MiCCS4Mobile

Middleware para Comunicação e Coordenação Segura em Redes Ad-hoc com Participantes Desconhecidos

David Matos, Emanuel Alves, Nuno Neves e Alysson Bessani

Universidade de Lisboa, Faculdade Ciências
{ealves,dmatos}@lasige.di.fc.ul.pt,
{neves,bessani}@di.fc.ul.pt

Resumo A proliferação de dispositivos sem-fios de variadas capacidades (e.g., *smartphones* e *tablets*) levaram ao aumento de desenvolvimento de aplicações móveis. Algumas aplicações exploram protocolos de redes sem fios e a mobilidade dos dispositivos para aumentar as suas funcionalidades. A classe de protocolos de redes sem fios com dispositivos móveis chama-se *MANET*. Devido ao meio de comunicação, as *MANETs* têm um conjunto de vulnerabilidades e limitações que dificultam a concretização de aplicações distribuídas. Contudo, o uso de um *middleware* que suporte a comunicação e coordenação entre participantes (dispositivos), ainda que na presença de participantes bizantinos e participantes desconhecidos, seria extremamente útil para construção destas aplicações. O artigo propõe uma pilha protocolar com protocolos de comunicação e acordo em *MANETs*. A pilha protocolar foi avaliada no simulador ns-3, simulando vários ambientes de execução, mostrando que a pilha protocolar pode efetivamente ser executada num conjunto interessante de cenários.

Keywords: MANET, Filiação, Acordo mútuo binário, Participantes bizantinos, Participantes desconhecidos

1 Introdução

A proliferação de dispositivos sem-fios de variadas capacidades (e.g., *smartphones* e *tablets*) levaram ao aumento do desenvolvimento de aplicações móveis. Estes dispositivos têm um conjunto de características em comum: interfaces de comunicação sem-fios de curto alcance, poder computacional reduzido, apresentam alguma mobilidade e são relativamente baratos. Juntamente com a proliferação destes dispositivos, surgiram aplicações interessantes como plataformas de disseminação de apresentações, calendários e outras aplicações para trabalho cooperativo (e.g. [1]). Algumas destas aplicações dependem da comunicação com sistemas distantes do utilizador para efetuar operações, como disseminação e trabalho cooperativo, que exigem comunicação e coordenação (acordo) que não é possível efetuar diretamente entre participantes. Por vezes a comunicação apenas é possível através de redes convencionais encontradas em habitações, nos

locais de trabalho ou em locais públicos. Estas redes convencionais requerem uma infraestrutura fixa com administração centralizada, que exigem um esforço não desprezível de configuração e manutenção. A solução para estas limitações consiste em criar aplicações distribuídas sobre redes ad-hoc móveis, i.e., Mobile Ad-hoc Network (MANET).

Uma das vantagens de protocolos de *MANETs* é a flexibilidade face a alterações de topologia de rede, que ocorre por vezes devido à mobilidade dos dispositivos (podendo sair do alcance de comunicação de outros dispositivos). A mobilidade dos dispositivos torna extremamente difícil a criação de uma filiação coerente a todos participantes (dispositivos) da rede, o que impossibilita a execução de protocolos de acordo. Por acréscimo, o meio de comunicação de sem fios das *MANETs* estão sujeitas a uma vasta quantidade de ataques.

Na literatura foram propostas soluções de algoritmos teórico de filiação tolerantes a faltas por paragem com participantes desconhecidos para execução de protocolos de acordo [2,3]. Alchieri et al. [4] descreve um algoritmo teórico de filiação tolerante a faltas bizantinas [5] com participantes desconhecidos. Moniz et al. [6] descreve um algoritmo que resolve o problema k-acordo (i.e., k-consensus [7]). Este algoritmo contorna a impossibilidade FLP [8] e de Santoro-Widmayer [9], recorrendo à aleatoriedade, e garantindo a terminação com uma probabilidade de 1.

O artigo faz uma proposta inicial de um middleware que ofereça primitivas de comunicação, filiação e acordo - MICCS4Mobile –, no intuito de enriquecer a construção de aplicações distribuídas e cooperativas. O MICCS4Mobile é composto por implementações dos algoritmos Alchieri et al. [4] e Moniz et la. [6] adaptados para *MANETs*. O artigo também inclui uma descrição das modificações aos algoritmos [4,6] para torná-los mais eficientes e robustos e uma avaliação da escalabilidade da pilha protocolar em cenários simulados.

