixeuil realizaram uma análise entre os requisitos de conectividade e sincronia necessários para se resolver o FT-CUP [Greve and Tixeuil 2007]. O seu interesse é identificar uma solução a partir das condições de sincronia minimais para resolver o consenso numa rede clássica, que são representadas pelo detector de falhas $\diamondsuit\mathcal{S}$. Portanto, considerando-se esse detector, buscou-se quais os requisitos mínimos relacionados à conectividade do grafo, definido pelo detector de participação, que possibilitariam resolver o consenso. Sendo assim, novas classes de detectores foram propostas:

- k-Conexo (k-CO). O grafo $\mathcal{G}$ não direcionado é k-conexo;
- k-Fortemente Conexo (k-SCO). O grafo $\mathcal{G}_{di}$ é k-fortemente conexo;
- k-Redutível a Único Poço (k-OSR). O grafo $\mathcal{G}$ é conexo. Reduzindo o grafo $\mathcal{G}_{di}$ às componentes k-fortemente conexas, existe apenas uma componente poço. Entre quaisquer pares de componentes existem pelo menos k caminhos disjuntos.

Greve e Tixeuil demonstram que o detector k-OSR é suficiente e necessário para resolver o consenso, na presença de $f < k < n$ falhas, e apresentam algoritmos que resolvem, não somente o consenso FT-CUP, mas também o FT-CUP uniforme com este detector. Para tanto, propõem-se três algoritmos: COLLECT,SINK,CONSENSUS. Cada um deles é executado em sequência, individualmente por cada processo do sistema.

COLLECT – é o primeiro algoritmo a ser executado. Ele é usado pelo processo p para expandir o seu conhecimento sobre os participantes da computação, a partir de uma busca no grafo. No início do algoritmo, o processo p consulta o seu detector de participação para obter PD_p. Em seguida, num procedimento de descoberta do grafo, p irá requisitar novas informações de conhecimento (visões) a novos processos identificados, até que nenhuma nova informação possa ser obtida. O algoritmo executa naturalmente em rodadas.

[1]Um poço é um nó com grau de saída 0.

Em cada rodada $r > 0$, p contata todos os nós que ele não conhece e que acabou de tomar conhecimento na rodada $r - 1$. Na rodada 0, p conhece apenas ele mesmo; na rodada 1, ele conhece todos os que são retornados pelo seu PD_p; na rodada 2, ele conhece quem os processos em PD_p conhecem e assim sucessivamente, até que não possa mais incrementar o seu conhecimento. Este conhecimento acumulado dos participantes é armazenado em Π_p e para evitar bloqueio, em cada rodada espera-se por $|\Pi_p| - f$ respostas dos nós. Ao término do algoritmo, ele terá armazenado em Π_p o conjunto maximal de participantes que ele pode alcançar no grafo de conhecimento.

SINK – é o segundo algoritmo a ser executado. Ele é usado para que cada nó possa saber se pertence à única componente poço (definida pelo grafo k-OSR). Nesse algoritmo, cada nó p envia mensagens para todos os nós em Π_p (encontrados pelo COLLECT), visando comparar o seu conjunto com o de cada processo em Π_p. Seja $q \in \Pi_p$ um desses processos. Se q possui o conjunto $\Pi_q = \Pi_p$, então q responde ACK à solicitação de p, senão ele responde NACK. Se $|\Pi_p| - f$ nós respondem com ACKs, o nó p deduz que ele pertencente à componente poço, já que apenas na componente poço os nós possuem a mesma visão do conjunto de processos no sistema. Se o nó p recebe algum NACK, ele deduz que não pertence à componente poço. Note que se o grafo for k-SCO, todos os nós devem ser considerados pertencentes à componente poço, pois esta componente coincide com o próprio grafo do sistema.

CONSENSUS – este último algoritmo a ser executado é responsável pela realização do acordo. Ele admite dois comportamentos para os nós. Num primeiro, apenas os nós identificados como participantes da componente poço executam um algoritmo de consenso clássico, com a ajuda de detectores de falhas não confiáveis [Chandra et al. 1996]. Num segundo, os demais nós enviam mensagens, requisitando a decisão obtida pelos nós da componente poço. Quando o consenso clássico convergir para algum valor, esta decisão é propagada para os nós que enviaram requisição.

As restrições para a convergência desta solução são as seguintes. Devem existir ao menos $2f + 1$ processos na componente poço, $f < k$. Tais restrições decorrem das condições para solucionar o consenso clássico com detectores não-confiáveis, que exigem uma maioria correta de processos no sistema [Chandra et al. 1996].

