UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

DISSEMINAÇÃO DE DADOS EM REDES DE SENSORES

Ricardo Samuel da Silva Mascarenhas

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2010

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

DISSEMINAÇÃO DE DADOS EM REDES DE SENSORES

Ricardo Samuel da Silva Mascarenhas

DISSERTAÇÃO

Projecto orientado pelo Prof. Doutor Hugo Alexandre Tavares Miranda

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2010

Agradecimentos

Durante o decorrer da minha formação, muitas foram as pessoas que contribuíram e me apoiaram das mais diversas formas. Cada uma à sua maneira contribuiu com um pouco de si.

Em primeiro expresso os meus mais sinceros agradecimentos ao Prof. Doutor Hugo Miranda pela orientação, apoio, disponibilidade e dedicação demonstrada durante o decorrer da concretização deste trabalho. Estes factores proporcionaram agradáveis diálogos, donde resultaram ideias que contribuíram para enriquecer a concretização deste trabalho.

Aos meus colegas e amigos com quem partilhei diversas discussões interessantes, muito trabalho, alguns momentos de lazer e acima de tudo, muito companheirismo. Sem uma ordem especial um grande obrigado para alguns dos colegas e amigos com quem tive o prazer de trabalhar e aprender com as suas virtudes e aptidões: Filipe Rodrigues, João Craveiro, Marco Lourenço, Daniel Policarpo e Tiago Martins.

Aos docentes que me acompanharam na longa jornada que uma formação implica, expresso a minha gratidão.

Ao Departamento de Informática e ao LaSIGE, uma palavra de apreço pelas condições de trabalho proporcionadas.

Finalmente, mas não em último lugar, um agradecimento muito especial ao meu pai Eugénio, à minha mãe Manuela, às minhas irmãs Ana e Cátia e à minha namorada Ana, que privados de muitas horas do meu convívio sempre tiveram uma palavra de incentivo e apoio, sendo seguro dizer que foram a minha fonte de inspiração.

This work has been partially supported by FCT project REDICO (PTDC/EIA/71752/2006) through POSI and FEDER.

À minha família

Resumo

As redes de sensores sem fios (RSSF) são compostas por um grande número de pequenos dispositivos que monitorizam o ambiente em que estão inseridas. Estes dispositivos são caracterizados pelas restrições a nível energético, poder de processamento e de memória. As fortes restrições a que os dispositivos estão sujeitos obrigam à utilização de paradigmas de comunicação específicos, que tenham em consideração uma utilização racional dos recursos. Com a especialização dos nós receptores surge a necessidade destes conseguirem expressar o interesse em informação que considerem relevante, independentemente dos restantes participantes. O paradigma de comunicação usado tipicamente nas RSSF não considera os interesses que os nós receptores têm na informação que está a ser transmitida, pelo que pode ser enviado um grande número de mensagens nas quais nenhum nó receptor está interessado. O envio de mensagens desnecessárias é particularmente penalizador nas RSSF, uma vez que a actividade do rádio é responsável pelo consumo de grande parte da energia que os dispositivos dispõem. Sendo a memória um recurso escasso, importa também que o paradigma de comunicação seja igualmente eficiente em termos da quantidade de memória utilizada.

Este trabalho apresenta e avalia experimentalmente um sistema Publicador/Subscritor que tem em conta o interesse dos nós em determinados tipos de informação e suprime o envio de informação sem relevância. Para além disso, o sistema adapta-se às restrições de energia e de memória dos dispositivos.

Adicionalmente é apresentada uma concretização dum algoritmo de disseminação para o sistema operativo TinyOS que sugere que o sistema apresentado é exequível em sensores reais.

Palavras-chave: Disseminação, Publicador/Subscritor, Redes de Sensores Sem Fios, Subscrições Geograficamente Distribuídas

Abstract

Wireless sensor networks (WSN) are composed of a large number of small devices which monitor the environment in which they operate. These devices are characterized by restrictions like energy, processing and memory. The strong restrictions on the devices enforce the use of paradigms of communication which take into account a rational use of resources. With the specialization of the receptor nodes arises the need for them to be able express interest in information they deem relevant, regardless the interests of other participants. The paradigm of communication typically used in WSNs does not consider the receivers' interests in information being transmitted, so it is possible to send a large number of messages in which no receiver node is interested. Sending unnecessary messages is particularly expensive in WSNs, since the activity of the radio is responsible for consuming much of the power the devices have. Since memory is a scarce resource, the paradigm of communication must also be efficient in the amount of memory used.

This work presents and experimentally evaluates a Publisher/Subscriber system which takes into account the interest of the nodes in certain types of information and suppresses the sending of information without relevance. In addition, the system adapts to the resource constraints of devices such as reduced energy capacity and scarce memory space.

Additionally, an implementation of an dissemination algorithm to the TinyOS operating system is presented. This implementation suggests that the presented system is feasible in real sensors.

Keywords: Dissemination, Publish/Subscribe, Wireless Sensor Networks, Geographically Distributed Subscriptions

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

As redes sem fios são hoje em dia um importante meio de comunicação, e a sua utilidade aumenta a cada dia em número e em cenários de utilização. As redes sem fios típicas baseiam-se numa infra-estrutura que fornece as funções de gestão necessárias à comunicação, como por exemplo, controlo de acesso ao meio e encaminhamento. Uma infra-estrutura pode ser composta por um ou vários nós tipicamente denominados por pontos de acesso. Nestas redes, para os participantes comunicarem entre si, é necessário que cada participante estabeleça uma ligação com pelo menos um dos nós que compõem a infra-estrutura. Apesar destas redes serem usadas em inúmeras aplicações práticas, existem cenários em que as redes que se baseiam numa infra-estrutura não são a solução mais viável dado que por vezes se verifica a impossibilidade de proceder à instalação da mesma. Alguns factores que condicionam viabilidade da aplicação destas redes são:

Planeamento Instalar os nós que compõem a infra-estrutura de forma a terem o desempenho e cobertura esperada requer um estudo do terreno onde a rede vai ser instalada que, permite saber quais os melhores locais para a colocação dos nós que irão formar a infra-estrutura. Por um lado, os locais escolhidos para a colocação destes nós devem ter em conta a possibilidade de furto. Por outro, o número de ligações, a cobertura, velocidade e nível de sinal são também factores a ter em conta no planeamento da instalação da infra-estrutura, os quais nem sempre podem ser garantidos. A instalação de uma infra-estrutura em locais hostis, remotos, ou em cenários de emergência são alguns dos casos típicos onde é difícil garantir os factores mencionados em cima.

Tempo O planeamento é necessário à instalação de uma boa infra-estrutura de suporte. Porém as tarefas de planeamento e instalação da infra-estrutura são demasiado morosas. Em casos de emergência urge a necessidade da existência de uma rede funcional para auxiliar as equipas de emergência, tarefa que não é

possível realizar em tempo útil se a rede for infra-estruturada.

Tolerância a faltas As infra-estruturas que são compostas por um único ponto de acesso têm como desvantagem um ponto único de falha. Por exemplo, na presença de uma avaria ou de interferências externas este deixa de fornecer os serviços necessários à comunicação. Para fornecer um nível de tolerância a faltas mais elevado seria necessário um maior investimento monetário e temporal para instalação de mais pontos de acesso.

Custo Todos os factores referidos anteriormente têm um custo associado. Porém, há um conjunto de custos que não estão contabilizados nos pontos anteriores, como é o caso do preço da matéria prima (ex. cabos e hardware) e da mão de obra.

As adversidades descritas em cima alertam para a necessidade de estabelecer redes espontâneas, que são caracterizadas pelos dispositivos cooperantes que fornecem os serviços essenciais à comunicação em detrimento da necessidade de instalação de uma infra-estrutura. Assim sendo, as adversidades em cima descritas tendem a desvanecer-se ou mesmo desaparecer. Em particular, as redes que operam em grandes áreas e que são compostas por um elevado número de dispositivos beneficiam do estabelecimento de redes espontâneas, uma vez que a instalação de uma infra-estrutura é uma tarefa complexa e morosa, especialmente para redes de grande dimensão.

Com a evolução da tecnologia e consequente diminuição do tamanho dos dispositivos, as redes de sensores sem fios (RSSF) tornaram-se extremamente apelativas. As RSSF não necessitam de uma infra-estrutura de suporte à comunicação, e são tipicamente compostas por dispositivos de pequena dimensão que dispõem de uma unidade de processamento, memória, uma interface de comunicação sem fios e uma fonte de energia. De modo implícito, a cooperação entre os dispositivos forma uma infra-estrutura, que possibilita a formação de RSSF espontâneas.

O objectivo primário das RSSF é a recolha e transmissão de dados recolhidos da área de observação, ou seja, monitorização do meio em que estão inseridas. Neste tipo de redes existem essencialmente duas entidades, os nós sensores e os nós *sink*. Os nós sensores estão equipados com tecnologia que lhes permite obter informação sobre o ambiente em que estão inseridos (ex. temperatura, humidade, pressão, movimento). Os sensores estão tipicamente dispersos pela área de observação onde recolhem dados, e após algum processamento, os enviam para um ou mais nós *sink*, que se encarregam de fazer chegar a informação ao

utilizador final.

A utilização de RSSF é apropriada para um vasto número de aplicações, como por exemplo, monitorização de habitats, agricultura de precisão, monitorização de edifícios e aplicações militares [1, 2, 3].

O nós sensores são caracterizados pelo limitado poder de processamento e espaço de armazenamento exíguo, bem como a fonte de energia que é tipicamente uma bateria de capacidade reduzida. Em muitos cenários de aplicação os sensores podem não estar facilmente acessíveis devido aos locais onde operam, ou devido ao seu grande número. Em qualquer um dos casos, a substituição de baterias pode ser impraticável, o que sugere que o dispêndio energético deve ser tão reduzido quanto possível de forma a prolongar o tempo de vida da rede. Para poupar energia, alguns dispositivos desligam-se periodicamente.

Os nós *sink* dispõem normalmente de recursos adequados às suas necessidades, nomeadamente, maior poder de processamento e uma maior capacidade energética.

A interface de rede sem fios permite aos sensores enviar informação directamente para os nós que se encontrem dentro do seu raio de transmissão. A entrega de mensagens a nós mais distantes e cuja localização é potencialmente desconhecida é conseguida por retransmissões sucessivas, realizadas por diferentes dispositivos. A descoberta de uma sequência de nós entre a origem e o destino da mensagem, bem como o envio das mensagens ponto-a-ponto é realizado por protocolos de encaminhamento para redes não infra-estruturadas. A sequência de nós utilizada para enviar as mensagens desde o nó emissor até ao nó destinatário é denominada por rota. Alguns nós podem falhar por exemplo, devido ao enfraquecimento da bateria ou devido a eventos destrutivos. Nestes casos é responsabilidade dos restantes nós da rede assegurarem os serviços de comunicação.

Os nós sensores são programados em tempo de produção para utilizar a sua limitada capacidade de processamento na avaliação da relevância da informação obtida, ou seja, se a sua retransmissão justifica o consumo de energia necessário para a entrega ao nó *sink* mais próximo. A avaliação da informação considerada como relevante é tipicamente imparcial, ou seja, é seguido o mesmo critério para toda a informação independentemente do nó receptor. Com o aumento do número de sensores, aumenta também o volume e diversidade de informação recolhida. Adicionalmente, abre-se caminho à especialização dos nós *sink*, que poderão seleccionar, de forma independente dos restantes, os tipos de informação que consideram relevantes, o que sugere que o modelo de comunicação típico não é adequado. Neste cenário os sensores passam a ter que identificar correctamente o(s) *sink(s)* a quem a informação deve ser entregue. Impõe-se por isso

a necessidade de definição de novos modelos de comunicação, que contemplem adequadamente casos em que nenhum nó *sink* esteja interessado em algumas das amostras. A incapacidade de incorporar a selecção de informação de interesse no modelo de comunicação pode levar ao envio desnecessário de um elevado número de mensagens, que implicam um gasto energético não negligenciável e consequente diminuição de longevidade da rede [4]. Estudos apresentados em [5, 6] mostram que o envio de dados é mais dispendioso em termos de recursos energéticos do que o processamento de dados. Em particular, o consumo de energia resultante do envio de um único bit é aproximadamente o mesmo que o consumido no processamento de 1000 operações num nó sensor comum. Assim, surge a necessidade de tomar medidas para a redução do consumo de energia que vão para além dos dispositivos se desligarem periodicamente, e que passam pela redução do número de mensagens transmitidas.

1.1 Paradigma de comunicação Publicador/Subscritor

O Publicador/Subscritor [7] (Pub/Sub) é um paradigma de comunicação que fornece as funcionalidades necessárias à especificação de interesses em determinados tipos de informação. Alguns cenários de aplicação do modelo de comunicação Pub/Sub às RSSF são:

Equipas de salvamento Em casos de emergência como, desabamentos, maremotos, erupções, inundações ou tempestades, onde é necessária uma acção imediata das equipas de salvamento a dispensa de infra-estrutura é uma mais valia para a formação espontânea de redes. As equipas de salvamento como bombeiros e médicos expressam o seu interesse em informação relevante para o cumprimento eficiente da respectiva função. Os bombeiros podem expressar o interesse em informação relativa a temperatura (ex. para detecção de incêndios e/ou risco de explosões), qualidade do ar (ex. de forma a detectar agentes nocivos à saúde) e movimento (ex. pessoas soterradas). O pessoal médico pode expressar o interesse em informação relacionada com sinais vitais para detectar eventuais feridos.

Monitorização de eventos naturais Supondo uma rede global de sensores, onde existem sensores a monitorizar vulcões, encostas com neve, icebergues, oceanos, terrenos propícios a terramotos, entre outros, diversas instituições poderiam expressar o seu interesse em informação de acordo com a sua área de acção.

Contexto militar No campo militar as funções de cada equipa são bem delineadas e definidas, onde diferentes equipas desempenham tarefas especificas,

pelo que podem expressar o interesse em informação que sirva melhor os seus propósitos. Por exemplo, as equipas médicas podem expressar o interesse em informação relativa à localização e estado das tropas, as equipas de intervenção podem expressar o interesse em informação relativa à detecção de tropas inimigas ou veículos, e todas as equipas podem expressar o interesse em informação que permite detecção de ataques biológicos.

Num sistema Pub/Sub, os *publicadores* produzem informação através de "publicações". A informação é produzida em forma de *eventos*, unidades de informação não endereçadas que são entregues ao sistema. Os *subscritores*, por sua vez, expressam perante o sistema o seu interesse em publicações com determinadas características através de "subscrições" e esperam ser notificados sempre que seja publicada informação que satisfaça o interesse expressado. O envio da informação relevante aos subscritores é tipicamente denotada pelo termo *evento*, e o acto de entrega pelo termo *notificação*.

Cabe à operação de *correspondência* verificar se uma determinada publicação p satisfaz o interesse expressado numa subscrição s. Dizemos que uma publicação p satisfaz o interesse expressado na subscrição s se e só se satisfizer todas as restrições declaradas em s. Após a execução da operação de correspondência, o sistema Pub/Sub é responsável por entregar os eventos aos subscritores que neles estejam interessados.

Existem essencialmente duas abordagens para a concretização de sistemas Pub/Sub. Uma baseia-se na concretização centralizada que delega algumas das tarefas do sistema em nós especiais bem conhecidos. A outra abordagem, denominada por concretização descentralizada, não assume a existência de nós especiais e portanto, todos os participantes partilham as responsabilidades de executar as tarefas necessárias. As concretizações descentralizadas de sistemas Pub/-Sub para redes sem fios poderão ser posicionadas entre dois extremos. Um consiste na inundação das subscrições, onde todos os nós guardam e retransmitem as subscrições recebidas pela primeira vez. Deste modo, as subscrições chegam a todos nós da rede permitindo que os publicadores possam determinar localmente quais os subscritores a quem os eventos deverão ser entregues. O outro consiste na inundação dos eventos, delegando nos nós a responsabilidade de retransmitirem os eventos e de verificarem se o evento satisfaz as suas subscrições. Contudo, nenhuma das soluções se adequa às restrições impostas pelos dispositivos que compõem as RSSF uma vez que, por um lado o número de mensagens requeridas pela inundação de eventos é elevado, e cada transmissão/recepção efectuada tem um custo energético não negligenciável associado [4]. Por outro, a inundação de subscrições determina que todos os dispositivos devem guardar

uma cópia de cada subscrição, o que não concorda com as restrições de memória dos dispositivos.

A concretização de sistemas Pub/Sub em RSSF é particularmente desafiante devido às restrições de recursos impostas pelos sensores. Neste sentido importa encontrar soluções que reduzam simultaneamente o número de operações de transmissão a um mínimo de forma a reduzir o gasto energético, e a memória utilizada pelo paradigma de comunicação.

1.2 Contribuições

As contribuições deste trabalho de investigação incidem sobre sistemas Pub/Sub para RSSF. Em particular, a investigação descrita neste trabalho é direccionada ao estudo de estratégias de concretização de sistemas Pub/Sub que permitem minimizar simultaneamente o consumo de bateria e a quantidade de memória utilizada pelos dispositivos. Embora a redução de utilização destes dois recursos seja o objectivo, este é atingido através de uma solução de compromisso que consiste na replicação parcial da informação relativa à subscrições. A importância atribuída à minimização da quantidade de recursos utilizados deve-se ao consequente aumento do tempo de vida da rede. Os resultados obtidos através de simulações permitem afirmar que o sistema Pub/Sub proposto reduz efectivamente a quantidade de recursos utilizados, pelo que contribui para o aumento do tempo de vida da rede.

Adicionalmente é apresentada uma implementação para sensores de um algoritmo de disseminação de dados. A concretização foi levada a cabo para o sistema operativo TinyOS com o intuito de verificar parcialmente a exequibilidade do sistema Pub/Sub proposto em sensores reais.

1.3 Estrutura do documento

O relatório está estruturado da forma descrita em seguida. O Cap. 2 enquadra o trabalho relacionado, onde são detalhados os pormenores relativos ao paradigma de comunicação Pub/Sub, alguns sistemas Pub/Sub e respectivas abordagens. As ferramentas necessárias à concretização do sistema são expostas no Cap. 3. No Cap. 4 são discutidas diferentes alternativas para a concretização de sistemas Pub/Sub. Neste capítulo é também apresentado um novo sistema Pub/Sub para RSSF. O Cap. 5 apresenta uma implementação do algoritmo de disseminação de dados PAMPA para o sistema operativo de sensores TinyOS. No Cap. 6 são dados a conhecer os resultados dos testes relativos à concretização do algoritmo PAMPA em sensores reais. São também dados a conhecer os resultados obtidos

através de simulação do sistema Pub/Sub proposto, em comparação com uma abordagem onde os eventos são disseminados para a rede através do algoritmo PAMPA. Finalmente o Cap. 7 conclui o trabalho apresentando as conclusões mais importantes do trabalho realizado e uma discussão do trabalho futuro.

1.4 Publicações

O artigo "Um Sistema Publicador/subscritor com Subscrições Geograficamente Distribuídas para RSSFs" que resume este trabalho, foi aceite para publicação no Simpósio de Informática, INForum 2010.

Capítulo 2

Trabalho relacionado

Neste capítulo são abordados alguns aspectos do paradigma de comunicação Pub/Sub que fornecem um contexto mais abrangente e que permitem compreender melhor o funcionamento dos sistemas bem como a importância e impacto de algumas das decisões tomadas neste trabalho. Adicionalmente, este capítulo contém uma visão global do trabalho já realizado em sistemas Pub/Sub em diversos contextos, e é discutido a razão pela qual os trabalhos já realizados não podem ser aplicados às RSSF. O capítulo está organizado do modo descrito em seguida. Nas Sec. 2.1 e 2.2 é fornecido algum contexto adicional sobre sistemas Pub/Sub. De seguida nas Sec. 2.3 e 2.4 são apresentados alguns sistemas Pub/-Sub no contexto das redes com fios tradicionais e nas redes sem infra-estrutura, respectivamente. Finalmente na Sec. 2.5 é apresentado um resumo do trabalho relacionado.

2.1 Estratégias para concretização de sistemas Pub/-Sub

Existem diversas estratégias de concretização de sistemas Pub/Sub [7]. Alguns sistemas centralizam a correspondência entre as publicações e os subscritores em nós bem conhecidos, denominados mediadores (*brokers*) [8, 9, 10]. Os *brokers* estão incumbidos de agregar informação sobre as subscrições dos nós participantes, de efectuar a operação de correspondência, e enviar os eventos para os nós interessados. Os *brokers* têm normalmente recursos apropriados às suas funções e atenuam o gasto de recursos nos nós clientes, em particular, a memória uma vez que não têm que armazenar informação relativas a subscrições, o processamento dado que não necessitam de efectuar a operação de correspondência, e a bateria porque o esforço do envio dos eventos é suportado pelos *brokers*.

A arquitectura dos *brokers* pode ser centralizada, caso em que existe apenas um nó responsável por tratar de toda a correspondência [11, 12], ou distribuída,

onde o esforço é distribuído pelos diversos *brokers*. De forma a aumentar a eficiência no envio dos eventos para os nós subscritores, muitos sistemas [8, 9, 10] optam por dispor os *brokers* numa topologia em forma de árvore. A discussão levada a cabo em [13] mostra que a estrutura em forma de árvore simplifica a tarefa de encaminhamento e permite realizar optimizações que têm como objectivo diminuir o número de mensagens. Apesar de simplificarem consideravelmente a complexidade do sistema, a forte dependência num conjunto limitado de dispositivos, e os recursos computacionais e energéticos requeridos tornam os *brokers* inadequados para os cenários de aplicação das RSSF.

Os sistemas descentralizados dispensam os serviços fornecidos pelos *brokers* pelo que delegam todas as operações nos nós participantes. A descentralização do sistema Pub/Sub é uma mais valia, no sentido em que se adapta à formação de RSSF espontâneas, mas apresenta um conjunto de desafios acrescidos que resultam do aumento da complexidade resultante da distribuição. No caso das RSSF é importante encontrar soluções que requeiram simultaneamente baixa utilização da memória e a transmissão de um número limitado de mensagens por todos os participantes.

2.2 Modelos de subscrição

Uma das características distintivas dos diferentes sistemas Pub/Sub existentes é a expressividade com que um subscritor consegue indicar os seus interesses.

No modelo baseado em tópicos (por exemplo, [14, 15]) quer as publicações quer as subscrições são identificadas por um assunto. Os subscritores são notificados sempre que o assunto da informação publicada satisfaça o assunto especificado nas subscrições. Este modelo é facilmente projectado num modelo de difusão selectiva (*multicast*), com cada tópico a corresponder a um endereço distinto e tem como principal desvantagem a fraca expressividade que oferece aos subscritores.

O modelo baseado em conteúdos oferece uma maior expressividade já que permite a utilização de operadores sobre os atributos das subscrições e não apenas sobre o tópico. Neste modelo assume-se que um subscritor está interessado numa publicação se e só se todas as restrições declaradas nos atributos da subscrição são satisfeitas. Este modelo fornece aos subscritores um maior nível de refinamento dos seus interesses. Por exemplo, uma aplicação para detecção de incêndios estaria exclusivamente interessada em publicações registando temperatura superior a 60° e humidade inferior a 70%. Este modelo requer processamento adicional em relação ao modelo baseado em tópicos uma vez que as operações de correspondência são mais complexas [16]. Alguns sistemas que

usam este modelo estão descritos em [9, 10, 6, 17].

No modelo baseado em tipos [18], os eventos são objectos que podem conter atributos e/ou métodos. O nível de expressividade deste modelo fica entre o nível de expressividade oferecido pelos modelos baseados em tópicos e baseados em conteúdos [7]. O modelo baseado em tópicos pode ser emulado, subscrevendo apenas o tipo do objecto, enquanto que o modelo baseado em conteúdos pode ser emulado expressando interesses sobre os atributos do objecto.

Alguns estudos [19, 20] descrevem sistemas Pub/Sub que se baseiam em documentos XML, e que por isso oferecem um modelo de dados semi-estruturado permitindo o uso de hierarquias. Por esta razão, este modelo permite uma maior flexibilidade em relação ao modelo baseado em conteúdos [7]. Este modelo apresenta como vantagem o facto de ser facilmente extensível e independente da implementação. Contudo, os algoritmos de correspondência requerem um maior processamento, mesmo em relação ao modelo baseado em conteúdos, pelo que não se adequa às RSSF.

Os modelos baseados em localização [21, 22] são tipicamente usados em ambientes móveis, e permitem exprimir interesses de forma a que os nós sejam notificados quando estão próximos de um determinado local, ou serviço. Este modelo não é adequado ao contexto deste trabalho uma vez que para o seu funcionamento seria necessário um mecanismo de localização (ex. GPS) que tende a consumir um quantidade de recursos energéticos não negligenciáveis.

2.3 Sistemas Pub/Sub em redes com fios

A área dos sistemas Pub/Sub mais estudada é a que envolve as redes tradicionais (redes com fios). Os sistemas TIB/RV [23], iBUS [24], SCRIBE [14] e Bayeux [15] são alguns exemplos que usam o modelo de subscrição baseado em tópicos.

O modelo de subscrição baseado em conteúdos é o mais popular dada a sua maior expressividade, sendo usado em sistemas de referência como por exemplo, SIENA [9], JEDI [10], LeSubscribe [6], Ready [25], Hermes [17] e Elvin [26].

A lista de sistemas existentes para as redes tradicionais é extensa, porém, como não adequados para as RSSF serão descritos com mais detalhe apenas três dos sistemas Pub/Sub mencionados em cima.