O artigo está estruturado da seguinte forma. A Secção 2 apresenta o modelo de sistema. A Secção 3 ilustra a pilha protocolar e a relação entre protocolos com maior detalhe. A Secção 4 inclui uma avaliação de resultados dos protocolos. Finalmente, a Secção 5 são discutidos os resultados obtidos e concluído o artigo.

2 Modelo do Sistema

O sistema distribuído é composto por um conjunto finito $\Pi = \{p_1, p_2, p_3, ..., p_n\}$ de participantes desconhecidos. Considera-se que um conjunto de participantes conhecidos é aquele em que $\forall i \in \Pi : \Pi_i = \Pi$, sendo Π_i os participantes que um participante i conhece. Todos os conjuntos que não sejam compostos por participantes desconhecidos são considerados como conjuntos de participantes desconhecidos.

Cada participante possui um, e só um, identificador no universo de Π. Assume-se a inexistência de ataques de personificação (i.e., *sybil attacks* [10]). Esta assunção pode ser substanciada com protocolos de encaminhamento resiliente a participantes bizantinas (e.g., [11]). A comunicação entre participantes é baseada em mensagens, transmitidas por canais que garantem autenticação e integridade.

Um participante que siga o algoritmo é considerado um participante correto. Todos os participantes que não são corretos ou que saíram da rede durante execução do protocolos são considerados como participantes bizantinos [5]. O sistema distribuído permanece correto enquanto existir, no máximo, f participantes bizantinos. Apesar do conjunto ser composto por participantes desconhecidos, assume-se a possibilidade de estimar $|\Pi|$ para definir f.

O conjunto Π_i inicial é dado pelo protocolo *participant discovery* [4], posteriormente é expandido durante a execução dos protocolos de filiação [4] da seguinte forma. Um participante i comunica com j apenas se conhecer j, i.e., $j \in \Pi_i$. Se o participante j recebe a mensagem de i e $i \notin \Pi_j$, então i é adicionado ao conjunto Π_j.

Não existe assunção sobre a velocidade de processamento dos participantes ou sobre o atraso das mensagens no sistema distribuído, ou seja, assume-se o modelo assíncrono. Contudo, a mobilidade dos participantes não pode modificar a topologia de rede, durante execução dos protocolos.

3 Pilha Protocolar

A pilha de protocolos, ilustrada na Figura 1, é dividida em duas camadas: a camada de Suporte e a camada de Acordo. A camada de Suporte responde às necessidades de comunicação e filiação dos protocolos de acordo. A camada de Acordo executa o protocolo de acordo tolerante a participantes bizantinas [6] sobre a filiação obtida pela camada de Suporte. As próximas subseções apresentam uma breve descrição dos principais algoritmos concretizados. Para uma descrição detalhada e provas ver [4,6].

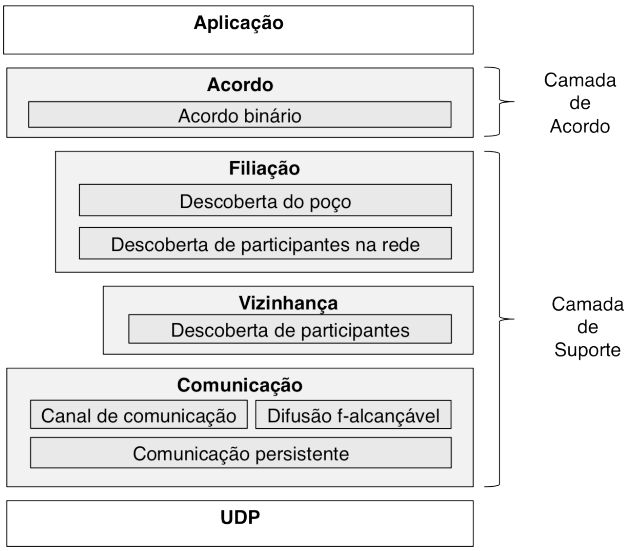
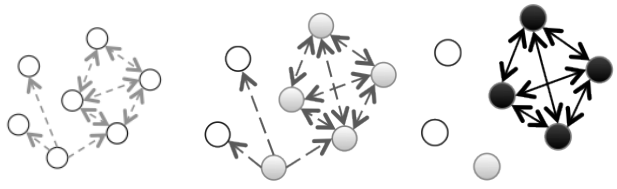


Figura 1: Pilha protocolar MICCS4Mobile.