O custo computacional dos algoritmos (quantidade de mensagens e latência para decidir) depende do grafo de conhecimento $\mathcal{G}_{di}$ formado pelos processos e este depende da topologia da rede. Embora uma análise teórica, no pior caso, possa ser efetuada, ela só terá significado se acompanhada de uma análise real, baseada numa topologia específica (Manets). Assim, COLLECT tem latência máxima equivalente ao diâmetro do grafo; a qtde de mensagens será proporcional ao número de nós alcançáveis em $\mathcal{G}_{di}$. Em SINK, esses valores também dependem do número de nós alcançáveis. Logo a complexidade de mensagens fica em $O(n^2)$ para esses algoritmos. O CONSENSUS tem a sua complexidade dependente do consenso clássico subjacente. O impacto real dessa complexidade poderá ser apreciado na seção 4.2 .

3. Modelo de Implementação

Cada algoritmo que compõe o FT-CUP corresponde a um módulo implementado no ambiente de simulação. A figura 1 representa a interação entre esses algoritmos dentro do ambiente de simulação. Existem oito módulos: APPLICATION, que representa a aplicação que utiliza o consenso, o PD, módulo do detector de participação;

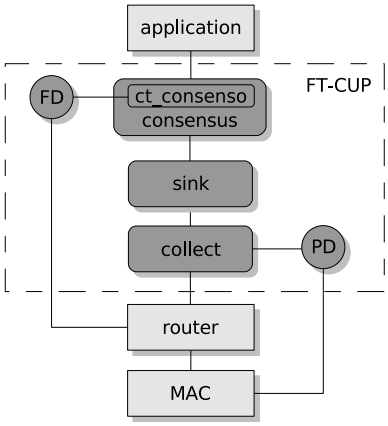


Figura 1. Algoritmos do FT-CUP

COLLECT, SINK e CONSENSUS, que correspondem aos algoritmos existentes no FT-CUP de [Greve and Tixeuil 2007], o FD, módulo de detecção de falhas usado para resolver o consenso clássico; nesse caso, foi implementado o consenso proposto em [Mostéfaoui and Raynal 1999] representado por CT_CONSENSO dentro do módulo CONSENSUS, e finalmente ROUTER, módulo que contém o protocolo de roteamento de mensagens e o módulo MAC que representa a camada de enlace. Nos algoritmos COLLECT, SINK e CONSENSUS, os processos precisam enviar mensagens *unicast*. Sendo assim, foi necessário o uso de um algoritmo de roteamento para entregar corretamente as mensagens aos seus destinos. Como qualquer algoritmo pode ser usado, optou-se pelo *plain flooding* para a entrega de mensagens *unicast*, uma vez que esse protocolo obtém uma taxa de entrega relativamente alta, mesmo tendo uma baixa eficiência [Oliveira et al. 2006]. Sem perder a generalidade do problema, foi utilizada uma camada de enlace que simula o ambiente *wireless* sem perda de mensagens, para ser fiel ao modelo de canais confiáveis[2].

Detectores de Participação. A implementação do detector de participação é fundamental para o funcionamento do FT-CUP. Porém, não existem propostas de implementação de detectores que garantam as propriedades especificadas, a saber, grafo k-SCO, k-OSR, etc. Neste trabalho, propõem-se detectores com premissas simples, que não oferecem garantias em relação as propriedades da classe implementada. Dois detectores foram utilizados, o PD-1hop e PD-2hop, sendo que ambos podem ser facilmente implementados em redes móveis *ad-hoc*. Este trabalho analisa se o desempenho do FT-CUP com estes detectores é satisfatório para o conjunto de parâmetros definido nas simulações.

O detector PD-1hop retorna para os processos a lista dos nós que se encontram no seu alcance de transmissão (vizinhos). O detector inicia enviando mensagens do tipo "hello". Os vizinhos vão receber a mensagem e adicionar o nó fonte ao seu conjunto Π estimado. Esse algoritmo termina quando um tempo (*timeout*) é atingido e a quantidade de participantes encontrada é maior ou igual a k. Durante este tempo os nós podem emitir novas mensagens "hello" em resposta a mensagens "hello" recebidas por outros nós. Garante-se assim que o detector associado a cada participante p irá retornar o conjunto de vizinhos com os quais p se comunica e esse conjunto terá cardinalidade no mínimo k.

O detector PD-2hop realiza os mesmos passos do primeiro, exceto que os nós

[2]Canais confiáveis podem ser implementados usando retransmissões e mensagens de confirmação.