SIENA

O SIENA [9] é um sistema Pub/Sub baseado em conteúdos que se foca num balanço entre a escalabilidade e a expressividade. Uma rede de servidores (*brokers*) dispostos numa topologia de árvore é responsável por receber e retransmitir

eventos. Os nós podem participar no papel de subscritores ou no de publicadores. Os publicadores devem proceder ao envio de mensagens *advertisement*, que informam o sistema sobre o tipo de informação que o publicador consegue produzir. Baseado nestas mensagens *advertisement* e nas subscrições os servidores conseguem optimizar o encaminhamento dos eventos que é baseado no modelo baseado em *filtros* [27]. Pelo facto dos servidores conterem a informação sobre as subscrições e serem responsáveis por enviar os eventos, estes podem decidir não enviar eventos se não existirem nós interessados na informação.

Contudo, o envio das mensagens *advertise* no SIENA baseia-se numa operação de disseminação global, ou seja, uma disseminação através de toda a rede, que limita a escalabilidade do sistema e aumenta o número de mensagens necessárias. Para além disto, as subscrições são enviadas para todos os *brokers*, sendo que todos guardam a mesma informação. Por estas razões o SIENA não é adequado para as RSSF.

Hermes

O sistema Pub/Sub Hermes [17] foi desenvolvido na Universidade de Cambridge e tem como principal motivação a escalabilidade de tal modo que pode ser aplicado no contexto da Internet, porém, é genérico o suficiente para suportar aplicações que operam noutros cenários. Este sistema utiliza o modelo de subscrição baseado em tipos. Antes de um nó publicador poder publicar informação, deve enviar mensagens *advertise* que informam o sistema sobre o tipo de informação que este produz. Os nós participantes (publicadores ou subscritores) devem estabelecer uma ligação com um dos vários *brokers* que efectuam as tarefas de encaminhamento. Os *brokers* são baseados no modelo *Rendezvous* [28] que gere as mensagens *advertise* e de publicação. Este modelo permite que cada *broker* armazene subscrições apenas de um tipo. Deste modo, e contrariamente ao SIENA, existe uma única cópia da subscrição no sistema.

Embora este sistema diminua consideravelmente a memória utilizada para guardar subscrições, recorre ao uso de *brokers*, pelo que não é adequado para as RSSF.

JEDI

O JEDI (Java Event-based Distributed Infrastructure) [10] foi desenvolvido no Politécnico de Milão e Cefriel e foca-se na escalabilidade e na gestão da mobilidade dos clientes. À semelhança dos sistemas descritos anteriormente, este usa também um conjunto de *brokers* dispostos em forma de árvore, que coope-

ram entre si coleccionando as subscrições efectuadas pelos participantes e são também responsáveis por enviar os eventos aos respectivos nós interessados na informação. Cada um dos clientes deve estabelecer uma ligação a qualquer um dos diversos *brokers*, e usa-o para realizar as subscrições, publicações e inclusive para receber os eventos. Os *brokers* simplificam a tarefa de encaminhamento, em particular, cada um mantém uma tabela de encaminhamento com rotas para os eventuais nós interessados em cada uma das publicações. Contudo, a formação da topologia em forma de árvore é efectuada após a eleição de um líder de grupo. Cada líder deve anunciar a sua presença para que todos os *brokers* conheçam os lideres de grupo, sendo que este processo dificulta a escalabilidade do sistema.

Para suportar a mobilidade dos clientes, o JEDI suporta de forma nativa a desconexão e o estabelecimento de uma nova ligação, contudo os nós são responsáveis por avisar o sistema da sua intenção de desconexão e posteriormente, na sua nova localização, devem informar explicitamente o sistema da intenção de participar novamente na comunicação. Durante a fase de desconexão os nós participantes são livres de se mover e de se ligarem a outros *brokers*, que mantêm armazenadas temporariamente todas as mensagens que circularam pela rede durante o período de desconexão. O JEDI garante a entrega das mensagens numa sequência que respeita a ordenação causal.

Este sistema não se adequa ao modelo das RSSF uma vez que também faz uso de um conjunto de *brokers*.

2.4 Sistemas Pub/Sub para redes não infra-estruturadas

Nesta secção são descritos alguns sistemas Pub/Sub para redes não infra-estruturadas. Depois de apresentado um sistema Pub/Sub para redes Mesh na parte inicial da secção, são referidos alguns dos sistemas Pub/Sub utilizados em MANETs. No final da secção são apresentados dois sistemas Pub/Sub para RSSF.

Redes Mesh

As redes Mesh são compostas por uma mistura de nós fixos e de nós móveis. Os nós fixos denominados por *routers mesh* não têm restrições de energia e fornecem uma infra-estrutura central de suporte à comunicação. Adicionalmente, os *routers mesh* facilitam o acesso a outras redes como por exemplo a Internet. Os clientes, normalmente compostos por pequenos dispositivos como por exemplo um PDA ou um computador portátil participam na rede através de tecnologia sem fios ou mesmo através de cabos. Os nós clientes participam na rede estabelecendo uma ligação com os *routers mesh* caso se encontrem ao alcance destes, ou

através de outros clientes. Neste último caso, os nós clientes mais próximos dos *routes mesh* são responsáveis por fornecer o serviço a nós clientes mais distantes retransmitindo a informação até aos nós destinatários.

O SPINE [29] é um sistema de Pub/Sub para redes parcialmente infra-estruturadas (*Mesh*). O sistema aproveita o facto de a rede dispor de alguns dispositivos sem restrições de energia e com uma localização fixa, os quais designa de *routers*, utilizando-os para armazenar as subscrições. O SPINE assume que os *routers* estão dispostos em matriz. As subscrições são disseminadas pelos routers que se encontrem na mesma linha que o subscritor e as publicações pelos routers que se encontram na mesma coluna. Sempre que os routers recebem uma publicação, verificam se há subscritores interessados na sua linha e encaminham-na para o subscritor. Apesar da sua simplicidade, a aplicação deste sistema a RSSF não é trivial já que obrigaria a que os nós sensores tivessem o conhecimento da sua localização geográfica.

Redes ad hoc móveis

As redes ad hoc móveis (MANETs) partilham algumas características das redes Mesh e das RSSF. As MANETs assemelham-se às RSSF no sentido em que não necessitam de uma infra-estrutura de suporte à comunicação, o que permite a criação de redes espontâneas, e às redes Mesh em relação à mobilidade dos nós participantes. Os dispositivos que compõem uma MANET são semelhantes aos nós clientes que compõem uma rede Mesh. Embora tenham restrições ao nível energético (alimentados por bateria), as restrições impostas pelos sensores são mais acentuadas em termos energéticos, capacidade de processamento e memória.

Para tornar os sistemas Pub/Sub eficientes no contexto das MANETs, é essencial construir uma infra-estrutura eficaz que seja capaz de encaminhar os eventos dos publicadores para os subscritores com um custo – em termos do número de mensagens – reduzido. Tipicamente essa infra-estrutura é composta por *brokers*, que coleccionam todas as subscrições, efectuam a operação de correspondência, tratam da operação de encaminhamento que inclui a descoberta de rotas, e do envio dos eventos para os nós subscritores. Desta forma, os dispositivos clientes não têm que utilizar a sua memória para guardar informação sobre as subscrições, vêm a necessidade de processamento ser reduzida, e reduzem o consumo de bateria uma vez que não necessitam de trocar mensagens para efectuar o encaminhamento dos eventos. A eficiência desejada é conseguida dispondo os *brokers* numa topologia de árvore que simplifica a tarefa de encaminhamento.

Algumas propostas para sistemas Pub/Sub estão descritas em [30, 31, 32, 33]

e todas usam uma infra-estrutura de *brokers* para reduzir o gasto de recursos nos nós clientes distinguindo-se pelas optimizações efectuadas a nível de encaminhamento e nos modelos de subscrição. A aplicação destes sistemas a RSSF não é desejável uma vez que a forte dependência num conjunto limitado de dispositivos, e os recursos computacionais e energéticos requeridos tornam os *brokers* inadequados para os cenários de aplicação das RSSF.

Redes de sensores sem fios

Os dispositivos que compõem as RSSF são os que mais restrições impõem a nível dos recursos, em particular, poder de processamento, memória e capacidade energética. Por este motivo é particularmente importante encontrar soluções que façam uma gestão mais rigorosa dos recursos. De seguida são apresentados alguns sistemas desenhados especificamente para RSSF, e são discutidas as vantagens e desvantagens de cada solução.

O sistema Mires [34] utiliza o modelo de subscrição baseado em tópicos. Os publicadores devem enviar mensagens *advertise* que indicam aos nós *sink* que tipo de tópicos são produzidos (ex. temperatura e humidade). Desta forma os nós *sink* não subscrevem tópicos que não tenham sido anunciados pelos publicadores. Estas mensagens chegam aos nós *sink* através de um protocolo de encaminhamento. As aplicações que executam no nó *sink* escolhem o tipo de informação que desejam monitorizar com base nas mensagens *advertise* e enviam as respectivas subscrições para a rede pelo mecanismo de inundação, ou seja, todos os nós sensores da rede recebem e guardam as subscrições. Todos os nós sensores publicam a informação recolhida da área de monitorização enviando os eventos para os respectivos subscritores – os nós *sink* interessados – através de um algoritmo de encaminhamento. O sistema utiliza o modelo de subscrição baseado em tópicos, que requer pouco processamento para a operação de correspondência. Contudo o nível reduzido de expressividade pode-se revelar um problema, no sentido em que o nó *sink* pode estar interessado apenas num sub-conjunto da informação algo que o modelo baseado em tópicos não permite devido à sua expressividade limitada e que pode implicar um envio de mensagens desnecessárias. Como foi discutido na Sec. 1, o envio de mensagens implica um maior gasto energético que o necessário para o processamento de informação, pelo que a incapacidade de refinar os interesses é penalizadora devido ao envio de mensagens desnecessárias. A aplicação do modelo de subscrição baseado em conteúdos requer mais processamento, mas potencialmente menos mensagens e por tanto permite um menor gasto de energia.

Este sistema enquadra-se nos modelos descentralizados em que se efectua a

disseminação de subscrições e reúne dois factores indesejáveis. O primeiro factor deve-se ao facto de a informação de subscrições ser armazenada em todos os nós, o que sugere um elevado uso de memória dos dispositivos. O segundo factor deve-se ao mecanismo de inundação utilizado na disseminação das subscrições, que requer um elevado número de mensagens que tem como consequência um elevado consumo energético.

Para resolver o problema das restrições de memória dos dispositivos, alguns sistemas usam o mecanismo Data Centric Storage (DCS) [35] cujo objectivo é distribuir e localizar informação numa rede não infra-estruturada. No DCS os dados são identificados por uma chave e a esta é associado um par de coordenadas geográficas através de uma função determinista. Os participantes responsáveis por salvaguardar cada item de dados são aqueles que circundam a coordenada geográfica determinada para o item. Uma aplicação possível do DCS é atribuir uma chave a cada tópico de um sistema Pub/Sub, nomeando assim os dispositivos mais próximos da localização como mediadores [36].

Este sistema enquadra-se na concretização descentralizada que difunde as subscrições, porém a informação de subscrição é armazenada apenas num conjunto limitado de nós. Tendo as subscrições armazenadas em locais específicos, os publicadores antes de publicarem informação devem recolher as subscrições de modo a verificarem a existência de nós interessados na informação. Esta verificação é crucial para a redução do número de mensagens, uma vez que não deverão ser enviadas mensagens que contenham informação na qual nenhum nó expressou o interesse.

Contudo, este sistema apresenta diversas desvantagens. Em particular, a sua inadequação a um modelo baseado em conteúdos, onde não é trivial definir critérios para associação de subscrições. Adicionalmente, o DCS requer que todos os nós tenham conhecimento das suas coordenadas geográficas e dos limites da área de rede, o que diminui a aplicabilidade do sistema.

2.5 Resumo

O paradigma de comunicação Pub/Sub permite que os nós consigam especificar o tipo de informação em que estão interessados, reduzindo o número de mensagens desnecessárias.

Os sistemas Pub/Sub que recorrem a uma infra-estrutura de *brokers* libertam os seus clientes de funções como armazenamento de subscrições, operação de correspondência e envio dos eventos para os nós subscritores interessados. A utilização de *brokers* verifica-se inapropriado para RSSF devido aos cenários de

aplicação destas redes (ex. hostis, remotos) e também aos recursos computacionais e energéticos requeridos.

O modelo de subscrição utilizado por cada sistema influencia o nível de expressividade com que os subscritores podem expressar os seus interesses. Os modelos mais expressivos permitem aos subscritores expressar os seus interesses com grande exactidão, isto é, permite que os subscritores recebam apenas os eventos em que estão interessados. O modelo baseado em tópicos oferece um nível de expressividade reduzido, e requer pouco processamento para a operação de correspondência. Por oposição, o modelo baseado em conteúdos oferece um maior nível de expressividade, e requer mais processamento uma vez que as operações de correspondência são mais complexas. Como o envio de mensagens é energeticamente mais dispendioso que o processamento de dados, a quantidade de energia requerida pelo processamento adicional do modelo baseado em conteúdos pode ser compensado com a redução do número de mensagens que um maior nível de expressividade pode proporcionar.

Os sistemas apresentados para as redes tradicionais não foram concebidos para serem executados em dispositivos com recursos fortemente limitados. Adicionalmente, já foi por nós constatado que a utilização de *brokers* não é apropriada para os cenários de aplicação das RSSF.

Os sistemas descritos para redes Mesh e para MANETs requerem também os serviços prestados pelos *brokers*. Esta é a forma encontrada para reduzir o número de operações que os nós clientes devem executar, passando essa responsabilidade para os *brokers*, que estão munidos de recursos apropriados que permitem a realização das suas operações. Deste modo o uso dos recursos nos nós clientes é reduzido, aumentando o tempo de vida da rede. Porém, como foi discutido anteriormente, o uso de *brokers* não é apropriado para os cenários de aplicação das RSSF.

Os sistemas propostos para RSSF não recorrem ao uso de *brokers* pelo que são completamente descentralizados. Contudo, o sistema Mires utiliza o modelo de subscrição baseado em tópicos, que não permite refinar os interesses dos utilizadores e por tanto pode levar a um envio de mensagens desnecessário que tem como consequência um maior gasto energético. O Mires guarda a informação sobre as subscrições em todos os sensores, o que pode esgotar rapidamente o espaço disponível em memória. O sistema baseado no DCS, que tenta reduzir o uso de memória dos nós sensores, porém, requer que os nós tenham o conhecimento da sua localização e dos limites da área de observação, o que diminui a aplicabilidade do sistema.

Das soluções apresentadas em cima, em particular as direccionadas para RSSF, nenhuma reúne o conjunto de características que permitem reduzir simultanea-

mente o gasto energético e a memória utilizada nas operações de comunicação.

Capítulo 3

Distribuição, localização e envio de recursos

Os dispositivos que compõem as RSSF são caracterizados por terem recursos energéticos escassos, baixo poder computacional e uma capacidade de armazenamento diminuta. Desta forma é importante encontrar modelos de comunicação que minimizem o número de mensagens necessárias de forma reduzir o gasto energético, e ao mesmo tempo diminuir o uso de memória necessária para as operações de comunicação.

O paradigma de comunicação Publicador/Subscritor permite aos nós que exerçam o papel de subscritores expressar interesses em determinados tipos de informação, e portanto reduzir o número de mensagens. Para reduzir a memória utilizada, a informação guardada na memória dos dispositivos deve ser minimizada tanto quanto possível.

Este capítulo apresenta as ferramentas usadas para atingir o objectivo de compromisso entre o número de mensagens necessárias e a memória utilizada. É também fornecida uma explicação sobre o funcionamento de cada uma das ferramentas apresentadas.

O capítulo está estruturado da forma descrita em seguida. A Sec. 3.1 apresenta um dos protocolos de encaminhamento mais popular para redes sem fios não infra-estruturadas. Na Sec. 3.2 é introduzido um mecanismo de replicação para redes sem fios não infra-estruturadas. A Sec. 3.3 resume o capítulo.

3.1 AODV

Os protocolos de encaminhamento têm como objectivo descobrir um conjunto de nós entre o emissor e o receptor de uma mensagem, pelos quais a mensagem é encaminhada. A esta sequência de nós é dado o nome de rota. Após a descoberta de rota, o protocolo é responsável por fazer chegar a mensagem ao destinatário.

O AODV (Ad hoc On-demand Distance Vector) [37] é um algoritmo de encaminhamento para redes sem fios não infra-estruturadas extremamente popular. Neste sentido foi o algoritmo de encaminhamento elegido para incorporar a concretização do sistema proposto.

Este é responsável por descobrir uma rota e enviar os eventos do publicador para os subscritores interessados. De seguida é detalhado o funcionamento do protocolo de encaminhamento, bem como as suas vantagens e desvantagens.

No AODV a informação sobre rotas é guardada numa estrutura de dados denominada por tabela de encaminhamento. A tabela de encaminhamento mantém (entre outra) a seguinte informação: <destino, próximo nó, número de saltos, tempo de expiração da rota>. Cada nó mantém uma tabela de encaminhamento, onde consta no máximo uma entrada por destinatário. As mensagens são enviadas com base na informação contida na tabela de encaminhamento de cada um dos nós que compõem a rota. Assim, se um nó A enviar uma mensagem com destino ao nó B, o AODV acede à tabela de encaminhamento do nó A e envia para o nó seguinte. Este por sua vez decide com base na sua tabela de encaminhamento para que nó a mensagem deve ser enviada a seguir. O processo repete-se até que a mensagem chegue a B. Para cada rota é associado um tempo ao fim do qual a rota é considerada inválida. Contudo, sempre que a rota é utilizada dentro do tempo em que é considerada válida, o seu tempo de expiração é renovado. Este mecanismo é uma mais valia quando os nós que compõe a rede apresentam um elevado nível de mobilidade, uma vez que se a rota não for usada durante um determinado tempo é provável que a rota se tenha quebrado. Neste caso, em vez de ser enviada uma mensagem que nunca chegará ao destinatário, procede-se de imediato a um novo pedido de rota, diminuindo o número de mensagens necessárias.

A operação de descoberta de rota tem como objectivo actualizar a tabela de encaminhamento não só do nó emissor, como também dos nós intermédios, ou seja, os nós que vão encaminhar a mensagem até ao nó destinatário. A operação é despoletada pelo envio de uma mensagem *route request* que é composta pela seguinte estrutura: <endereço do emissor, número de sequência do emissor, identificador de difusão, endereço de destino, número de sequência do destino, número de saltos>. Na sua versão original a mensagem é enviada para a rede através do algoritmo de inundação, que é bastante dispendiosa em termos do número de mensagens que requer para fazer chegar a mensagem a todos os nós da rede. O par <endereço do emissor, identificador de difusão> identifica univocamente um pedido *route request* e é usado para detectar pedidos de rota repetidos, os quais são descartados. Durante o processo de descoberta de rotas, cada nó mantém informação temporária que cria uma rota denominada por *caminho inverso* por

onde são enviadas eventuais respostas para o emissor.

Quando eventualmente o pedido de descoberta de rota chega a um nó (possivelmente o próprio destinatário) que contém uma rota para o destinatário, é verificado se a rota é suficientemente actual para que possa ser enviada uma resposta, que contém entre outros, um campo que regista o número de saltos. Uma resposta *route reply* é enviada para o emissor se e só se o "número de sequência do destino" presente na mensagem *route request* é menor ou igual do que o número de sequência mantido na tabela de encaminhamento do nó receptor, caso contrário o nó retransmite o pedido. A rota de *caminho inverso* é utilizada para enviar a mensagem de resposta ponto-a-ponto, que segue o *caminho inverso* ao percorrido pela mensagem *route request*. Desta forma é evitada mais uma dispendiosa operação de inundação. Se um nó receber mais que um *route reply* do mesmo emissor para o mesmo destino actualiza a informação de encaminhamento e retransmite a resposta se e só se: $i)$ o campo "número de sequência do destino" da mensagem *route reply* for maior que o do *route reply* anterior; $ii)$ ambos os *route reply* têm o mesmo "número de sequência do destino", mas com um número de saltos inferior. Esta actualização permite escolher a rota mais curta em número de saltos.

Uma quebra de rota verifica-se quando uma mensagem não chega ao destinatário devido à incapacidade de qualquer nó intermédio entregar a mensagem ao próximo nó da rota. A natureza dinâmica de algumas redes, a falha das ligações, períodos de inactividade do rádio são alguns exemplos que podem quebrar uma ou mais rotas. Os nós que verifiquem uma quebra de rota devem gerar uma mensagem *route error* que avisa o emissor do sucedido. Na grande maioria dos casos, quando o emissor de dados recebe uma mensagem deste tipo, difunde um novo pedido de rota.

O AODV é um protocolo reactivo, que por oposição aos protocolos pró-activos, inicia o processo de descoberta de rota apenas quando é necessário, isto é, quando um nó necessita de enviar mensagens para destinatários cujas rotas não constem na tabela de encaminhamento ou cuja validade tenha expirado. A vantagem destes protocolos verifica-se no número reduzido de mensagens que requerem, uma vez que cada nó inicia a descoberta de rotas para outros nós se e só se tiver mensagens para transmitir. Pelo facto deste protocolo não transmitir mensagens potencialmente desnecessárias o gasto energético é menor, aumentando a longevidade da rede. A desvantagem é perceptível ao nível do tempo de envio, uma vez que quando um nó tem que enviar uma mensagem para um destinatário que não se encontre na tabela de encaminhamento deve esperar que o processo de descoberta de rota termine para então proceder ao envio da mensagem.

3.2 PCACHE

A PCACHE [38] é um módulo de código intermédio para redes sem fios não estruturadas que replica os dados de tal forma que as réplicas estejam suficientemente distantes para prevenir o excesso de redundância, mas simultaneamente permanecem suficientemente perto de forma a que a informação possa ser obtida por qualquer nó com um número de mensagens reduzido. O funcionamento da PCACHE não se baseia em informação de localização como o GPS, requerendo apenas que os dispositivos tenham a capacidade de obter a força de sinal (RSSI – Received Signal Strength Indication) das mensagens recebidas.

Nesta secção é descrito o funcionamento da PCACHE de um modo genérico, bem como as características que fizeram da PCACHE a ferramenta ideal para concretizar um sistema Pub/Sub económico em termos dos recursos consumidos. O funcionamento da PCACHE em sintonia com o sistema Pub/Sub é abordado no Cap. 4.

Distribuição geográfica dos dados. Na PCACHE a disseminação de dados é realizada por difusão, utilizando o algoritmo PAMPA [39], e é despoletada pelo envio de uma mensagem de *registo*. A mensagem de *registo* tem um campo de controlo Time From Storage (TFS) que indica a distância (em número de mensagens) do nó que envia a mensagem até à cópia mais próxima. A constante *Distance Between Copies* (DbC) dita o valor máximo do campo TFS e implicitamente o grau de replicação dos dados. No contexto deste trabalho usámos para as constantes PCACHE os mesmos valores usados na sua avaliação em [38], em particular $DbC = 2$. Os dados são armazenados a cada $DbC + 1$ saltos, ou seja, 3 saltos.

Quando um nó recebe uma mensagem de *registo* atrasa a sua retransmissão com base no tempo ditado pelo algoritmo PAMPA. Durante o período de espera a PCACHE contabiliza o número de cópias recebidas da mensagem *registo*, e para cada uma computa o menor valor TFS recebido e coloca-o na variável $mTFS$. Findo o tempo de espera ditado pelo algoritmo PAMPA, é realizada uma de três acções: $i)$ se $mTFS = DbC$, o nó guarda a informação na sua memória numa zona à qual chamámos *zona de dados*, actualiza o valor $TFS = 0$, e retransmite a mensagem; $ii)$ se foram recebidas cópias suficientes da mensagem durante o intervalo de espera a retransmissão da mensagem é cancelada (disseminação assegurada pelos outros nós); $iii)$ o número de cópias recebidas não é suficiente para cancelar a retransmissão, nem $mTFS = DbC$, e por tanto o nó deve proceder à actualização do campo TFS para $mTFS + 1$ e retransmitir a mensagem de *registo*;

Quando um nó recebe uma mensagem com $TFS = 0$ reconhece que tem uma

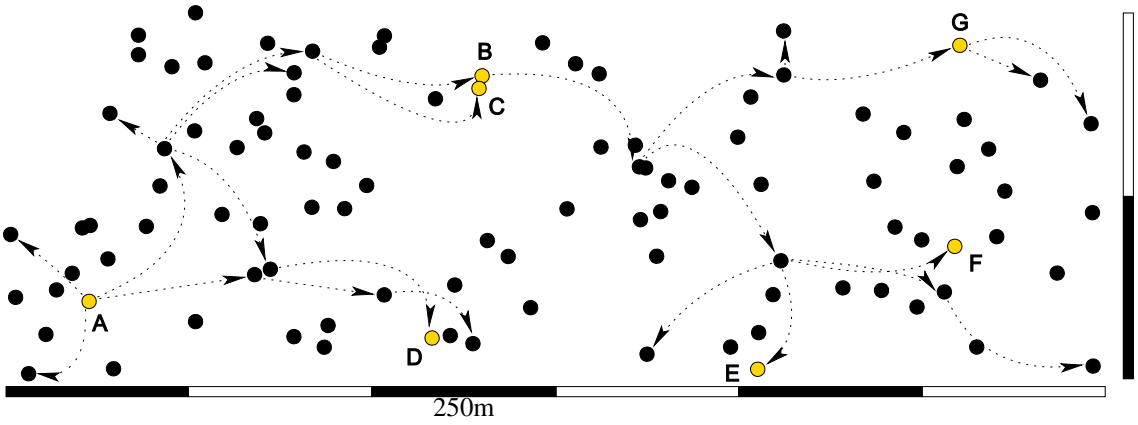


Figura 3.1: Disseminação de uma subscrição

cópia da informação a um salto de distância.