3.1 Camada de Suporte

O protocolo de *comunicação persistente* utiliza canais de comunicação UDP. A motivação da utilização do protocolo UDP é que este permite a difusão local de mensagens, permitindo reduzir número de transmissões necessárias de uma mensagem. Os protocolos da camada de filiação dependem da garantia de fiabilidade da entrega de mensagens. A forma de garantir essa propriedade através de canais não fiáveis (e.g., UDP) é retransmitir continuamente em períodos de tempo aleatórios todas as mensagens, i.e., *stubborn channels* em inglês. O protocolo de *canal de comunicação* é uma abstração básica que elimina a retransmissão infinita, implementando confirmação de recepção de mensagens. O protocolo é composto pelos seguintes passos de comunicação: 1) emissor envia a mensagem para o receptor, 2) o receptor envia a confirmação da recepção da mensagem e 3) o emissor repete o passo 1 até receber a confirmação. O protocolo de *difusão f-alcançável* é um protocolo de difusão em que um receptor apenas entrega uma mensagem recebida, se esta veio por $f + 1$ caminhos disjuntos. O protocolo tem por objetivo garantir que a mensagem entregue é fidedigna, i.e., que veio por, pelo menos, um caminho composto por participantes corretos.

O protocolo *descoberta de participantes*, ilustrado na Figura 2a, assume o papel de *participant discovery* de [4].Este protocolo efetua uma difusão local informando a presença de um participante. O protocolo de *descoberta de participantes na rede*, ilustrado na Figura 2b, utiliza o protocolo de difusão f-alcançável para enviar um pedido de visões parciais aos participantes assegurando que este é fidedigno. O *descoberta de participantes na rede* reúne os conjuntos de participantes conhecidos até que a soma do número destes conjuntos reunidos com os pedidos de conjuntos pendentes seja menor que f. O resultado final é um conjunto de participantes que estão presentes em mais de f conjuntos reunidos. Este conjunto possibilita obter uma visão fidedigna sobre participantes existentes na rede, i.e., G_i. O protocolo de *descoberta de poço*, ilustrado na Figura 2c, usa o protocolo de *canal de comunicação*, para determinar se os restantes participantes da visão, obtida pelo protocolo *descoberta de participantes na rede*, contêm uma proposta de filiação igual, i.e., $G_i = G_j$ sendo i e j dois participantes. Se reunir mais de $|G_i| - f$ respostas afirmativas, o participante i assume que pertence à filiação final: caso reúna mais de f respostas negativas, assume que não pertence à filiação final.



(a) Descoberta de participantes

(b) Descoberta de participantes na rede

(c) Descoberta do poço

Figura 2: Execução da camada de suporte com $f = 1$

3.2 Camada de Acordo

O protocolo de *acordo binário* é executado pelos participante da filiação. Cada participante i tem um estado interno $s_i = \{\phi_i, v_i, V_i, status_i\}$, em que ϕ_i é o número da fase do protocolo, v_i é o valor proposto, V_i é o vector de mensagens recebidas e $status_i \in \{DECIDED, UNDECIDED\}$ consiste no estado de decisão.

Na Figura 3 apresenta-se um exemplo de uma execução do protocolo de acordo. No exemplo temos $n = 4$, $f = 1$ e o quorum = 3. Um dos processos (P4) é bizantino e não segue a execução normal e o processo P1 não recebe a mensagem difundida por P2. Neste caso, como as mensagens enviadas por P4 não são congruentes com as mensagens anteriores, todas as mensagens de P4 na fase de *LOCK* e *DECIDE* são descartadas no processo de validação. Os restantes processos (P1, P2 e P3) decidem o valor correto, 1. Embora P4 decida um valor diferente, isto não afeta a validade do algoritmo porque P4 não é um processo correto.