Tabela 1. Parâmetros de simulação

Parâmetros de simulação	
Simulador	OMNeT++ (3.4)
Quantidade de nós	10 à 30
Área	300 x 300 m^2
Alcance *wireless*	100 m
Tempo de simulação	10 s
Repetições	10
Padrão de Mobilidade	Random Waypoint
Velocidade dos nós	de 20 à 50 m/s
Tempo de pausa	até $2s$

anexam nas mensagens "hello" o conjunto de participantes encontrado até o momento. Desta forma, ao receber a mensagem, o nó adiciona ao seu conjunto de conhecimento o nó fonte e o conjunto enviado pelo mesmo. Isto faz com que o conhecimento seja pelo menos de dois níveis (2 hops) em relação ao grafo de comunicação.

Modelo de Falhas. No modelo de falhas adotado na implementação, o processo falha por parada. Nesse caso, a falha ocorre depois da execução do detector de participação. Se o nó falha antes ou durante a execução do detector, seria como se o nó nunca tivesse existido na rede, fazendo com que os algoritmos do FT-CUP não fossem afetados. Se o nó falha depois da detecção, significa que outros nós da rede conhecem o nó falho e que esse nó deveria participar da execução dos algoritmos. Desta forma, buscando avaliar o impacto das falhas no FT-CUP, considera-se que a falha do nó ocorre após a execução dos detectores de participação. O detector de falhas necessário para o FT-CUP é o $\diamondsuit\mathcal{S}$. Nesse caso, optou-se por utilizar a implementação para o detector da classe $\diamondsuit\mathcal{P}$, descrita em [Chandra and Toueg 1996]. Esse detector satisfaz também as propriedades do detector $\diamondsuit\mathcal{S}$, já que a classe $\diamondsuit\mathcal{P}$ é redutível à classe $\diamondsuit\mathcal{S}$.

4. Simulações

As simulações foram executadas sobre a ferramenta OMNeT++ [Varga 2001], um ambiente de simulações modular orientado a eventos discretos. Para implementar o modelo das redes Manets, foi utilizado o Mobility Framework, desenvolvido especificamente para o uso em conjunto com o OMNET++. Cada simulação é executada por um tempo definido, sendo que, em cada uma delas, apenas um consenso FT-CUP é executado. Esse consenso é iniciado por todos os nós da rede e quando uma proposta é solicitada à um nó específico, esse nó propõe o valor que corresponde ao seu identificador.

4.1. Cenários

Ambiente da Manet. A escolha dos parâmetros associados ao ambiente de simulação (redes móveis *ad-hoc*) baseou-se nas considerações recentemente realizadas por [Kurkowski et al. 2005], que faz uma crítica aos trabalhos que simulam algoritmos para Manets, mostrando como devem ser selecionados os parâmetros para que esses sejam realistas. A tabela 1 descreve os parâmetros gerais utilizados nas simulações. A *quantidade de nós*, que varia de 10 à 30 nós, a *área* de 300x$300m^2$ e o *alcance do dispositivo de comunicação wireless* com raio de $100m$ foram selecionados visando definir a densidade

da rede. Testes preliminares foram realizados com uma rede mais esparsa (área e alcance de transmissão maiores com menor número de nós). Entretanto, nesse cenário o consenso não apresentou boas taxas de convergência, dado que frequentemente ocorria o particionamento da rede. Esses valores selecionados para a área e alcance de transmissão permitem uma conectividade entre os nós suficiente para evitar esse particionamento. Por exemplo, para 10 nós, espera-se que cada nó tenha em média 3.49 vizinhos. Para 30 nós, a média é de 10.47 vizinhos. O *tempo de duração* de cada simulação é de 10 segundos, tempo próximo ao máximo obtido nas simulações preliminares do FT-CUP para que houvesse uma convergência para um valor de decisão. O *padrão de mobilidade* usado é o *Random Waypoint* com a *velocidade* variando de 20 à 50 m/s e *tempo de pausa* de até $2s$. Esta velocidade selecionada não interfere na convergência do consenso devido à densidade da rede, que reduz a possibilidade de particionamento da rede, fazendo com que o algoritmo de roteamento execute corretamente e entregue as mensagens de forma confiável. Porém, esta interfere na latência, pois um nó pode mover-se para fora do alcance de transmissão de outro nó fazendo com que o algoritmo de roteamento demore um tempo maior para entregar a mensagem.