A decisão de armazenar uma determinada informação é local a cada nó mas condicionado pelo conteúdo das mensagens transmitidas pelos vizinhos e pela memória disponível no dispositivo.

No contexto dos sistemas Pub/Sub a PCACHE pode ser utilizada por exemplo para disseminar a informação relativa às subscrições efectuadas pelos nós subscritores. A Fig. 3.1 (retirada de [40]) ilustra a disseminação de uma subscrição numa a rede com 100 nós uniformemente distribuídos por um área de $1500m \times 500m$ e onde os dispositivos têm um raio de transmissão de 250m. A disseminação da subscrição tem inicio no nó A que se encontra no canto inferior esquerdo. A mensagem de subscrição é retransmitida pelos nós que são destino de uma seta e é guardada cada três saltos ($DbC = 2$) nos mais claros que se encontram alfabeticamente nomeados. Como pode ser observado, a subscrição é armazenada num conjunto muito limitado de nós, sendo que nenhuma replica está a mais de 3 saltos de nenhum nó. A imagem ilustra dois nós adjacentes, B e C, que armazenam a informação de subscrição. Este é um comportamento identificado e discutido em [40], e é atribuído à proximidade dos nós. A reduzida distância entre os nós leva o algoritmo PAMPA a ditar um tempo de expiração muito semelhante, o que faz com que os dois nós guardem informação concorrentemente sem saberem da decisão um do outro.

De notar que o nível de replicação é ditado com base nos valores escolhidos na configuração da PCACHE.

Operação de pesquisa. Para recolher a informação dispersa pela zona de dados dos diversos nós é difundida, também através do algoritmo PAMPA, uma mensagem de interrogação, num mecanismo de difusão limitado a alguns saltos.

O raio de procura da operação de pesquisa é limitado pela variável $qTTL$,

que é inicializada com o valor [$((DbC + 1)/2)$+1]. Este valor é suficiente para que a mensagem de interrogação chegue ao conjunto dos nós mais próximos que deverá conter o conhecimento completo de toda a informação disponível.

Quando os nós recebem a mensagem de interrogação independentemente de terem dados na sua memória que a satisfaçam esperam um determinado período de tempo ditado pelo PAMPA, de forma a eleger os nós que retransmitem o pedido de interrogação e os que cancelam a retransmissão do pedido de interrogação. Os nós que tenham informação que satisfaça a interrogação e cuja retransmissão do pedido de interrogação foi cancelada, enviam a resposta para o nó do qual receberam o pedido de interrogação. Os nós elegidos para retransmitirem o pedido de interrogação, independentemente de terem dados que satisfaçam a interrogação não respondem aquando da retransmissão do pedido, em vez disso esperam novamente durante um determinado tempo ditado pela distância a que se encontram do nó emissor da mensagem de interrogação. Por outras palavras, os nós mais longe do emissor – com menor $qTTL$ – respondem primeiro que os nós mais próximos. Este mecanismo permite que os nós retransmissores do pedido de interrogação mais próximos possam receber as respostas dos nós mais distantes antes de submeterem as suas próprias respostas, o que permite reduzir o número de mensagens necessárias através da agregação de respostas, e ao mesmo tempo, possibilita eliminar a informação redundante presente nas respostas.

Os nós que retransmitem o pedido de interrogação juntam o seu endereço a um campo da mensagem de $registo$, formando uma *rota de resposta*, por onde as respostas são enviadas.

No contexto dos sistemas Pub/Sub a operação de pesquisa permite aos nós publicadores reunirem a informação relativa às subscrições de forma a terem conhecimento sobre o conjunto de subscritores aos quais os eventos devem ser enviados. O algoritmo de pesquisa está representado na Fig. 3.2. Na figura, a operação de pesquisa é iniciada pelo nó publicador P e propagada pelos nós assinalados com H, os quais se encontram no centro de uma circunferência que representa o respectivo raio de transmissão. Os nós não assinalados que sejam origem de uma seta tracejada contêm informação relevante. Ao receber a primeira cópia da mensagem, cada nó verifica a sua zona de dados local comparando a subscrição com a publicação, e caso disponha de subscrições que sejam satisfeitas pela publicação de P envia a informação disponível para o nó H do qual recebeu o primeiro pedido de pesquisa. A ordem pela qual os nós respondem é ditada pelo algoritmo PAMPA juntamente com uma função de atraso que tem em conta a distância ao emissor em número de saltos, ou seja, dos mais distantes para os mais próximos, em relação ao nó P. Os nós H mais próximos

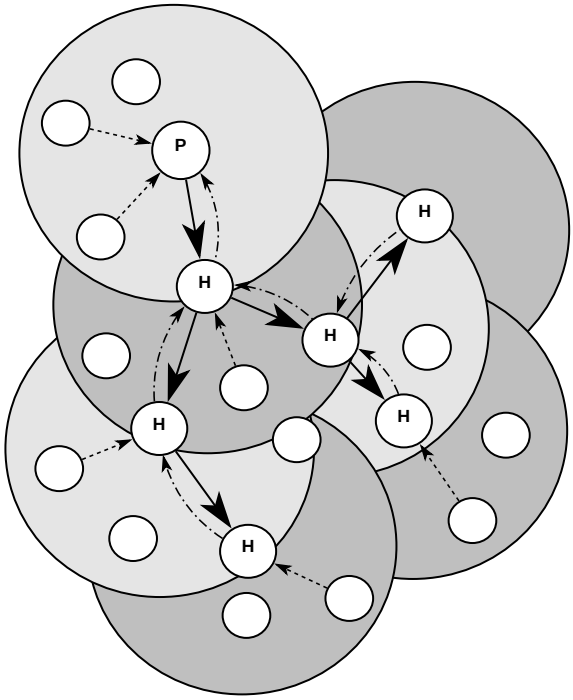


Figura 3.2: Algoritmo de pesquisa de subscrições

de P agregam os dados recebidos dos nós mais distantes e enviam-nos ao nó H anterior. O processo é repetido até que o publicador receba as respostas com a informação sobre os nós interessados. De notar que $DbC = 2$, o que implica que a informação é guardada de 3 em 3 saltos. Como pode ser observado, a operação de pesquisa abrange apenas uma distância em número de saltos limitada, ou seja 3 saltos, que representam o conjunto dos nós mais próximos que reúnem toda a informação.

A PCACHE não faz qualquer interpretação sobre os tipos de dados armazenados, assumindo na sua versão mais simples que estes são constituídos por um par <chave,valor>, sendo a chave utilizada para a identificação de duplicados e como critério de pesquisa.

De notar que a PCACHE não garante que as respostas de nós mais distantes cheguem aos nós mais próximos do emissor antes destes enviarem a sua resposta. Por exemplo, seja A um nó próximo do emissor do pedido de interrogação que aguarda o tempo suficiente que permite receber as respostas de nós mais distantes. No fim do período de espera, não é garantido que todas as respostas de nós mais distantes tenham chegado a A. Nestes casos o nó A responde com a informação que tem em sua posse, e mais tarde se receber uma resposta atrasada procede à sua retransmissão em direcção ao emissor sem mais demora.

Gestão de memória. A escolha dos nós que devem guardar a informação é influenciada pela topologia e pelos nós que originam a disseminação da informação. A PCACHE dispõe de um mecanismo que tenta manter o mesmo rácio de ocupação de memória em todos os dispositivos independentemente da localização dos nós de modo a evitar situações em que alguns nós guardam muita informação, e outros guardam pouca. Este mecanismo permite também que dispositivos com diferentes capacidades de memória possam coexistir na mesma rede. Para tal é preciso garantir que os dispositivos com menos capacidade de memória não ocupem toda a sua memória enquanto outros com mais capacidade dispõem de quantidades consideráveis de memória livre. O mecanismo é accionado apenas quando a utilização de memória excede um valor pré-definido. A intuição deste mecanismo é simples e consiste em incentivar os nós com menor ocupação de memória a expirarem o seu tempo antes dos nós que têm uma maior ocupação de memória, na esperança que sejam os nós com menor ocupação de memória a guardar a informação. Para tal, é introduzido um tempo de espera adicional ao PAMPA, proporcional ao rácio de ocupação do espaço de armazenamento de cada nó de forma a que os nós com mais memória ocupada esperem mais tempo. No caso de grande parte dos nós terem esgotado o espaço de armazenamento, são removidos dados aleatoriamente para permitir acomodar a nova informação. A politica de escolha da informação a ser removida é aleatória dado que uma eleição determinista pode eleger para remoção o mesmo item em todos os nós, caso em que a PCACHE deixava de beneficiar da redundância da informação.

Adicionalmente, a PCACHE tolera o movimento ocasional e/ou a falha (temporária ou não) de alguns nós, sem que isso afecte o seu desempenho [40].

3.3 Sumário

Neste capítulo foram apresentadas as ferramentas usadas na concretização do sistema Pub/Sub cujas contribuições são relevantes para atingir os objectivos propostos.

Em termos de algoritmos de encaminhamento foi apresentado o AODV, um algoritmo reactivo mas cuja operação de descoberta de rotas é extremamente dispendiosa em termos do número de mensagens necessárias para popular a tabela de encaminhamento, uma vez que usa o método de inundação. Contudo, por ser reactivo não necessita de enviar mensagens de controlo desnecessárias, o que permite reduzir os gastos energéticos.

A PCACHE permite distribuir geograficamente qualquer tipo de dados. No contexto dos sistemas Pub/Sub os dados distribuídos geograficamente são por exemplo relativos às subscrições dos nós da rede. Assim, cada nó guarda ape-

nas uma pequena porção das subscrições efectuadas no sistema, o que permite reduzir a utilização de memória dos dispositivos. A informação relativa às subscrições pode ser posteriormente obtida com um reduzido número de mensagens, independentemente da localização do publicador. Adicionalmente, a PCACHE permite a falha e/ou movimento ocasional de alguns dos nós. Não são feitas assumpções relativamente à existência de mecanismos de localização, nem sobre a quantidade de memória que cada dispositivo deve disponibilizar, factores que facilitam a escolha e a coexistência de diferentes dispositivos na mesma rede.

Capítulo 4

Sistema Pub/Sub com subscrições geograficamente distribuídas

O paradigma de comunicação Publicador/Subscritor permite aos nós expressar interesses em determinados tipos de mensagens, e portanto reduz o número de mensagens necessárias uma vez que mensagens nas quais nenhum nó expressa interesse não são enviadas. As aplicações que executam nos nós sensores normalmente necessitam de memória que permita a sua própria execução, pelo que reduzir a memória gasta no armazenamento das subscrições é também importante. Neste capítulo são discutidos algumas abordagens de concretização de sistemas Pub/Sub e é explicada a intuição de uma abordagem que permite atingir os objectivos propostos. É ainda apresentado um novo sistema Pub/Sub eficiente em termos energéticos e em termos de memória consumida.

O capítulo está organizado da seguinte forma: Na Sec. 4.1 são expostas as características que um sistema Pub/Sub deve apresentar. Na Sec. 4.2 são apresentadas algumas abordagens possíveis para a concretização de sistemas Pub/Sub. A Sec. 4.3 apresenta e descreve um novo sistema Pub/Sub para RSSF. É também explicado qual a contribuição das ferramentas descritas no capítulo anterior para o sistema como um todo. Finalmente na Sec. 4.4 é apresentado um resumo do capítulo.

4.1 Requisitos para um Sistema Pub/Sub em RSSF

Os sistemas Pub/Sub devem apresentar as seguintes características:

Dinamismo O sistema deve tolerar ligações intermitentes, falhas e/ou períodos de inactividade dos participantes.

Escalabilidade A degradação do desempenho do algoritmo não deve ser notada com a alteração do número de nós na rede, com o tamanho da rede (em termos de área), número de publicadores ou número de subscritores.

Eficiência O sistema deve apresentar uma elevada taxa de entrega, ou seja, entregar os eventos a todos os subscritores interessados.

Porém os sistemas Pub/Sub para RSSF, para além de partilharem as características acima descritas, devem ainda considerar um conjunto de outras particularidades:

Descentralização Os nós devem ser cooperantes de forma a que a existência de uma infra-estrutura seja dispensável.

Consumo de energia Como os sensores têm uma reserva de energia limitada, o número de transmissões efectuadas pelos nós deve ser tão reduzido quanto possível.

Restrições de hardware As restrições de recursos devem ser tidas em conta na concretização do sistema.

4.2 Algumas abordagens possíveis para concretizar sistemas Pub/Sub

Existem diversas estratégias de concretização de sistemas Pub/Sub [7], porém apenas as descentralizadas são indicadas para os cenários de aplicação das RSSF, uma vez que não requerem a existência de nós especiais que limitam a aplicabilidade do sistema Pub/Sub. Nesta secção serão descritos os dois extremos das estratégias descentralizadas para a concretização de sistemas Pub/Sub e uma terceira solução que permite obter ganhos a nível da utilização de recursos.

4.2.1 Inundação de eventos

A estratégia de inundação de eventos consiste no envio de todos os eventos por um mecanismo de difusão, independentemente dos interesses de eventuais subscritores na informação.

Os nós interessados em receber informação devem esperar que os eventos difundidos cheguem até si, altura em que cada nó deve verificar se o evento satisfaz algum dos seus interesses. Deste modo, não são necessárias mensagens de subscrição o que beneficia esta estratégia em termos de recursos energéticos, mas apenas na operação de subscrição.

Este extremo é preferível para redes onde se espera um número muito reduzido de publicações comparativamente ao número de subscrições, por exemplo aquelas cujo objectivo é a notificação de um único evento, como um fogo florestal.

A vantagem desta solução é que não requer a utilização da memória dos dispositivos para guardar as subscrições efectuadas no sistema, uma vez que cada nó guarda localmente os seus interesses. Contudo, nesta aproximação o número de mensagens cresce linearmente com o número de publicações efectuadas, o que reduz a longevidade da rede. Adicionalmente, com o número crescente de mensagens acresce o número de colisões, que resultam numa maior perda de mensagens.

4.2.2 Inundação de subscrições

Esta abordagem consiste em disseminar todas as subscrições pela rede de forma a que todos os nós armazenem a mesma informação reunindo o conhecimento completo dos interesses.

A vantagem desta abordagem é que os publicadores reúnem o conhecimento necessário para decidir se devem enviar os eventos, e para que nós os devem enviar. Em particular, um publicador pode decidir não enviar um determinado evento se não tiver conhecimento de subscrições que o satisfaçam.

Pelas suas características, este extremo terá maior eficiência num cenário onde importa optimizar o custo da entrega de publicações, ou seja, quando se espera que o número de publicações exceda o número de subscrições. Redes de monitorização de longa duração com uma afiliação aproximadamente constante ao longo do tempo são um bom exemplo deste tipo de aplicação.

Porém, tendo em conta a reduzida quantidade de memória dos sensores, esta abordagem pode ser um problema no sentido em que consoante o número de subscrições, a memória pode esgotar rapidamente, impedindo o armazenamento de novos interesses ou descartando interesses mais antigos. A incapacidade de guardar todos os interesses pode-se reflectir na taxa de entrega uma vez que um nó publicador no instante de publicar um evento pode não ter conhecimento de subscrições que satisfaçam o evento, ou por outro lado, pode ter apenas um conhecimento parcial dos interesses.

4.2.3 Disseminação geográfica das subscrições

A disseminação geográfica das subscrições é semelhante à abordagem anterior no objectivo e nos cenários de aplicação. Porém, a estratégia de armazenamento é diferente, e consiste em guardar as subscrições apenas numa pequena percentagem dos nós da rede. Os nós interessados em publicar dados reúnem a informação relativa às subscrições que se encontra dispersa pelos nós da rede. Após reunirem a informação, têm em sua posse o conhecimento sobre os nós interessados na informação.

A desvantagem desta abordagem em relação à inundação de subscrições reside no número de mensagens necessárias para as operações de comunicação. Nesta abordagem o número de mensagens é superior ao número de mensagens requeridas pela inundação de subscrições uma vez que é necessário reunir a informação relativa às subscrições cada vez que um nó publica dados.

Porém, esta solução representa uma abordagem de compromisso entre a memória utilizada nos sensores e o número de mensagens necessárias para as operações de comunicação.

4.2.4 Comparação entre as abordagens descritas

Na tabela 4.1 é efectuada uma comparação das três abordagens descritas anteriormente. A comparação é efectuada em termos do número de mensagens necessárias por publicação, do número de mensagens necessárias por subscrição e pela quantidade de memória necessária. O valor n denota o número de nós da rede e s o número de nós que subscreveram uma determinada informação.

	Inundação de eventos	Inundação de subscrições	Distribuição geográfica de subscrições
Retransmissões por publicação	$O(n)$	$O(s)$	$O(s)$
Retransmissões por subscrição	0	$O(n)$	$O(n)$
Memória requerida	0	$O(s)$	$< O(s)$

Tabela 4.1: Comparação entre as abordagens de concretização de sistemas Pub/-Sub

Em termos do número de mensagens necessárias por publicação a abordagem de inundação de publicações requer um número de mensagens igual ao número de nós existentes na rede, sendo que a inundação de subscrições e a distribuição geográfica de subscrições requerem tantas mensagens como o número de subscritores existentes para uma determinada publicação. De notar que para as duas ultimas abordagens é necessário contar com o número de mensagens necessárias para a entrega dos eventos, que deverá ser menor que o número de mensagens requeridas pela inundação de eventos.

O número de mensagens de subscrição requeridas pela inundação de eventos é nula, e igual para a inundação de subscrições e para a distribuição geográfica

de subscrições.

A inundação de eventos não requer qualquer utilização de memória para guardar subscrições. A distribuição geográfica de subscrições apresenta vantagem em termos da memória utilizada para guardar as subscrições uma vez que cada dispositivo guarda um número de subscrições inferior ao número total de subscrições efectuadas no sistema. Na inundação de subscrições são armazenadas em cada dispositivo todas as subscrições efectuadas no sistema.

4.3 Sistema Pub/sub

Esta secção apresenta um sistema Pub/Sub descentralizado que utiliza o modelo de subscrição baseado em conteúdos e que tem em conta as restrições de recursos impostas pelos sensores. A aproximação seguida assenta numa replicação parcial das subscrições sendo que as réplicas se encontram distribuídas geograficamente. Enquanto a replicação parcial diminui a quantidade de memória utilizada em cada dispositivo, a distribuição geográfica das réplicas contribui para um baixo consumo energético na obtenção das subscrições, dado que o conjunto integral das subscrições pode ser obtido dos vizinhos de qualquer nó da rede. No nosso sistema, cabe aos publicadores recolher as subscrições e entregar os eventos aos subscritores, utilizando encaminhamento ponto-a-ponto.

No sistema apresentado é esperado que as publicações ocorram em maior número comparativamente às subscrições.

4.3.1 Arquitectura

A Fig. 4.1 ilustra a arquitectura do sistema Pub/Sub proposto. A interacção entre a aplicação e o sistema de Pub/Sub é efectuada através das primitivas *Subscrever* e *Publicar*. A *Notificação* dá a conhecer à aplicação os eventos que satisfazem a subscrição realizada. O nível Pub/Sub utiliza os serviços dos módulos PCACHE e AODV. O serviço de replicação parcial com distribuição geográfica para redes não infra-estruturadas denominado PCACHE [38], está encarregue da replicação e recolecção das subscrições. As subscrições são armazenadas em memória numa estrutura de dados denominada *dados*. O protocolo de encaminhamento AODV [37] é encarregue do envio dos eventos aos respectivos subscritores.

Tanto a PCACHE como o AODV disponibilizam ao módulo de Pub/Sub o acesso para leitura das estruturas de dados mantidas por cada um. A estrutura *dados* mantida pela PCACHE é acedida pelo módulo de Pub/Sub para identificação da lista parcial de subscrições armazenadas que satisfazem uma publicação. A tabela de encaminhamento do AODV é utilizada pelo módulo de Pub/Sub

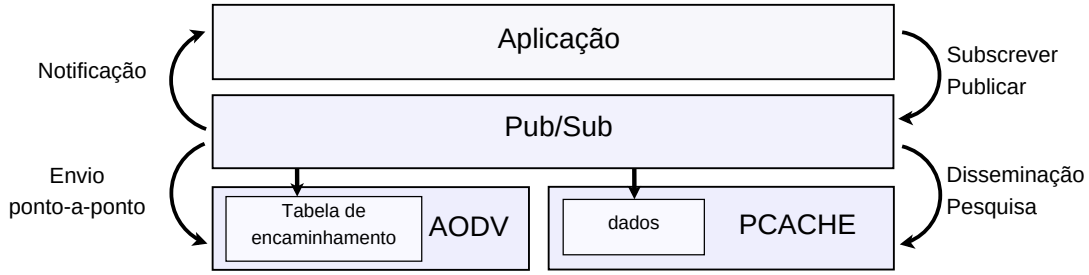


Figura 4.1: Arquitectura do sistema

para efectuar uma optimização que consiste em identificar troços comuns nas rotas para os subscritores e que será abordada mais adiante neste capítulo.

4.3.2 Subscrições

Os pedidos de subscrição são recebidos no nível Pub/Sub. Este por sua vez contacta o nível PCACHE que efectua a disseminação das subscrições, assegurando a distribuição geográfica das réplicas.

A informação associada a cada subscrição é salvaguardada no nível PCACHE e é constituída por um filtro de pesquisa e pelo endereço do subscritor. Quando comparado com uma publicação o filtro de pesquisa permite identificar se a publicação satisfaz ou não os critérios da subscrição de acordo com o modelo baseado em conteúdos.

4.3.3 Publicações

A operação de publicação é decomposta em duas fases: identificação das subscrições relevantes e entrega dos eventos. Ambas são geridas pelo nó publicador.

Recolha de Subscrições A recolha de subscrições é iniciada pelos nós publicadores sempre que publicam informação. Esta operação tem como objectivo identificar os nós interessados na publicação, através da congregação – no nó publicador – dos endereços dos subscritores interessados na publicação. Para tal, é utilizada a operação de pesquisa da PCACHE, passando o publicador a própria publicação como critério de pesquisa. A PCACHE fornece um serviço de pesquisa em melhor esforço. Em particular, em nenhum instante é possível assegurar ao participante que realiza a operação de recolha de subscrições que $i)$ todos os itens de dados que satisfazem a pesquisa foram retornados; $ii)$ que não serão recebidas mais respostas depois de um determinado instante.

Os mecanismos de difusão e pesquisa da PCACHE são ortogonais ao tipo de dados utilizado. Por essa razão, a operação de verificação de correspondência,

entre os dados guardados em cada nó pela PCACHE e a publicação, é delegada pela PCACHE na instância local do módulo de Pub/Sub que com base nos filtros verifica se existem subscrições cujo interesse seja satisfeito pelas publicações. Contudo, cabe à PCACHE o envio das respostas, que respeita o algoritmo de consulta original. Este algoritmo assegura um consumo mínimo de energia, por exemplo por agregar as respostas de diferentes participantes nos nós intermédios e removendo duplicados.

De notar que a operação de pesquisa não informa o publicador do conjunto total de subscrições mantidas pelos nós no raio de pesquisa. O publicador recebe apenas informação relativa a subscrições cujo interesse é satisfeito pela publicação que irá ser emitida. Deste modo, o tráfego na rede é reduzido para o mínimo necessário, e uma vez que só é enviada informação relevante a PCACHE não incorre num desperdício de energia.

Após o nível Pub/Sub do nó publicador ter recebido um pedido de pesquisa, este espera pelo conjunto de respostas durante um período de tempo ao fim do qual é assumido que o conjunto de todas as subscrições foi recolhido com sucesso. O tempo de espera é calculado da mesma forma que a PCACHE calcula os tempos de resposta à operação de pesquisa, que tem em conta a distância dos nós. Ou seja, o nó que é a origem do pedido de pesquisa, irá esperar mais tempo que o tempo necessário para a resposta dos nós incumbidos de retransmitirem o pedido de pesquisa. Deste modo, o instante de expiração no nó publicador acontece depois das respostas à operação de pesquisa terem chegado, sendo este o motivo pelo qual é assumido que no instante de expiração o publicador tem em sua possa toda a informação relevante existente sobre as subscrições. De notar que a PCACHE não garante a entrega das respostas durante o tempo de espera. Findo o período de tempo de espera, os eventos são enviados para os respectivos subscritores.

Durante o tempo de espera todas as respostas recebidas são temporariamente armazenadas. A PCACHE garante que no conteúdo de cada resposta não existe informação redundante, contudo, o conjunto de todas as respostas podem conter informação redundante resultante da duplicação de registos em nós distintos da rede. Neste sentido, o conteúdo de cada resposta recebida é verificado pelo módulo Pub/Sub e comparado face a outras respostas recebidas de modo a verificar e remover informação redundante, ou seja, subscrições efectuadas pelo mesmo nó. Este processo é essencial para evitar que um subscritor receba múltiplos eventos relativos à mesma publicação.