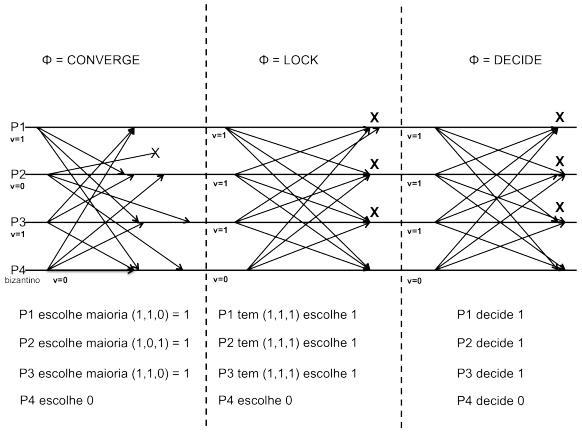


Figura 3: Exemplo de execução da camada acordo com $f = 1$

3.3 Modificações nos algoritmos originais

Como referido anteriormente na Subseção 3.1, o protocolo de comunicação persistente utiliza canais de comunicação UDP dado que estes suportam difusão local. É possível melhorar o desempenho recorrendo a difusão, uma vez que permite reduzir o número de mensagens. Os protocolo de *difusão f-alcançável* e o *descoberta do poço* exploram a difusão local para diminuir número de mensagens necessárias. O protocolo de *difusão f-alcançável* é um protocolo de *flooding* que utiliza o protocolo *comunicação persistente* assegurando a entrega de mensagens. O protocolo *difusão f-alcançável* não retransmite a mensagem m se $|\Pi_i| - f$ transmitir m. Assim não são necessárias mensagens de confirmação de entrega. O protocolo de *descoberta do poço* difunde localmente o seu pedido de comparação e mais tarde enviar inclusive para cada $q \in G_i \setminus \Pi_i$. Este mecanismo reduz o número de mensagens caso $G_i \subseteq \Pi_i$.

O protocolo de *acordo binário* segue a descrição original do algoritmo [6]. Foi necessário tornar o processo de validação mais robusto no caso em que ocorrem

participantes bizantinas que afetam o valor proposto e a variável de *status*. O novo processo de validação tem apenas em conta as mensagens da fase anterior (o algoritmo original recorria a mensagens até duas fases anteriores) para validar a mensagem recebida, o que simplifica o processo de validação.

4 Implementação

A vantagem de utilizar um simulador de rede é que permite analisar desempenho da pilha protocolar em cenários de mobilidade, maior número de participantes, aumento de participantes corretos e bizantinos. O simulador de rede escolhido foi ns-3, por sobressair nas comparações entre simuladores (e.g., [12]). A pilha protocolar foi concretizada como um módulo C++ do simulador ns-3, respeitando o desenho protocolar da Figura 1. Os detalhes de implementação do módulo ns-3 concretizado estão na Tabela 1.

A interface do módulo contêm o método *Propose* e *SetCallback*. O método *Propose* inicia protocolo de acordo com a proposta de valor passado como argumento. O *SetCallback* define o método a chamar quando protocolo terminar com valor acordado. O ns-3 obriga a utilização de objetos (e.g., UdpSockets) da sua API, que modificaram ligeiramente os protocolos de comunicação da pilha protocolar. Os canais de comunicação são criados por uma fábrica de objetos *UdpSocketFactory* do ns-3, que contêm uma API diferente dos *sockets* POSIX. O objeto UdpSocket não suporta esperas bloqueates nos sockets, obrigando a definir *callbacks* para recepção de pacotes. O agendamento de alarmes é feito pelo escalonador do ns-3.

Camada	*Número de classes*	*Número de linhas de código*
Acordo	8	2923
Suporte	15	5519

Tabela 1: Concretização do MICCS4Mobile em módulo ns-3

5 Avaliação

A rede ad-hoc sem fios simulada foi configurada com protocolo de comunicação 802.11g e o protocolo de encaminhamento *Ad-hoc On Demand Distance Vector* (AODV). Consideramos que o raio de alcance dos meios de comunicação varia entre 0 a 38 metros e as retransmissões de mensagens são feitas entre 0 a 1.6 segundos.

O comportamento da pilha protocolar foi simulado em dois cenários: grelha e aleatório. O cenário grelha é uma representação genérica de uma sala de aula. O cenário aleatório é uma representação genérica de um recinto fechado sem obstáculos, onde os participantes se deslocam de livre arbítrio.