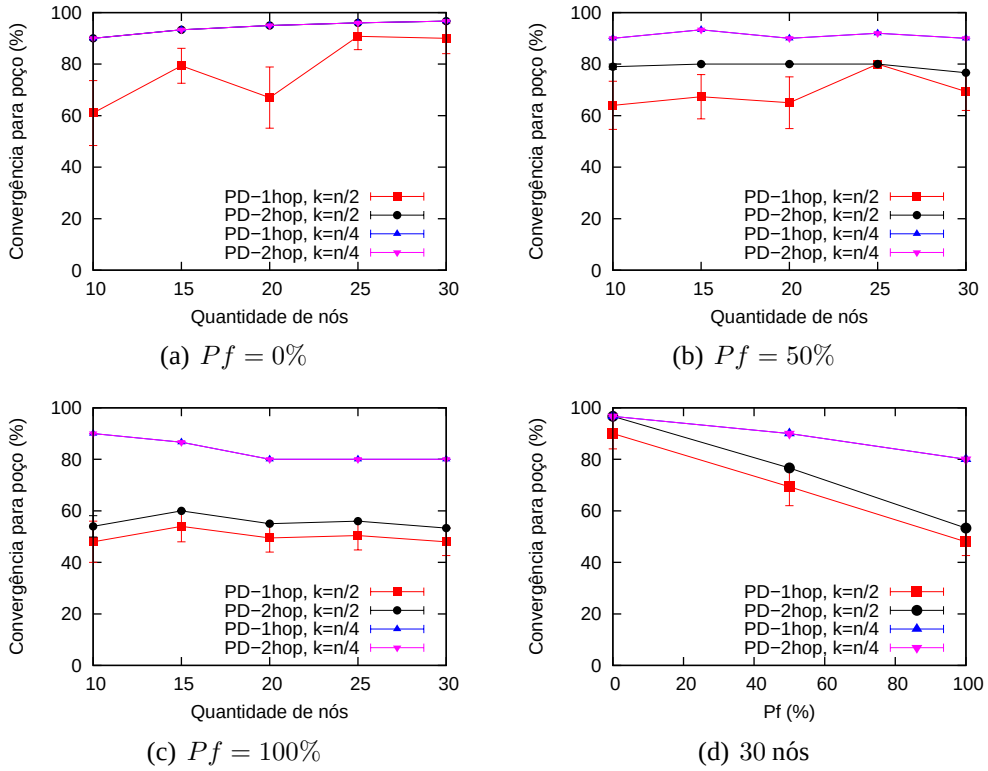
Ambiente do FT-CUP. Para os parâmetros específicos do FT-CUP, k (*conectividade do grafo*) e f (*máximo de falhas tolerável*), considera-se uma combinação de valores, para que se possa analisar seu impacto no FT-CUP e determinar as melhores configurações para o consenso, obedecendo à restrição $f < k < n$. Além desses parâmetros, o Pf, porcentagem real de falhas, é definido. Para k, considera-se os valores $k = n/2$ e $k = n/4$. O valor máximo para k no FT-CUP é $k = n - 1$; porém, esse valor não foi utilizado por causar a conectividade completa do grafo, levando o problema à características similares ao consenso clássico. Os valores $k = n/2$ e $k = n/4$ fazem com que a rede possua conectividade média e baixa, respectivamente.

O parâmetro f foi definido em função de k, através da relação $f = k - 1$, que considera a quantidade de falhas máxima para o FT-CUP. O Pf foi definido como um percentual em relação à f. Os valores considerados foram de 0%, 50% e 100% em relação à f, fazendo com que existam cenários sem falhas ($Pf = 0$), com metade das falhas possíveis ($Pf = (k - 1)/2$) e com o máximo de falhas tolerável ($Pf = k - 1$). Os detectores utilizados nas simulações são os mesmos definidos na seção 3, o PD-1hop e PD-2hop, usando um *timeout* de $100ms$. A combinação dos parâmetros k, Pf e do detector de participação definem conjuntos de simulações. Cada conjunto é simulado por 10 vezes, sendo que o que se vê nos gráficos é a média dessas 10 repetições.

4.2. Resultados

Para a análise dos resultados, foram escolhidas quatro métricas. A *convergência do FT-CUP* refere-se à porcentagem de nós que terminam o consenso com uma decisão. A métrica *convergência para poço* representa a porcentagem dos nós que fazem parte do poço e que conseguem determinar que estão na componente poço. A *latência do FT-CUP* mostra a média dos tempos gastos para a convergência do FT-CUP e a *latência do* SINK, a média dos tempos para execução do algoritmo SINK.