Envio dos eventos Após expirado o período de tempo – no publicador – durante o qual são acumuladas as respostas à operação de pesquisa, o publica-

dor tem em princípio toda a informação sobre eventuais subscritores, isto é, o endereço dos nós interessados na publicação. O módulo Pub/Sub garante que qualquer eventual informação redundante foi removida durante o processo de recolha de subscrições. Nestas circunstâncias o módulo Pub/Sub está em condições de proceder ao envio dos eventos ponto-a-ponto, recorrendo ao protocolo de encaminhamento para redes não infra-estruturadas AODV [37], depois de aplicada uma optimização que reduz o número de mensagens necessárias para entrega dos eventos.

A utilidade do AODV no contexto deste sistema consiste em entregar os eventos aos subscritores cujo módulo Pub/Sub tem conhecimento. A forma mais intuitiva de envio dos eventos consiste em enviar um evento por cada subscritor interessado na publicação. Porém, é possível reduzir o número de mensagens necessárias para fazer chegar os eventos ao mesmo conjunto de subscritores. Para tal, é efectuada uma optimização onde o módulo Pub/Sub acede à tabela de encaminhamento do AODV, e faz uso da informação obtida aquando dos pedidos de descoberta de rota. Adicionalmente, o Pub/Sub usufrui do conhecimento completo do conjunto de subscritores para efectuar a optimização. A tabela de encaminhamento mantida em cada nó pelo AODV tem uma entrada por destinatário onde, para além de outra informação de controlo, como a actualidade da rota, consta o endereço do próximo nó da rota para o destinatário. Em vez de enviar um evento por subscritor, o módulo Pub/Sub utiliza a tabela de encaminhamento do AODV para agregar os eventos de acordo com o próximo nó da rota para o destino. Por outras palavras, para cada subscritor interessado na publicação o módulo Pub/Sub verifica junto da tabela de encaminhamento qual o nó seguinte na rota. Para subscritores que partilhem o mesmo nó seguinte é enviado um único evento que contém os endereços dos subscritores em questão. Esta optimização foi denominada "agregação de eventos". Caso os subscritores não partilhem o mesmo nó seguinte, o envio do evento é delegado no AODV, através de uma mensagem endereçada ao subscritor.

Cada nó que receba uma mensagem com agregação de eventos faz a mesma verificação com base na sua tabela de encaminhamento em relação ao conjunto de subscritores presentes no evento. Quando um nó eventualmente verifica que um dos subscritores não partilha o nó seguinte com os demais, o endereço do subscritor é removido do conjunto de subscritores presente no evento e é enviado um novo evento com destino ao subscritor em questão. No limite, este comportamento é adoptado para todo o conjunto de subscritores, se não existirem nós seguintes comuns para nenhum subscritor. Nesta situação é enviado um evento para cada subscritor.

De notar que aquando da desagregação de um evento, um novo evento é endereçado ao subscritor, sendo que a entrega do evento é inteiramente delegada no AODV deixando o módulo Pub/Sub de inspeccionar o evento em cada nó seguinte, uma vez que não são efectuadas mais operações de optimização.

As respostas a um pedido de pesquisa entregues tardiamente pela PCACHE ao módulo Pub/Sub não beneficiam da optimização de agregação de eventos. A optimização também não é aplicada aos destinatários para os quais não exista uma entrada na tabela de encaminhamento do publicador. Nestes casos, é criada uma mensagem endereçada ao destinatário e o envio é delegado no AODV.

Aquando do envio de um evento, o AODV verifica se tem rota para o destinatário, se tiver envia o evento sem mais demora, caso contrário procede à operação de descoberta de rota. A operação de descoberta de rota requer um elevado número de mensagens, sendo que na presença de um elevado número de operações de descoberta de rota num curto intervalo de tempo, pode surgir instabilidade na rede, um cenário vulgarmente denominado por *Broadcast Storm* [41]. Para reduzir o risco da sua ocorrência, os eventos são entregues ao módulo AODV a um ritmo cadenciado, permitindo a estabilização da rede entre cada uma das operações. Para tal, em vez dos eventos serem enviados imediatamente após a expiração do temporizador do publicador, são colocados numa fila de espera. Para os eventos com destino a nós para os quais não exista uma rota na tabela de encaminhamento é aplicado um tempo de espera maior de 1.5 segundos, uma vez que o AODV terá que proceder à operação de descoberta de rota – composta pela operação de inundação – e portanto requer um maior número de mensagens. Quando o evento tem como destino um nó para o qual existe uma rota, o atraso aplicado é menor, 0.2 segundos. Tal deve-se ao número de mensagens relativamente reduzido que o envio de mensagens ponto-a-ponto requer.

4.4 Sumário

Foram apresentadas três estratégias de concretização de sistemas Pub/Sub das quais duas representam extremos opostos, nomeadamente, a inundação de eventos e a inundação de subscrições. Os cenários de operação indicados para as duas estratégias são distintos. A inundação de eventos deve ser aplicada quando se espera um menor número de publicações comparativamente ao número de subscrições e não necessita de memória para armazenar subscrições. A inundação de subscrições tem vantagens quando o número de publicações superior ao número de subscrições, e requer que todos os nós guardem todas as subscrições efec-

tuadas no sistema Pub/Sub, o que pode ser incomportável. A terceira estratégia apresentada consiste na disseminação geográfica das subscrições. Esta estratégia é semelhante à inundação de subscrições, porém requer uma menor utilização da memória dos dispositivos. O custo desta redução de memória é um aumento do número de mensagens necessárias para as operações de comunicação uma vez que é necessário recolher as subscrições.

Foi proposto um novo sistema Pub/Sub que tem como objectivo encontrar uma solução de balanço entre a memória necessária para as operações de comunicação e o número de mensagens requeridas pelas operações de comunicação. O sistema proposto utiliza a disseminação geográfica de subscrições para reduzir a memória dos dispositivos. A disseminação das subscrições bem como a sua recolha é efectuada pela PCACHE, que requer um número de mensagens reduzido para a sua operação.

Os eventos são enviados ponto-a-ponto aos respectivos subscritores através do algoritmo de encaminhamento AODV. Duas optimizações foram concretizadas para tentar reduzir o número de mensagens necessárias às operações de comunicação. A primeira consiste em enviar as mensagens a um ritmo cadenciado de forma a evitar o problema de *Broadcast Storm*. A segunda, consiste em agrupar os eventos que partilhem um mesmo troço da rota aquando do envio para os respectivos subscritores.

Capítulo 5

Concretização do PAMPA

Neste capítulo é detalhado o funcionamento do algoritmo de disseminação PAMPA e é apresentada uma concretização do mesmo algoritmo para o sistema operativo TinyOS. Como foi observado no capítulo anterior, o PAMPA é a fundação para o funcionamento da PCACHE. Com a implementação do PAMPA pretende-se por um lado passar do plano teórico e ideal para o plano real de modo a ter uma noção das reais dificuldades impostas pelas restrições de recursos. Por outro, pretende-se também ter uma noção parcial da exequibilidade do sistema Pub/Sub proposto no capítulo anterior em sensores reais. O termo "noção parcial" surge devido ao facto de ter sido concretizado apenas a fundação da PCACHE que é uma das ferramentas necessárias ao sistema Pub/Sub.

A Sec. 5.1 descreve o algoritmo de difusão PAMPA. A Sec. 5.2 introduz alguns conceitos do sistema operativo TinyOS, bem como uma síntese do seu modo de funcionamento. Na Sec. 5.3 é ilustrada a arquitectura da concretização do algoritmo PAMPA numa perspectiva de componentes necessários à concretização do algoritmo. São também descritas as relações entre os componentes. Na Sec. 5.4 são descritas as decisões de concretização mais relevantes bem como as dificuldades introduzidas pelas particularidades do TinyOS. Na Sec. 5.5 é descrito de um modo geral o simulador utilizado como depurador durante a concretização do algoritmo Pampa. Finalmente na Sec. 5.6 é apresentado o sumário do capítulo.

Detalhes sobre o trabalho realizado, nomeadamente uma discussão das ferramentas e do ambiente de desenvolvimento, bem como uma breve descrição dos sensores utilizados no decorrer da concretização são apresentados no Apêndice A.

5.1 Algoritmos de difusão

O propósito dos algoritmos de difusão de dados é a entrega da informação, em melhor esforço, ao maior número possível de participantes. O algoritmo de inundação é um dos mais simples algoritmos de difusão de dados. O funcionamento

do algoritmo consiste na retransmissão por cada nó de cada mensagem recebida pela primeira vez. O processo de retransmissão termina quando todos os nós retransmitiram. A entrega em melhor esforço não garante que todos os nós recebem a mensagem (ex. existência de partições ou interferências), contudo na ausência de condições adversas, é esperada uma elevada taxa de entrega. Este algoritmo requer um elevado número de mensagens para o processo de disseminação, dado que são necessárias tantas transmissões quanto o número de nós que compõem a rede.

A principal desvantagem deste algoritmo é que muitas das mensagens enviadas não contribuem para a continuação da disseminação por serem duplicadas. Adicionalmente, o elevado número de mensagens enviadas pode gerar um elevado número de colisões, um cenário denominado por *Broadcast Storm* [41]. É importante relembrar que o envio de cada mensagem tem um custo energético associado, facto que torna este algoritmo dispendioso em termos de recursos energéticos. Contudo, este algoritmo é usado por muitos protocolos de encaminhamento para aquisição e manutenção de informação de encaminhamento, utilizada no envio de mensagens.

O PAMPA (Power-Aware Message Propagation Algorithm) [39] é um algoritmo de difusão que tenta ser conservador no uso dos recursos dos dispositivos.

O PAMPA assume que os dispositivos são capazes de obter a intensidade da força de sinal (Received Signal Strength Indication – RSSI) aquando da recepção de cada mensagem. O RSSI é usado como indicador da distância da qual o nó receptor se encontra do nó emissor, e é crucial para a decisão de retransmissão de uma determinada mensagem. A intenção subjacente é que devem ser os dispositivos mais distantes do nó emissor a assegurar a continuidade da disseminação de forma a que as retransmissões consigam abranger um novo conjunto de nós tão grande quanto possível e desta forma maximizar o número de nós que recebem a retransmissão pela primeira vez.

Ao receber uma mensagem, os nós não procedem imediatamente à sua retransmissão. Em vez disso, cada nó adia a retransmissão com base num tempo t ditado pelo produto do RSSI obtido – localmente em cada nó – da mensagem por uma constante k pré-definida. O RSSI decai com o aumento da distância. Quanto mais longe o receptor estiver do emissor menor será o tempo de adiamento da retransmissão, e por tanto, mais depressa irá retransmitir. Devido à dispersão dos nós que compõem a rede, os nós receptores situam-se a distâncias diferentes de qualquer nó emissor, o que implica tempos de espera distintos para a retransmissão da mensagem. Cada nó no fim do tempo de adiamento, procede à retransmissão agendada.

Para reduzir o número de retransmissões, o PAMPA conta com um algoritmo de cancelamento. Cada nó guarda o número de cópias que já recebeu de cada mensagem. Uma constante n define o número máximo de cópias que cada nó deve receber para cancelar a respectiva retransmissão. Quando o número de cópias recebidas atinge o limite de cópias pré-definido n, o nó reconhece que a disseminação foi assegurada pelos restantes nós participantes e cancela a retransmissão.

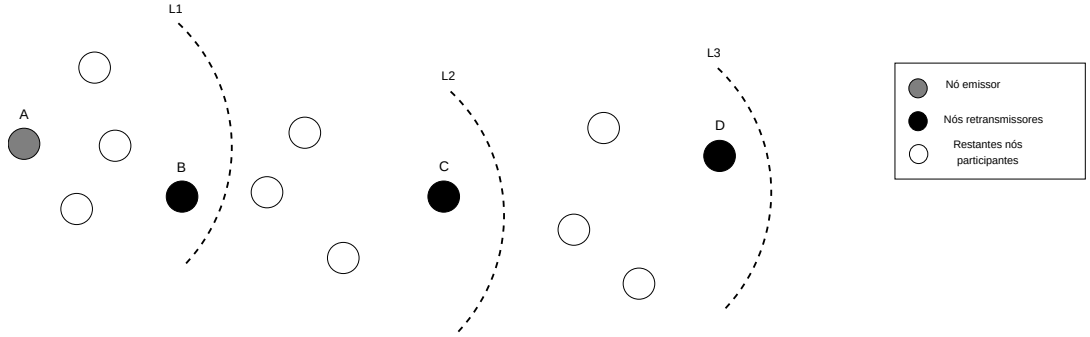


Figura 5.1: Envio e retransmissão de uma mensagem

Na Fig. 5.1 é ilustrada a disseminação de uma mensagem efectuada pelo algoritmo PAMPA, para $n = 1$. As fronteiras L1, L2 e L3 representam o raio de alcance dos nós A, B e C, respectivamente. O nó emissor A envia a mensagem, todos os nós à esquerda de L1 recebem e agendam a retransmissão da mensagem com base nos valores do RSSI e de k. O tempo t do nó B é inferior ao dos restantes nós por ser o que se encontra mais distante do emissor, e como tal é o primeiro a retransmitir a mensagem. Quando B retransmite a mensagem tanto os nós à esquerda de L1 como os nós entre L1 e L2 recebem a mensagem. Os nós à esquerda de L1 apercebem-se que se trata de uma retransmissão e cancelam a retransmissão agendada ($n = 1$), os nós entre L1 e L2 que recebem a mensagem pela primeira vez agendam a retransmissão. O nó C é o primeiro a retransmitir dado que se encontra mais distante do nó retransmissor, pelo que todos os nós entre L1 e L2 cancelam a retransmissão ao ouvir a retransmissão realizada por C, enquanto que os nós entre L2 e L3 agendam a retransmissão. Analogamente, D é o primeiro a retransmitir, pelo que os demais nós entre L2 e L3 cancelam a retransmissão agendada.

Desta forma, e na ausência de efeitos anormais na propagação de mensagens, o PAMPA garante que os nós que retransmitem são os que têm uma maior área de cobertura, reduzindo o número de retransmissões necessárias já que os nós mais perto do emissor ao ouvir novamente a mensagem cancelam a retransmissão. Assim o PAMPA consegue reduzir consideravelmente o número de mensagens necessárias quando comparado com o algoritmo de inundação e consequentemente

a quantidade de energia gasta na operação.

5.2 TinyOS e nesC

O TinyOS [42, 43] é um sistema operativo de código aberto, especialmente direccionado para RSSF. Este sistema operativo surgiu no contexto de um projecto de investigação da Universidade da California, Berkley e desde então tem vindo a crescer e a ganhar maturidade, razão pela qual é amplamente adoptado para a programação em redes de sensores.

O sistema operativo TinyOS está implementado em nesC (network embedded system C). O nesC [42], é também a linguagem de programação usada para concretizar aplicações para o TinyOS e surge de uma extensão da linguagem de programação C. Esta linguagem de programação é baseada em componentes e segue o modelo de execução baseado em eventos. Neste modelo de execução é a ocorrência de eventos que determina o fluxo de execução do programa.

No TinyOS existe a noção de *interfaces* e *componentes*, sendo que os componentes incluem *módulos* e *configurações*. As interfaces são usadas na comunicação entre componentes. Os módulos representam as implementações que fornecem a funcionalidade a uma ou mais interfaces. Os módulos implementam um ou mais serviços e são nomeados com sufixo "P" que denota uma abstracção privada ou com sufixo "C", que denota uma abstracção usável. Porém, ficheiros com nomes compostos pelo sufixo "C" podem também representar configurações. Por convenção, as configurações interligam os diversos componentes. O modo de interligação é explicado mais adiante nesta secção.

De um modo geral os módulos são semelhantes a objectos, no sentido em que encapsulam estado e agregam esse estado a uma ou mais funcionalidades. A diferença entre módulos e objectos está no âmbito de acção do espaço de nomes. Ao contrário do C++ e do Java que se referem a funções e variáveis num espaço de nomes global, os componentes nesC têm acesso a um espaço de nomes puramente local. Por esta razão, os diversos módulos comunicam entre si através de interfaces. Neste sentido cada módulo deve indicar os serviços que requer de outros módulos, e os serviços que fornece a outros módulos. O processo de associação entre o utilizador de um serviço (ex. módulo que usa um serviço de outro módulo) e o fornecedor do serviço (ex. módulo que implementa e fornece o serviço) é obrigatório, sendo que deve ser realizado nas configurações através de interfaces, e é denominado por *wiring*. Por convenção o *wiring* pode ser feito a módulos ou configurações, porém, não deve ser feito a módulos cujo nome contenha o sufixo "P", uma vez que denotam módulos privados. Assim, o *wiring* deve ser efectuado a abstracções que contenham o sufixo "C".

Os componentes dispõem de três abstracções computacionais, *comandos*, *eventos* e *tasks*. Os componentes comunicam entre si através de interfaces recorrendo aos comandos e aos eventos, enquanto que as tasks são utilizadas como um mecanismo de gestão de concorrência entre os componentes.

Um *comando* representa um pedido de um serviço a um módulo. O pedido efectuado pelo *comando* não bloqueia à espera de uma resposta relativa à execução do serviço, em vez disso recebe uma resposta apenas com o estado do pedido (ex. pedido aceite, pedido não aceite). Caso o pedido não tenha sido aceite cabe ao emissor do pedido decidir o que fazer. Caso o pedido tenha sido aceite, mais tarde, um evento é sinalizado para indicar o resultado da execução do pedido (ex. pedido efectuado com sucesso, impossível processar o pedido, entre outros). Uma vez que existe apenas um fio de execução único, este mecanismo permite que computações mais longas possam executar sem que seja necessário bloquear à espera das respostas à execução do pedido.

Os eventos podem ser também sinalizados de forma assíncrona por exemplo, devido a interrupções de hardware, à recepção de mensagens, à expiração de temporizadores, entre outros.

As *tasks* permitem adiar a execução de trechos de código cujo processamento não seja necessário de imediato. Implicitamente oferecem um modo de gerir a concorrência (ex. acesso a estruturas de dados), no sentido em que não são interrompidas por outras *tasks*. Quando uma *task* é agendada para execução é colocada numa fila que obedece a uma ordenação FIFO. O sistema operativo é encarregue de escalonar a sua execução assim que possível. Porém uma *task* também não deve executar durante longos períodos de tempo, já que há partes do sistema operativo que também recorrem a este mecanismo, o que pode fazer com que o sistema operativo tenha um desempenho inferior ao esperado. Para evitar longas computações as *tasks* podem-se propor a elas próprias para executar mais tarde. Quando são novamente escalonadas para execução continuam a operação que estavam a executar anteriormente. O tempo de resposta do sistema operativo pode-se manter reduzido aumentando o número de *tasks* e simultaneamente reduzindo o número de instruções que têm que executar. Porém, cada *task* necessita memória para armazenar o seu estado, pelo que deve haver um balanço entre o número de instruções computadas numa *task* e o espaço utilizado para manter o seu estado. Outra particularidade das *tasks* é que não recebem argumentos, pelo que toda a informação a ser computada dentro das *tasks* deve estar armazenada em variáveis globais.

Embora a execução de uma *task* não seja interrompida por outra *task*, a sinalização de um evento interrompe a execução da *task* e executa o código das rotinas de tratamento de interrupções. As rotinas de tratamento de interrupções

executam apenas código assíncrono, que tem uma prioridade de escalonamento superior à das *tasks*. Desta forma é possível por exemplo, receber mensagens ou ter conhecimento da expiração de um temporizador durante a execução de uma *task*.

O nesC é optimizado para as limitações de memória impostas pelos sensores. O processo de compilação cria um ficheiro binário onde constam apenas os componentes estritamente necessários à execução da aplicação numa dada plataforma. Deste modo o ficheiro binário gerado é tão reduzido quanto possível.

5.3 Arquitectura

O componente Pampa é composto por vários componentes. Na Fig. 5.2 são ilustrados de forma gráfica os componentes necessários à operação do algoritmo Pampa. As caixas com os limites a tracejado denotam componentes genéricos, ou seja, componentes que denotam abstracções que recebem argumentos numéricos (ex. uma dimensão para inicialização de uma estrutura de dados) e de tipo (ex. segundos, milissegundos). As caixas simples representam um módulo, e uma caixa dupla uma configuração. As entidades ovais denotam interfaces e as setas ilustram o *wiring*. As entidades que são origem de uma seta, utilizam os serviços das entidades que são destino da respectiva seta. O diagrama mostra que a implementação da interface Pampa é concretizada pelo módulo PampaP, que por sua vez requer os serviços que são fornecidos através das configurações LedsC, RF230ActiveMessageC, PampaLinkedListC e TimerMilliC. De seguida são brevemente descritos os serviços que cada configuração fornece, e quais as suas contribuições.

LedsC. Configuração fornecida pelo TinyOS que permite controlar os leds do sensor (ex. acender, apagar).

RF230ActiveMessageC. Configuração fornecida com o TinyOS que provê ao componente PampaP as funcionalidades das interfaces SplitControl, AMSend, Receive, Packet, e PacketField<uint8_t>.

A funcionalidade fornecida pela interface SplitControl permite activar e inicializar todos os serviços que estejam especificados no *wiring*, como por exemplo, o serviço rádio. Sem os serviços desta interface todas a tentativas de envio e/ou recepção de mensagens são infrutíferas. Permite também desligar os serviços quando estes não são necessários, de forma a reduzir o consumo de energia. As interfaces AMSend e Receive fornecem o serviço de envio e recepção de mensa-

gens, respectivamente. A interface Packet facilita e permite a manipulação das mensagens recebidas como por exemplo, acesso à zona de dados da mensagem. Finalmente a interface PacketField<uint8_t> permite obter o valor do RSSI de cada uma das mensagens recebidas.

PampaLinkedListC. Estrutura de dados que permite guardar temporariamente as mensagens recebidas e que são candidatas a uma potencial retransmissão.

TimerMilliC. Configuração fornecida com o TinyOS que provê a funcionalidade de um temporizador utilizada para indicar o instante em que uma mensagem deve ser transmitida de forma a dar continuação à disseminação.

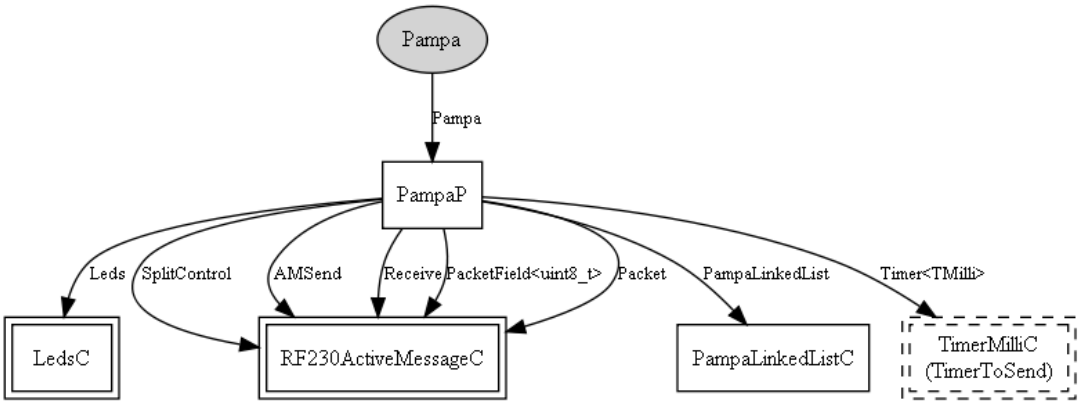


Figura 5.2: Arquitectura do módulo Pampa

Na figura 5.3 é ilustrado o *wiring* necessário do ponto de vista de uma aplicação cliente TestPampaC, para utilizar os serviços fornecidos pelo componente Pampa. É visível que o *wiring* é efectuado à configuração PampaC. Do ponto de vista da aplicação cliente, este é o único passo a efectuar para poder utilizar o serviço do algoritmo Pampa. A configuração PampaC trata de todo o processo de *wiring* que o algoritmo Pampa necessita. Pelo facto de o *wiring* ser feito à configuração e não ao módulo privado PampaP, a tarefa de interligação dos componentes necessários à operação do algoritmo Pampa torna-se transparente para o módulo cliente. Adicionalmente, a aplicação cliente, requer os serviços fornecidos pelas configurações MainC e TimerMilliC. A configuração MainC [44] sinaliza um evento aquando da inicialização de todos os componentes do sistema operativo. O componente genérico, representa um temporizador usado na aplicação de teste.

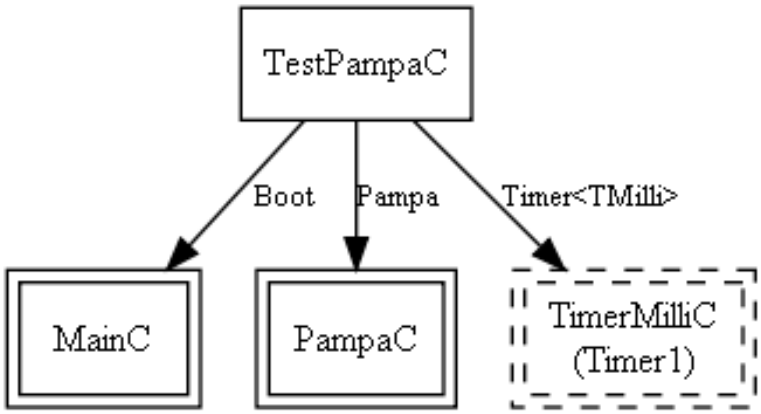


Figura 5.3: Arquitectura da aplicação cliente do módulo Pampa

5.4 Concretização

Esta secção contribui com uma explicação de alto nível em relação à concretização do algoritmo no TinyOS, bem como uma discussão das opções tomadas mais relevantes.