Durante a simulação destes cenários foram utilizados os seguintes parâmetros:

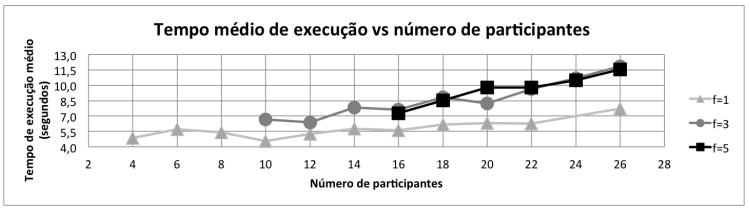
– **grelha**: os participantes estão dispersos em grelha com intervalos de 0,3 metros na horizontal e 0,5 metros na vertical. Os participantes não apresentam mobilidade.
– **aleatório**: os participantes são colocados em posições aleatórias numa área de 250 metros quadrados. A velocidade dos participantes varia entre 0 a 5 metros por segundos, podendo parar 2 segundos no máximo. O modelo de mobilidade dos participantes escolhido foi o *Random Waypoint*.

As próximas subseções expliquem as parametrizações das simulações e análises dos resultados obtidos.

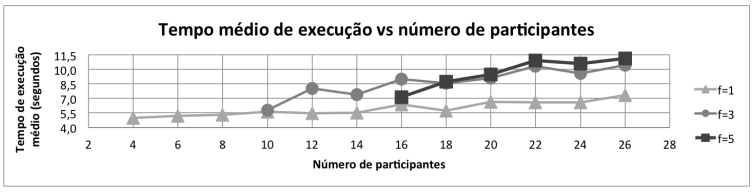
5.1 Simulações da camada de Suporte

As simulações da camada de Suporte tinham como objetivo estudar o desempenho da camada quando a densidade de participantes e os participantes bizantinos aumentam e comparar os resultados em diferentes cenários. Os dados recolhidos foram o tempo médio e o desvio de execução da camada de Suporte.

Os resultados das Figuras 4b e 4a apresentam semelhanças interessantes de tempos médios de execução, concluindo que a mobilidade dos participantes não altera impacto consideráveis no desempenho da camada. Estes resultados comprovam que há benefícios na utilização da difusão local na camada Suporte.



(a) Simulação no cenário grelha

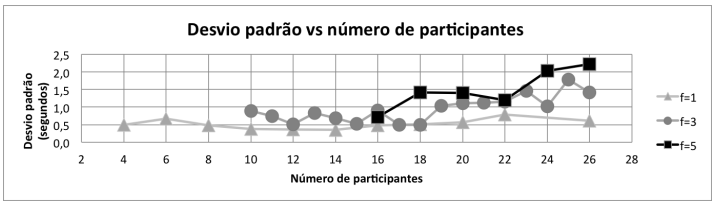


(b) Simulação no cenário aleatório

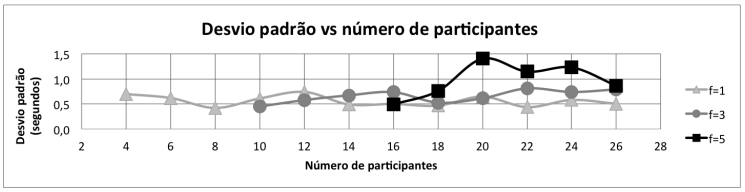
Figura 4: Tempo médio de execução da camada de Suporte (segundos)

Comparando os desvios padrões dos tempos de execução das Figuras 5b e 5b, existe um aumento no desvio padrão no cenário aleatório em simulações com mais de 20 participantes. O desvio padrão poderá ser um fator prejudicial ao desempenho de protocolos dependentes da camada de Suporte porque agrava o assincronismo. Uma solução para diminuir assincronismo é definir uma regra em

que os protocolos, por omissão, guardam mensagens recebidas mesmo que não estejam prontos a responder ou computá-las.



(a) Simulação no cenário grelha



(b) Simulação no cenário aleatório

Figura 5: Desvio padrão de execução da camada de Suporte (segundos)

5.2 Simulações da camada de Acordo

As simulações ao protocolo de acordo tinham como objetivo estudar o desempenho do algoritmo quando a densidade de participantes aumenta e comparar os dados obtidos quando os dispositivos estão dispostos em grelha e quando estão dispostos aleatoriamente. Os dados recolhidos foram o número de fases e o tempo estimado para a conclusão do acordo.

Foram executadas simulações num cenário em que todos os processos são correctos e em outro cenário em que f processos assumem um comportamento bizantino (i.e. votam num valor diferente do que é suposto). Nas simulações em que se recolheram o número de fases na execução do algoritmo os processos votam em valores divergentes (metade dos processos votam em 1 e a outra metade em 0).