Convergência para Poço. A figura 2 representa a porcentagem de nós pertencentes à componente poço. No gráfico 2(a) ($Pf = 0\%$), os resultados para $k = n/4$ e $k = n/2$ com PD-2hop foram em média de 95%. Para $k = n/2$ e PD-1hop os valores ficaram em



(a) $Pf = 0\%$ (b) $Pf = 50\%$

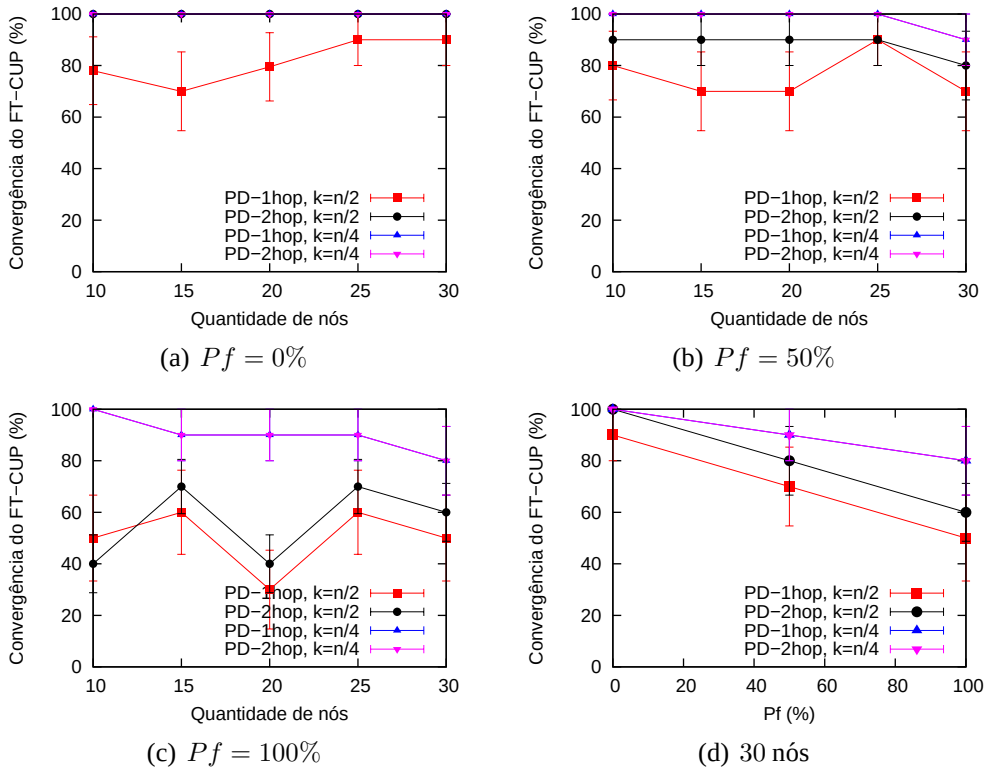
(c) $Pf = 100\%$ (d) 30 nós

Figura 2. Convergência para Poço com $Pf = 0\%$ (a), $Pf = 50\%$ (b), $Pf = 100\%$ (c) e em função da quantidade de falhas para 30 nós (d).

torno de 80%. A limitação da informação retornada pelo detector PD-1hop faz com que as visões dos processos pertencentes a uma mesma componente não sejam idênticas após a execução do COLLECT. Nesse caso, pode acontecer do SINK indicar erradamente que o nó não pertence à componente poço. A terminação do algoritmo COLLECT acontece quando o nó p recebe $|\Pi_p| - f$ visões dos nós conhecidos, sendo que, mesmo sem falhas reais ($Pf = 0\%$), o nó não espera por todas as respostas e sempre ignora f visões de processos. Como neste trabalho o grafo formado pela rede não é garantidamente k-OSR, pode acontecer de algum subconjunto de Π_p, dentre os f nós ignorados, conter novos processos, fazendo com que dois processos tenham visões diferentes e que o SINK convirja erradamente. Uma possível melhoria, a ser incorporada ao COLLECT, para evitar esse fenômeno, seria a substituição da espera por $|\Pi_p| - f$ processos pelo uso do detector de falhas. Ou seja, espera-se respostas dos processos, enquanto esses não são suspeitos.

Para $k = n/4$, a porcentagem de convergência é superior à de $k = n/2$. Isto pode estar ocorrendo pelo fato da quantidade de falhas suportadas (em termos absolutos) ser menor. Ou seja, como adota-se ($f \leq k - 1$), tem-se ($f \leq n/4 - 1$), no lugar de ($f \leq n/2 - 1$). Assim, tem-se que cada nó p recebe uma quantidade maior de visões no COLLECT ($|\Pi_p| - f$), fazendo com que a possibilidade de ocorrerem visões divergentes na rede seja menor. Os resultados alcançados para ambos os detectores em $k = n/4$ foram coincidentes. Conclui-se, desta forma, que o nível de informação trazido pelo PD-1hop é suficiente para a resolução do problema com $k = n/4$ e os parâmetros especificados na seção 4.1. Comparando os gráficos 2(a), 2(b) e 2(c), percebe-se que a porcentagem cai à medida que as falhas aumentam. Esta queda é melhor visualizada através do gráfico 2(d).