5.4.1 Interface do algoritmo Pampa

A interface do algoritmo Pampa disponibiliza um conjunto de serviços acessíveis aos módulos clientes.

O mecanismo que permite activar o serviço Pampa é disponibilizado através do comando *command error_t start()*. O tipo *error_t* reporta o estado do pedido de activação. Se este não for inicializado a concretização do algoritmo Pampa não fornece as funcionalidades propostas. O evento *event void startDone(error_t error)* é sinalizado para reportar o resultado *error_t* relativo à execução do pedido de activação do serviço.

O envio de mensagens é efectuado através do comando *command error_t sendMessage(pampa_msg_t * msg)*, sendo o argumento msg um apontador para a mensagem a ser enviada. O tipo *error_t* reporta o estado do pedido de envio da mensagem. O evento *event void sendDone(pampa_msg_t * msg, error_t error)* é sinalizado reportando o resultado *error_t* da execução do envio da mensagem msg.

Aquando da recepção de uma mensagem o evento *event void messageReceived(pampa_msg_t * msgRec)* é sinalizado. O argumento $msgRec$ é um apontador para a mensagem recebida.

De modo a evitar gastos desnecessários de energia é disponibilizado um serviço que permite terminar o serviço Pampa. O serviço é fornecido pelo comando *command error_t stop()*. O evento *event void stopDone(error_t error)* é sinalizado para reportar o resultado *error_t* relativo à execução do pedido de terminação do

serviço.

A interface detalhada do algoritmo Pampa encontra-se descrita no Apêndice B.

5.4.2 Estruturas de dados

O algoritmo de disseminação Pampa requer que as mensagens recebidas sejam armazenadas temporariamente para que possam eventualmente ser retransmitidas, pelo que é necessária uma estrutura de dados que guarde temporariamente as mensagens recebidas. Adicionalmente é necessária outra estrutura de dados que guarde temporariamente informação sobre mensagens que já foram enviadas, retransmitidas ou cuja retransmissão foi cancelada.

Nesta secção são discutidas as opções tomadas em relação às duas estruturas de dados necessárias à concretização do algoritmo Pampa. As estruturas de dados têm objectivos distintos e devem coexistir tendo em conta os escassos recursos dos sensores.

Foram ponderadas duas opções para a concretização das estruturas de dados: a primeira opção seria recorrer a uma estrutura de dados com memória estática que pudesse ser alocada em tempo de compilação; a segunda opção passaria por usar memória dinâmica. Ambas as opções têm vantagens e desvantagens. Esta secção discute resumidamente quais as vantagens e desvantagens de cada uma das abordagens e qual a abordagem utilizada em cada uma das estruturas de dados.

As estruturas de dados que recorrem ao uso de memória estática são vantajosas em termos de processamento, uma vez que não são necessárias operações de gestão de memória que só por si requerem recursos computacionais. O espaço necessário é alocado em tempo de compilação sem qualquer necessidade de processamento. Esta abordagem tem como desvantagem a necessidade de configuração adicional, no sentido em que será necessário definir em tempo de compilação o número de elementos que cada estrutura de dados comporta. Esta determinação não é trivial, uma vez que o número de mensagens que a estrutura de dados deverá comportar não é conhecido antecipadamente, pois o volume de mensagens na rede é incerto. Como consequência, se o tamanho da estrutura não for adequado às condições da rede poderá ocorrer um de dois cenários distintos: $i)$ a estrutura de dados não tem capacidade para comportar todas as mensagens recebidas e como consequência algumas serão descartadas; $ii)$ a estrutura de dados tem uma dimensão excessivamente grande, que resulta num desperdício de memória do dispositivo.

A utilização de memória dinâmica tem um impacto negativo a nível de de-

sempenho, uma vez que as funções de gestão de memória (ex. alocar e libertar memória) requerem recursos computacionais. As estruturas que recorrem a memória dinâmica podem comportar tantos dados quanto o espaço livre em memória permitir. Embora esta seja a vantagem da utilização de memória dinâmica, em sensores poderá representar uma desvantagem, dado que o TinyOS não oferece qualquer protecção de memória [42]. A ocorrência de um cenário em que a estrutura de dados pode crescer até esgotar a memória do sensor é real e no limite, a ocorrência deste cenário pode parar a execução do sensor comprometendo o funcionamento da rede. Este problema de crescimento "descontrolado" seria facilmente mitigado limitando a recepção do número de mensagens com um valor pré-definido. No caso do valor pré-definido ser atingido, as mensagens subsequentes seriam descartadas à semelhança da abordagem com memória estática. Porém, a definição de um número de mensagens máximo que a estrutura de memória dinâmica pode guardar difere do problema introduzido pela abordagem com memória estática uma vez que se o valor escolhido for grande, esta estrutura de dados não incorre num desperdício de memória. Contudo, definir um valor para limitar o crescimento da estrutura de dados é algo complexo, pois depende de factores como memória disponibilizada pelo sensor, tamanho das mensagens, memória utilizada por outros componentes (ex. sistema operativo), entre outros. Por outro lado, tipicamente a informação que é recolhida e transmitida pelos sensores não é contínua mas sim periódica, pelo que existem períodos de tempo entre disseminações que são uma mais valia para que algumas das disseminações tenham tempo de terminar antes de outras começarem. Desta forma a probabilidade da ocorrência do problema de crescimento "descontrolado" é mitigado e consequentemente também a perda de mensagens.

Lista de mensagens a retransmitir

Esta estrutura de dados tem como função alojar temporariamente as mensagens recebidas e que serão potencialmente retransmitidas. Com base na discussão relativa à comparação entre memória dinâmica e memória estática, o modelo escolhido para a concretização desta estrutura de dados foi o modelo de memória dinâmica. A razão desta escolha deve-se ao facto de a memória dinâmica ser mais indicada para satisfazer as necessidades de um serviço de disseminação de mensagens, onde que o objectivo é fazer com que todas as mensagens cheguem a todos os nós, e portanto a possibilidade de perder mensagens deverá ser tão reduzida quanto possível.

A estrutura de dados é semelhante a uma lista ligada e foi concretizada para atender às necessidades de concretização do Pampa, não com o objectivo de for-

necer o funcionamento de uma lista ligada genérica. Por motivos de eficiência as mensagens são adicionadas à lista ligada ordenadas por tempo de retransmissão, obedecendo a uma ordenação crescente. Ou seja, a primeira mensagem a ser retransmitida encontra-se à cabeça da lista, e a última na cauda da lista. As vantagens de uma inserção ordenada são: $i)$ sempre que uma nova mensagem é recebida o temporizador que indica o instante em que uma mensagem deve ser retransmitida é reconfigurado. Se esta lista não estivesse ordenada este processo implicaria uma procura com complexidade $O(n)$ cada vez que uma nova mensagem é recebida. Uma vez que a lista se encontra ordenada a reconfiguração do temporizador é uma operação trivial e com complexidade $O(1)$; $ii)$ no instante de proceder a uma retransmissão se a lista não estiver ordenada é necessário proceder a uma nova pesquisa a fim de encontrar a mensagem que deve ser retransmitida. No caso de a lista estar ordenada não só é trivial saber qual a mensagem que irá ser retransmitida – a que está à cabeça da lista –, como incorremos numa operação com complexidade $O(1)$; $iii)$ o processo de remoção da mensagem retransmitida da lista ligada é análogo ao ponto $ii)$. Contudo, a remoção de uma mensagem cuja retransmissão foi cancelada continua com uma complexidade $O(n)$, uma vez que pode estar em qualquer posição da lista. Porém, uma vez que não é esperado um crescimento descontrolado, a operação de remoção de uma mensagem cancelada deverá ter um impacto moderado.

Outra optimização simples, mas relevante, no consumo de recursos foi introduzida e é relativa ao cálculo do número de elementos da lista. Normalmente, o número de elementos de uma lista é obtido através da travessia completa, onde o número de elementos é contabilizado. Esta é uma operação de complexidade $O(n)$. De forma a tentar melhorar o tempo necessário à obtenção desta informação bem como a redução do processamento necessário, o tamanho da lista é mantido por uma variável.

Vector de mensagens tratadas

Esta estrutura de dados tem como função alojar temporariamente as mensagens tratadas. Mensagens "tratadas" são aquelas que foram enviadas, retransmitidas ou cuja retransmissão foi cancelada pelo nó em questão. Esta estrutura de dados evita a retransmissão de mensagens cuja contribuição é nula para a continuação da disseminação. Com base na discussão relativa a memória dinâmica em relação à memória estática levada a cabo no inicio desta secção, o modelo escolhido para a concretização desta estrutura de dados foi o modelo de memória estática. A razão da escolha deste modelo baseia-se na simplicidade da escolha do tamanho desta estrutura de dados no sentido em que, mesmo que o valor

para o tamanho da estrutura de dados seja mal calculado, o algoritmo não deixa de fornecer o seu objectivo principal, poderá no entanto fornecer o serviço a um maior custo (ex. mais retransmissões) se o tamanho da estrutura de dados for muito reduzido.

O número de mensagens mantidas é configurável em tempo de compilação. A estrutura tem o comportamento de um vector circular, ou seja, após o preenchimento da última posição, o próximo local e insersão de dados é a primeira posição. Deste modo, os dados são mantidos enquanto não forem sobrepostos por novos dados.

De modo a minimizar o espaço ocupado pela estrutura de dados, a única informação armazenada, é o UID das mensagens recebidas composto pelos campos "identificador do nó" e "número de sequência do nó", de 16 e 8 bits respectivamente. É importante realçar que a escolha do tamanho desta estrutura pode afectar o comportamento esperado do algoritmo. Em particular, para o cenário em que existe apenas um nó sensor a enviar informação, o tamanho máximo desta estrutura de dados não pode exceder 255 posições ($2^8 - 1$). Se este valor máximo não for respeitado, para além do espaço necessário pode dar-se o caso de todas as mensagens posteriores às 255 primeiras mensagens sejam assumidas erradamente como sendo cópias. Este cenário acontece dado que todas as mensagens enviadas se encontram guardadas na estrutura "mensagens tratadas" e porque número de sequência se volta a repetir.

5.4.3 Envio de uma mensagem

Os clientes do módulo PAMPA podem enviar uma ou mais mensagens cujo conteúdo é independente do algoritmo.

Para um nó enviar uma mensagem deve efectuar o respectivo pedido através da interface Pampa fornecida para o efeito. Quando o pedido de envio da mensagem é submetido pelo módulo cliente, este recebe em forma de retorno o estado do pedido, isto é, se o pedido de envio foi aceite ou recusado. Se foi recusado, é da responsabilidade do módulo cliente tentar novamente o envio se assim o desejar. Se foi aceite, não significa que a mensagem foi/irá ser enviada. Mais tarde será sinalizado um evento com o estado do envio da mensagem. Se o evento for sinalizado com um valor afirmativo, o cliente pode nesta altura ter a certeza que a mensagem foi enviada com sucesso, caso contrário é retornado um erro apropriado que dá informação ao cliente.

O TinyOS permite apenas o envio de uma mensagem de cada vez, pelo que enquanto o evento com o resultado da execução do pedido de envio não for sinalizado, não é possível enviar outras mensagens. É o programador que deve garantir esta propriedade para evitar a corrupção da mensagem.

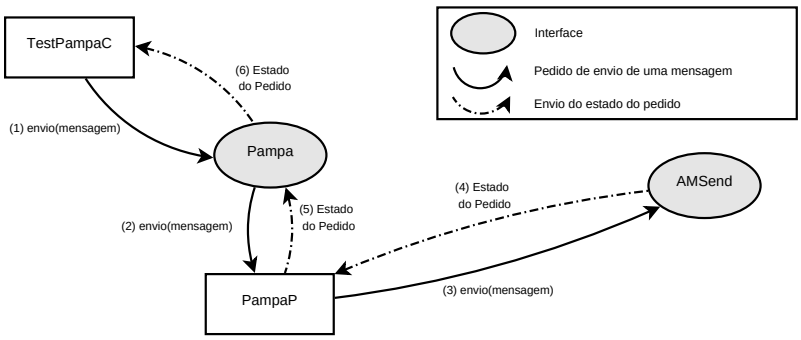


Figura 5.4: Sequência de pedidos de envio de uma mensagem e respectivos estados

A Fig. 5.4 ilustra o processo de envio de uma mensagem do ponto de vista do módulo TestPampaC. O pedido de envio é efectuado ao módulo PampaP através da interface Pampa. O módulo PampaP valida o tamanho da mensagem e verifica se existe alguma mensagem cujo envio esteja pendente. Caso a validação não suceda, é retornado imediatamente um erro que descreve o tipo de ocorrência. Caso a validação suceda e não exista nenhuma mensagem cujo envio esteja pendente o módulo PampaP efectua o pedido de envio da mensagem ao componente fornecido pelo sistema operativo através da interface AMSend.

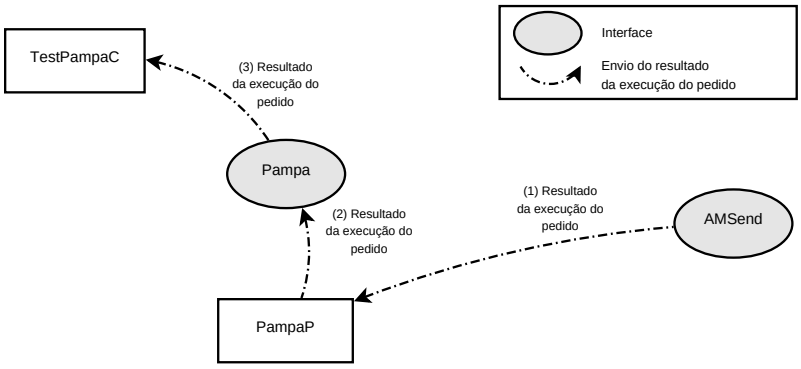


Figura 5.5: Sequência de sinalização de eventos referentes ao pedido de envio de uma mensagem

Caso o resultado da execução do pedido seja positivo, é sinalizado um evento com o resultado da execução do envio da mensagem. A Fig. 5.5 ilustra a sequência de confirmações efectuadas. O módulo PampaP recebe a confirmação da execução do envio da mensagem através da interface AMSend e informa o módulo TestPampaC. Nesta altura o UID da mensagem enviada é armazenado na estrutura de dados "mensagens tratadas" e o número de sequência é actualizado. Caso o resultado da execução do pedido seja negativo cabe ao módulo cliente decidir o que fazer.

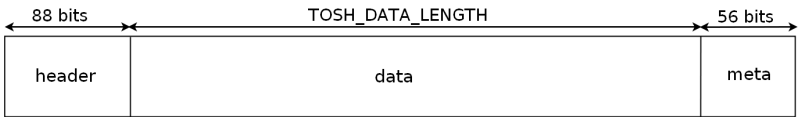


Figura 5.6: Formato da mensagem AM

O módulo Pampa lida com o formato de mensagens normalizado no TinyOS, as mensagens do tipo AM (Active Message) [45], que é ilustrado na Fig. 5.6. O campo *header* contém entre outra informação o endereço do nó, o tamanho da mensagem e o tipo de mensagens. No campo *data* são transportados os dados, sendo que *TOSH_DATA_LENGTH* representa o tamanho máximo que a camada de comunicação consegue fornecer. No campo *meta* está a informação relativa à força de sinal, entre outra.

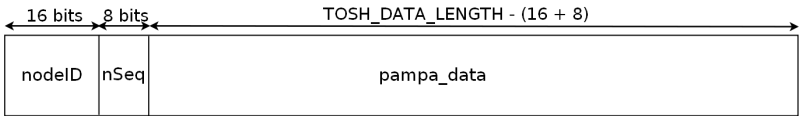


Figura 5.7: Formato da mensagem Pampa

A Fig. 5.7 ilustra a mensagem Pampa, ou seja, a mensagem usada pelo módulo cliente. Como pode ser observado na figura, os campos de controlo usado pelo algoritmo Pampa são constituídos pelos dois campos que representam o UID da mensagem. A cada nó é atribuído um identificador único e é inicializado um número de sequência que é local a cada nó. Desta forma cada nó pode enviar 256 (2^8) mensagens antes do número de sequência se voltar a repetir. O tamanho da mensagem que o cliente do módulo Pampa pode usufruir é restringido pelos campos de controlo usados pelo próprio algoritmo, e pelo tamanho *TOSH_DATA_LENGTH*.

5.4.4 Recepção e cancelamento de retransmissão de mensagens

Esta secção descreve o comportamento do algoritmo perante a recepção de mensagens. De relembrar que uma mensagem "tratada" denota uma mensagem enviada, retransmitida ou cuja retransmissão foi cancelada.

Aquando da recepção de mensagens duas verificações são efectuadas com base no UID da mensagem. A primeira consiste em verificar se a mensagem recebida já existe na estrutura de dados de mensagens tratadas. No caso de existir, a mensagem é descartada, caso contrário é verificado se a mensagem já se encontra na lista de mensagens a retransmitir. No caso da mensagem já existir na lista de mensagens a retransmitir, é incrementado o número de cópias recebidas da men-

sagem, caso contrário é tratada como sendo uma nova mensagem e é adicionada à lista com indicação do instante em que deve ser retransmitida.

No caso da mensagem recebida ser uma cópia, após o incremento do número de cópias é verificado se o número de cópias recebidas é suficientes para cancelar a retransmissão. Se o número de cópias recebidas é suficiente para cancelar a retransmissão da mensagem, a mensagem é removida da lista de mensagens a retransmitir juntamente com o número de cópias recebidas, caso contrário nenhuma acção é realizada.

Sempre que uma nova mensagem é adicionada ou removida da lista de mensagens a retransmitir, o temporizador é reconfigurado para expirar no instante indicado pela próxima mensagem a ser retransmitida.

5.4.5 Retransmissão de mensagens

Sendo o Pampa um algoritmo de disseminação, a retransmissão das mensagens recebidas é essencial para assegurar a continuação da disseminação. Nesta secção será discutido o processo de retransmissão de mensagens.

O processo de retransmissão é auxiliado por um temporizador local a cada nó que dispensa a troca de mensagens para a sincronização com os relógios dos demais nós participantes da rede. Aquando da recepção de uma mensagem, esta é adicionada à lista de mensagens a retransmitir de forma ordenada tal que a próxima mensagem a ser retransmitida se encontra à cabeça da lista. Neste processo o temporizador é renovado utilizando o tempo da mensagem que está à cabeça da lista. Quando o temporizador expira, a mensagem que está à cabeça da lista é retransmitida. No processo de retransmissão de uma mensagem há dois casos que podem impedir o envio da mensagem, os quais são tidos em conta pela concretização: $i)$ no instante da retransmissão outra mensagem está a ser enviada – relembrar que não podem ser enviadas duas mensagem simultaneamente –, pelo que a retransmissão é agendada para mais tarde. Como este processo é efectuado recorrendo a *tasks*, a retransmissão será tentada novamente tão cedo quanto o sistema operativo escalonar a *task* para execução; $ii)$ o componente do sistema operativo que trata do envio da mensagem retorna um erro ao pedido de envio da mensagem. Também neste caso a retransmissão é agendada para mais tarde do mesmo modo que o ponto anterior.

A ocorrência de qualquer um destes dois casos resulta num atraso na retransmissão da mensagem relativamente ao que era esperado. Outra causa para o atraso de uma retransmissão ocorrer é se duas mensagens (seja A e B) tiverem o mesmo instante de retransmissão, ou se por outro lado, se tiverem um instante de retransmissão tão próximo que seja impossível de processar a retransmissão da

mensagem A antes do instante de retransmissão da mensagem B ocorrer. O único problema da ocorrência de qualquer destes casos de atraso na retransmissão é a impossibilidade da disseminação da mensagem ocorrer conforme seria esperado. Perante estes casos não são efectuadas mais retransmissões no sentido em que uma mensagem cuja retransmissão foi adiada pode ainda ser cancelada, se o nó onde o atraso ocorreu receber o número de cópias suficiente para que o cancelamento da retransmissão da mensagem suceda.

Quando a mensagem é retransmitida, esta é adicionada ao vector de mensagens tratadas por forma a evitar que esta seja novamente recebida e proposta para retransmissão pelo mesmo nó. Por fim, a mensagem é removida da lista de mensagens a retransmitir e o temporizador é configurado com o instante de retransmissão da próxima mensagem a ser retransmitida.

5.5 Simulador

A única forma de interacção directa com o utilizador que normalmente os sensores dispõem, é através de *leds*. Esta forma de interacção é muito limitada o que torna as tarefas de análise e depuração em sensores reais extremamente complexas e difíceis. Outro grande entrave à depuração em sensores reais é que os testes não são reproduzíveis, ou seja, um problema ocorrido numa iteração de teste pode não voltar a manifestar-se em execuções subsequentes. Neste sentido, os simuladores são ferramentas indispensáveis em redes de sensores.

Os simuladores são ferramentas úteis que facilitam o estudo dos sistemas num ambiente controlado, onde podem ser observadas as interacções que ocorrem. O TOSSIM (TinyOS SIMulator) [46, 47] é um simulador desenvolvido com o propósito de simular aplicações TinyOS com alta fidelidade e escalabilidade. Normalmente, as representações dos algoritmos avaliadas em simuladores são diferentes das usadas nas implementações reais [46]. O TOSSIM permite simular a mesma concretização que é usada nos sensores reais, a menos da alteração de uns componentes de baixo nível que fornecem uma abstracção de hardware. Este simulador fornece um ambiente controlado e reproduzível bem como um conjunto de ferramentas de depuração. Este conjunto de funcionalidades potencia uma depuração eficiente que permite encontrar erros que de outra forma seriam difíceis de detectar. Deste modo, a transição para sensores reais pode ser adiada até que o código fonte esteja convenientemente testado e os algoritmos estejam compreendidos.

Durante a concretização do algoritmo Pampa este foi o simulador utilizado para depuração de erros, e que permitiu perceber as interacções e o comportamento entre os diversos módulos. O TOSSIM permitiu também perceber a

interacção e o comportamento do algoritmo entre diversos sensores, sem que para isso fosse necessário instalar e configurar uma RSSF.

Embora a contribuição do TOSSIM no desenvolvimento de aplicações para RSSF seja extremamente importante, há um conjunto de factores no seu funcionamento que não correspondem ao comportamento esperado em sensores reais. Por exemplo, no TOSSIM as rotinas de tratamento de eventos são não-bloqueantes, comportamento que é oposto ao dos sensores reais. Isto significa que nos sensores reais a rotina de tratamento de eventos pode interromper outra rotina de tratamento de eventos, o que pode originar um acesso concorrente a alguns dos dados. Outro dos problemas é que na simulação o tempo de processamento não é tido em conta, pelo que duas operações que devam ser executadas em instantes muito próximos, ou no limite, instantes consecutivos (ex. envio de duas mensagens), são executadas sem problemas aparentes.

Por esta razão o código que executa aparentemente sem problemas no simulador pode não executar do mesmo modo num sensor real. Assim, o TOSSIM é tido como sendo uma ferramenta de extrema importância, mas não deve ser usado para obter conclusões finais [47]. Por esta razão as conclusões finais devem sempre ser retiradas recorrendo testes realizados em sensores reais.

5.6 Sumário

A inundação é um algoritmo de difusão de dados que requer um elevado número de mensagens para a disseminação de cada mensagem, pelo que é bastante dispendioso em termos do consumo de bateria, uma vez que cada transmissão/recepção de mensagens tem um custo energético associado. Contudo, a sua importância não é menosprezada, sendo que é usado em alguns dos mais populares protocolos de encaminhamento para redes não infra-estruturadas.

O PAMPA é um algoritmo de difusão eficiente em termos do número de mensagens quando comparado com o algoritmo de inundação, o que o torna menos dispendioso em termos energéticos. O PAMPA assume que os dispositivos conseguem obter o RSSI das mensagens recebidas. Este algoritmo é a base de funcionamento do módulo de código intermédio PCACHE.

A concretização do algoritmo PAMPA para sensores reais teve como objectivo por um lado passar do plano teórico e ideal para o plano real de modo a ter uma noção das reais dificuldades impostas pelas restrições de recursos. Por outro, pretende-se também ter uma noção parcial da exequibilidade do sistema Pub/Sub proposto no capítulo anterior em sensores reais.

O sistema operativo TinyOS foi apresentado e acompanhado de uma descrição de algumas das particularidades que destacam a complexidade de concretização

de aplicações para sensores. Foram também discutidos os problemas e dificuldades impostos pelas restrições de recursos dos sensores e pelas particularidades do TinyOS como por exemplo, aspectos relacionados com o envio consecutivo de mensagens, e também com memória dinâmica e memória estática.

As conclusões retiradas da concretização do algoritmo são descritas no capítulo seguinte, juntamente com a descrição dos testes efectuados com os sensores reais.

Capítulo 6

Avaliação

Este capítulo apresenta a avaliação efectuada à concretização do algoritmo Pampa em sensores reais, que permite tirar conclusões sobre a exequibilidade do sistema Pub/Sub proposto em sensores reais. São também apresentados os resultados do sistema Pub/Sub proposto obtidos através de simulações, que permitem verificar se os objectivos da redução do uso de recursos foram atingidos com sucesso.

6.1 Concretização do Pampa em sensores

Para testar a concretização do algoritmo Pampa em sensores reais foi concretizada uma aplicação simples denominada por TestPampaC. A aplicação é responsável por inicializar o serviço Pampa e posteriormente proceder ao envio periódico de mensagens. A aplicação envia vinte mensagens, uma a cada 3 segundos. O algoritmo Pampa foi configurado para cancelar a retransmissão de mensagens aquando da recepção de 3 cópias. A constante k que juntamente com o RSSI lido dita o tempo de retransmissão foi configurada com o valor $k = 15000$. Este valor adia a retransmissão por aproximadamente três segundos após a recepção da mensagem quando os sensores estão situados aproximadamente a 50 cm um do outro e sem obstáculos pelo meio. Os valores escolhidos permitem uma visualização clara do funcionamento do algoritmo.