Como se pode observar no gráfico da Figura 7, na presença de participantes bizantinos o algoritmo perde um pouco de desempenho, mas no geral, apresenta um resultado aceitável. Observou-se em algumas execuções sem participantes bizantinas um pior desempenho do que com participantes. Isto deve-se ao facto de as mensagens bizantinas serem descartadas, facilitando a convergência dos participantes para o valor decidido.

Quando os dispositivos estão dispostos aleatoriamente e com mobilidade os resultados não variam muito da disposição em grelha (ver Figuras 6 e 7). Existem alguns picos que se devem à perda de mensagens causada pela densidade de nós. Neste cenário é visível que em muitos casos os participantes bizantinos aceleram, ao invés de atrasar, a execução do algoritmo de consenso. Isto deve-se ao facto

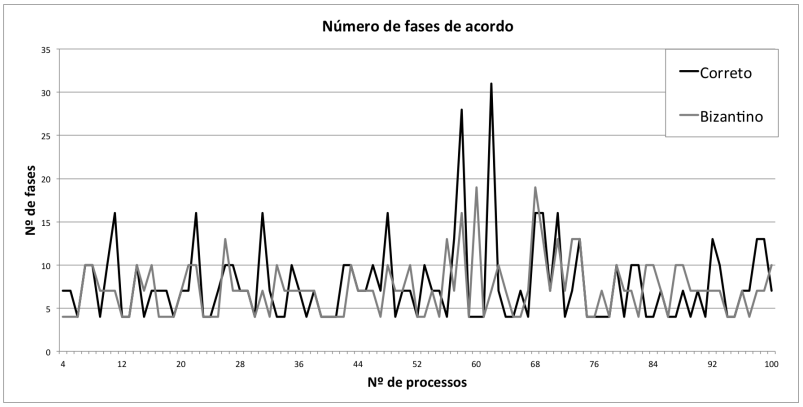


Figura 6: Número de fases simulando ambiente com disposição aleatória

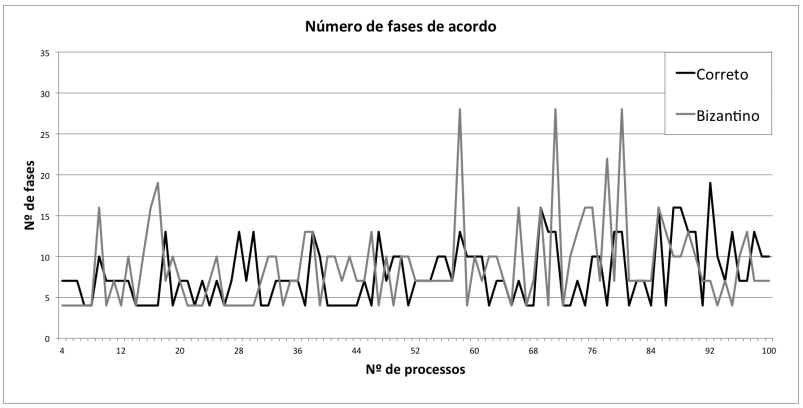


Figura 7: Número de fases simulando ambiente com disposição em grelha

das mensagens serem descartadas na validação o que permite lidar com menos mensagens em cada fase ($n - f$ ao invés de n).

Outro resultado interessante das simulações é que em todos os casos $n - f$ participantes chegaram a acordo, em nenhum caso apenas k participantes terminaram, o que não significa necessariamente que tal não aconteça, mas sim que é extremamente raro.