Para $k = n/2$, a incidência de falhas é maior, por isso a queda na convergência em relação ao aumento das falhas é mais acentuada para este valor de k.



(a) $Pf = 0\%$ (b) $Pf = 50\%$

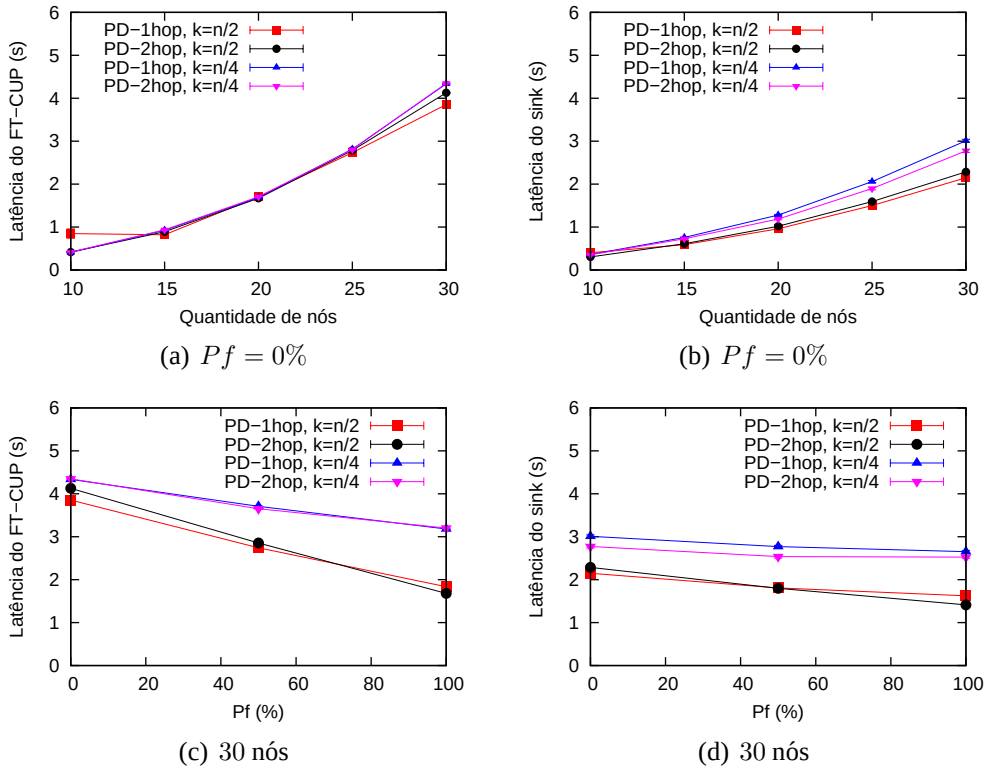
(c) $Pf = 100\%$ (d) 30 nós

Figura 3. Convergência do FT-CUP para $Pf = 0\%$ (a), $Pf = 50\%$ (b), $Pf = 100\%$ (c) e em função da quantidade de falhas para 30 nós (d).

Convergência do FT-CUP. Na figura 3 é mostrada a porcentagem de nós que decidem por um valor, representando a taxa de convergência do FT-CUP. Mesmo na ausência de falhas (gráfico 3(a)), para $k = n/2$ e PD-1hop, a convergência oscilou em torno de 80%. Conclui-se que o detector PD-1hop é inadequado para $k = n/2$, ou seja, para ambientes com maiores necessidades de conectividade, deve ser usado um detector mais forte, que oferece mais garantias sobre as propriedades da classe a que pertence. Para $k = n/4$, o desempenho foi similar em ambos os detectores, ficando exatamente em 100% de convergência (na ausência de falhas). A causa disto é o bom desempenho na convergência para poço (representado na figura 2), reforçando a hipótese de que, para uma conectividade menor, um detector simples possa ser utilizado. Constata-se ainda, pelos mesmos motivos, que o desempenho para $k = n/4$ é sempre maior do que para $k = n/2$.