No teste do algoritmo foram usados dois sensores. Um dos sensores foi seleccionado para enviar mensagens, enquanto que o outro tem como função receber e retransmitir mensagens.

Importa também referir a quantidade de espaço de armazenamento necessário em cada memória do sensor. Aquando da compilação é gerado um ficheiro binário com 38832 bytes que contém as instruções necessárias à execução do algoritmo. Este ficheiro é posteriormente enviado para a memória FLASH do sensor. A memória ROM é preenchida com as instruções necessárias para a execução do algoritmo, sendo que são utilizados 13800 bytes desta memória. Para a memória

RAM são necessários 594 bytes. A soma do espaço utilizado na memória ROM e RAM é menor que o espaço utilizado no ficheiro binário uma vez que este contém informação adicional, por exemplo, endereços de memória onde as instruções devem ser colocadas em memória e informação de controlo como por exemplo *checksums*.

6.1.1 Modo de depuração.

Os sensores usados (ver Apêndice A) dispõem apenas de um led que indica que está a ser fornecida energia, não sendo possível alterar o seu estado. Assim, estes sensores em particular não têm uma forma de interacção directa com o exterior. Por esta razão foi necessário encontrar uma forma alternativa de perceber o funcionamento do algoritmo nos sensores reais. Para ferramenta de depuração foi adoptado um mecanismo que envia mensagens do sensor para um terminal através da interface USB. Esta foi a única solução que permitiu servir os nossos propósitos. Porém, esta solução apresenta um conjuntos de problemas: $i)$ as mensagens têm um tamanho limitado; $ii)$ necessita de componentes adicionais, que por si só colocam uma carga extra sobre os recursos dos sensores e deste modo reduz o desempenho do algoritmo; $iii)$ as mensagens enviadas para o terminal usam o mesmo mecanismo de envio das mensagens que são destinadas à rede. Do ponto de vista do rádio e dos componentes que o controlam não há nenhuma diferença entre uma mensagem destinada à rede e uma mensagem de depuração destinada à consola. Devido a esta razão e tendo em conta que só pode ser enviada uma mensagem de cada vez, as mensagens que se destinam à rede podem interferir com as mensagens de depuração e vice-versa. Em particular, a mensagem destinada à rede pode deixar de ser enviada devido à interferência da mensagem de depuração, alterando o comportamento espectável do algoritmo. Por outro lado, a mensagem que é destinada à rede pode interferir com a mensagem de depuração, alterando a percepção do funcionamento do algoritmo. Todos estes factores tornaram a depuração em sensores reais complexa e morosa.

Numa tentativa de reduzir a ocorrência dos dois últimos casos referidos, a transmissão da mensagem que potencialmente vai interferir com a mensagem que já se encontra no processo de envio é adiada. Este mecanismo é efectuado recorrendo a uma *task*. O mecanismo permite apenas contornar o problema no caso particular em que durante o processamento de uma mensagem para envio é proposta para envio uma outra mensagem. Neste caso, uma vez que as mensagens são atrasadas torna-se difícil ter uma percepção exacta do funcionamento do algoritmo. Se for proposta para envio mais que uma mensagem, a informação das variáveis globais da *task* é sobreposta, levando à perda de mensagens o que altera o funcionamento do algoritmo, ou deturpa o modo como o funcionamento

do algoritmo é percepcionado.

Ambos os sensores executam o mesmo algoritmo, porém, as mensagens de depuração ocorrem em momentos diferentes no emissor e no receptor. Embora ambos os nós funcionem simultaneamente como emissores e receptores, na explicação desta secção assumimos que emissor é o nó que inicia a disseminação e receptor é o nó que retransmite as mensagens. O sensor emissor envia mensagens de depuração para a consola quando procede ao envio de mensagens para o sensor receptor e também quando recebe as retransmissões efectuadas. O sensor receptor envia mensagens de depuração para a consola quando recebe uma transmissão do sensor emissor e quando retransmite as mensagens. As mensagens enviadas para a rede transportam informação aleatória no campo de dados.

6.1.2 Testes realizados

Nesta secção são descritos os testes que permitiram tirar conclusões relativamente à exequibilidade do algoritmo Pampa em sensores reais.

Primeiro teste. Envio de vinte mensagens com os sensores estacionários. Uma vez que o UID das mensagens enviadas nunca é repetido o mecanismo de cancelamento de retransmissão não é activado. Neste cenário todas as mensagens recebidas foram retransmitidas pelo sensor receptor. A ordem de retransmissão foi a mesma pela qual as mensagens foram recebidas. Tal deve-se ao facto do valor RSSI não ter variado.

Segundo teste. Para testar a ordenação da lista de mensagens a retransmitir e o cálculo do tempo de retransmissão, é necessário fazer variar o RSSI das mensagens recebidas. Para este teste os sensores estão inicialmente estacionários. Sensivelmente a meio da transmissão das vinte mensagens o sensor emissor é movimentado para o lado oposto relativamente ao sensor receptor. Pelo meio está uma torre de computador. Sensivelmente após a transmissão da mensagem número 15 o sensor emissor volta à posição inicial. Neste teste, o obstáculo diminuiu o valor do RSSI das mensagens recebidas no sensor receptor, o que significa que estava mais longe. Assim, as mensagens que foram transmitidas enquanto havia um obstáculos entre os sensores foram as primeiras a serem retransmitidas, reflectindo o comportamento esperado.

Terceiro teste. Para testar o cancelamento das retransmissões, e as verificações das estruturas de dados, os números de sequência foram manipulados de forma a serem repetidos. A repetição dos números de sequência emula a recepção de cópias no sensor receptor. O tempo de retransmissão foi estendido para que

# mensagem	# sequência	Descrição
1	1	
2	1	cópia da mensagem 1
3	3	
4	4	
5	1	cópia da mensagem 1
6	6	
7	7	
8	8	
9	8	cópia da mensagem 8
10	10	
11	11	
12	11	cópia da mensagem 11
13	8	cópia da mensagem 8
14	14	
15	14	cópia da mensagem 14
16	1	cópia da mensagem 1 – cancelamento da retransmissão da mensagem 1
17	6	cópia da mensagem 6
18	18	
19	14	cópia da mensagem 14
20	8	cópia da mensagem 8 – cancelamento da retransmissão da mensagem 8

Tabela 6.1: Resultado da execução do Pampa num cenário propício ao cancelamento das retransmissões

seja possível receber o número de mensagens suficientes para cancelar a retransmissão.

A tabela 6.1 ilustra o resultado da execução do algoritmo Pampa configurado para cancelar a retransmissão das mensagens após a recepção de 3 cópias. A primeira coluna contém o número da mensagem recebida. A segunda coluna contém o número de sequência manipulado de cada mensagem recebida. De notar que o emissor é sempre o mesmo, pelo que mensagens com números de sequência iguais denotam mensagens com o mesmo UID. Por exemplo, a mensagem número 2 é tida como sendo uma cópia da mensagem número 1, pelo que no contexto deste teste, o receptor não recebeu uma mensagem 2 mas sim uma cópia da mensagem 1. As mensagens cuja retransmissão foi cancelada devido à recepção de cópias são representadas com as linhas de fundo preenchido. A tabela mostra que são recebidas várias cópias para diferentes mensagens. Porém, só as mensagens número 1 e 8 é que recebem as três cópias necessárias para que o mecanismo de cancelamento da retransmissão seja activado.

No final do tempo de espera são retransmitidas as mensagens para as quais não foram recebidas cópias suficientes para proceder ao cancelamento: 3, 4 , 6, 7, 10, 11, 14 e 18.

Quarto teste. O quarto teste consistiu em reduzir o tempo de retransmissão das mensagens de tal forma que fosse possível o nó emissor estar a enviar mensagens e o nó receptor a retransmitir as mensagens recebidas simultaneamente. O objectivo do teste consiste em testar eventuais problemas de conflito entre as mensagens enviadas e as mensagens recebidas. Não se verificaram conflitos, todas as mensagens foram enviadas e retransmitidas correctamente.

Quinto teste. Os testes foram repetidos com a diminuição do intervalo de tempo entre o envio de cada mensagem. Por vezes, para valores inferiores a 0.03 segundos algumas das mensagens não são imprimidas na consola do emissor aquando do envio, mas são imprimidas na consola do receptor aquando da recepção. Este cenário espelha a interferência que existe entre as mensagens de rede e as mensagens de depuração. A redução do intervalo de tempo foi levada a 0.008 segundos. O algoritmo revelou-se funcional para este valor, mas com o mesmo problema de impressão de mensagens. Para este valor, existe ainda um outro caso, mensagens que por vezes não são imprimidas em nenhuma circunstância. Para estes casos não é possível saber se foi apenas uma falha na impressão da mensagem, ou se realmente em algum ponto do envio a mensagem enviada para a rede sofreu interferência da mensagem de depuração.

Sexto teste. Para testar eventuais fugas de memória, o algoritmo foi deixado a executar até chegar aproximadamente às 60000 mensagens enviadas e retransmitidas. A execução correu sem problemas aparentes, uma vez que o comportamento do algoritmo foi o esperado até ser terminado.

6.1.3 Resultados

O número de sensores usados na experimentação desviam-se um pouco do que seria a realidade de centenas ou mesmo milhares de sensores a operarem na mesma rede. Também os testes não demonstram uma grande fidelidade, contudo, são suficientes para concluir que o Pampa é exequível em sensores reais. Uma vez que o Pampa é a base da PCACHE, é possível ter também uma noção da exequibilidade da PCACHE, sendo que esta fica condicionada apenas pela quantidade de memória disponível no sensor. Do mesmo modo, uma vez que a PCACHE é a base do sistema Pub/Sub proposto no Cap. 4, é possível assumir a

exequibilidade do sistema Pub/Sub em sensores reais, sendo que esta é condicionada apenas pela memória disponibilizada pelos sensores.

De notar que é espectável que o desempenho do algoritmo aumente em função da redução do uso de recursos originada pela remoção das mensagens de depuração e respectivos componentes.

6.2 Sistema Pub/Sub

Nesta secção apresentamos e discutimos a avaliação do sistema Pub/Sub. O desempenho da solução proposta é comparado com uma versão melhorada da inundação de eventos que usa o algoritmo de difusão PAMPA. As aproximações por inundação de eventos têm custo nulo na difusão das subscrições mas um custo que se espera mais elevado por publicação. Importa por isso perceber o ponto a partir do qual o equilíbrio da nossa solução apresenta um custo inferior ao da inundação, sabendo que a solução apresentada terá um custo superior (em número de transmissões) à inundação de subscrições. No entanto, importa referir que a inundação de subscrições troca a replicação parcial pela replicação total das subscrições, o que representa um aumento do consumo de memória que pode não ser comportável face aos limitados recursos dos dispositivos.

A avaliação é realizada utilizando um protótipo do sistema Pub/Sub estudado desenvolvido para a v. 2.34 do simulador de redes *ns-2* [48].

O mesmo protótipo é utilizado para avaliar o desempenho da inundação de eventos, através da contabilização do número médio de retransmissões realizadas pelo algoritmo PAMPA para a disseminação das subscrições. A contabilização do número médio de retransmissões é efectuado para cada uma das diferentes áreas, uma vez que a densidade da rede influencia os resultados obtidos. Desta forma é garantida a equidade dos valores obtidos entre a abordagem de inundação de eventos e o sistema Pub/Sub.

O protótipo utiliza a concretização do AODV que acompanha o simulador. Na Sec. 3.1 foi descrito um mecanismo que invalida as rotas após um determinado período de tempo. O mecanismo foi desactivado por não apresentar vantagens em cenários sem movimento, uma vez que, mesmo após longos períodos de tempo os nós se mantêm no mesmo local e por isso é pouco provável que se suceda uma quebra de rota.

6.2.1 Métricas

De modo a estudar os resultados face aos objectivos propostos foram utilizadas quatro métricas, as quais são descritas em seguida:

Número de mensagens transmitidas O número de mensagens transmitidas é uma métrica importante dado que é um factor que afecta significativamente o tempo de vida dos dispositivos, o que influência o tempo de vida da rede. Esta métrica reflecte o número acumulado de transmissões efectuadas por todos os nós da rede.

Taxa de entrega A taxa de entrega é definida como sendo a proporção de eventos que são entregues aos nós subscritores interessados. Esta métrica permite aferir a eficiência do sistema Pub/Sub em termos do seu objectivo básico: fazer chegar a informação a todos os nós interessados.

Latência Esta métrica permite perceber qual o atraso na entrega das subscrições. Mais concretamente, esta métrica traduz o tempo em segundos decorrido deste o inicio da operação de recolha de subscrições até à recepção dos eventos em cada um dos nós interessados.

Reutilização de rotas A reutilização de rotas refere-se à optimização realizada no sistema Pub/Sub que tem como objectivo reduzir o número de mensagens enviadas através do agrupamento de eventos. Esta métrica possibilita perceber qual o impacto da optimização em termos da redução do número de mensagens.

6.2.2 Cenários de simulação

Os cenários de simulação são compostos por 100 nós uniformemente dispostos por regiões com 8 dimensões distintas, definindo desta forma redes com diferentes densidades. A densidade da rede diminui com o aumento da área de simulação. Em todos os gráficos, as áreas são arrumadas por ordem crescente. Cada participante dispõe de uma interface de rede sem fios IEEE 802.11b a 11Mb/s, com um raio de transmissão fixo de 250 metros. Desta forma, as diferentes dimensões da rede simulada apresentam também comprimentos de rotas distintos.

Em todos os testes efectuados, os eventos comportam 100 bytes de informação. Os eventos enviados têm um tamanho base de 107 bytes, correspondentes à informação e aos cabeçalhos do algoritmo Pub/Sub. Por cada subscritor na lista de subscritores acrescem 4 bytes ao tamanho base.

Para garantir uma maior fiabilidade dos resultados, estes resultam da execução de 20 simulações para cada área de simulação, contabilizando um total de 160 simulações. Cada simulação combina diferentes disposições dos nós na área de simulação, diferentes escolhas dos atributos nas subscrições e nas publicações, e momentos de subscrição e publicação distintos.

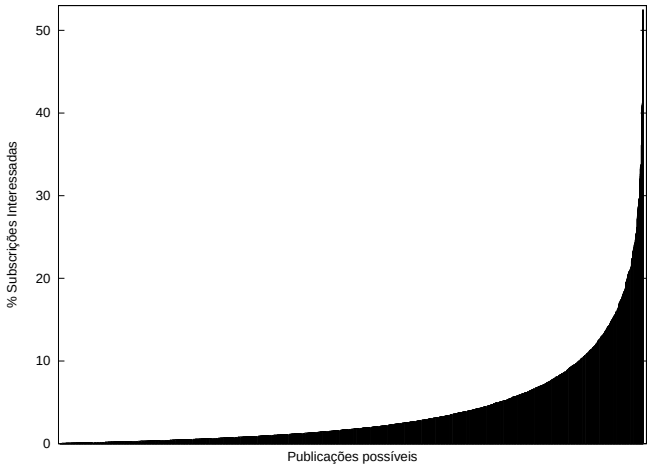


Figura 6.1: Interesse das subscrições face às publicações produzidas

Cada simulação tem a duração de 87120 segundos e é decomposta em 2 fases. Nos primeiros 660 segundos, 50 nós são incumbidos de realizarem uma subscrição. O momento de cada subscrição é seleccionado aleatoriamente, reservando-se os últimos 60s para a terminação das operações de subscrição. A partir dos 660s, são efectuadas 1440 publicações a uma média de uma publicação por minuto. Uma vez mais, não são iniciadas publicações nos últimos 60s, assegurando desta forma o tempo necessário para a entrega das publicações a todos os subscritores.

Para simular um modelo baseado em conteúdos, cada publicação é representada por um tuplo de 4 atributos da forma $< x_1, x_2, x_3, x_4 >, x_i \in [1, 9]$. Os valores de cada atributo são independentes, gerados aleatoriamente com uma distribuição uniforme. As subscrições por sua vez impõem restrições aos 4 atributos, na forma $< [y_1, y_1 + 3], [y_2, y_2 + 3], [y_3, y_3 + 3], [y_4, y_4 + 3] >, y_i \in [1, 6]$, com y_i a ser determinado aleatoriamente de acordo com uma distribuição zipf. Esta distribuição foi escolhida por ter sido demonstrado que representa por exemplo, índices de popularidade de web sites [49], cenário que se aproxima das aplicações antecipadas para os sistemas Pub/Sub. A combinação da distribuição uniforme das publicações com a distribuição zipf das subscrições é ilustrada na Fig. 6.1, onde para cada uma das 6561 (9^4) publicações possíveis é apresentada a proporção de 10000 subscrições que satisfaz.

De notar que nos testes realizados os nós não apresentam períodos de inactividade.

6.2.3 Parâmetros de configuração

Os valores utilizados para a configuração da PCACHE são os mesmos utilizados em [38], com a excepção da memória que a PCACHE assume ter disponível. Para não prejudicar a avaliação da qualidade de distribuição em condições ideais, o protótipo assume que os dispositivos têm memória suficiente para guardar todas as subscrições que a PCACHE determina, sabendo-se que serão sempre uma pequena fracção do total [38].

Os eventos são enviados para a rede de forma cadenciada de acordo com os mecanismos descritos na Sec. 4.3. Os valores dos atrasos aplicados em cada situação são também discutidos na secção anteriormente mencionada e iguais em todas as simulações. Importa perceber que o envio de forma cadenciada tem como objectivo diminui a ocorrência de situações anómalas na rede, resultante de tráfego excessivo, mas cujo efeito colateral resulta num aumento do tempo decorrido desde a operação de pesquisa até à entrega do evento. Tendo as vantagens e desvantagens desta optimização em mente, os valores discutidos na Sec. 4.3.3 parecem ser equilibrados em relação ao tempo de espera e consequente aumento da latência.

6.2.4 Resultados

Esta secção mostra e discute os resultados obtidos na simulação do sistema Pub/-Sub proposto. Nesta secção o algoritmo PAMPA denota a abordagem de inundação de eventos.

Número de mensagens enviadas

A Fig. 6.2 mostra o número de mensagens enviadas em função da área de simulação. Para o sistema Pub/Sub, é ainda representada a contribuição dos diferentes protocolos utilizados para o número total de transmissões. A figura mostra que o número total de transmissões das duas soluções avaliadas evolui em sentidos opostos com a densidade. Em particular, é notória a vantagem do sistema Pub/Sub (em termos do número de mensagens necessárias) a partir da área de rede $2000m \times 1000m$. Este resultado é atribuído simultaneamente ao aumento verificado do número de transmissões PAMPA e à redução do número de mensagens AODV.

O PAMPA regista um aumento do número de transmissões com a diminuição da densidade da rede, uma vez que este adapta a proporção de nós que retransmitem as mensagens à densidade da rede [39]. Por outras palavras, para redes menos densas o PAMPA requer mais transmissões de forma a que as mensagens

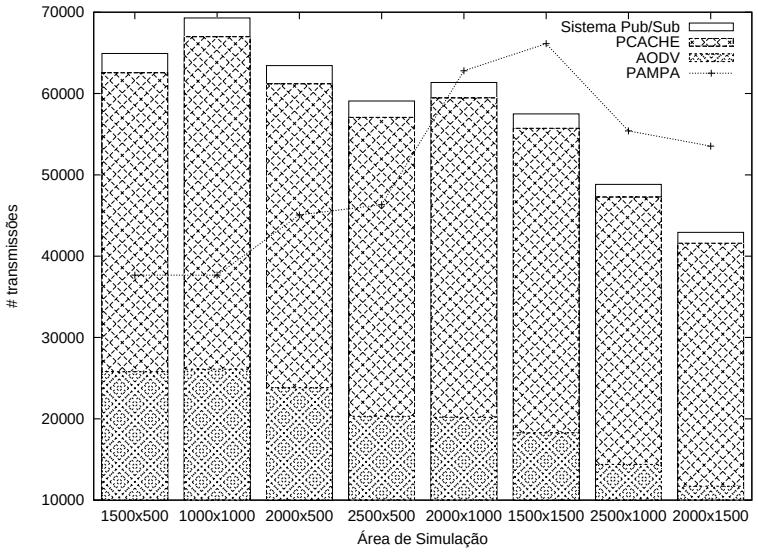


Figura 6.2: Número total de transmissões

possam chegar aos nós que com a redução da densidade se encontram mais afastados.

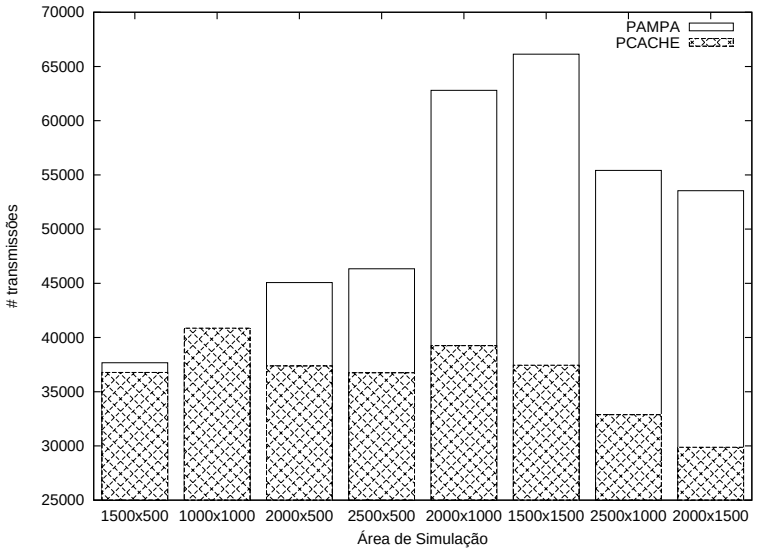


Figura 6.3: Comparação de transmissões entre a PCACHE e o PAMPA

É importante observar que para a área $1500m \times 500m$, o número de mensagens PCACHE por si só, é semelhante ao número de mensagens PAMPA. A Fig. 6.3 ilustra este cenário com mais pormenor apresentando uma comparação directa entre o número de transmissões requeridas pelo PAMPA e o número de transmissões requeridas pela PCACHE. De notar que o aumento do número de mensagens PAMPA influência o número de mensagens PCACHE, nomeadamente, nas operações de disseminação e pesquisa de subscrições.

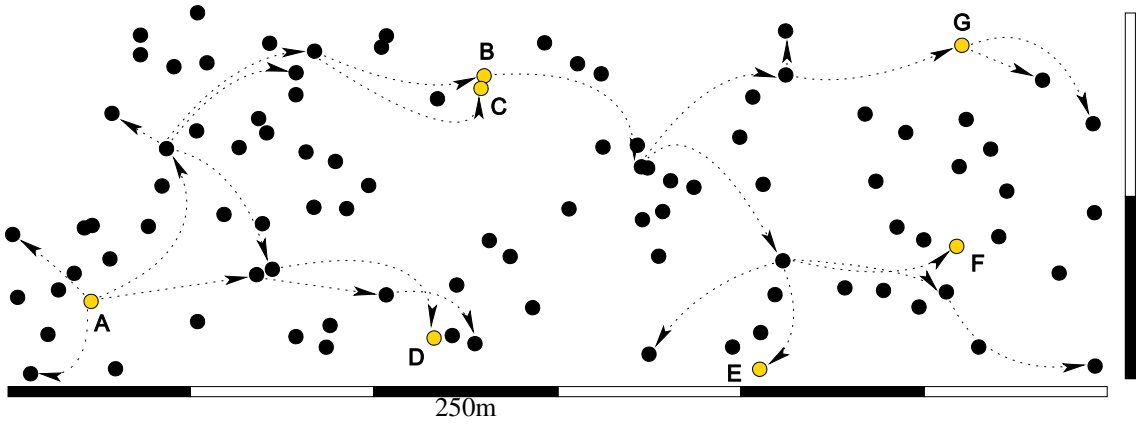


Figura 6.4: Abrangência da operação de pesquisa de subscrições para uma área de $1500m \times 500m$

O facto do número de mensagens ser tão semelhante para a área $1500m \times 500m$ deve-se ao número de saltos necessários para efectuar a operação de pesquisa aliado a uma área de rede reduzida. Para justificar esta afirmação importa recordar a Fig. 6.4 apresentada inicialmente na Sec. 4.3.2 onde foi apresentado um cenário retirado de uma simulação que ilustra a disseminação de subscrições para as mesmas condições de rede, nomeadamente, uma área de $1500m \times 500m$ e os mesmos 100 nós. A mesma figura pode ilustrar a operação de pesquisa, uma vez que está configurada para realizar o mesmo número de saltos, os nós têm o mesmo raio de alcance e é usado o mesmo mecanismo de difusão. No contexto da operação de pesquisa de subscrições com três saltos se um dos nós B ou C iniciar a operação de pesquisa é perceptível que esta abrange uma grande maioria dos nós da rede. Por esta razão, o número de mensagens necessárias para a operação de pesquisa deverá ser muito semelhante ao número de mensagens que a inundação de eventos requer, o que explica o facto de a PCACHE só por si ter um desempenho semelhante (em número de mensagens) em relação ao PAMPA.

Observando a Fig. 6.3 com mais detalhe surgem duas questões: a área de $1000m \times 1000m$ apesar de ser maior que a área $1500m \times 500m$, requer mais mensagens PCACHE; a área de $1000m \times 1000m$ apesar de ter uma densidade igual à área $2000m \times 500m$, requer mais mensagens PCACHE. Esta situação não invalida a explicação anterior, e é atribuída ao facto de os nós situados em áreas quadradas conseguirem chegar a mais nós com o mesmo número de retransmissões. Esta afirmação é suportada pelo aumento do esforço do algoritmo PAMPA (em número de transmissões) verificado da área $1000m \times 1000m$ para a área $2000m \times 500m$, apesar de representarem a mesma área de simulação efectiva.

Porém, com a diminuição da densidade da rede, o número de saltos limitado da operação de pesquisa tenderá a surtir efeito na redução do número de mensagens, uma vez que os três saltos irão abranger uma área progressivamente mais

limitada da rede. Esta afirmação é suportada pelo crescimento aproximadamente linear do número de mensagens que a PCACHE exibe quando comparada com o PAMPA, até às duas maiores áreas. A redução do número de mensagens observada nos testes realizados nas duas menores densidades é atribuído a partições de rede pelo que os resultados não são discutidos.

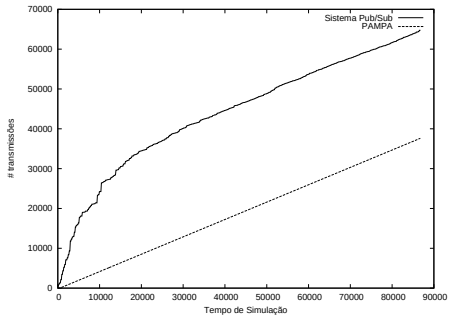
A Fig. 6.5 ilustra com mais detalhe o número de mensagens requeridas pelo sistema Pub/Sub e pela inundação dos eventos ao longo do tempo de simulação, para cada uma das áreas de simulação. Nesta imagem é perceptível a convergência do número de mensagens requeridas pelo sistema Pub/Sub com o número de mensagens requeridas pela abordagem de inundação de eventos à medida que a densidade da rede diminui.

O PAMPA começa com uma vantagem em qualquer dos cenários, uma vez que não requer a transmissão de mensagens durante o período de tempo destinado ao envio de subscrições. O sistema Pub/Sub por oposição inicia o seu funcionamento pelo envio de subscrições.

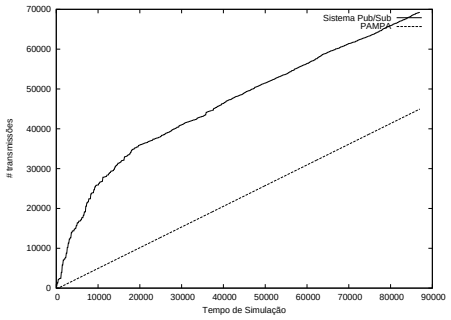
O número de mensagens Pub/Sub sofre um aumento abrupto de mensagens no inicio de cada simulação. Este acontecimento é atribuído às dispendiosas operações de descoberta de rota realizadas pelo AODV sempre que um evento é enviado pela primeira vez a um determinado destinatário. Esta operação é executada predominantemente nos primeiros 10000 segundos de simulação, uma vez que é efectuada apenas na primeira vez que um evento é enviado para um novo destinatário. Deste modo, eventos com destino a nós que já tenham despoletado uma operação de descoberta de rota não necessitam de nova informação de encaminhamento e portanto requerem um menor número de mensagens.

Enquanto a solução de inundação de eventos apresenta um custo (em número de mensagens) fixo por publicação para cada densidade, o sistema proposto tende a reduzir o custo com o aumento do número de publicações uma vez que o custo da descoberta de rotas é progressivamente menor. Em particular, uma vez que os nós publicadores conheçam uma rota para todos os subscritores a operação deixa de ser efectuada. Este facto juntamente com o reduzido número de mensagens necessárias para enviar os eventos ponto-a-ponto sugere que o sistema proposto poderá obter resultados (em número de mensagens) melhores do que os apresentados, mesmo para densidades mais elevadas.

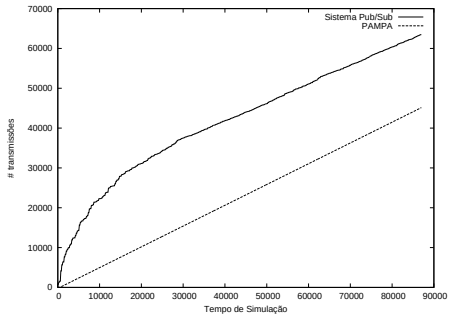
Para as quatro densidades mais elevadas representadas nas Fig. 6.5(e), 6.5(f), 6.5(g) e 6.5(h) o sistema Pub/Sub proporciona uma vantagem na quantidade de transmissões ao fim de um determinado tempo observado de simulação, que tende a reduzir com a diminuição da densidade.



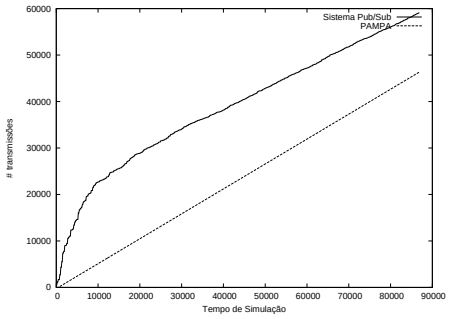
(a) [Transmissões para a área $1500m \times 500m$



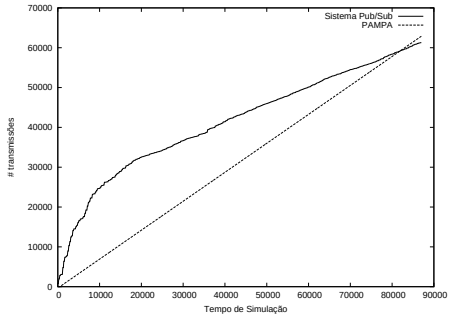
(b) Transmissões para a área $1500m \times 500m$



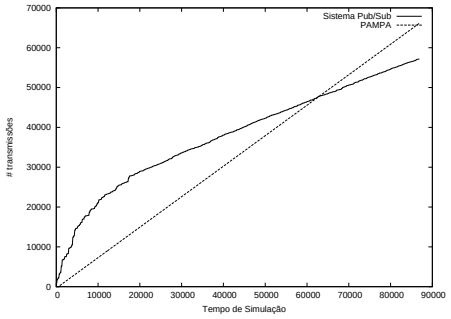
(c) Transmissões para a área $2000m \times 500m$



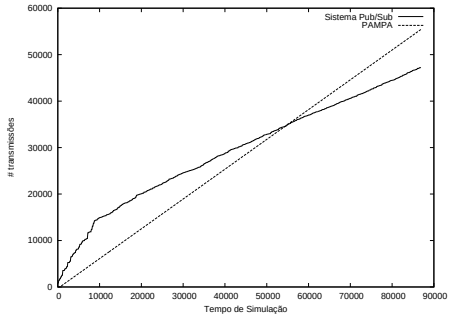
(d) Transmissões para a área $2500m \times 500m$



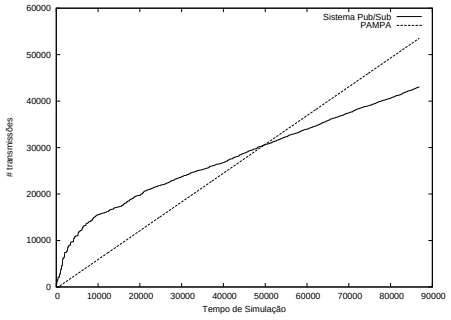
(e) Transmissões para a área $2000m \times 1000m$



(f) Transmissões para a área $1500m \times 1500m$



(g) Transmissões para a área $2500m \times 1000m$



(h) Transmissões para a área $2000m \times 1500m$

Figura 6.5: Comparação do número de transmissões do sistema Pub/Sub em relação ao PAMPA

Reutilização de rotas

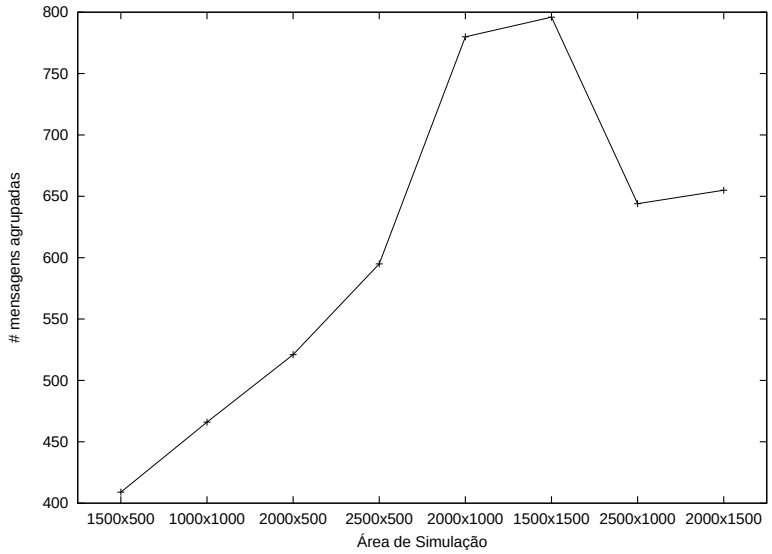


Figura 6.6: Reutilização das rotas

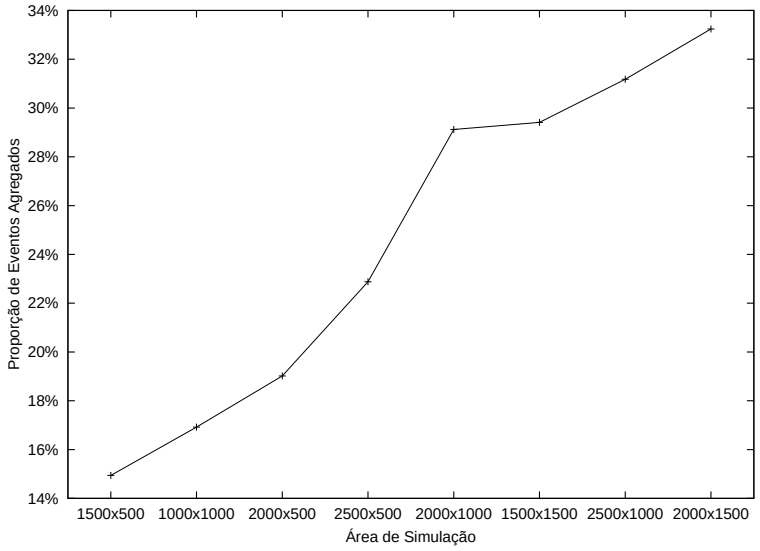


Figura 6.7: Proporção de eventos agrupados

A Fig. 6.6 permite visualizar o número de eventos agregados, que reflecte o número de mensagens que não foram enviadas pelo publicador devido à optimização concretizada.

A redução do número de mensagens enviadas para a rede, traduz-se numa diminuição de transmissões AODV que justifica a redução do número de mensagens verificado na Fig. 6.2.

Como pode ser observado na Fig 6.6, a diminuição da densidade da rede é acompanhada de um aumento do número de eventos agregados. Com o aumento da distância entre os dispositivos, o número de vizinhos de cada nó diminui. Deste modo, o número de nós que o AODV pode seleccionar para a construção de rotas também diminui aumentando a probabilidade das rotas partilharem o mesmo nó seguinte. De relembrar que o factor que possibilita a agregação de eventos, é a partilha do mesmo nó seguinte. Deste modo, o aumento da probabilidade de duas ou mais rotas partilharem o mesmo nó seguinte resulta num aumento de eventos agregados.

Embora a ocorrência de partições nas áreas $2500m \times 1000m$ e $2000m \times 1500m$ não permitam retirar conclusões directas, é possível retirar conclusões através do cálculo da proporção entre o número de eventos enviados, e o número de eventos agregados. A Fig. 6.7 reflecte este cálculo e clarifica o constante aumento do número de eventos agregados à medida que a densidade diminui. Para a menor densidade testada é verificado um agrupamento que excede os 33%.

Para um aumento do número de publicações é espectável que a agregação de eventos também aumente. Porém, a proporção de eventos agregados deve manter-se estável para cada densidade, uma vez que é a densidade da rede que mais influencia a proporção de eventos agregados.

Taxa de entrega

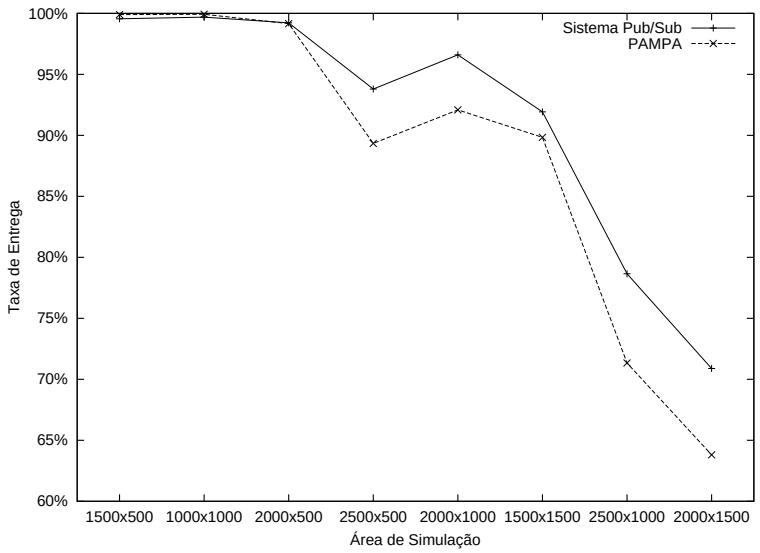


Figura 6.8: Taxa de entrega

A Fig. 6.8 apresenta a taxa de entrega de eventos para o sistema Pub/Sub e a taxa de entrega de subscrições conseguida pelo PAMPA.

É perceptível na imagem um padrão de decaimento da taxa de entrega dos eventos à medida que a taxa de entrega do PAMPA também decresce. A quebra progressiva da taxa de entrega com a diminuição da densidade é atribuída ao isolamento de alguns dispositivos. Na presença de dispositivos isolados a disseminação das subscrições não é realizada apropriadamente, impedindo que a informação seja armazenada segundo a distribuição esperada. No limite, um subscritor pode estar isolado dos restantes nós da rede, impossibilitando sequer que algum nó na rede tenha conhecimento da sua subscrição.

A quebra na taxa de entrega do sistema Pub/Sub deve-se à incapacidade da operação de recolha de subscrições em reunir toda a informação sobre as subscrições uma vez que a sua disseminação não foi correctamente realizada. Como consequência os nós publicadores ficam apenas com um conhecimento parcial sobre os subscritores, impedindo o envio de eventos para os subscritores dos quais não obteve qualquer informação de subscrição. Este cenário é reflectido na Fig. 6.9, que ilustra a taxa de entrega de subscrições recolhidas pela PCACHE na operação de recolha de subscrições. A taxa de entrega de subscrições representa a percentagem de subscrições correctamente recolhidas na operação de pesquisa e entregues ao sistema Pub/Sub, em relação àquele que sabemos ser o conhecimento completo. É possível observar que com o aumento da área de simulação a PCACHE tem mais dificuldade em informar o sistema Pub/Sub sobre o conjunto ideal de subscrições, devido ao processo de subscrição. A figura mostra também a taxa de entrega do AODV, que reflecte a percentagem de eventos correctamente entregues aos respectivos subscritores. A taxa de entrega do AODV é bastante elevada, sendo que a influência do AODV na taxa de entrega do sistema Pub/Sub é mínima. Porém, importa perceber que a causa da taxa de entrega do AODV ser tão elevada deve-se ao facto de o AODV só ter que entregar eventos a nós que a PCACHE conseguiu contactar, ou seja, os que não se encontram isolados.

Como era esperado, o padrão do conjunto de subscrições comunicadas ao sistema Pub/Sub pela PCACHE juntamente com os eventos que foram entregues pelo AODV aos subscritores é igual ao padrão verificado para a taxa de entrega do sistema Pub/Sub.

Em condições nas quais o isolamento de nós não ocorra a taxa de entrega é superior a 90%. De salientar que a taxa de entrega do sistema Pub/Sub é superior à taxa de entrega do algoritmo PAMPA. Este resultado é atribuído por um lado ao número mais elevado de entregas que o PAMPA tem que realizar. Por outro, pode ocorrer um problema conhecido como *overcancellation* [50], que consiste no cancelamento precoce da disseminação em nós chave para a continuação da disseminação, impedido que a disseminação da informação chegue a todos os

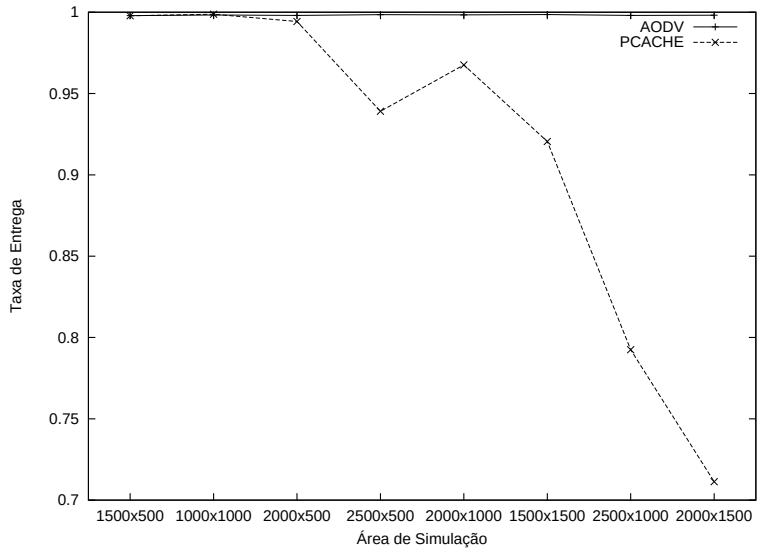


Figura 6.9: Taxa de entrega das subscrições pela PCACHE e dos eventos pelo AODV

nós. Este problema não é notado na disseminação de subscrições, uma vez que a informação acaba por ser guardada noutros nós.

Em áreas maiores cuja densidade da rede seja próxima da observada nas áreas mais pequenas dos cenários simulados, é esperado que o sistema Pub/Sub mantenha a taxa de entrega acima dos 90% uma vez que o processo da PCACHE não é afectado pelo número de nós na rede. Em particular, serão precisas mais retransmissões para a operação de subscrição e as mesmas transmissões para a operação de pesquisa, que na ausência de nós isolados não têm qualquer influência na taxa de entrega. É previsto que o AODV mantenha também um bom desempenho para uma rede com mais nós, sendo que é potencialmente afectado na fase inicial relativamente ao maior número de mensagens necessárias na operação de descoberta de rotas.

Latência

O processo de recolha de subscrições levado a cabo pelo sistema Pub/Sub introduz naturalmente algum atraso que não é possível colmatar. O publicador introduz também um atraso no envio dos eventos quando espera pelo conjunto de todas as subscrições para começar a enviar eventos. O processo de envio de eventos é o terceiro e último mecanismo que introduz um atraso no envio dos eventos, desta feita para proceder ao envio cadenciado da informação de forma a evitar perturbações na rede. Ainda relativamente ao processo de envio dos eventos é importante referir que a operação de descoberta de rota introduz também um atraso no envio dos eventos. Os resultados apresentados para a latência da

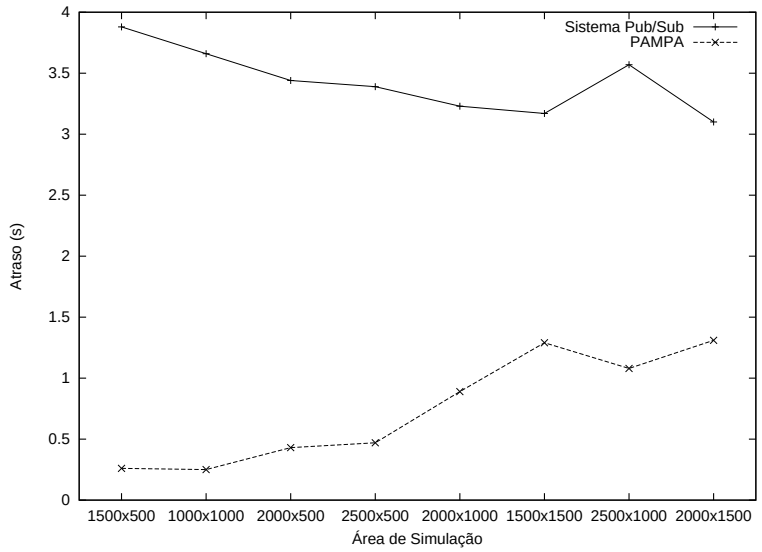


Figura 6.10: Latência

disseminação na Fig. 6.10, confirmam estas expectativas.

Os valores para o sistema Pub/Sub medem o tempo médio decorrido o início da operação de recolha de subscrições e a entrega dos eventos a cada subscritor. No caso do PAMPA, mede-se o tempo decorrido entre o inicio da disseminação dos eventos e a sua entrega ao último nó da rede.

O gráfico sugere a existência de uma convergência entre as duas aproximações avaliadas. No caso do sistema Pub/Sub, a diminuição consecutiva da latência à medida que a densidade diminui é atribuída à reutilização de rotas. Tal deve-se ao facto de cada evento ser sujeito a um atraso para assegurar o envio cadenciado de mensagens. Os eventos agregados perante esta optimização de envio cadenciado contam como sendo um evento, e deste modo à medida que as agregações aumentam, diminuem os eventos sujeitos ao atraso de optimização. A excepção verifica-se para a área $2500m \times 1000m$, onde a latência aumenta contrariando a tendência. A causa desta excepção não é clara, sendo que é um ponto a marcar para estudos futuros.

No caso do PAMPA, o crescimento resulta do aumento do número de transmissões e do algoritmo executado pelo protocolo, uma vez que cada dispositivo aplica algum atraso à sua própria retransmissão [39].

6.3 Sumário

Este capítulo apresentou os testes realizados em sensores reais que servem como base para uma determinação da exequibilidade do algoritmo Pampa. Os testes realizados à implementação do algoritmo PAMPA para o sistema operativo

TinyOS divergem em relação ao número de sensores esperado nas RSSF. Divergem também no rigor que seria esperado, devido ao facto do próprio mecanismo de depuração interferir no funcionamento do algoritmo. Porém, as condições reunidas e os resultados dos testes efectuados, são suficientes para afirmar que o PAMPA é exequível em sensores reais. Uma vez que o PAMPA é o mecanismo base da PCACHE e sendo o PAMPA exequível em sensores reais, é possível inferir também a exequibilidade da PCACHE e do sistema Pub/Sub em sensores reais, sendo que poderá ser apenas condicionada pela memória necessária.

A abordagem de inundação de subscrições tem um funcionamento semelhante à distribuição geográfica de subscrições mas requer um menor número de mensagens para realizar as mesmas operações de comunicação. Porém, o consumo de memória é mais elevado na inundação de subscrições, facto que pode tornar o sistema incomportável.

A avaliação entre o extremo de inundação de eventos e a disseminação geográfica de subscrições foi realizada recorrendo a simulações. A avaliação do sistema Pub/Sub proposto que recorre à disseminação geográfica de subscrições, demonstrou que para redes com menor densidade ou com uma maior área o sistema Pub/Sub requer menos mensagens que a abordagem que realiza inundação de eventos. Ao contrário da inundação de eventos, o sistema Pub/Sub não requer um aumento linear de mensagens, pelo que sugere ser adequado para redes de longa duração, como é o caso das redes de sensores.

Tendo atingido a redução do número de mensagens, é seguro afirmar que em algumas situações o sistema Pub/Sub com disseminação geográfica das subscrições é mais económico em termos do gasto energético requerido, o que proporciona um aumento do tempo de vida da rede. Importa referir que a redução da utilização de memória comparativamente à inundação de subscrições é atingida com sucesso através da disseminação geográfica das subscrições.

Relativamente à taxa de entrega, a avaliação demonstrou que na ausência de partições de rede, o sistema proposto apresenta um bom desempenho sendo a taxa de entrega superior a 90%, uma taxa de entrega em alguns casos superior à conseguia pelo algoritmo PAMPA. O isolamento de alguns dispositivos revelou a incapacidade de uma disseminação apropriada das subscrições, o que proporcionou aos subscritores um conhecimento incompleto das subscrições. Este cenário impediu os publicadores de enviarem eventos para os subscritores dos quais não tinham conhecimento. O AODV também revelou ter uma influência na taxa de entrega, ainda que mínima. Deste modo, a grande causa da quebra da taxa de entrega deve-se ao isolamento de alguns dos nós. Deste modo é esperado que para

redes com um maior número de nós e com uma densidade moderada o sistema tenha ainda uma taxa de entrega superior a 90%.

A optimização relativa à agregação de eventos revelou-se uma mais valia para a obtenção dos objectivos propostos. Esta optimização permite reduzir o número de mensagens enviadas para a rede. Observaram-se reduções do número de mensagens na ordem dos 33%, sendo que estes valores são directamente reflectidos na redução do número de mensagens requeridas pelo sistema Pub/Sub.

O sistema Pub/Sub está sujeito a uma série de mecanismos que por motivos de optimizações em termos do número de mensagens acabam por aumentar a latência do sistema. O PAMPA por sua vez conta apenas com o atraso que é inerente ao seu funcionamento. Assim, a latência apresentada pelo sistema Pub/Sub é bastante superior à do PAMPA, facto que não constituiu uma surpresa. Porém, a optimização de agregação de eventos revelou ter uma importante contribuição na redução da latência do sistema Pub/Sub.

Capítulo 7

Conclusão

As RSSF são tipicamente compostas por dispositivos com restrições a nível do poder de processamento, memória e energia. A cooperação entre os dispositivos que compõem as RSSF permite que estas dispensem a existência de uma infra-estrutura de suporte à comunicação. Estas têm como objectivo primário a monitorização, que é realizada através da recolha e envio dos dados obtidos do ambiente onde estão inseridas para os nós *sink*. Em muitos cenários de aplicação os sensores podem não estar facilmente acessíveis devido aos locais onde operam, ou devido ao seu grande número. Em qualquer um dos casos, a substituição de baterias pode ser impraticável, o que sugere que o dispêndio energético deve ser tão reduzido quanto possível de forma a prolongar o tempo de vida da rede.

Os nós sensores normalmente usam a sua limitada capacidade de processamento para avaliação da relevância da informação obtida. Porém, esta avaliação é imparcial, ou seja, é seguido o mesmo critério de avaliação para toda a informação independentemente do nó receptor.

Com o aumento do número de sensores, aumenta também o volume e a diversidade de informação recolhida. Adicionalmente, com a especialização dos nós *sink* surge a necessidade de conseguirem seleccionar informação independentemente dos demais nós da rede. A avaliação imparcial é configurada em tempo de produção, pelo que não é adequada à necessidade de especificação independente de interesses. A incapacidade de incorporar a selecção de informação de interesse no modelo de comunicação pode levar ao envio desnecessário de um elevado número de mensagens nos casos em que nenhum nó está interessado na informação enviada, o que implica um gasto energético não negligenciável e consequente diminuição de longevidade da rede.

O paradigma de comunicação Pub/Sub permite colmatar as necessidades de especificação de interesses dos nós *sink*. Neste modelo de comunicação os nós interessados na informação, ou seja, os nós "subscritores" indicam os seus interesses ao sistema através de subscrições. Os nós "publicadores" enviam exclusi-

vamente informação de interesse na forma de eventos através de publicações. Os sistemas Pub/Sub ao permitirem a especificação de interesses reduzem o número de mensagens necessárias, uma vez que publicações para as quais não existam subscrições não implicam o envio de eventos.

Existem sistemas Pub/Sub centralizados que necessitam dos serviços de nós especiais bem conhecidos, e sistemas descentralizados, que recorrem exclusivamente aos serviços fornecidos pelos demais participantes da rede. Relativamente às opções descentralizadas existem dois extremos, um recorre à inundação de eventos e outro à inundação de subscrições. A inundação de eventos não necessita do envio de mensagens de subscrição, os nós interessados em receber informação esperam por uma eventual recepção de eventos que satisfaçam os seus interesses. A inundação de subscrições requer um elevado espaço de armazenamento na memória dos dispositivos de forma a armazenar todas as subscrições efectuadas no sistema, o que pode ser incomportável. Uma terceira abordagem foi considerada e consiste na disseminação geográfica das subscrições. Esta abordagem é semelhante à inundação de subscrições, contudo, requer uma menor utilização de memória dos dispositivos uma vez que as subscrições são armazenadas apenas num sub-conjunto dos nós que compõem a rede.

Este trabalho propõe um sistema Pub/Sub para RSSF que se adapta às restrições dos recursos impostas pelos sensores. O sistema proposto é descentralizado e utiliza o modelo de subscrição baseado em conteúdos, que provê um nível de expressividade elevado. Os eventos são enviados ponto-a-ponto através do protocolo de encaminhamento AODV, sendo que este foi adaptado aos objectivos do trabalho.

O sistema Pub/Sub proposto foi estudado através de simulações que permitiram obter resultados claros do seu benefício em termos dos recursos necessários às operações de comunicação. A redução do número de mensagens deve-se em parte às optimizações concretizadas, que revelaram ser uma mais valia para atingir os objectivos da redução do número de mensagens. Em particular, a optimização que permite o agrupamento de eventos faculta uma redução do número de transmissões até 33%. Com o aumento do número de publicações é espectável que a proporção de eventos agrupados se mantenha estável para cada densidade da rede. Com base nos resultados obtidos é seguro afirmar que o sistema Pub/Sub proposto é económico em termos do número de mensagens requeridas para as operações de comunicação relativamente à inundação de eventos. Porém, o sistema Pub/Sub é mais económico relativamente ao número de mensagens apenas em cenários nos quais as redes têm uma dimensão considerável como é o caso das RSSF, e cuja densidade da rede varia de moderada a reduzida.

Para reduzir a quantidade de memória utilizada nos dispositivos foi utilizada a estratégia que efectua a disseminação geográfica das publicações. A PCACHE foi a ferramenta utilizada para realizar as operações de disseminação e recolecção das subscrições. Era esperado à partida que a solução adoptada na concretização do sistema Pub/Sub requeresse mais mensagens que a abordagem de inundação de subscrições devido à necessidade de recolha de subscrições. O aumento do número de mensagens revelou ser um custo colateral inevitável que permite reduzir a quantidade da memória utilizada. Porém, mesmo com a necessidade de um maior número de mensagens em relação à inundação de subscrições verificou-se experimentalmente que o sistema Pub/Sub para algumas configurações de rede requer menos mensagens que a inundação de eventos.

Apesar da redução do número de mensagens necessárias, a taxa de entrega conseguida é superior a 90%. Este valor é considerado apenas para a ausência de partições na rede, onde o isolamento de alguns nós pode comprometer a distribuição da informação de subscrição e consequentemente a taxa de entrega. Porém, é de salientar que o sistema Pub/Sub tem uma taxa de entrega superior relativamente à abordagem de disseminação de eventos.

Os testes realizados relativamente à latência são elucidativos, e mostram uma vantagem da abordagem de disseminação de eventos sobre o sistema Pub/Sub. Os resultados obtidos sugerem que o sistema Pub/Sub não é adequado para fortes restrições temporais.

Este trabalho apresenta também uma concretização do algoritmo PAMPA para o sistema operativo TinyOS. Com a concretização do algoritmo PAMPA pretende-se por um lado passar do plano teórico e ideal para o plano real de modo a ter uma noção das reais dificuldades impostas pelas restrições de recursos. Por outro, pretende-se também ter uma noção parcial da exequibilidade do sistema Pub/Sub proposto. Com base em testes efectuados em sensores reais, concluiu-se que o algoritmo PAMPA é exequível. Uma vez que o algoritmo PAMPA é a base da PCACHE, e esta por sua vez é a base do sistema Pub/Sub, é esperado que o sistema Pub/Sub seja também exequível em sensores reais, ficando esta condicionada pela memória disponível no sensor.

Para trabalho futuro, é nosso objectivo optimizar o código fonte da concretização do PAMPA em sensores de forma a aumentar o desempenho e a reduzir o espaço necessário em memória. Seria também interessante experimentar a concretização numa *testbed* que permitisse constatar o desempenho real do algoritmo em sensores reais numa rede de larga escala. Relativamente ao estudo levado a cabo com o sistema Pub/Sub, planeamos variar alguns parâmetros e de-

senvolver outras optimizações, nomeadamente, estratégias para diminuição da latência do sistema e para uma maior redução do número de mensagens requeridas pelas operações de comunicação. O estudo do impacto de outros protocolos de encaminhamento e da exploração de outras sinergias com o AODV fazem também parte dos planos para futuros melhoramentos.

Apêndice A

Particularidades da concretização

A.1 Ambiente de desenvolvimento

Algumas das ferramentas de desenvolvimento propostas para a elaboração deste projecto são específicas para o desenvolvimento de aplicações TinyOS, outras são tipicamente usadas no desenvolvimento de aplicações para sistemas embebidos independentes de um sistema operativo, como é o caso das Atmel AVR Tools.

Actualmente o *TinyOS Core Working Group* suporta o TinyOS em duas plataformas: Cygwin(em ambiente Windows) e Linux. A plataforma de desenvolvimento adoptada foi a Cygwin para ambiente Windows. Tal escolha deve-se ao facto do fabricante dos sensores disponibilizar apenas ferramentas para este sistema operativo, algumas cruciais para a progressão dos trabalhos como por exemplo, o ICload. De seguida são descritas com maior detalhe as ferramentas utilizadas no desenvolvimento do algoritmo Pampa.

- Microsoft Windows XP Professional x86 versão 2002, com Service Pack 2
- **Cygwin versão 1.5.24** — fornece um ambiente semelhante em termos de funcionalidades ao Unix, nomeadamente uma *shell* e várias ferramentas como por exemplo perl e shell scripts essenciais para o ambiente TinyOS.

A.2 Compiladores nativos

O objectivo dos compiladores nativos baseia-se em compilar o código fonte para microcontroladores de baixo consumo energética, pelo que cabe aos compiladores nativos gerar o código assembly apropriado para as plataformas alvo.

Atmel AVR Tools

As Atmel AVR Tools são compostas pelos seguintes pacotes de software:

- **avr-binutils** — avr-binutils-2.17tinyos-3.cygwin.i386.rpm

- **avr-gcc** — avr-gcc-4.1.2-1.cygwin.i386.rpm
- **avr-libc** — avr-libc-1.4.7-1.cygwin.i386.rpm
- **avarice** — avarice-2.4-1.cygwin.i386.rpm
- **insight** — avr-insight-6.3-1.cygwin.i386.rpm
- **avrdude** — avrdude-tinyos-5.6cvs-1.cygwin.i386.rpm

PXA27x Tools (iMote2)

As PXA27x Tools são compostas pelos seguintes pacotes de software:

- **xscale-elf-binutils** — xscale-elf-binutils-2.15tinyos-1.cygwin.i386.rpm
- **xscale-elf-gcc** — xscale-elf-gcc-3.4.3-1.cygwin.i386.rpm
- **xscale-elf-newlibc** — xscale-elf-newlib-1.11.0tinyos-1.cygwin.i386.rpm
- **jtag**

TI MSP430 Tools

As TI MSP430 Tools são compostas pelos seguintes pacotes de software:

- **base** — msp430tools-base-0.1-20050607.cygwin.i386.rpm
- **python tools** — msp430tools-python-tools-1.0-1.cygwin.noarch.rpm
- **binutils** — msp430tools-binutils-2.16-20050607.cygwin.i386.rpm
- **gcc** — msp430tools-gcc-3.2.3-20050607.cygwin.i386.rpm
- **libc** — msp430tools-libc-20080808-1.cygwin.i386.rpm

A.3 TinyOS toolchain

As TinyOS-specific Tools são compostas pelo compilador nesC e por um conjunto de ferramentas desenvolvidas para o TinyOS. O compilador nesC é independente da plataforma para a qual o código fonte se destina: o seu resultado é passado para os compiladores nativos para que possa ser portado para uma plataforma específica devidamente optimizado.

TinyOS-specific Tools

- **nesC** — nesc-1.3.0-1.cygwin.i386.rpm
- **Deputy** — tinyos-deputy-1.1-1.cygwin.i386.rpm
- **tinyos-tools** — tinyos-tools-1.3.0-1.cygwin.i386.rpm

A.4 TinyOS 2.x source tree

Para compilar as aplicações TinyOS é necessário ter o código fonte deste sistema operativo. Ao compilar a aplicação, os componentes do sistema operativo necessários à execução da aplicação são também eles compilados. Da compilação é gerado um ficheiro binário que contém a o sistema operativo e a aplicação.

- **TinyOS** — tinyos-2.0.2-2.cygwin.noarch.rpm

A.5 Sensores utilizados

Os sensores utilizados no decorrer da concretização do algoritmo PAMPA para o sistema operativo TinyOS foram os *ICradio Stick 2.4G*.

De seguida apresenta-se a descrição do hardware dos sensores:

Microcontrolador: Atmel, ATmega1281 de 8-Bits a 16 MHz

Memória flash: 64K/128K/256K Bytes

EEPROM: 4K Bytes

SRAM: 8K Bytes

Requisitos energéticos: 0 - 8 MHz @ 2.7 - 5.5V, 0 - 16 MHz @ 4.5 - 5.5V

Comunicação: Chip rádio Atmel AT86RF230, ZigBee / IEEE 802.15.4 2.4GHz com sensibilidade até 103dBm, taxa de transferência até 250 kbps e alcance superior a 50m

Dimensões Físicas: 55.9 x 16.8 x 4.5 mm (incluindo conector USB)

Temperatura: -40.C to +85.C

Programador: JTAG

Interfaces: A interface do ATmega1281 está ligada à USB Bridge CP2102 Após a instalação de drivers permite a interacção com o dispositivo a através de um porto COM virtual com taxas de transferência até 1MBit

Bootloader: Está equipado com um bootloader que permite o envio de aplicações para o dispositivo através do porto COM

A.6 Outras ferramentas

- **ICload** – ICload V12, permite colocar o ficheiro binário resultante da compilação na memória de programa.

Apêndice B

Interface da concretização Pampa

B.1 Interface Pampa

```
1  /**
   * Interface of the PAMPA algorithm.
   *
   * @author Ricardo Mascarenhas
   */
6
   interface Pampa{

     /**
11     * Send a packet with a data payload to address
       * AM_BROADCAST_ADDR. To determine the maximum available size, use the
       * Pampa interface, maxMessageLenght() command. If send
       * returns SUCCESS, then the component will signal the sendDone
       * event in the future; if send returns an error, it will not
16     * signal the event.  Note that a component may accept a send
       * request which it later finds it cannot satisfy; in this case, it
       * will signal sendDone with error code.
       *
       * @param <b>msg</b>     the message
21     * @return        SUCCESS if the request to send succeeded and a
       *                sendDone will be signaled later, EBUSY if the
       *                abstraction cannot send now but will be able to
       *                later, FAIL if the communication layer is not
       *                in a state that can send (e.g., off), or ESIZE
26     *        if the <tt>msg</tt> length exceed the maxPayloadLength.
       *
       * @see           sendDone
       */
     command error_t sendMessage(pampa_msg_t * msg);
31
     /**
       * Signaled in response to an accepted send request. <tt>msg</tt> is
       * the message buffer sent, and <tt>error</tt> indicates whether
       * the send was successful.
36     *
       * @param  <b>msg</b>  the packet which was submitted as a send request
       * @param  <b>error</b> SUCCESS if it was sent successfully, FAIL if it was not,
       *                ECANCEL if it was cancelled
       *
41     */
     event void sendDone(pampa_msg_t * msg, error_t error);

     /**
       * Receive a packet buffer, returning a buffer for the signaling
46   * component to use for the next reception. The return value
     * can be the same as <tt>msg</tt>, as long as the handling
```

```
    * component copies out the data it needs.
    * Copies of messages will not be signaled, only the new will.
    *
    * <b>Note</b> The message received will always have the message size set
    * in the constant PAMPA_MESSAGE_SIZE which is on file PampaMsg.h
    *
    * @param  <b>msgRec</b>  the received message
    * @return          a message buffer for the stack to use for the next
    *                  received packet.
    */
    event void messageReceived(pampa_msg_t * msgRec);

    /**
    * Return the maximum length that can be used by PAMPA
    * message (pampa_msg_t). This command behaves identically to
    * <tt>Packet.maxPayloadLength</tt>.
    *
    * <b>Note</b> The size returned is not the maximum size that the communication
    * layer can provide
    *
    * @return the maximum message length
    */
    command uint8_t maxMessageLength();

    /**
    * Start this component and all of its subcomponents.  Return
    * values of SUCCESS will always result in a <code>startDone()</code>
    * event being signalled.
    *
    * @return SUCCESS if the device is already in the process of
    *         starting or the device was off and the device is now ready to turn
    *         on.  After receiving this return value, you should expect a
    *         <code>startDone</code> event in the near future.<br>
    *         EBUSY if the component is in the middle of powering down
    *               i.e. a <code>stop()</code> command has been called,
    *               and a <code>stopDone()</code> event is pending<br>
    *         EALREADY if the device is already on <br>
    *         FAIL Otherwise
    */
    command error_t start();

    /**
    * Notify caller that the component has been started and is ready to
    * receive other commands.
    *
    * @param <b>error</b> — SUCCESS if the component was successfully
    *                       turned on, FAIL otherwise
    */
    event void startDone(error_t error);

    /**
    * Stop this component and all of its subcomponents.  Return
    * values of SUCCESS will always result in a <code>stopDone()</code>
    * event being signalled.
    *
    * @return   SUCCESS if the device is already in the process of stopping or the device
        was on and the device is now ready to turn off. After receiving this return value,
         you should expect a stopDone event in the near future.
    *       EBUSY if the component is in the middle of powering up i.e. a start() command
        has been called, and a startDone() event is pending
    *       EALREADY if the device is already off
    *       FAIL Otherwise
    */
    command error_t stop();

    /**
    * Notify caller that the component has been stopped.
    *
    * @param <b>error</b> — SUCCESS if the component was successfully
    *                       turned off, FAIL otherwise
```

```
    */
116 event void stopDone(error_t error);

}
```

Listing B.1: Interface Pampa

Bibliografia

[1] I. F. A., W. S., Y. S., and E. C., "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: application driver for wireless communications technology," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 2 supplement, pp. 20–41, 2001.

[3] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[4] L. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," *Proc. of the 20th Conf. of the IEEE Computer and Communications Societies (INFOCOM 2001). Proceedings.*, pp. 1548–1557, 2001.

[5] R. Vijay, S. Curt, P. Sung, and B. S. Mani, "Energy-aware wireless microsensor networks," in *IEEE Signal Processing Magazine*, 2002.

[6] J. Pereira, F. Fabret, F. Llirbat, R. Preotiuc-Pietro, K. A. Ross, and D. Shasha, "Publish/subscribe on the web at extreme speed," in *VLDB*, 2000, pp. 627–630.

[7] B. Garbinato, H. Miranda, and L. Rodrigues, *Middleware for Network Eccentric and Mobile Applications*. Springer, 2009.

[8] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen, "The information bus: an architecture for extensible distributed systems," *SIGOPS Oper. Syst. Rev.*, vol. 27, no. 5, pp. 58–68, 1993.

[9] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, 2001.

[10] G. Cugola, E. Di Nitto, and A. Fuggetta, "Exploiting an event-based infrastructure to develop complex distributed systems," in *ICSE '98: Proceedings of the 20th international conference on Software engineering*, 1998, pp. 261–270.

[11] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha, "Filtering algorithms and implementation for very fast publish/subscribe systems," in *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 115–126.

[12] T. W. Yan and H. Garcia-Molina, "The sift information dissemination system," *ACM Trans. Database Syst.*, vol. 24, no. 4, pp. 529–565, 1999.

[13] M. L., C. G., and P. G., "Tree overlays for publish-subscribe in mobile ad hoc networks," Politecnico di Milano, Tech. Rep., 2005.

[14] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale event notification infrastructure," in *Lecture Notes in Computer Science*. Springer, 2001, pp. 30–43.

[15] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz, "Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination," in *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, 2001, pp. 11–20.

[16] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Achieving scalability and expressiveness in an internet-scale event notification service," in *PODC '00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM, 2000, pp. 219–227.

[17] P. R. Pietzuch and J. M. Bacon, "Hermes: A distributed event-based middleware architecture," *Distributed Computing Systems Workshops, International Conference on*, 2002.

[18] P. T. Eugster, R. Guerraoui, and C. H. Damm, "On objects and events," in *OOPSLA '01: Proceedings of the 16th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. New York, NY, USA: ACM, 2001, pp. 254–269.

[19] C. Raphaël and F. Pascal, "Semantic peer-to-peer overlays for publish/subscribe networks," in *Euro-Par 2005 Parallel Processing*, 2005, pp. 1194–1204.

[20] S. Thirunavukkarasu, B. Gordon, and C. Geoff, "Green: A configurable and re-configurable publish-subscribe middleware for pervasive computing," in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, 2005, pp. 732–749.

[21] M. Ren and C. Vinny, "Steam: Event-based middleware for wireless ad hoc networks," *Distributed Computing Systems Workshops, International Conference on*, vol. 0, p. 639, 2002.

[22] C. Gianpaolo, E. Jose, and d. C. Munoz, "On introducing location awareness in publish-subscribe middleware," in *ICDCS Workshops*, 2005, pp. 377–382.

[23] B. Oki, M. Pfluegl, A. Siegel, and D. Skeen, "The information bus: an architecture for extensible distributed systems," in *SOSP '93: Proceedings of the fourteenth ACM symposium on Operating systems principles*. New York, NY, USA: ACM, 1993, pp. 58–68.

[24] M. Altherr, M. Erzberger, and S. Maffeis, "ibus–a software bus middleware for the java platform," in *In Proceedings of the International Workshop on Reliable Middleware Systems*.

[25] R. Gruber, B. Krishnamurthy, and E. Panagos, "The architecture of the ready event notification service," *Distributed Computing Systems, International Conference on*, vol. 0, p. 0108, 1999.

[26] B. Segall and D. Arnold, "Elvin has left the building: A publish/subscribe notification service with quenching," in *In Proceedings of the Australian UNIX and Open Systems User Group Conference*, 1997.

[27] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, no. 3, pp. 332–383, 2001.

[28] H. Jafarpour, B. Hore, S. Mehrotra, and N. Venkatasubramanian, "Subscription subsumption evaluation for content-based publish/subscribe systems."

[29] J. A. Briones, B. Koldehofe, and K. Rothermel, "SPINE: Publish/Subscribe for Wireless Mesh Networks through Self-Managed Intersecting Paths," in *Proceedings of the 8th International Conference on Innovative Internet Community Systems (I2CS 2008)*, Schoelcher, Martinique, 2008.

[30] E. Anceaume, A. K. Datta, M. Gradinariu, and G. Simon, "Publish/subscribe scheme for mobile networks," in *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*. New York, NY, USA: ACM, 2002, pp. 74–81.

[31] Y. Huang and H. Garcia-Molina, "Publish/subscribe in a mobile environment," *Wirel. Netw.*, vol. 10, no. 6, pp. 643–652, 2004.

[32] "Enhanced cross-layer based middleware for mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 32, no. 2, pp. 490 – 499, 2009.

[33] L. Mottola, G. Cugola, and G. Picco, "A self-repairing tree topology enabling content-based routing in mobile ad hoc networks," *Mobile Computing, IEEE Transactions on*, vol. 7, no. 8, pp. 946 –960, aug. 2008.

[34] E. Souto, G. Guimarães, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, and J. Kelner, "Mires: a publish/subscribe middleware for sensor networks," *Personal Ubiquitous Comput.*, vol. 10, no. 1, pp. 37–44, 2005.

[35] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with ght, a geographic hash table," *Mob. Netw. Appl.*, vol. 8, no. 4, pp. 427–442, 2003.

[36] M. Albano and S. Chessa, "Publish/subscribe in wireless sensor networks based on data centric storage," in *CAMS '09: Proceedings of the 1st International Workshop on Context-Aware Middleware and Services*, 2009, pp. 37–42.

[37] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," *Mobile Computing Systems and Applications, IEEE Workshop on*, vol. 0, p. 90, 1999.

[38] H. Miranda, S. Leggio, L. Rodrigues, and K. Raatikainen, "An algorithm for dissemination and retrieval of information in wireless ad hoc networks," in *Proceedings of the 13th International Euro-Par Conference, Euro-Par 2007*, ser. Lecture Notes in Computer Science, A.-M. Kermarrec, L. Bougé, and T. Priol, Eds., vol. 4641. Springer, 2007, pp. 891–900.

[39] ——, "A power-aware broadcasting algorithm," *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium. Proceedings.*, 2006.

[40] H. Miranda, "Gossip-based data distribution in mobile ad hoc networks," Ph.D. dissertation, Universidade de Lisboa, Campo Grande, 1749-016 Lisboa - Portugal, 2007.

[41] Y. Tseng, S. Ni, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wirel. Netw.*, vol. 8, no. 2/3, pp. 153–167, 2002.

[42] P. Levis, "TinyOS programming," Disponível em: "http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf", Acedido em: 02/09/2009.

[43] "TinyOS," http://www.tinyos.net/.

[44] P. Levis, "TEP: 107," Disponível em: "http://www.tinyos.net/tinyos-2.x/doc/html/tep107.html", Acedido em: 12/01/2010.

[45] ——, "TEP: 111," Disponível em: "http://www.tinyos.net/tinyos-2.x/doc/html/tep111.html", Acedido em: 12/01/2010.

[46] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," Disponível em: "http://www.eecs.berkeley.edu/~pal/pubs/tossim-sensys03.pdf", Acedido em: 03/09/2009.

[47] P. Levis and N. Lee, "TOSSIM: A simulator for tinyos networks," Disponível em: "http://www.cs.berkeley.edu/~pal/pubs/nido.pdf", Acedido em: 03/09/2009.

[48] "The network simulator - ns-2." [Online]. Available: http://www.isi.edu/nsnam/ns/

[49] L. B. P. Cao, "Web caching and zipf-like distributions: Evidence and implications," in *In INFOCOM*, 1999, pp. 126–134.

[50] H. Miranda, C. Ellis, and F. Taiani, "Count on me: Lightweight ad-hoc broadcasting in heterogeneous topologies," *Int. Workshop on Middleware for Pervasive Mobile and Embedded Computing (M-MPAC 2009), in collocation with the ACM/IFIP/USENIX 10th Int. Middleware Conf. (Middleware 2009), November 30th 2009, Urbana Champaign, Illinois, USA*.