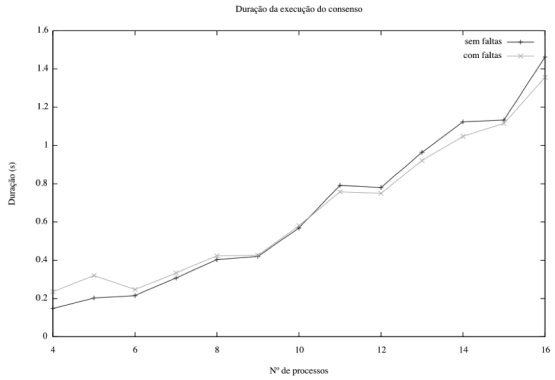


Figura 8: Duração(s) da execução do consenso disposição aleatória

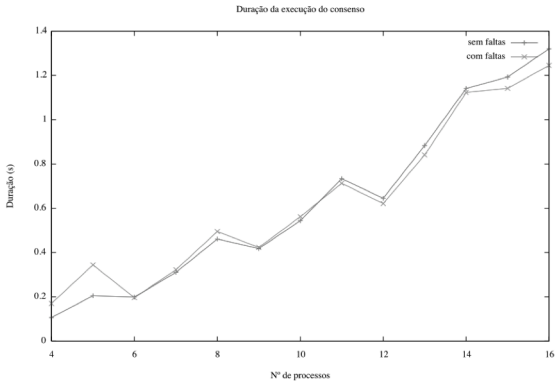


Figura 9: Duração(s) da execução do consenso disposição em grelha

As figuras 8 e 9 apresentam a duração média de dez execuções do protocolo de acordo em ambos os cenários testados e assumindo a presença (ou não) de participantes bizantinos. Como se pode observar pelos resultados, a presença de participantes bizantinos não afecta o desempenho do protocolo. A disposição dos processos também não afecta o desempenho do algoritmo. Os resultados obtidos revelam que o algoritmo de acordo escolhido é o mais indicado para este tipo de redes pois apresenta resultados satisfatórios neste ambiente.

6 Conclusões e Trabalho Futuro

O artigo estuda os algoritmos a serem usados na pilha de protocolos e apresenta os resultados das simulações das modificações propostas. A camada de Suporte e de Acordo em aplicações móveis que necessitam de acordo *seguro* durante uma

computação distribuída, um exemplo útil será acordar a terminação de um protocolo de votação *seguro* das melhores apresentações durante uma conferência. As simulações demonstram que é possível obter uma decisão em 8 segundos, pelo menos, num cenário com 16 participantes móveis e desconhecidos num recinto fechado com a presença de $f = 5$ participantes bizantinos. Este resultado demonstra a potencialidade da pilha protocolar do artigo sobre *MANETs*, motivando uma continuar da investigação nesta área.

Futuramente, pretendemos estender a camada de Acordo com protocolos de acordo multi-valor e o acordo de vectores. E também pretendemos concretizar a pilha protocolar MICCS4Mobile e um protocolo de encaminhamento seguro de MANETs para dispositivos com sistema operativo *Android*.

Agradecimentos. Este trabalho foi parcialmente suportado pela CE através do projeto FP7-257475 (MASSIF), pela FCT através do programa Multianual (LaSIGE) e pelo projeto PTDC/EIA-EIA/113729/2009 (SITAN).

Referências

1. Slideshark: Slideshark (June 2013)
2. David Cavin, Y.S.: Reaching Agreement with Unknown Participants in Mobile Self-Organized Networks in Spite of Process Crashes. Technical Report IC/2005/026 (2005)
3. Greve, F., Tixeuil, S.: Knowledge Connectivity vs. Synchrony Requirements for Fault-Tolerant Agreement in Unknown Networks. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), IEEE (June 2007) 82–91
4. Alchieri, E.A., Bessani, A.N., Silva Fraga, J., Greve, F.: Principles of Distributed Systems. Volume 5401 of Lecture Notes in Computer Science., Berlin, Heidelberg (December 2008)
5. Lamport, L., Shostak, R., Pease, M.: The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems **4**(3) (July 1982) 382–401
6. Moniz, H., Neves, N.F., Correia, M.: Byzantine Fault-tolerant Consensus in Wireless Ad hoc Networks. IEEE Transactions on Mobile Computing (November 2012) 1–1
7. De Prisco, R., Malkhi, D., Reiter, M.K.: On k-set consensus problems in asynchronous systems. In: Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing. PODC '99, New York, NY, USA, ACM (1999) 257–265
8. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. Journal of the ACM **32**(2) (April 1985) 374–382
9. N. Santoro, P.W.: Time is not a healer. Lecture Notes in Computer Science **349** (1989) Volume 349, pp 304–313
10. Douceur, J.R.: The Sybil Attack. Proceedings of the 1st International Workshop on Peer-to-Peer Systems (March 2002) 251–260
11. Yu, M.Y.M., Zhou, M.Z.M., Su, W.S.W.: A Secure Routing Protocol Against Byzantine Attacks for MANETs in Adversarial Environments (2009)

12. Weingartner, E., vom Lehn, H., Wehrle, K.: A performance comparison of recent network simulators. In: Communications, 2009. ICC '09. IEEE International Conference on. (2009) 1–5