No gráfico 3(b) são mostrados os resultados para $Pf = 50\%$. Nesse gráfico, a convergência cai para 90% com $k = n/2$ e PD-2hop, mantendo-se em 100% para $k = n/4$ até 25 nós. Para $k = n/2$ e PD-1hop, a convergência oscila em torno de 70%. Apenas no gráfico 3(c), onde $Pf = 100\%$, a taxa de convergência para $k = n/4$ diminui significativamente, assim como para $k = n/2$. O gráfico 3(d) representa a taxa de convergência em função da porcentagem de falhas, levando-se em consideração 30 nós da rede. Por esse gráfico pode-se observar que, à medida que a porcentagem de falhas aumenta, há uma tendência de queda na taxa de convergência. Esta queda segue o mesmo comportamento observado na convergência para poço através do gráfico 2(d).

A baixa convergência do FT-CUP na ocorrência de muitas falhas ou quando $k = n/2$, pode estar ocorrendo devido a um bloqueio do protocolo do CONSENSUS, ocasionado pela baixa convergência para poço. Neste caso, um processo p, que deveria participar ativamente do consenso, porque pertence à componente poço, não participa, porque não se considera como pertencente ao poço (decisão errônea do protocolo SINK) e atua como nó passivo, esperando a mensagem de decisão. Os demais processos no poço, o consideram como ativo e esperam pela sua mensagem (seja de proposição, de ACK, etc.). Se o processo p não é suspeito pelo detector de falhas, os demais processos na componente poço podem ficar bloqueados esperando pela sua mensagem, que nunca virá. Uma possível solução para isso seria fazer com que o detector de falhas suspeite dos nós que não se consideram pertencentes à componente poço.



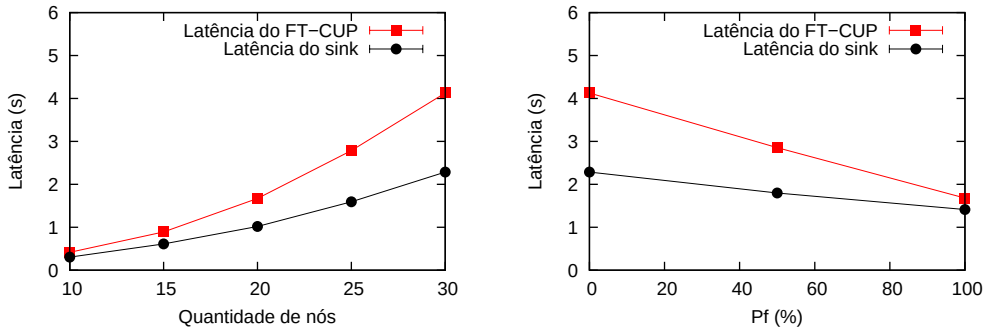
(a) $Pf = 0\%$ (b) $Pf = 0\%$

(c) 30 nós (d) 30 nós

Figura 4. Latência do FT-CUP (a) e SINK (b) para $Pf = 0\%$, e latência do FT-CUP (c) e SINK (d) em função da quantidade de falhas para 30 nós.

Latência. As figuras 4 e 5 representam os resultados de latência obtidos nas simulações. Os gráficos 4(a) e 4(b) mostram, respectivamente, a latência do FT-CUP e do SINK, em relação à quantidade de nós para $Pf = 0\%$. Os resultados para as outras porcentagens de falhas são similares, sendo que apenas são mostrados para 30 nós, nos gráficos 4(c) e 4(d). O comportamento da latência do SINK é similar à latência do FT-CUP. Observe que a latência aumenta à medida que aumenta a quantidade de nós na rede. Isto se deve à alta complexidade de mensagens, $O(n^2)$, dos três algoritmos envolvidos no FT-CUP: COLLECT, SINK e CONSENSUS.

A diferença da latência é considerável quando comparados os resultados de $k = n/2$ e $k = n/4$ nos gráficos 4(c) e 4(d), sendo aquela para $k = n/4$ a que apresenta maiores valores. Para este valor de k, o detector de participação retorna um conjunto de



(a) PD-2hop, $k = n/2$, $Pf = 0\%$ (b) PD-2hop, $k = n/2$, 30 nós

Figura 5. Latência do FT-CUP e SINK com PD-2hop e $k = n/2$ para $Pf = 0\%$ (a) e em função da quantidade de falhas para 30 nós (b).

participantes menor, fazendo com que o algoritmo COLLECT efetue mais etapas de busca no grafo, exigindo mais tempo para convergir para o conjunto maximal de processos alcançáveis na rede. Nesses gráficos pode-se observar também que a latência diminui com o aumento da porcentagem de falhas, pois os nós falhos não participam dos algoritmos do FT-CUP, o que faz com que a complexidade de mensagens diminua e reduza a latência.

Pode-se observar que o SINK é responsável por parte considerável do tempo gasto para a convergência do consenso pelo gráfico 5(a), que representa a latência para $Pf = 0\%$ com $k = n/2$ e PD-2hop. A diferença entre as duas curvas nos gráficos representa a latência do algoritmo CT_CONSENSO, que é constituido basicamente pelo tempo gasto no consenso clássico [Mostéfaoui and Raynal 1999]. Pelo gráfico 5(b), têm-se que, com o aumento de Pf, diminui a latência do consenso clássico. Este comportamento é consequência da complexidade de mensagens ser relacionada com a quantidade de nós que participam do consenso. Mais falhas na rede indicam que menos nós participam do consenso clássico, fazendo com que a latência diminua.

5. Conclusões

O FT-CUP, consenso tolerante a falhas para redes desconhecidas, é uma nova variação do problema do consenso, levando em consideração o não conhecimento dos participantes da rede, fato característico de redes móveis *ad-hoc*. Ele utiliza a abstração dos detectores de participação para obter um conhecimento parcial dos participantes da rede. Este trabalho analisou os aspectos práticos do FT-CUP proposto em [Greve and Tixeuil 2007]. Foram sugeridas implementações para os algoritmos e para os detectores de participação, sendo propostos os detectores PD-1hop e PD-2hop. Simulações foram executadas com esses algoritmos, variando os parâmetros de conectividade (k) e de falhas (f) para o FT-CUP.

Os resultados das simulações evidenciaram que o detector de participação e o parâmetro de conectividade k são fundamentais para a convergência do consenso. Nas simulações realizadas, percebeu-se que o uso dos detectores simples PD-1hop e PD-2hop deu origem a um grafo com um único poço. Além disto, verificou-se que a latência do SINK e do FT-CUP crescem com o aumento da quantidade de nós da rede. Os resultados evidenciaram que a latência do SINK constitui uma parte considerável da latência total do FT-CUP. Para $k = n/2$ e o detector PD-1hop, os resultados de convergência foram consideravelmente abaixo dos resultados encontrados com PD-2hop. Nos resultados com

o detector PD-1hop com $k = n/2$, o FT-CUP não obteve 100% de convergência, mesmo na ausência de falhas. Usando o detector PD-2hop com $k = n/2$, aumenta-se a relação de conectividade dos nós, o que melhorou a taxa de convergência do procotolo, chegando aos 100% no cenário sem falhas. Em cenários com uma menor capacidade de tolerar falhas, como em $k = n/4$, os resultados de convergência para PD-1hop e PD-2hop não mostraram diferenças relevantes, levando à conclusão de que para cenários com poucas falhas um detector simples é satisfatório. A taxa de convergência foi sempre maior com $k = n/4$ do que com $k = n/2$. Conclui-se que, para tolerar maiores quantidades de falhas, é preciso um detector de participação com premissas mais fortes. Em cenários onde a capacidade de tolerar falhas é menor, os resultados para o FT-CUP foram satisfatórios, mesmo usando detectores de participação com premissas simples.

Referências

Basile, C., Killijian, M., and Powell, D. (2003). A survey of dependability issues in mobile wireless networks. Technical report, LAAS CNRS, Toulouse, France.

Cavin, D., Sasson, Y., and Schiper, A. (2004). Consensus with unknown participants or fundamental self-organization. In *Third Int. Conf. on Ad hoc Net. and Wireless (ADHOC-NOW 2004)*, pages 135–148, Vancouver, Canada.

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, Ecole Polytechnique Federale de Lausanne.

Chandra, T. D., Hadzilacos, V., and Toueg, S. (1996). The weakest failure detector for solving consensus. *J. ACM*, 43(4):685–722.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382.

Greve, F. and Tixeuil, S. (2007). Knowledge conectivity vs. synchrony requirements for fault-tolerant agreement in unknown networks. DSN 2007 Int. Conf. Dependable Systems and Networks, to be published.

Kurkowski, S., Camp, T., and Colagrosso, M. (2005). Manet simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61.

Larrea, M., Fernandez, A., and Arevalo, S. (2004). On the implementation of unreliable failure detectors in partially synchronous systems. *IEEE Transactions on Computers*, 53(7):815–828.

Mostéfaoui, A. and Raynal, M. (1999). Solving consensus using Chandra-Toueg's unreliable failure detectors: A general quorum-based approach. In *13th Int. Symp. on Distributed Computing (DISC'99)*, volume 1693 of *LNCS*, pages 49–63.

Oliveira, T., Costa, V., Greve, F., and Schnitman, L. (2006). Evaluating the impact of faults on broadcasting protocols for manets. In *WTF 2006 VII Workshop de Testes e Tolerância a Falhas*, pages 49–60.

Varga, A. (2001). The OMNeT++ discrete event simulation system. In *European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic.