Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Mapa de Energia baseado em Predição para Redes de Sensores Sem Fio

Raquel Aparecida de Freitas Mini

Tese apresentada ao Curso de Pós-Graduação em Ciência Computação da Universidade Federal de Minas Gerais como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Antonio Alfredo Ferreira Loureiro

Co-orientador: Prof. Badri Nath (Rutgers University)

Belo Horizonte, Janeiro de 2004

Agradecimentos

Inicialmente gostaria de agradecer ao professor Loureiro pelo paciente trabalho de orientação. Sou muito grata pelas valiosas discussões e pelo tempo dedicado a este trabalho. Ao professor Badri Nath, pela oportunidade de realizar um estágio na Universidade de Rutgers, EUA.

Gostaria de agradecer aos professores Djamel, Rezende, José Marcos e Robson por terem participado da banca examinadora. Muito obrigada pelas críticas e sugestões feitas a este trabalho. Muito obrigada também aos demais professores do DCC que contribuíram para a minha formação pessoal e intelectual. Em especial, gostaria de agradecer ao professor Mário Campos que me orientou na graduação e no mestrado. Sou muito grata pelos seus importantes ensinamentos.

Aos alunos do grupo de pesquisa Sensornet do DCC/UFMG, em especial ao Max por ter contribuído com este trabalho. Gostaria de agradecer também aos alunos do grupo DATAMAN de Rutgers.

Gostaria de agradecer ao apoio financeiro do CNPq durante o meu doutorado sanduíche. Agradeço também à PUC Minas pelo auxílio financeiro concedido através do Programa Permanente de Capacitação de Docente.

Obrigada a você, Marcos, pelo apoio, companheirismo e por ter enfrentado comigo o desafio de morar um ano nos EUA. Muito obrigada pelo carinho e enorme paciência nos vários momentos de estresse. Obrigada à minha mãe, Terezinha. Não tenho palavras para expressar o quanto sou grata a você. Seu incentivo constante foi fundamental durante todo o doutorado.

Obrigada aos familiares, amigos e colegas que de alguma forma se fizeram presentes durante o desenvolvimento deste trabalho.

Finalmente, agradeço a Deus pela oportunidade de ter vivido cada momento.

Resumo Estendido

Introdução

Na última década, houve um grande avanço tecnológico nas áreas de sensores, circuitos integrados e comunicação sem fio. Este avanço levou ao surgimento das redes de sensores sem fio, que são aquelas nas quais os dispositivos computacionais são sensores de baixo custo que podem se comunicar através de uma interface sem fio e possuem capacidade limitada de computação e de memória. Estes sensores podem produzir respostas às mudanças de condições físicas, tais como temperatura, umidade ou campo magnético. O principal objetivo destas redes é realizar tarefas de sensoriamento distribuído, principalmente em aplicações como monitoração de ambientes, espaços inteligentes e sistemas médicos. Estas redes formam um novo tipo de redes *ad hoc* com um novo conjunto de características e desafios [Loureiro et al., 2003].

Ao contrário das redes *ad hoc* convencionais, as redes de sensores sem fio potencialmente possuem centenas de milhares de nós [Sohrabi et al., 2000]. Os sensores devem operar em ambientes inóspitos e, para alcançar boa resolução de sensoriamento, estas redes devem ser densas. Portanto, em redes de sensores, escalabilidade em relação ao número de nós é um fator essencial. De forma diferente dos nós de uma rede *ad hoc* convencional, os nós sensores geralmente permanecem estacionários após sua deposição. Entretanto, mesmo os nós sendo estacionários, estas redes ainda assim possuem topologia dinâmica. Durante períodos de baixa atividade, a rede deve entrar em estado de dormência na qual muitos nós devem se desligar para economizar energia. Além disso, os nós podem ficar fora de serviço quando a energia de suas baterias acabarem ou quando um evento destrutivo acontecer. Outra característica que distingue estas redes das demais é que os sensores possuem recursos bastante restritos, tais como limitada capacidade computacional, de memória e de energia. Estes sensores serão equipados com bateria e, em muitas aplicações, eles serão colocados em áreas remotas, impossibilitando o acesso a esses elementos para manutenção. Neste cenário, o tempo de vida da rede depende da quantidade de energia disponível nos nós sensores e, por isso, esses nós devem balancear seus recursos limitados com o objetivo de aumentar o tempo de vida da rede. Portanto, a conservação de energia é um dos aspectos mais importantes a serem considerados no projeto das redes de sensores.

A informação sobre a quantidade de energia disponível em cada parte da rede é chamada de mapa de energia e esta informação pode auxiliar a prolongar o tempo de vida da rede. O mapa de energia de uma rede de sensores pode ser representado como uma imagem em níveis de cinza na qual áreas claras representam regiões com mais energia disponível, e regiões com pouca energia são representadas por áreas escuras. De posse do mapa de energia, é possível determinar se alguma parte da rede está na iminência de falhar devido a falta de energia. O conhecimento sobre áreas com pouca energia disponível pode ajudar também na tarefa de deposição de novos nós. Sensores adicionais podem ser colocados seletivamente nas regiões com pouca energia disponível. A escolha da melhor localização para o nó sorvedouro pode também ser feita com base no mapa de energia. Nó sorvedouro é um nó especial da rede responsável pela coleta das informações sensoriadas pelos nós sensores. É provável que nós próximos ao nó sorvedouro irão gastar mais energia porque eles serão utilizados mais freqüentemente para transmitir pacotes para o nó sorvedouro. Conseqüentemente, se o nó sorvedouro for movido para áreas com maior quantidade de energia disponível, é possível que o tempo de vida da rede seja prolongado. Protocolos de roteamento também podem beneficiar-se da informação sobre a quantidade de energia disponível em cada parte da rede. Um algoritmo de roteamento pode fazer melhor uso das reservas de energia se este seletivamente escolher rotas que utilizam nós com maior quantidade de energia disponível, de tal forma que partes da rede com poucas reservas de energia possam ser preservadas. Esses algoritmos de roteamento podem também criar um *backbone* virtual conectando ilhas com grandes quantidades de energia. Outras possíveis aplicações que podem utilizar o mapa de energia são algoritmos reconfiguráveis e fusão de dados. De fato, é difícil pensar em alguma aplicação e/ou algoritmo que não se beneficiaria com o uso do mapa de energia.

O objetivo deste trabalho é construir o mapa de energia de uma rede de sensores sem fio utilizando técnicas de predição. Em mapas de energia baseados em predição, cada nó sensor utiliza algoritmos de predição para modelar sua dissipação de energia com o objetivo de prever

seu consumo de energia no futuro. Nesta situação, ao invés de enviar para o nó sorvedouro apenas sua quantidade de energia disponível, cada sensor envia também os parâmetros do modelo que descreve seu consumo de energia. O nó sorvedouro pode atualizar constantemente a informação do mapa de energia utilizando os parâmetros do modelo de predição recebidos de cada nó sensor. Neste cenário, os nós sensores só precisarão enviar outra informação de energia para o nó sorvedouro quando os parâmetros do último modelo enviado não mais representar satisfatoriamente o seu consumo de energia. Desta forma, os nós sensores enviarão uma quantidade menor de pacotes de energia para o nó sorvedouro, diminuindo assim a energia gasta na construção do mapa de energia.

Neste trabalho, a construção do mapa de energia baseada em predição é abordada através de duas formas bastante diferentes. Na primeira, é definida a precisão na qual o mapa deve ser construído, e nenhuma restrição com relação à quantidade de energia gasta nesta construção é especificada. Nesta abordagem, é investigada também a possibilidade de se amostrar a informação de energia em alguns nós sensores. Resultados mostram que o uso de amostragem produz curvas de erro mais constantes. Na segunda abordagem, devido à extrema importância da conservação de energia, é especificada a quantidade máxima de energia que cada nó pode gastar na construção do mapa de energia, e o melhor mapa deve ser construído utilizando apenas a quantidade de energia especificada. Esta abordagem é denominada modelo de orçamento de energia e é um novo paradigma para se projetar soluções para as redes de sensores sem fio. Neste novo paradigma, cada atividade da rede deve alcançar seu melhor desempenho gastando apenas uma quantidade finita de energia.

O desempenho das duas abordagens para construir o mapa de energia é avaliado através de simulação. Quando simulação é utilizada para avaliar o desempenho da construção do mapa de energia ou de qualquer outro problema relacionado à energia, é necessário conhecer a forma como acontece a dissipação de energia nos nós sensores. Com este objetivo, é apresentado também um modelo de dissipação de energia que é utilizado para simular o comportamento dos nós sensores em termos do consumo de energia. O modelo de dissipação aqui proposto é comparado com um modelo de dissipação ideal no qual todos os nós possuem conhecimento global sobre a rede. Utilizando o modelo de dissipação de energia proposto, resultados de simulação indicam que a utilização de técnicas de predição na construção do mapa de energia das redes de sensores sem fio é mais eficiente do que uma solução na qual nenhuma predição é

utilizada. Resultados mostram também que é possível alcançar os limites de desempenho na construção do mapa de energia utilizando o modelo de orçamento de energia proposto.

Revisão Bibliográfica

O conhecimento sobre a quantidade de energia disponível em cada parte da rede é uma informação importante para as redes de sensores sem fio. Uma forma simples de construir o mapa de energia é estabelecer que periodicamente cada nó deve enviar para o nó sorvedouro sua energia disponível. O problema desta abordagem é que na maioria das aplicações previstas para as redes de sensores sem fio, estas terão muitos nós com recursos limitados, fazendo com que a quantidade de energia gasta nesta solução seja proibitiva. Por esta razão, técnicas mais eficientes em termos de consumo de energia devem ser propostas para obter a quantidade de energia disponível em cada parte da rede.

O trabalho proposto em [Zhao et al., 2002] obtém o mapa de energia de uma rede de sensores utilizando uma abordagem baseada em agregação. Um nó sensor apenas precisa enviar para o nó sorvedouro sua energia local quando existe uma queda significativa quando comparada com a última vez que o nó reportou sua energia disponível. Ao longo do caminho para o nó sorvedouro, os nós que receberem duas ou mais informações de energia podem agregá-las de acordo com várias regras. Se as informações de energia são de áreas topologicamente adjacentes e se têm níveis de energia semelhantes, elas podem ser agregadas. O objetivo da agregação é reduzir o custo de coletar a informação de energia, mantendo a qualidade da informação obtida. Em [Zhao et al., 2002], são apresentados resultados de simulação que comparam as abordagens propostas com uma abordagem centralizada. Entretanto, nas simulações, não é levado em consideração o custo da atualização periódica da árvore de agregação.

Neste trabalho são apresentadas técnicas para construção do mapa de energia utilizando predição. A idéia é fazer com que cada nó envie para o nó sorvedouro os parâmetros do modelo que descreve o seu consumo de energia. O nó sorvedouro utiliza estes dados para atualizar localmente a informação da energia disponível em cada nó. A motivação desta abordagem é que se um nó é capaz de prever a quantidade de energia que será gasta, ele pode enviar esta informação para o nó sorvedouro e nenhuma outra informação de energia será enviada enquanto o modelo descrever satisfatoriamente o seu consumo de energia. Assim, se um nó pode eficientemente predizer a quantidade de energia que ele irá gastar no futuro, é possível economizar energia no processo de construção do mapa.

A principal diferença entre a abordagem baseada em agregação e a baseada em predição é que na primeira, cada nó envia para o nó sorvedouro apenas sua energia disponível. Na segunda, são enviados também os parâmetros de um modelo que descreve o consumo de energia no futuro. Neste caso, cada nó envia para o nó sorvedouro sua energia disponível e também os parâmetros do modelo escolhido para representar seu consumo de energia. Com estes parâmetros, o nó sorvedouro pode atualizar localmente a energia disponível em cada nó da rede, diminuindo a quantidade de pacotes de energia necessários na construção do mapa.

Taxonomia para as Redes de Sensores Sem Fio

Em redes de sensores, o desempenho dos protocolos está intimamente relacionado com as características destas redes, tais como: modelo de entrega dos dados, topologia, densidade e dinamismo dos sensores, fenômenos e observadores. Portanto, é importante especificar o tipo de rede em que cada protocolo será utilizado. Com este objetivo, neste capítulo, é apresentada uma taxonomia para estas redes na qual estas são organizadas de acordo com suas infra-estruturas e com as características dos eventos e do observador. Esta taxonomia irá auxiliar na definição do tipo de rede que será tratada nos próximos capítulos.

A tabela 1 ilustra a taxonomia proposta. O restante deste trabalho se baseia em um tipo de rede de sensores na qual os nós são estáticos e homogêneos, a troca ou recarga da bateria é inviável ou impossível, e também que existe apenas um nó sorvedouro sem restrição de energia. Além disso, os nós são depositados de forma aleatória formando uma rede densa em uma topologia plana. Com relação ao modelo de entrega de dados, é simulada uma rede baseada em eventos na qual os nós reportam informação somente se um evento de interesse ocorrer. Neste caso, o nó sorvedouro está interessado apenas na ocorrência de um evento específico ou em um conjunto de eventos. O modelo de comunicação é o modelo cooperativo no qual os sensores comunicam entre si para disseminar informações relacionadas ao evento.

Infra-estrutura		
Sensores	Dinamismo	estático dinâmico
	Heterogeneidade	homogêneos heterogêneos
	Bateria	com recarga sem recarga
Estratégia de Deposição		aleatória planejada
Densidade		baixa média alta
Topologia		plana hierárquica
Modelos de Comunicação		cooperativo não-cooperativo
Fenômeno		
Dinamismo		estático dinâmico
Duração		instantâneo duradouro infinito
Observador		
Dinamismo		estático dinâmico
Modelos de Entrega de Dados		contínuo guiado por eventos iniciado no observador híbrido
Número de Observadores		um múltiplos

Tabela 1: Taxonomia para as redes de sensores sem fio.

Modelo de Dissipação de Energia

Quando simulação é utilizada para avaliar o desempenho da construção do mapa de energia ou de qualquer outro problema relacionado à energia, é necessário conhecer como ocorre a dissipação de energia em cada nó sensor. Para isto, neste trabalho, é proposto o "Modelo de Dissipação de Energia baseado em Estados" (*State-based Energy Dissipation Model - SEDM*). Este modelo descreve o comportamento dos nós sensores em termos do consumo de energia. Neste modelo, a chegada dos eventos na rede segue um processo de Poisson ou uma distribuição de Pareto. Além disso, três outros modelos são usados para descrever o comportamento dos eventos. No primeiro, os eventos são estáticos e possuem tamanho fixo. No segundo modelo, os eventos se movem no campo de sensoriamento e, finalmente, no terceiro modelo, os eventos são estáticos, mas suas áreas de influência aumentam a uma determinada taxa.

O SEDM é baseado em uma estrutura na qual os nós possuem vários modos de operação com diferentes níveis de ativação e, conseqüentemente, com diferentes níveis de consumo de energia. Neste modelo, cada nó possui quatro modos de operação:

- Modo 1: sensor desligado, rádio desligado;
- Modo 2: sensor ligado, rádio desligado;
- Modo 3: sensor ligado, rádio recebendo;
- Modo 4: sensor ligado, rádio transmitindo.

As transições entre estes modos são descritas através do diagrama da figura 1. Neste diagrama, os modos de operação são representados pelos estados 1, 2, 3 e 4. Além disso, foi necessário representar mais dois estados $2'$ e $3'$. O estado i' também representa o modo de operação i. Como exemplo, ambos os estados 2 e $2'$ representam o modo de operação 2, a única diferença é que quando um nó vai para o estado 2, ele sempre inicia um temporizador, enquanto que, no estado $2'$, ele verifica se existe algum evento para ele. Portanto, em termos de consumo de energia, o estado i é exatamente igual ao estado i'. A única diferença é que o comportamento de um nó que vai para o estado i é diferente do comportamento de um nó que vai para o estado i'.

Com o objetivo de avaliar o SEDM, este é comparado com um modelo ideal de dissipação de energia no qual todos os nós possuem conhecimento global sobre rede. Como exemplo, no modelo ideal, mesmo se um nó estiver no modo 1, ele pode verificar se existe algum evento para ele ou se existe algum outro nó tentando se comunicar com ele. Nestas situações, este nó pode acordar imediatamente, sem a sobrecarga de ficar acordando periodicamente para verificar se existe algum evento. Portanto, no modelo ideal, todos os nós possuem conhecimento global sobre a rede e por isto eles podem fazer o melhor para realizar as tarefas de sensoriamento. Isto implica que a energia gasta pelo modelo ideal representa a menor quantidade de energia necessária para o tipo de rede de sensores analisado neste trabalho. Portanto, o modelo de

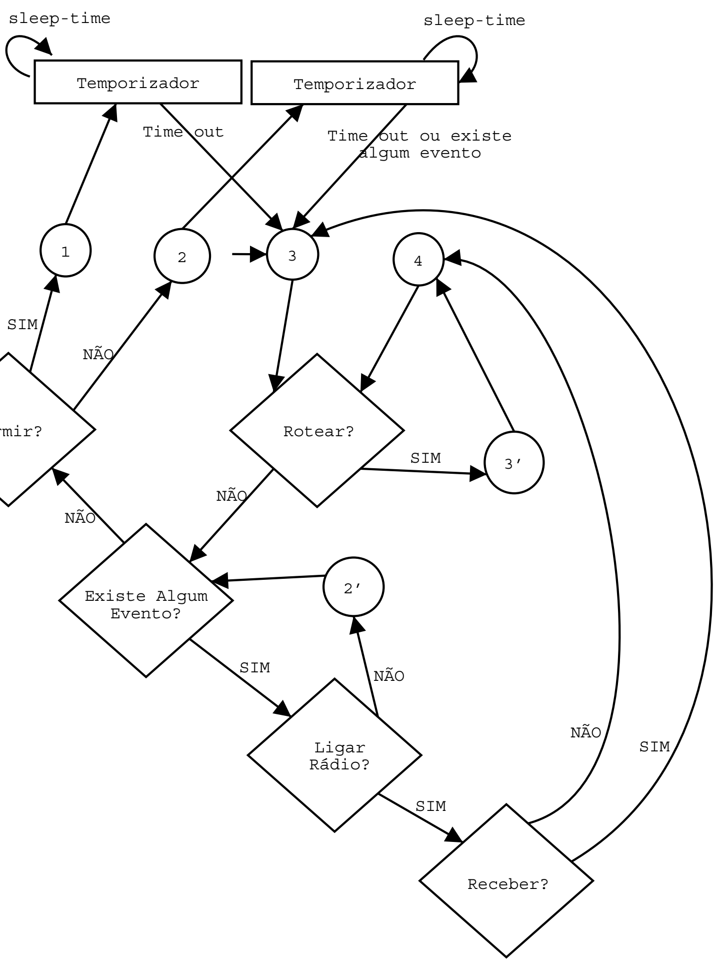


Figura 1: Diagrama de estados do SEDM.

dissipação de energia proposto é comparado com o limite inferior em termos do consumo de energia.

Mapa de Energia baseado em Predição

Como descrito anteriormente, o conhecimento sobre a quantidade de energia disponível em cada parte da rede é uma informação importante para as redes de sensores. Uma solução ingênua para construir o mapa de energia é programar cada nó para enviar periodicamente seu nível de energia para o nó sorvedouro. Como uma rede de sensores pode possuir muitos nós com recursos limitados, a quantidade de energia gasta por esta abordagem é proibitiva. Por esta razão, técnicas mais eficientes em termos de consumo de energia devem ser projetadas para obter a informação sobre a quantidade de energia disponível em cada parte da rede.

Neste capítulo, são discutidas as possibilidades de se construir o mapa de energia utilizando abordagens baseadas em predição. Basicamente, cada nó envia para o nó sorvedouro os parâmetros do modelo que descreve seu consumo de energia e o nó sorvedouro utiliza estes parâmetros para atualizar localmente a informação sobre a quantidade de energia disponível em cada nó. A motivação para este abordagem é que se um nó é capaz de predizer a quantidade de energia que ele irá gastar, ele pode enviar esta informação para o nó sorvedouro e nenhuma outra informação de energia será enviada durante o período em que o modelo descrever satisfatoriamente sua dissipação de energia. Portanto, se um nó pode eficientemente predizer a quantidade de energia que ele irá gastar no futuro, é possível economizar energia no processo de construir o mapa de energia de uma rede de sensores.

O requisito mais importante das técnicas de predição para nós sensores é que elas devem ser simples, não somente em relação à complexidade computacional, mas principalmente em relação ao número de mensagens que devem ser transmitidas para o nó sorvedouro. Em redes de sensores sem fio, a quantidade de energia gasta para realizar comunicação é muito maior do que a gasta para fazer computação. Estudos indicam que a execução de 3000 instruções gasta a mesma quantidade de energia que enviar 1 bit a 100 m via rádio [Pottie and Kaiser, 2000a]. Isto indica que as técnicas de predição devem necessitar do menor número de comunicações possível. Além disso, os algoritmos de predição devem ser escolhidos levando-se em consideração as limitações da capacidade computacional e de memória dos nós sensores.

Com o objetivo de prever a energia dissipada, dois modelos são estudados. O primeiro é um modelo probabilístico baseado nas cadeias de Markov. No segundo modelo, a energia é representada por uma série no tempo e é utilizado o modelo estatístico ARIMA (*Autoregressive Integrated Moving Average*) para prever séries no tempo. Estes dois modelos baseados em predição são comparados com uma abordagem ingênua para construir o mapa de energia. Resultados de simulação mostram que a utilização de técnicas baseadas em predição é mais eficiente em termos do consumo de energia do que a abordagem que não utiliza predição.

Neste capítulo também é analisada a possibilidade de se utilizar amostragem para construir o mapa de energia. A suposição básica desta técnica é que nós vizinhos tendem a gastar energia de forma similar. Portanto, a informação enviada por um nó pode ser utilizada para atualizar a taxa do consumo de energia dos seus nós vizinhos. Resultados mostram que modelos de amostragem podem diminuir o número de pacotes de energia enviados e que sua principal vantagem é produzir curvas de erro mais constantes.

A construção do mapa de energia apresentada neste capítulo foi projetada levando-se em consideração a precisão desejada no mapa. Nesta abordagem, o mapa de energia é construído levando-se em consideração apenas o erro máximo permitido e nenhuma restrição com relação à quantidade máxima de energia que pode ser gasta nesta construção é definida. No próximo capítulo, o problema da construção do mapa de energia será visto de forma diferente. Ao invés de se definir a precisão desejada no mapa, será definida a quantidade máxima de energia que poderá ser gasta nesta construção. O objetivo será construir o melhor mapa de energia possível sob a restrição de que cada nó pode gastar no máximo uma certa quantidade de energia.

Modelo de Orçamento de Energia

A utilização em larga escala das redes de sensores sem fio depende do projeto de uma arquitetura escalável e de baixo custo. Além disso, o projeto deve considerar a conservação de energia um problema fundamental e propor mecanismos para estender o tempo de vida da rede. Este cenário nos leva naturalmente para o seguinte problema: qual é o melhor desempenho que um protocolo pode atingir dado que ele pode gastar apenas uma quantidade finita de energia? Utilizando esta idéia, é possível associar um orçamento finito de energia para cada atividade da rede e pedir que esta atividade alcance seu melhor desempenho utilizando apenas seu orçamento de energia. Esta é uma nova forma de lidar com problemas relacionados a rede, e deveria ser considerada um novo paradigma para projetar algoritmos para redes que utilizam bateria, especialmente para as redes de sensores sem fio.

O modelo de orçamento de energia é altamente aplicável à construção do mapa de energia de uma rede de sensores. Seria inútil construir o melhor mapa de energia gastando toda a energia disponível. Neste capítulo, a construção do mapa de energia é estendida para situações nas quais um orçamento de energia é definido para esta construção. O objetivo é construir o melhor mapa de energia sob a restrição de que cada nó pode gastar apenas uma certa quantidade de energia nesta construção. Resultados indicam que é possível atingir o limite de desempenho na construção do mapa de energia utilizando o modelo de orçamento finito proposto neste trabalho.

Conclusões e Direções Futuras

Neste trabalho, o estado da arte foi estendido em três direções principais. Primeiramente, foi proposto o "Modelo de Dissipação de Energia baseado em Estados" que representa o comportamento de um nó sensor em termos do consumo de energia. A segunda contribuição consiste na proposição do uso de algoritmos de predição para construir o mapa de energia das redes de sensores. É mostrado como nós sensores podem predizer seus consumos de energia com o objetivo de construir um mapa de energia. A terceira contribuição é a proposta de um novo paradigma, denominado modelo de orçamento de energia, para projetar soluções para as redes de sensores sem fio. É mostrado também como este novo paradigma pode ser aplicado na construção de mapas de energia com restrições.

Este trabalho pode ser estendido em uma variedade de direções. A primeira seria a avaliação de outros algoritmos para prever o consumo de energia em um nó sensor. Outras técnicas de predição poderiam ser comparadas com o modelo de Markov analisado neste trabalho. Outra possível extensão deste trabalho seria utilizar o mapa de energia para auxiliar a tarefa de roteamento e/ou disseminação de informação na rede. Pretende-se também investigar o uso do paradigma de orçamento de energia em outras atividades da rede. Como exemplo, poderia-se projetar mecanismos para disseminar informação para a maior quantidade de nós possível sob a restrição de que se gaste apenas uma determinada quantidade de energia. Outra possibilidade seria projetar funções de gerenciamento de rede para alcançar seus melhores desempenhos utilizando apenas uma quantidade finita de energia.

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Prediction-based Energy Map for Wireless Sensor Networks

Raquel Aparecida de Freitas Mini

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Prof. Antonio Alfredo Ferreira Loureiro

Co-Advisor: Prof. Badri Nath (Rutgers University)

Belo Horizonte, January, 2004

Acknowledgment

This work has been partially supported by CNPq, Brazil, under process number 55.2111/2002-3, DARPA under contract number N-666001-00-1-8953, and a grant from CISCO systems. We would like to thank the comments and suggestions of Prof. Narayan Mandayam[1], the members of the DATAMAN group of Rutgers University, and also the members of the Sensornet project of DCC/UFMG.

[1]Associate Professor of ECE Department at Rutgers University and Associate Director at WINLAB.

Abstract

A fundamental issue in the design of a wireless sensor network is to devise mechanisms to make efficient use of its energy, and thus, extend its lifetime. The information about the amount of available energy in each part of the network is called the energy map and can be useful to increase the lifetime of the network. In this work, we address the problem of constructing the energy map of a wireless sensor network using prediction-based techniques. We deal with the energy map construction problem using two different approaches. In the first one, we define the accuracy in which the map should be constructed, and no restriction regarding the amount of energy spent in this construction is defined. We also investigate the possibility of sampling the energy information in some nodes in the network. Results show that the use of sampling techniques produce more constant error curves. In the second approach, due to the paramount importance of energy conservation, we specify the maximum amount of energy each node can spend in the energy map construction, and we build the best energy map using only the specified energy. This is a new paradigm to design solutions for wireless sensor networks in which we should associate a finite energy budget for each network activity, and ask this activity to achieve its best performance using only its budget. We show that we can approach the performance limits on the energy map construction using the proposed finite energy budget model. The performance of the two approaches to construct the energy map is evaluated through simulation. When simulation is used to evaluate the performance of the energy map construction or any other energy related problem, we have to know how the energy dissipation happens in sensor nodes. To this end, we also present an energy dissipation model that is used to simulate the behavior of a sensor node in terms of energy consumption. Simulation results indicate that prediction-based techniques are more energy-efficient than a solution in which no prediction is used. The approaches studied in this work deal with the construction of an energy map of a wireless sensor network in two novel ways: the restrictive and the non-restrictive energy map construction.

Contents

List of Figures **xxv**

List of Tables **xxix**

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Objectives and Contributions . . . 3

1.3 Organization of this work . . . 5

2 Background **7**

2.1 Introduction . . . 7

2.2 Characteristics and Challenges of Sensor Networks . . . 10

2.3 Physical Layer Issues . . . 12

2.4 Link Layer Protocols . . . 13

2.5 Network Layer Protocols . . . 14

2.5.1 Routing . . . 14

2.5.2 Addressing Mechanisms . . . 16

2.6 Distributed Algorithms . . . 17

2.6.1 Location Mechanisms . . . 18

2.6.2 Time Synchronization . . . 19

2.6.3 Coverage Calculation . . . 20

2.6.4 Topology Discovery . . . 20

2.6.5 Energy Map . . . 21

2.7 Applications . . . 22

2.8 Concluding Remarks . . . 25

3 Taxonomy for Wireless Sensor Networks **27**

3.1 Introduction . . . 27

3.2 Infrastructure . . . 27

3.3 Phenomenon . . . 30

3.4 Observer . . . 31

3.5 Taxonomy . . . 32

4 Energy Dissipation Model **35**

4.1 Introduction . . . 35

4.2 State-based Energy Dissipation Model . . . 37

4.2.1 Modeling . . . 37
4.2.2 Event Arrival Models . . . 42
4.2.3 Event Generation Models . . . 44
4.3 Ideal Energy Dissipation Model . . . 45
4.4 Simulation Results . . . 46
4.4.1 Basic Operation . . . 47
4.4.2 Final Energy Map . . . 55
4.4.3 Undetected Events . . . 60
4.5 Concluding Remarks . . . 63

5 Prediction-based Energy Map 67
5.1 Introduction . . . 67
5.2 Probabilistic Model . . . 68
5.3 Statistical Model . . . 72
5.4 Simulation Results . . . 76
5.4.1 Basic Operation . . . 76
5.4.2 Changing the Number of Events . . . 78
5.4.3 Changing the Energy Map Precision . . . 83
5.5 Energy Map Construction using Sampling Technique . . . 87
5.5.1 Sampling Model . . . 87
5.5.2 Simulation Results . . . 89
5.6 The Role of the Energy Map in the Protocol Design of Wireless Sensor Networks 96
5.7 Concluding Remarks . . . 100

6 Finite Energy Budget 103
6.1 Introduction . . . 103
6.2 Non-Adaptive Energy Map Construction under a Finite Energy Budget . . . 104
6.3 Simulation Results for the Non-Adaptive Energy Map Construction . . . 109
6.4 Adaptive Energy Map Construction under a Finite Energy Budget . . . 113
6.4.1 Changing the Value of c . . . 113
6.4.2 Choosing the Initial Value of c . . . 118
6.5 Simulation Results for the Adaptive Energy Map Construction . . . 125
6.6 Concluding Remarks . . . 126

7 Conclusions and Future Directions 131
7.1 Summary of Accomplished Work . . . 131
7.2 Future Directions . . . 133

A ACF and PACF 141
A.1 Autocorrelation Functions . . . 141
A.2 Partial Autocorrelation Functions . . . 142

B Changing the Number of Events in the Finite Energy Budget 145
B.1 Poisson Arrival Model . . . 145
B.2 Pareto Arrival Model . . . 150

List of Figures

1 Diagrama de estados do SEDM. xii

1.1 Example of an energy map of a wireless sensor network. 2

4.1 Diagram of the State-based Energy Dissipation Model. 40
4.2 Scheme of the event generation. 45
4.3 Diagram of the Ideal Energy Dissipation Model. 46
4.4 Comparison between the SEDM and the Ideal model for $\lambda = 0.01$. 49
4.5 Comparison between the SEDM and the Ideal model for $\lambda = 0.04$. 50
4.6 Comparison between the SEDM and the Ideal model for $\lambda = 0.1$. 51
4.7 Comparison between the SEDM and the Ideal model for $\lambda = 0.2$. 52
4.8 Comparison between the SEDM and the Ideal model for $\lambda = 0.3$. 53
4.9 Comparison between the SEDM and the Ideal model for $\lambda = 0.01$ during 1000 seconds of simulation. 53
4.10 Comparison between the SEDM and the Ideal model for $\lambda = 0.04$ during 1000 seconds of simulation. 54
4.11 Comparison between the SEDM and the Ideal model for $\lambda = 0.2$ during 1000 seconds of simulation. 54
4.12 Final energy map for 500 seconds of simulation, $\lambda = 0.04$. 55
4.13 Final energy map for 1000 seconds of simulation, $\lambda = 0.04$. 56
4.14 Final energy map for 2000 seconds of simulation, $\lambda = 0.04$. 56
4.15 Final energy map for 3000 seconds of simulation, $\lambda = 0.04$. 57
4.16 Scale used in the gray level image construction. 57
4.17 Geographical location of nodes in the field of size 50×50 m^2. 57
4.18 Final energy map when the sensing radius is 5 m. 58
4.19 Final energy map when the sensing radius is 10 m. 58
4.20 Final energy map when the sensing radius is 15 m. 59
4.21 Instant of time in which the 67 events were generated. 60
4.22 Non-detected events in the Static Events Model. 62
4.23 Non-detected events in the Dynamic Event Model. 63
4.24 Non-detected events in the Static and Increasing Event Model. 64

5.1 Example of a time series that represents the energy drop of a sensor node. . . 72
5.2 Difference of the time series of Figure 5.1. 73
5.3 The correct available energy in a sensor node and the values found using the naive, Markov and ARIMA models for different values of λ. 78

5.4 Number of energy information packets the node of simulations of Figure 5.3 had to send. 79
5.5 Average number of events when the parameters λ and a are changed. 79
5.6 Average number of packets for different values of λ, *threshold* $= 3\%$. 80
5.7 Average number of bytes for different values of λ, *threshold* $= 3\%$. 82
5.8 Average number of packets for different values of a, *threshold* $= 3\%$. 83
5.9 Average number of bytes for different values of a, *threshold* $= 3\%$. 84
5.10 Average number of packets for different values of *threshold*, $\lambda = 0.2$. 85
5.11 Average number of bytes for different values of *threshold*, $\lambda = 0.2$. 86
5.12 Probability p for $d = 0.4$. 88
5.13 Number of energy packets sent using the static and dynamic sampling approaches. 90
5.14 Error in percentage using the static and dynamic sampling approaches. 92
5.15 The influence of the interpolation phase in sampling techniques. 93
5.16 Comparison between sampling models with the original Markov using a great value of *threshold*. 94
5.17 Number of energy packets sent using the static and dynamic sampling approaches. 95
5.18 Error in percentage using the static and dynamic sampling approaches. 96
5.19 The influence of the interpolation phase in sampling techniques. 97
5.20 Comparison between sampling models with the original Markov using a great value of *threshold*. 98
5.21 Protocol architecture for a wireless sensor network with a power management plane. 99

6.1 Function defined by equation (6.3) for different values of c. 107
6.2 Function p' for $p = 0.5$, $c = 2$ and different values of k. 108
6.3 Function p' for $p = 0.5$, $k = \log_c 1000$ and different values of c. 108
6.4 Function p' for $p = 0.5$, $k = \log_c 1000$ and different values for c_1 and c_2. 109
6.5 Changing the value of c when each node can spend 2 packets with energy information, $k = \log_c 1000$ and $\lambda = 0.1$. 111
6.6 Changing the value of c when each node can spend 4 packets with energy information, $k = \log_c 1000$ and $\lambda = 0.1$. 113
6.7 hanging the value of c when each node can spend 8 packets with energy information, $k = \log_c 1000$ and $\lambda = 0.1$. 114
6.8 hanging the value of c when each node can spend 16 packets with energy information, $k = \log_c 1000$ and $\lambda = 0.1$. 115
6.9 Increase and decrease function. 117
6.10 Increasing the value of c in 10%. 118
6.11 The problem of using a small budget in a 2000 second simulation when the budget size is 1 packet per node, $k = \log_c 1000$ and $\lambda = 0.1$. 119
6.12 Adaptive Algorithm 120
6.13 Changing the initial value of c when the budget size is 1 packet, $k = \log_c 1000$ and $\lambda = 0.1$. 122
6.14 Changing the initial value of c when the budget size is 4 packets, $k = \log_c 1000$ and $\lambda = 0.1$. 123
6.15 Changing the initial value of c when the budget size is 10 packets, $k = \log_c 1000$ and $\lambda = 0.1$. 124

6.16 Changing the budget size in a 500 second simulation, $k = \log_c 1000$ and $\lambda = 0.1$. 126
6.17 Function defined by equation (6.8) and the budget of size 7 of Figure 6.16. . . 127
6.18 Changing the budget size in a 1000 second simulation, $k = \log_c 1000$ and $\lambda = 0.1$. 128
6.19 Changing the budget size in a 1500 second simulation, $k = \log_c 1000$ and $\lambda = 0.1$. 129
6.20 Changing the budget size in a 2000 second simulation, $k = \log_c 1000$ and $\lambda = 0.1$. 130

A.1 Autocorrelation functions of the time series of Figure 5.2. 142
A.2 Sample partial autocorrelation functions of the time series of Figure 5.2. 143

B.1 Changing the value of λ when the number of packets is 2 in a 1000 second simulation, $k = \log_c 1000$. 146
B.2 Changing the value of λ when the number of packets is 4 in a 1000 second simulation, $k = \log_c 1000$. 147
B.3 Changing the value of λ when the number of packets is 6 in a 1000 second simulation, $k = \log_c 1000$. 148
B.4 Changing the value of λ when the number of packets is 8 in a 1000 second simulation, $k = \log_c 1000$. 149
B.5 Changing the value of a when the number of packets is 2 in a 1000 second simulation, $k = \log_c 1000$. 150
B.6 Changing the value of a when the number of packets is 4 in a 1000 second simulation, $k = \log_c 1000$. 151
B.7 Changing the value of a when the number of packets is 6 in a 1000 second simulation, $k = \log_c 1000$. 152
B.8 Changing the value of a when the number of packets is 8 in a 1000 second simulation, $k = \log_c 1000$. 153

List of Tables

1 Taxonomia para as redes de sensores sem fio. x

3.1 Taxonomy for wireless sensor networks. 33

4.1 Current per hardware component of a sensor node [Hill et al., 2000]. 38
4.2 Activity in each operation mode. 38
4.3 Default values used in simulations 47

5.1 Default values used in simulations. 77
5.2 Total number of energy packets sent using the static and dynamic sampling approaches. 91
5.3 Total number of energy packets sent using the static and dynamic sampling approaches. 95

6.1 Default values used in simulations. 110
6.2 Initial value of constant c. 121

Chapter 1

Introduction

1.1 Motivation

Wireless sensor networks are composed of low-cost sensor nodes that can communicate with each other in a wireless manner, have limited computing capability and memory and operate with limited battery power. These sensors can produce a measurable response to changes in physical conditions, such as temperature or magnetic field. The main goal of such networks is to perform distributed sensing tasks, particularly for applications like environmental monitoring, smart spaces and medical systems. These networks form a new kind of ad hoc network with a new set of characteristics and challenges.

Unlike conventional wireless ad hoc networks, a wireless sensor network potentially comprises of hundreds to thousands of nodes [Sohrabi et al., 2000]. Sensors have to operate in noisy environments and, in order to achieve good sensing resolution, higher densities are required. Therefore, in a sensor network, scalability is a crucial factor. Different from nodes of a customary ad hoc network, sensor nodes are generally stationary after deployment. Although nodes are static, these networks still have dynamic network topology. During periods of low activity, the network may enter a dormant state in which nodes go to sleep to conserve energy. Also, nodes go out of service when the energy of the battery runs out, or when a destructive event takes place [Park et al., 2000]. Another characteristic of these networks is that sensors have limited resources, such as limited computing capability, memory and energy supplies, and they must balance these restricted resources in order to increase the lifetime of the network.

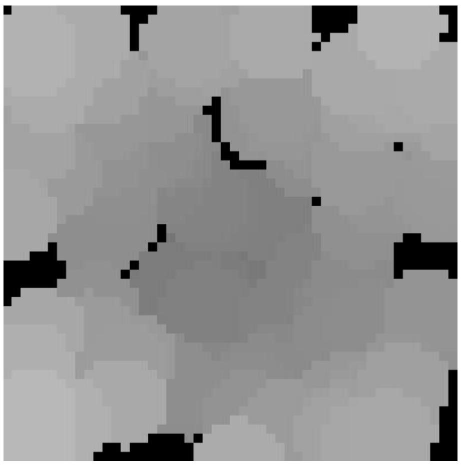


Figure 1.1: Example of an energy map of a wireless sensor network.

In addition, sensors will be battery powered and it is often very difficult to change or recharge batteries for these nodes. Therefore, in sensor networks, we are interested in prolonging the network lifetime, and thus the energy conservation is one of the most important aspects to be considered in the design of these networks.

The information about the remaining available energy in each part of the network is called the *energy map* and could aid in prolonging the lifetime of the network. We could represent the energy map of a sensor network as a gray level image as the one illustrated in Figure 1.1, in which light shaded areas represent regions with more remaining energy, and regions short of energy are represented by dark shaded areas. Using the energy map, a user may be able to determine if any part of the network is about to suffer system failures in near future due to depleted energy [Zhao et al., 2002]. The knowledge of low-energy areas can aid in incremental deployment of sensors because additional sensors can be placed selectively in those regions short of resources. The choice of the best location for the monitoring node can be made also based on the energy map. A monitoring node is a special node responsible for collecting information from sensor nodes. We know that nodes near the monitoring node probably will spend more energy because they are used more frequently to relay packets to the monitoring node. Therefore, if we move the monitoring node to areas with more remaining energy, we could prolong the lifetime of the network.

Other possible applications that could take advantage of the energy map are reconfiguration algorithms, query processing and data fusion. In fact, it is difficult to think of an application and/or an algorithm that does not need to use an energy map. Therefore, the energy map is an important information for sensor networks. However, the naive approach, in which each node sends periodically its available energy to the monitoring node, would waste so much energy due to communications that probably the utility of the energy information will not compensate the amount of energy spent in this process. For that reason, better energy-efficient techniques have to be devised to gather the information about the available energy in each part of a sensor network.

1.2 Objectives and Contributions

This work addresses the problem of constructing the energy map of a wireless sensor network. We are particularly interested in using prediction techniques to construct the energy map in such a way that sensor nodes construct models that represent their energy consumption in future time. We believe that, in many instances, a sensor node can predict its energy consumption based on its own past history. If a node can predict efficiently the amount of energy it will dissipate in the future, it will not be necessary to transmit frequently its available energy. This node can just send one message with its available energy and the parameters of the model that describes its energy dissipation. With this information, the monitoring node can often update its local information about the available energy of that node. Clearly, the effectiveness of this paradigm depends on the accuracy with which prediction models can be generated.

We deal with prediction-based energy map construction through two different perspectives. Firstly, the energy map is constructed taking into account a parameter that determines the maximum error allowed in the energy information. In this approach, no limitation concerning the amount of energy that can be spent in the map construction is defined. Secondly, we look at a restrictive energy map construction, in which, instead of defining the map accuracy, we define a finite energy budget that each sensor node can spend in the map construction. Our goal is to construct the best energy map under the constraint that each node uses only its energy budget. This is a new paradigm to design solutions for wireless sensor networks. The choice of the best approach to construct the energy map depends on the kind of wireless sensor network we are working on. If the map accuracy is more important than the amount of energy spent in its construction, we should use the non-restrictive energy map construction.

However, in most sensor network applications, the energy consumption is the most important aspect to be considered. In these situations, the restrictive energy map construction is highly applicable.

In order to analyze the approaches to construct the energy map, we have to have a clear idea of how is the energy drop in a sensor node. To this end, we propose a State-based Energy Dissipation Model (SEDM) that represents the behavior of a sensor node in terms of energy consumption. In the SEDM, we propose using various operation modes with different levels of activation and with different levels of energy consumption for sensor nodes. In the analysis of our model, we assume that the arrival of events in the network follows a Poisson process or a Pareto distribution. In addition, three other models are used to describe the behavior of events. In the first one, the event is static and has a fixed size. In the second model, the event moves around in the sensor field and, in the third one, the event is static but its area of influence increases with a given rate.

Using the SEDM, we analyze the performance of two ways of constructing the prediction-based energy map. Simulation results show that the use of prediction-based models decreases the amount of energy necessary to construct the energy map. Furthermore, results also indicate that we can approach the performance limits in the energy map construction using the proposed finite energy budget model.

Therefore, the main contributions of this work can be summarized as:

- The proposal of a State-based Energy Dissipation Model (SEDM) that represents more realistically the behavior of a sensor network in terms of energy dissipation. Based on this model, more realistic results can be obtained when evaluating solutions to wireless sensor networks that take into account energy aspects of sensor devices.

- The use of the idea of making prediction to construct the energy map of a wireless sensor network. We analyze through simulation the performance of two models to predict the energy drop in a sensor node and to construct a non-restrictive energy map. In the first model, sensor nodes are modeled as a Markov chain in which the states represent the operation modes of the node. In the second model, the energy drop is represented as a time series and we use forecasting techniques of time series to make the predictions. Using simulations, both approaches are evaluated and compared with a naive approach to construct the energy map in which no prediction is used. Simulation results show that the use of prediction-based models decreases the amount of energy necessary to construct the energy map of wireless sensor networks. We also investigate the energy map construction problem using sampling techniques in a way that it is not necessary that all nodes send their energy information to the monitoring node. The energy dissipation rate of a node that did not send its energy information packet is estimated using the information received from its neighboring nodes. In situations in which neighboring nodes spend their energy similarly, we can save energy sampling the energy information. Results show that the use of sampling techniques produce more constant error curves, and also that these approaches can reduce the number of energy information packets needed to construct the energy map.

- The proposal of using the finite energy budget model as a new paradigm for designing solutions for wireless sensor networks. In this paradigm, we specify the maximum amount of energy each network activity can spend, and ask this activity to achieve its best performance spending only the specified budget. We show how this new paradigm can be applied in the construction of a restrictive energy map. Due to the paramount importance of energy conservation in these networks, the idea of defining energy budget to each network activity is a very promising paradigm.

1.3 Organization of this work

This work is organized as follows. In Chapter 2, we briefly survey the related work. In Chapter 3, we present a taxonomy for wireless sensor networks in which these networks are classified according to different characteristics. In this chapter, we also specify the wireless sensor network we will be working in next chapters. Chapter 4 presents the model that we propose to describe the behavior of the energy dissipation model of a sensor node and, consequently, to simulate its energy drop. In Chapter 5, we describe two prediction-based techniques to construct a non-restrictive energy map for wireless sensor networks. We also investigate sampling techniques to diminish the number of packets needed to construct the map. In Chapter 6, we present the finite energy budget paradigm that is a new way of proposing solutions for wireless sensor networks. In this chapter, we also show how this paradigm can be applied in a restrictive energy map construction. Finally, in Chapter 7, we present our conclusions and give directions for future work.

Chapter 2

Background

2.1 Introduction

Since their emergence in the 1970s, wireless networks have become increasingly popular in the computing industry. Currently, there are two classes of wireless networks [Royer and Toh, 1999]. The first one is known as the infrastructured network. A mobile unit within these networks connects to and communicates with the nearest base station that is within its communication radius. Base stations are responsible for connecting this network to other networking elements. As the node travels out of range of one base station and into the range of another one, a hand-off occurs from one station to another one, and the node is able to continue its communication seamlessly throughout the network. Typical applications of this type of network include cellular network and office wireless local area network [Tang and Baker, 2000].

The second type of wireless network is the infrastructureless network, commonly known as ad hoc network [Manet, 2002]. An ad hoc network has no fixed routers, all nodes are capable of movement and can be connected dynamically in an arbitrary manner, acting as routers, discovering and maintaining routes to other nodes. Typically, each node is a personal information appliance such as a personal digital assistant (PDA) outfitted with a fairly sophisticated radio transceiver. The main goal of an ad hoc network is to form and maintain a connected multi-hop network capable of transporting a variety of traffic between nodes [Sohrabi et al., 2000].

Up to now, ad hoc networks have been studied considering mobility and more powerful nodes with typically tens to hundreds of communicating elements [Sohrabi et al., 2000]. However, recent advances in integrated circuit technology have contributed much to the introduction of wireless micro sensors which are low-cost devices that can communicate with each other in a wireless manner, have limited computing capability and memory and operate with limited battery power [Pottie and Kaiser, 2000b]. These sensors can produce a measurable response to changes in physical conditions, such as temperature, pressure, magnetic field, humidity, noise levels, lighting conditions and soil makeup. A network of such sensors, linked by a wireless medium or "wireless sensor network", forms a kind of ad hoc network with a new set of characteristics and challenges. The main goal of such network is to perform distributed sensing tasks particularly for applications like environmental monitoring. Their main advantage is their ability to be deployed in almost all kinds of terrain where it might not be possible to use traditional wired networks.

The basic operation of a sensor network can be described as follows. At the beginning, nodes are thrown or placed specifically in the area to be monitored. Next, nodes wake up and begin to establish routes to be used to send packets. When a target is detected, nodes collect information about it and, after making some computation on the data collected, send the processed data to a monitoring node. A monitoring node is a special node responsible for collecting information from sensor nodes. Typically, this node is named observer or end user and it is interested in obtaining information from sensor nodes about the phenomenon. The observer can indicate interests (or queries) to the network and receives responses to these queries. In this case, only information specified in the queries is sent to the monitoring node. Furthermore, sometimes, it can be necessary to accomplish a cooperative work in order to sense the target. In this case, sensors self-organize themselves in clusters to exchange information about the target. The combined information is sent to the monitoring node. The steps described above correspond to the general operation of a sensor network. In [Tilak et al., 2002], the authors describe other ways in which a sensor network could work.

Therefore, a wireless sensor network performs distributed sensing, wireless communication and distributed processing. These three characteristics are discussed in the following [Estrin et al., 2001]:

- "Why use distributed sensing?"– When the precise location of a signal of interest is unknown in a monitored region, distributed sensing allows one to place sensors closer to the phenomena being monitored rather than using a single sensor. Thus, the network of a large number of nodes enables high quality sensing information with the additional advantages of easy deployment and fault-tolerance. These characteristics make microsensors ideal for deployment in otherwise inaccessible environments where maintenance would be inconvenient or impossible [Min et al., 2001].

- "Why use wireless communication?"– In cases in which wired networking of distributed sensors can be easily achieved, and there are no other factors involved such as cost, it is often the most advantageous approach because nodes can be wired to energy sources simplifying the system design and operation. However, in many envisioned applications, the environment being monitored does not have installed infrastructure for either communications or energy, and therefore untethered nodes must rely on local, finite, and relatively small energy sources, as well as wireless communication channels.

- "Why use distributed processing?"– Although sensors are distributed to be close to the phenomena, one might still consider an architecture in which sensor outputs could be communicated back to a central processing unit. However, in the context of untethered nodes, the finite energy budget is a primary design constraint. Communication is a key energy consumer as the radio signal power in sensor networks drops up to around r^4 (where r is the communication radius) [Pottie and Kaiser, 2000b] due to ground reflections from short antenna heights. Therefore, one wants to process data as much as possible inside the network to reduce the number of bits transmitted, particularly over longer distances.

Thus, a sensor network is a distributed, wireless ad hoc network in which the processing is distributed along with the sensing.

The rest of this chapter is organized as follows. Section 2.2 describes the main characteristics and challenges of wireless sensor networks. Sections 2.3 to 2.7 describe, respectively, some of the researches that are being conducted in the following areas: Physical Layer, Link Layer, Network Layer, Distributed Algorithms and Applications for wireless sensor networks.

2.2 Characteristics and Challenges of Sensor Networks

Sensor networks are a new kind of ad hoc networks with some new characteristics and challenges. Unlike conventional wireless ad hoc networks, a sensor network potentially comprises hundreds to thousands of nodes [Sohrabi et al., 2000]. Sensors have to operate in noisy environments and higher densities are required to achieve a good sensing resolution. In some situations, it is necessary to cover large areas and, thus, networks of hundreds and, possibly, thousands of nodes are needed to provide sufficient coverage. Furthermore, since nodes are susceptible to failure, a higher degree of redundancy is also needed. Consequently, in a sensor network, scalability is a crucial factor.

Nodes of a wireless sensor network have limited resources such as limited computing capability, memory and energy supplies. In these networks, energy is typically more limited than in other wireless networks because of the nature of the sensing devices and the difficulty in recharging their batteries. Furthermore, sensor nodes can be deployed either by strategically placing them at specific locations or by air dropping them in a remote area. Often times, these sensors are placed in a hostile environments where human oversight is very difficult if not impossible. Thus, sensor nodes must balance these limited resources in order to increase the lifetime of the network [Park et al., 2000]. In addition, studies in the past have shown that 3000 instructions could be executed for the same energy cost as sending a bit 100 m by radio [Pottie and Kaiser, 2000a]. This indicates that the trade-off between communication and computation in sensor networks should be resolved in favor of computation.

Similarity to other wireless networks, sensor networks have dynamic network topology. However, in other types of wireless networks changes in topology are attributed to node movement. In typical ad hoc networks old links are broken and new ones are created as a node moves out of range of some nodes and enters the vicinity of other nodes. In cellular type networks, the topology changes as mobile nodes move and hand-off to different base stations. Unlike mobile nodes, sensor nodes are mostly stationary after deployment. Once positioned, sensor nodes will most likely remain stationary for the duration of their lifetime. Although nodes are static, the topology of a sensor network can constantly change. During periods of low activity, the network may enter a dormant state in which many nodes go to sleep to conserve energy. Also, nodes go out of service when the energy of the battery runs out or when a destructive event takes place [Park et al., 2000].

Distinct from the multimedia data streams of conventional wireless networks, sensor networks will likely have traffic with statistical properties [Sohrabi et al., 2000]. Although exact sensor data traffic properties are not known yet, it is clear that, due to the nature of the observed phenomena, the required bandwidth for sensor data will be low, on the order of 1-100 kbps [Pottie and Kaiser, 2000b].

Another difference between sensor networks and other kinds of wireless networks is that the main goal of the conventional wireless networks is providing high QoS (i.e., high throughput/low delay) and high bandwidth efficiency when mobility exists. For a sensor network, in contrast, we are interested in prolonging the lifetime of the network [Sohrabi et al., 2000]. Therefore, the conservation of energy is one of the most important aspect to be considered when designing algorithms for these environments.

Unlike traditional networks, wireless sensor networks are data-centric in a way that they are organized around the naming of data, not nodes [Estrin et al., 1999]. In these networks, nodes are neither unique nor reliable and applications express a need for a particular data element or type of data by naming it directly. Sensor networks are also application-specific in a way that they will be projected specially to the sensing task at hand. This means that intermediate nodes can perform application-specific data aggregation and caching.

The design of wireless sensor networks is a very challenging task. The characteristics described above impose new issues that have to be addressed. These issues potentially affect many aspects of the network design such as device-level communication primitives, routing, addressing mechanisms, application architectures, and security mechanisms [Estrin et al., 1999]. In [Estrin et al., 2001], the authors describe the following challenges to be faced by sensor networks designers:

- Untethered for energy and communication, requiring maximal focus on energy efficiency;
- Ad hoc deployment, requiring that the system identifies and copes with the resulting distribution and connectivity of nodes;
- Dynamic environmental conditions, requiring the system to adapt over time to changing connectivity and system stimuli;

- Unattended operation, requiring configuration and reconfiguration be automatic (self-configuration).

Furthermore, new metrics to evaluate these networks have to be defined. In [Tilak et al., 2002], the authors suggest the following ones: energy efficiency, latency, accuracy, fault-tolerance and scalability. In short, the design of wireless sensor networks is a very fertile area of research and more work needs to be done in order to set up this new class of network. In following sections, we describe some of the problems that are being researched.

2.3 Physical Layer Issues

Some studies are exploring the issues related to the design of sensors as energy-efficient as possible: [Bult et al., 1996, Asada et al., 1998, Kahn et al., 1999, Hill et al., 2000, Kahn et al., 2000, Pottie and Kaiser, 2000b, Rabaey et al., 2000, Shih et al., 2001]. These studies address the problem of designing hardware platforms that led to reduction in size, power consumption and cost of sensor nodes.

The project WINS [Asada et al., 1998, Pottie and Kaiser, 2000b] and the PicoRadio [Rabaey et al., 2000] are seeking ways to integrate sensing, signal processing, and radio elements onto a single integrated circuit that operates at micropower levels. Both projects have chosen to concentrate on radio frequency (RF) communications over short distances.

Researches involved in *Smart Dust* [Doherty et al., 2001b, Kahn et al., 2000, Kahn et al., 1999] explore the limits on size and power consumption in sensor nodes. Size reduction is important to make nodes as inexpensive and easy-to-deploy as possible and the power consumption is the paramount issue in the design of sensor nodes. This project incorporates the requisite sensing, communication (using optical devices) and computing hardware, along with the power supply, in a volume no more than a few cubic millimeters. However, the goal of this project is to create prototypes of Smart Dust that will be small enough to remain suspended in air, buoyed by air currents, sensing and communicating for hours or days.

In [Hill et al., 2000], the emphasis is on providing an initial exploration of system architectures for networked sensors. They describe a small sensor node and present a tiny microthreaded operating system, called TinyOS, to work in these nodes.

2.4 Link Layer Protocols

The limitations and characteristics of sensor networks certainly affect the design of media access control (MAC) protocols. A sensor network will be deployed in an ad hoc fashion, with individual nodes remaining largely inactive for long periods of time, but then becoming suddenly active when something is detected. The traffic tends to be variable and highly correlated. Over lengthy periods, there may be little activity or traffic, but, for short periods, the traffic may be very intense due to some abnormal event. Furthermore, there is very little buffering available on the node. Typically, a single packet is moving with only a few bits of buffering.

Another characteristic that affects the design of MAC protocols is that it is roughly the same cost per unit time to listen as to transmit or receive [Hill et al., 2000]. Every moment the radio is on, it consumes precious power. Thus, a key requirement is to turn the radio off whenever possible. In [Sinha and Chandrakasan, 2001], the authors use this idea to propose a dynamic power management to reduce system power consumption without significantly degrading performance. They propose an event-driven power consumption mechanism that a sensor node is shut down if no event occurs and wakes up when necessary.

The energy efficiency is the primary concern in designing a good MAC protocol for wireless sensor networks. In [Ye et al., 2002], the authors identify the major sources of energy waste in MAC protocols for sensor networks. The first one is *collision*. When a transmitted packet is corrupted it has to be discarded, and the follow-on retransmissions increase energy consumption. Collision increases latency as well. The second source is *overhearing*, meaning that a node picks up packets that are destined to other nodes. The third source is *control packet overhead*. Sending and receiving control packets consumes energy, and less useful data packets can be transmitted. The last major source of inefficiency is *idle listening*, listening to receive possible traffic that is not sent. Thus, a good MAC protocol has to reduce the waste of energy from all the above sources.

Another important attribute of a MAC protocol is its scalability in network size, node density and topology. A good MAC protocol should easily accommodate such network changes. Other important attributes include per-node fairness, latency, throughput and bandwidth utilization. As stated in [Ye et al., 2002], these attributes are generally the primary concerns in traditional wireless voice and data networks, but in sensor networks they are secondary. In [Woo and Culler, 2001, Ye et al., 2002], the authors propose new MAC protocols for wireless sensor networks to deal with specific characteristics of these networks.

2.5 Network Layer Protocols

In this section, we discuss some issues related to the network layer. In Section 2.5.1, we present some routing algorithms designed specifically for sensor networks and, in Section 2.5.2, we discuss some alternatives to the traditional network addressing mechanisms.

2.5.1 Routing

Sensor networks and other large-scale networks of small and embedded devices may require novel routing techniques for scalable and robust data dissemination [Ganesan et al., 2001]. Algorithms and protocols must be designed to provide a robust and energy-efficient communication mechanism to increase system lifetime. Another characteristic that affects the routing schemes is that, generally, in this kind of network, it is not feasible to have a unique address in each node. In Section 2.5.2, we will give more details about this issue.

In [Intanagonwiwat et al., 2000], it is proposed the *directed diffusion*, a new paradigm for communication between sensor nodes. The goal of directed diffusion is to establish efficient communication between sources and sinks. They introduce two new concepts. The first one is *data-centric* in which the data generated by sensor nodes is named by attribute-value pairs. A node requests data by sending interests for named data. Data matching the interest is then sent towards that node. The other concept is *data aggregation* in which intermediate nodes may aggregate several events into a single event to reduce transmissions and the amount of data size for system resource savings. The basic operation of the directed diffusion can be described as follows: a sensing task is disseminated throughout the sensor network as an interest for named data. This dissemination sets up gradients within the network designed to draw events. Events start flowing towards the originators of interests along multiple paths. The originator of the interest reinforces one, or a small number of these paths by which the data will be delivered. This work is extended in [Intanagonwiwat et al., 2002] in which a more energy-efficient approach is proposed to construct the data aggregation.

In [Heinzelman et al., 1999] and [Kulik et al., 1999], it is presented a family of adaptive protocols, called SPIN (Sensor Protocols for Information via Negotiation), that efficiently disseminates information among sensors in an energy-constrained wireless sensor network. The SPIN family of protocols incorporates two key innovations that overcome the deficiencies of traditional dissemination approaches: negotiation and resource-adaptation. Nodes negotiate with each other before transmitting data. This strategy helps ensure that only useful information will be transferred and then nodes will never waste energy on useless transmissions. Being aware of local energy resources allows sensors to cut back on activities whenever their energy resources are low to increase their longevity.

Some routing algorithms assume the existence of a localization system that enables each node to know its current position, like the Geographic and Energy Aware Routing (GEAR) proposed in [Yu et al., 2001]. GEAR uses energy and geographical information to route packets towards the target region. This protocol attempts to balance energy consumption to increase the network lifetime.

In [Heinzelman et al., 2000, Heinzelman, 2000], the authors propose the LEACH (Low-Energy Adaptive Clustering Hierarchy) algorithm, a clustering-based protocol that minimizes energy dissipation in sensor networks. This protocol utilizes randomized rotation of local cluster base stations (cluster-heads) to distribute the energy load among sensors in the network. Local compression is also used in order to reduce global communications.

A new routing algorithm is proposed in [Braginsky and Estrin, 2002]. In this work, the authors describe a method of routing queries to nodes that have observed a particular event. The basic idea is to create a path leading to each event and when a query is generated, it can be sent on a random walk until it finds the event path, instead of flooding it across the network. As soon as the query discovers the event path, it can be routed directly to the event. If the path cannot be found, the application can try re-submitting the query, or, as a last resort, flooding it. That work shows that under a wide range of conditions, it is possible to achieve an extremely high delivery rate, and flooding is a rare occurrence.

In [Xu et al., 2000], the authors present two algorithms for routing in energy-constrained ad hoc wireless networks. In these algorithms, nodes can trade off energy dissipation and data delivery quality according to application requirements. Algorithm analysis and simulation studies show that the energy-conserving algorithms can consume as little as 50% of the energy of an unmodified ad hoc routing protocol.

The concept of service differentiation as applicable to sensor networks is introduced in [Bhatnagar et al., 2001]. They argue that this kind of service is inherently required in sensor networks and also discuss various possible ways of providing service differentiation in sensor networks. In that work, it is proposed a simple forwarding algorithm called Adaptive Forwarding Scheme (AFS), which controls the reliability of a sensor network communication. A packet priority was used to determine the forwarding behavior for a packet. Packet with a higher priority will be forwarded multiple times. Thus, in AFS, redundancy is used as a basis to introduce reliability. Preliminary results indicate that the AFS is very promising.

In [Servetto and Barrenechea, 2002], the authors propose a probabilistic routing algorithm to achieve load balancing in large scale wireless sensor networks. They model a sensor network as a bidimensional grid in which each node has four neighbors. In that work, only one source and one destination is considered.

2.5.2 Addressing Mechanisms

In wireless sensor networks, the assumption that each node has a unique address can be a drawback. Specifically if the data rate is low compared to the address size and the number of transactions seen by an individual node is small compared to the number of existing nodes. As an alternative, we can configure the network with addresses that are locally unique. That is, each node on a network has an address that is unique with respect to its potential peers, based on the connectivity of the network or the scope of communication. Thus, devices that are mutually disconnected may share the same address at the same time. To maintain local addresses, a sensor network could use a protocol that dynamically assigns addresses to nodes based on the addresses of other nodes in the neighborhood. However, as the network topology becomes more dynamic, more work is required to keep addresses locally unique.

In [Elson and Estrin, 2000, Elson and Estrin, 2001a], the authors present an Address-Free Architecture (AFA) as an alternative to traditional statically-allocated, globally unique network addresses for sensor networks. In that work, instead of using statically assigned addresses that are guaranteed to be unique, nodes randomly select probabilistically unique identifiers for each new transaction. Because new identifiers are selected for each transaction, the AFA identifier alone is not sufficient to tell a receiver which node sent a packet, or even if two successive packets were sent by the same node. In situations like these, a specific node needs to be identified and then unique identifiers, statically assigned are used. In these cases, the unique identifier of a node can be sent as data, on demand, instead of having it in the header of every packet. The AFA scales well because the identifier size is tied to the average number of transactions that occur in the same place at the same time in a network, and this factor can remain constant in a network, even if it grows in size.

2.6 Distributed Algorithms

Sensor network applications are those in which sensor nodes cooperate to perform a higher-level sensing task. Clearly, this kind of coordination can be structured in a centralized manner. Individual sensors report their data to a central node, which then performs the computation required for the application. This centralized structure is not, in general, a good choice for several reasons [Estrin et al., 1999]: it provides a single point of failure, it can be energy inefficient, and it does not scale to large networks. Thus, it is natural to design sensor networks using distributed algorithms.

Some studies have been done in the direction of proposing new distributed algorithms specific for sensor networks. In [Estrin et al., 1999], a new paradigm is proposed for the design of these algorithms. They argue that sensor network coordination applications are better realized using *localized algorithms*. Localized algorithms are a distributed computation in which sensor nodes only communicate with sensors within their neighborhood, yet the overall computation achieves a desired global goal. Localized algorithms have two attractive properties. First, because each node communicates only with other nodes in some neighborhood, the communication overhead scales well with the increase in network size. Second, for similar reasons, these algorithms are robust to network partitions and node failures. The authors exemplify the use of localized algorithms by presenting a localized clustering for electing sensors to form the widest baseline for locating external objects. The goal of that algorithm is to elect cluster-head sensors such that each sensor in the multi-hop network is associated to a cluster-head sensor as its parent. The parent-child relationships are established only between sensors that are able to communicate with each other, thus preventing inconsistencies due to asymmetric communication. The task of this sensor network is to identify, in an energy-efficient manner, the exact location of some objects.

In Sections 2.6.1 to 2.6.5, we describe some proposals in the following areas: Location Mechanisms, Time Synchronization, Coverage Calculation, Topology Discovery and Energy Map, respectively.

2.6.1 Location Mechanisms

As many sensor networks are embedded to monitor or control the behavior of physical systems, nodes often need to label data or events with location information [Bulusu et al., 2001b]. Thus, in sensor networks, it is desirable that nodes can locate themselves in various environments, and on different distance scales. This problem is called *location* and is extremely crucial for many applications of a network of many devices. In some applications, nodes often need to determine their action based on their physical location (am I the right sensor to monitor a particular object?). In context-aware applications, location enables the intelligent selection of appropriate devices, and may support useful coordination among devices. In ad hoc networks, knowledge of node location can also assist in routing [Basagni et al., 1998, Câmara and Loureiro, 2000, Ko and Vaidya, 1998].

However, the incorporation of location awareness in wireless sensor networks is far from a trivial task [Savvides et al., 2001]. Since the network can consist of a large number of nodes that are deployed in an ad hoc fashion, the exact nodes locations are not known a priori. Besides, according to [Bulusu et al., 2000, Savvides et al., 2001], the straightforward solution of adding GPS to all nodes in the network is not practical since:

- The production cost factor of GPS can become an issue when a large number of nodes are to be produced.
- GPS does not work indoors, under water, or in the presence of overhead obstructions such as a dense foliage.
- The power consumption of GPS will reduce the battery life on sensor nodes, thus reducing the effective lifetime of the entire network.

- The sensor node is required to be small and unobtrusive, and GPS and its antenna would increase the sensor node form factor.

Thus, in sensor networks, it is necessary to design GPS-free algorithms to provide the location of each node.

Location algorithms typically rely on some form of communication between reference points with known positions and the receiver node that needs to be localized. If the number of reference points, or beacons, is large, each node can infer its proximity to those beacons by listening to broadcasts from a collection of nearby beacons. In [Bulusu et al., 2000], the authors present an idea in which a fixed number of nodes in the network, with overlapping regions of coverage, serve as reference points. These nodes are situated at known positions that form a regular mesh and transmit periodic beacon signals containing their respective positions. They assume that neighboring reference points can be synchronized so that their beacon signal transmissions do not overlap in time. Nodes use a simple connectivity metric to infer proximity to a given subset of these reference points and, thus, localize themselves to the centroid of the selected reference points. The location accuracy depends on the separation distance between two adjacent reference points and the transmission range of these reference points.

In other approaches, the density of beacons is not sufficient in some areas of sensor networks. In these situations, each beacon information is propagated through multiple hops to enable location estimation in an area of low beacon density [Doherty et al., 2001a, Niculescu and Badrinath, 2001, Savvides et al., 2001].

Another subject related to the problem of location is beacon placement. The location of beacons strongly affects the quality of spatial location. In [Bulusu et al., 2001b], the authors present the design, evaluation and analysis of three novel adaptive beacon placement algorithms based on RF-proximity.

2.6.2 Time Synchronization

In wireless sensor networks, the use of synchronized time can be useful in many tasks [Elson and Estrin, 2001b]: to integrate a time series of proximity detections into a velocity estimate; to measure the time-of-flight of sound for locating its source; or to suppress redundant messages by recognizing that they describe duplicate detections of the same event by different sensors. Existing time synchronization methods need to be extended to meet these new needs. In [Elson and Estrin, 2001b], the authors present an implementation of a sensor network time synchronization scheme that uses the idea of *post-facto synchronization*, a low-power method for synchronizing clocks in a local area when accurate timestamps are needed for specific events.

2.6.3 Coverage Calculation

Wireless sensor networks can provide a connection between the Internet and the physical world [Meguerdichian et al., 2001]. One of the fundamental problems in sensor networks is the calculation of coverage that is a measure of how well an object, moving on an arbitrary path, can be observed by the sensor network over a period of time. In these networks, a minimal coverage path provides valuable information about the worst case exposure-based coverage.

In [Meguerdichian et al., 2001], an algorithm is proposed for finding the minimal exposure path for any given distribution and characteristics of sensor networks. They give polynomial time centralized algorithms to solve the questions optimally. This problem is also addressed in [Li et al., 2003], where the authors consider a more general sensing model in which the sensing ability diminishes as the distance increases.

2.6.4 Topology Discovery

Network topology is an important model of the network state as it implicitly gives lots of information about the active nodes present and the connectivity/reachability map of the system [Deb et al., 2002]. Network topology can aid in network management and performance analysis. For any network, accurate knowledge of network topology is a prerequisite to many critical network management tasks, including proactive and reactive resource management and utilization, server siting, event correlation, root cause analysis, growth characteristics and even for use in simulation for networking research [Deb et al., 2003].

Many topology discovery algorithms exist for wired networks but most of them introduce a lot of additional traffic into the network. Some others use network models suitable for the

Internet based on power laws of Internet topology. These may not be useful for sensor networks. In [Deb et al., 2002], a topology discovery algorithm (*TopDisc*) for wireless sensor networks is described. *TopDisc* selects a set of distinguished nodes, and constructs a reachability map based on their information. *TopDisc* logically organizes the network in the form of clusters and forms a Tree of Clusters (*TreC*) rooted at the monitoring node. *TopDisc* is completely distributed and uses only local information.

2.6.5 Energy Map

The energy map gives the information about the remaining available energy in each part of the network. The work proposed in [Zhao et al., 2002] tries to obtain the energy map of sensor nodes using an aggregation-based approach. The process of constructing the energy map can be described as follows:

1. Determining local residual energy: at each node, the residual energy level is measured periodically. This measure can be obtained if each node gauges the battery voltage and uses a table, generated from battery manufacturer's data sheets, that tells how much battery is left. A sensor node only needs to report its local energy information when there is a significant energy level drop compared to the last time the node reported it.

2. Disseminating local energy information: the local energy information of each node must be disseminated across the network to compute the energy map of the entire network. For this to happen, the monitoring node sends a message requesting the energy information. This message propagates throughout the network by flooding. Upon receiving this message, each node sets the sender as its parent node leading towards the monitoring node. A tree is then constructed whose root is the monitoring node. Next, each node sends its local energy to its parent in the tree. This tree is refreshed periodically to adapt to network dynamics and failures.

3. Aggregation: along the path to the monitoring node, nodes that receive two or more energy information may aggregate them according to several rules. If the energy information are topologically adjacent and have the same or similar energy level, they can be aggregated into a tuple which contains a polygon that describes a collection of nodes,

and the range of the residual energy level at those nodes. The goal of aggregation is to reduce the messaging cost on collecting the energy information while losing little critical information content.

In that work one can find simulation results that compare the proposed approach with a centralized collection of individual residual energy. However, in simulations, they do not take into account the overhead due to the periodical refreshment in the aggregation tree.

An important difference between the approach proposed in [Zhao et al., 2002] and our work is that, in their approach, each node sends to the monitoring node only its available energy while in our work we also send the parameters of a model that tries to predict the energy consumption in the near future. Thus, in our approach, each node sends to the monitoring node its available energy and also the parameters of the model chosen to represent its energy drop. With these parameters, the monitoring node can update locally the current available energy in each node of the network, decreasing the number of energy information packets in the network.

2.7 Applications

As sensor networks are application-specific, the definition of the application is paramount in the design of these networks. Clearly, sensor networks can be used in a wide range of areas such as: biology, control, environment, military, security and traffic. Bellow, we describe some possible applications of these networks.

- **Biology:** sensor networks offer the promise of significant advances in biomedical areas. As stated in [Schwiebert et al., 2001], smart sensors are being considered for several biomedical applications. In this case, sensors are designed to be embedded and operate within the human body in order to aid in medical treatment. For example, wireless biomedical sensors may provide a more effective way to treat diabetes, by creating a more consistent, accurate and less invasive method for monitoring glucose levels. These sensors could be implanted in the patient once, and they would monitor the glucose level and transmit the results to a wristwatch display. This would allow for a less invasive way to monitor the glucose levels several times daily, as well as providing more accurate results.

Wireless biomedical sensor may also play a key role in early detection of cancer. Studies have shown that cancer cells produce and release nitric oxide, which affects the blood flow in the area surrounding a tumor. A sensor with the ability to detect these changes in the blood flow can be placed in suspect locations. It is likely that any abnormalities could be detected much sooner with sensors than without them [Schwiebert et al., 2001].

- **Control:** in industry, a group of sensors may be used to closely monitor an automated assembly line [Park et al., 2000]. They can sense a machine malfunction, monitor process transition from one stage of the assembly line to the next one and assist in quality control.

  In [Estrin et al., 1999], the authors describe a possible scenario in which a sensor network could aid in an inventory. Imagine that each item of inventory in a factory warehouse or office complex has, attached to it, a tag. Stick-on sensors, discreetly attached to walls, or embedded in floors and ceilings, track the location history and use of items. The sensor network can automatically locate items, report on those needing servicing, analyze long-term correlations between workflow and wear, report unexpected large-scale movements of items or significant changes in inventory levels. Some systems today (for example, those based on bar-codes) provide inventory tracking; full sensor-net based systems will eliminate manual scanning and provide more data than simply a location.

- **Environment:** we can use a sensor network to monitor environmental variables either indoor such as in buildings and homes or outdoor as forests and seas.

  In the indoor case, smart environments instrumented with sensor and wireless enhanced objects would be able to sense events and conditions about people and objects in the environment, and act upon the sensed information or use it as context when responding to queries and commands. An interesting application of such deeply instrumented physical environments is discussed in [Srivastava et al., 2001]. In that work, it is presented an application of sensor-based wireless networks to a "Smart Kindergarten" in which objects that children play with on regular basis will be wirelessly networked and have sensing capabilities. The main goal of this project is to enhance the education process by providing a childhood learning environment that is individualized to each child, adapts to the context, coordinates activities of multiple children, and allows continual evaluation of the learning process by the teacher.

In the outdoor environment, these networks will be able to monitor a wide variety of ambient conditions: temperature, pressure, humidity, soil makeup, noise levels, lighting conditions, the presence or absence of certain kinds of objects, mechanical stress levels on attached objects, weather forecast data, motion of a tornado, fire detection in a forest, pollution detection along beaches and monitoring species and ecosystems.

In [Estrin et al., 1999], the authors describe a possible scenario where a sensor network could be used. Thousand of disposable sensors are densely scattered over a disaster area. Some of them fall into regions affected by the disaster (as a fire for example) and are destroyed. The remaining sensors collectively map these affected regions, direct to the nearest emergency response teams to affected sites, or find safe evacuation paths. Disaster recovery today is by comparison very human intensive.

- **Military:** sensor nodes can be deployed in combat scenarios to track troop movements or to monitor the activities of the enemy by detecting motion and tracking targets. Also, sensors placed on small robots could detect land mine or the use of biological or chemical weapons and report their presence in time to protect troops.

- **Security:** sensor networks can also be used in security applications, like the one proposed in [Pottie and Kaiser, 2000b]. In that work, it is presented a security application designed to detect, identify and report threats within some geographic region.

- **Traffic:** the control of traffic conditions can be improved using sensor networks. We can, for example, have sensors attached to taxi cabs in a large metropolitan area to study the traffic conditions and plan routes effectively. In [Estrin et al., 1999], the authors describe a scenario where a sensor network aid in monitoring of automobile traffic in urban areas. Consider that every vehicle in a large metropolis has one or more attached sensors. These sensors are capable of detecting their location, vehicle sizes, speeds and densities, road conditions and so on. As vehicles pass each other, they exchange information summaries. These summaries eventually diffuse across sections of the metropolis. Drivers can plan alternate routes, estimate trip times, and be warned of dangerous driving conditions. Unlike the centralized systems sometimes seen today, one based on a local communication would scale as the number of vehicles grows and provides much greater local detail.

2.8 Concluding Remarks

Sensor networks are a new kind of ad hoc network with some new characteristics and challenges. These networks perform distributed sensing, wireless communication and distributed processing using very limited resources. The design of wireless sensor networks is a very challenge task and some new issues have to be addressed. These issues potentially affect many aspects of the network design such as device-level communication primitives, routing, addressing mechanisms, application architecture, and security mechanisms. The design of wireless sensor networks is a very fertile area of research and more work needs to be done in order to set up this new kind of network. An important point of all solutions for these networks is the efficient use of the limited energy resources.

Chapter 3

Taxonomy for Wireless Sensor Networks

3.1 Introduction

In wireless sensor networks, the performance of the protocols depends very much on the characteristics of the networks, such as the data delivery model, topology, density and dynamism of sensors, phenomenons and observers. Therefore, in order to determine the performance of a network protocol for these networks, it is important to specify in which kind of wireless sensor networks this protocol will be used. To this end, the goal of this chapter is to present a taxonomy for wireless sensor networks in which these networks are classified according to different characteristics. We intend to organize sensor networks taking into account their infrastructure, and the characteristics of the phenomenon and observer. This taxonomy will aid in identifying the kind of wireless sensor networks we will be working with in next chapters. In the following sections, we classify the different types of infrastructure, phenomenon and observer.

3.2 Infrastructure

The infrastructure consists of sensors and their current deployment status. More specifically, the infrastructure is influenced by the characteristics of the sensors and the deployment strategy [Tilak et al., 2002]. Initially, we discuss the physical characteristics of the sensors, the possible deployment strategy, and the density of wireless sensor networks. Next, these networks are classified taking into account their topologies and their communication models.

Sensors are the device that implements the physical sensing of environmental phenomena and reporting of measurements. Typically, it consists of five components: sensing hardware, memory, battery, embedded processor, and transceiver. The sensors can be classified according to its dynamism, heterogeneity, and characteristics of the battery. Bellow, we explain these features.

- Dynamism: in static sensor networks, there is no motion among sensors. Once the sensors are placed in the field, they stay still until the end of the network. A group of sensors spread for temperature sensing is an example of a static sensor network. On the other hand, if sensors are dynamic, they can move with respect to each other. For example, consider an application of traffic monitoring implemented by attaching sensor to taxis. As the taxis move, the attached sensors continuously communicate with each other about their own observations of the traffic conditions. In this case, the path between the sensors and the observer may fail and either the observer or the concerned sensors must take the initiative to rebuild a new path.

- Heterogeneity: wireless sensor networks can be homogeneous or heterogeneous with relation to the types, dimensions and functionalities of sensor nodes. In a homogeneous network, all nodes have the same physical characteristics, such as its dimension, battery capacity, and memory size. In a heterogeneous network, some nodes have a special hardware. As an example, in surveillance applications, some nodes can be responsible for collecting and processing image data. In this case, these nodes have to be equipped with more sophisticated hardware in order to be able to make this processing. Some hierarchical networks can also make use of heterogeneous nodes that are those responsible for doing special tasks. As an example, the battery of the cluster head should have more energy because this node will make more communication and processing than other sensor nodes. It is important to point out that, in wireless sensor networks, it is better to maintain hierarchy by hardware than by software. The overhead in communication needed to keep hierarchy in homogeneous network can be prohibitive in probably all kinds of networks. Thus, we support the idea that the hierarchy must be kept by hardware instead of software, meaning that a hierarchical network should use heterogeneous nodes.

- Battery: represents the source of energy and can be divided in two main groups. In the first one, we have batteries that do not provide any kind of reload, meaning that they have finite capacity. In the second group, we have batteries that can be reloaded. As in most sensor network applications, the replacement of a battery is unfeasible or impossible, the more plausible way of making battery reload is making sensor nodes extract energy from the environment (energy-scavenging). Solar cells can contribute up to 15 mW per square centimeter during direct sunlight hours, and up to 0.15 mW on cloudy days [Rabaey et al., 2000]. Averaging over daylight and nighttime hours, and considering nodes in the interior of the building or embedded in ceiling tiles, shows that solar cells can just barely serve as the sole energy source for sensor nodes. Therefore, additional sources of energy would be welcome. Harvesting energy from vibrations is a promising approach. Raised floors and dropped ceilings in most office buildings exhibit measurable vibrations that can be harnessed [Rabaey et al., 2000]. Advances in devices make integrated and tiny variable capacitors a reality. Power outputs between 10-100 μW per cubic centimeter are plausible from vibrations in a normal office building using existing technology [Rabaey et al., 2000].

The infrastructure of a wireless sensor network can also be classified according to the deployment strategy. The sensor nodes can be arranged in a random or planned way. When sensor nodes are deployed in a random topology, there is no a priori knowledge of location. In many scenarios, nodes will need to determine their relative positions and self-organize into a special coordinate system without relying on remote infrastructure such as GPS. On the other hand, when nodes are organized in a planned topology, the exact position of each node is carefully studied and known a priori.

Density is another important parameter in classifying sensor networks. We can express the network density $\mu(R)$ in terms of the number of nodes per nominal coverage area [Bulusu et al., 2001a]. If N nodes are scattered in a region of area A, and the nominal range of each node is R,

$$\mu(R) = \frac{N \times \pi \times R^2}{A}. \tag{3.1}$$

Various phenomena saturate at a certain critical network density particular to them. Beyond this critical node density, additional nodes do not necessarily provide additional sensing, communication or coverage fidelity. As an example, in [Kleinrock and Silvester, 1978], the authors show that in a wireless network with a uniform distribution of nodes, when $\mu(R)$ is 6 nodes, the probability that a node is connected reaches 1 regardless of actual node placement. Using this information, in this work, we consider that if the value $\mu(R)$ is smaller than 6, we have a low density network. If this value is between 6 and 15, we have a medium density network. On the other hand, networks with $\mu(R) > 15$ are a high density network. This classification is only a way of identifying the density of the network that is being analyzed.

The topology defines the way sensors communicate with each other. On a hierarchical topology, nodes are organized in groups, and some nodes have special roles in the network, working as a cluster-head or leader, as example. On the other hand, on a flat topology, the nodes are not organized in clusters, no infrastructure is defined, and no node has a special role in the network.

Defining the communication models is another important characteristic of sensor networks. Basically, there are two models of communication: cooperative and non-cooperative [Tilak et al., 2002]. Under the cooperative sensor model, sensors communicate with each other to realize the observer interest. This communication is beyond the relay function needed for routing. For example, in a clustering protocol, a cluster-head and sensor nodes communicate with each other for information dissemination related to the actual phenomenon. In-network data processing is another example of cooperative sensors. Non-cooperative sensors do not cooperate for information dissemination.

3.3 Phenomenon

Phenomenon, or event, is the entity of interest to the observer that is being sensed and potentially analyzed/filtered by the sensor network. Multiple phenomena may be under observation concurrently in the same network. The terms phenomenon and event are interchangeable in this work.

The phenomenon can be static or dynamic. A typical example of a moving phenomenon is sensors deployed for animal detection. Depending on the density of the phenomena, it will be inefficient if all sensor nodes are active all the time. Only sensors in the vicinity of the mobile phenomenon need to be active. The number of active sensors in the vicinity of the event can be determined by application specific goals, such as accuracy, latency, and energy efficiency [Tilak et al., 2002]. Sensors that are used to analyze the motion of a tornado form another example of dynamic phenomenon. On the other hand, some phenomena do not move. A group of sensors spread for temperature sensing is an example of a static phenomenon.

The events can also be classified according to their durations as instantaneous, lasting and forever. An example of an instantaneous event is the evaluation of the movement of a rare bird that appears and disappears in the sensor field very fast. Lasting events represent most of the phenomenon observed in the applications envisioned for sensor networks. These kinds of events are characterized as having a start and an ending time well defined. As an example, we have the application in which wireless biomedical sensors are responsible for monitoring glucose levels in diabetic patients. Other events can have a forever duration as sensors deployed in a forest for fire detection, or any other physical catastrophe detector.

3.4 Observer

Observer is the end user interested in obtaining information disseminated by sensor networks about the phenomenon. The observer may indicate interests (or queries) to the network and receive responses to these queries. Typically, the observer is named monitoring node, sink node or end user.

The observer can also be classified as static or dynamic. If the observer is dynamic, it is mobile with respect to the sensors and phenomena. As an example, consider sensors deployed in an inhospitable area for environment monitoring. A plane might fly over a field periodically to collect information from a sensor network. In this situation, the observer, in the plane, is moving in relation to the sensors and phenomena in the ground.

Sensor networks can be classified in terms of the data delivery required by the observer interest as [Tilak et al., 2002]: continuous, event-driven, observer-initiated, and hybrid. These models govern the generation of the application traffic. In the continuous model, sensors communicate their data continuously at a pre-specified rate. In an event-driven data model, sensors report information only if an event of interest occurs. In this case, the observer is interested only in the occurrence of a specific phenomenon or set of phenomena. A scenario where a sensor network is deployed for intrusion detection is an example of an event-driven data model. In this case, the event is an intruder entering the area. In the observer-initiated model, sensors only report their results in response to an explicit request from the observer, either directly, or indirectly through other sensors. Finally, when the three approaches can coexist in the same network, we refer to this model as the hybrid model.

We can classify a sensor network according to the number of observers, or the number of nodes that are requesting information from the network. Most of the proposals assume a single observer. Multiple observers can be supported as multiple instances of a single observer. More sophisticated protocols could also deal with multiple observers by merging their related interests.

3.5 Taxonomy

Table 3.1 summarizes the taxonomy proposed in this chapter. Throughout the remainder of this work, the kind of sensor network we will deal with is one in which nodes are static and homogeneous, the replacement or reload of battery is unfeasible or impossible, and also that there is only one static monitoring node with plenty of energy. We suppose that nodes are deployed randomly forming a high-density network in a flat topology. In relation to the data delivery model, we simulate an event-driven network in such a way that sensors report information only if an event of interest occurs. In this case, the monitoring node is interested only in the occurrence of a specific event or set of events. The communication model used is a cooperative sensor model in which sensors communicate with each other to disseminate information related to the event. Considerations about the phenomenon analyzed in this work will be given in Chapter 4.

Infrastructure		
Sensors	Dynamism	static dynamic
	Heterogeneity	homogeneous heterogeneous
	Battery	with reload without reload
Deployment Strategy		random planned
Density		low medium high
Topology		flat hierarchical
Communication Models		cooperative non-cooperative
Phenomenon		
Dynamism		static dynamic
Duration		instantaneous lasting forever
Observer		
Dynamism		static dynamic
Data Delivery Models		continuous event-driven observer-initiated hybrid
Number of Observers		one multiple

Table 3.1: Taxonomy for wireless sensor networks.

Chapter 4

Energy Dissipation Model

4.1 Introduction

When simulation is used to analyze the performance of the energy map construction or any other energy related problem, we have to know how the energy dissipation happens in sensor nodes. To this end, in this work, we propose an *energy dissipation model* that tries to model the energy drop in sensor nodes. To our knowledge, there is only one work that has addressed this problem [Zhao et al., 2002]. In that work, two energy dissipation models are proposed. The first one is the *uniform dissipation* model. During a sensing event, each node n in the network has a probability p of initiating a local sensing activity, and every node within a circle of radius r centered at node n consumes a fixed amount of energy e. The second one is the *hotspot dissipation* model. In this model, there are h fixed hotspots uniformly distributed on the sensor field. Each node n has a probability of $p = f(d)$ to initiate a local sensing activity, and every node within a circle of radius r centered at node n consumes a fixed amount of energy e, where f is a density function, and $d = min_{\forall i}\{|n - h_i|\}$ is the distance from node n to the nearest hotspot.

The main drawback of these models is that they consider that, when an event occurs, all nodes inside its area of influence will immediately see this event. This supposition can be plausible only in situations where all nodes in the sensor field keep their sensing on during all time. This may not be an appropriate approach to deal with sensor networks. As stated in [Hill et al., 2000], the best way to save energy is to make unused components inactive whenever possible. Thus, in sensor networks, nodes or parts of nodes that are not in use should be turned off to conserve energy.

Another limitation of the work proposed in [Zhao et al., 2002] is that they do not model any communication among sensors, neither between sensors and the monitoring node. In a sensor network, nodes have to communicate in two main situations. Firstly, they have to communicate in order to perform cooperative tasks that are those in which sensors communicate with each other to disseminate information related to the event. For example, nodes can exchange information about a moving object in order to send to the monitoring node the best approximation about the object position. Secondly, nodes have to communicate in order to route the sensed information to the monitoring node. In this case, nodes near the monitoring node are used more frequently to route information collected in the entire network. Neither of these communications are modeled by the *uniform dissipation model* nor the *hotspot dissipation model*. Other problems of these models include the assumption that all nodes working in a sensing event consume the same amount of energy, and that all events have the same radius of influence.

In this chapter, we propose a State-based Energy Dissipation Model (SEDM) that tries to represent more realistically the behavior of a sensor network in terms of energy dissipation. Based on this model, more pragmatic results can be obtained when evaluating solutions to wireless sensor networks that take into account energy aspects of sensor devices.

We make some considerations when studying and evaluating our model for wireless sensor networks. We propose the use of various operation modes with different levels of activation and with different levels of energy consumption for sensor nodes. In the evaluation of our model, we assume that the arrival of events in the network follows a Poisson process or a Pareto distribution. In addition, three other models are used to describe the behavior of events. In the first one, the event is static and has a fixed size. In the second model, the event moves around in the sensor field and, in the third one, the event is static but its area of influence increases with a given rate.

The kind of sensor network we will work is one in which the information about the events are sent to the monitoring node only at the end of each event. Also, we suppose that all nodes in the area of influence of one event have to work on that event.

This chapter is organized in the following way. Section 4.2 presents the energy dissipation model that we propose to describe the behavior of a sensor node and to simulate its energy drop. Section 4.3 describes an ideal energy dissipation model that is used as a comparison model to evaluate the state-based energy dissipation model, and Section 4.4 analyzes the performance of our approach. Finally, in Section 4.5, we present the conclusions of this chapter.

4.2 State-based Energy Dissipation Model

4.2.1 Modeling

The conservation of energy is the paramount issue in the design of sensor networks. According to [Hill et al., 2000], a wireless sensor network should embrace the philosophy of getting the work done as quickly as possible and going to sleep. This can be achieved in a framework in which the nodes have various operation modes with different levels of activation, and, consequently, with different levels of energy consumption. Besides, as soon as possible, nodes go to a mode that consumes less energy. Using this idea, we propose a State-based Energy Dissipation Model (SEDM) to describe how the energy is spent in a sensor node.

The energy consumption in a sensor node can be divided in three main parts: sensing, data processing and communication [Sohrabi et al., 2000]. In a wireless sensor network, communication is the major consumer of energy. Taking the example described in [Pottie and Kaiser, 2000b], the energy cost of transmitting 1 Kb over a distance of 100 m is approximately 3 J. By contrast, a general-purpose processor with 100 MIPS/W power could execute 300 million instructions for the same amount of energy. Thus, in wireless sensor networks, nodes have to do more processing, as opposed to exchanging raw data over the air. In this work, we model only the energy spent by communication and sensing of the node. Analyzing all possible combinations of performing sensing and communication, we find six operation modes:

- Mode 1: sensing off, radio off;
- Mode 2: sensing on, radio off;
- Mode 3: sensing off, radio receiving;

Components	**Active (mA)**	**Idle (mA)**	**Inactive (μA)**
MCU core (AT90S8535)	5	2	1
MCU pins	1.5	-	-
Co-proc (AT90LS2343)	2.4	0.5	1
EEPROM (24LC256)	3	-	1
Radio (RFM TR1000)	12 tx, 4.5 rx	-	5
Temperature Sensor(AD7416)	1	0.6	1.5

Table 4.1: Current per hardware component of a sensor node [Hill et al., 2000].

- Mode 4: sensing on, radio receiving;
- Mode 5: sensing off, radio transmitting;
- Mode 6: sensing on, radio transmitting.

The values of power consumption for each operation mode were calculated based on information presented in [Hill et al., 2000]. In that work, the authors describe the sensor network architecture of the WeC motes, and some physical specifications, such as the values of current required by each hardware component of the sensor node. These values are presented in Table 4.1.

Components	**Mode 1**	**Mode 2**	**Mode 3**	**Mode 4**	**Mode 5**	**Mode 6**
MCU core	inactive	active	active	active	active	active
MCU pins	inactive	active	active	active	active	active
Co-proc	inactive	active	active	active	active	active
EEPROM	idle	active	active	active	active	active
Radio	inactive	inactive	active (rx)	active (rx)	active (tx)	active (tx)
Temp. Sensor	inactive	active	inactive	active	inactive	active

Table 4.2: Activity in each operation mode.

Using the values of Table 4.1, we can find the power consumption of each operation mode. In Table 4.2, we have the specification of which part of the sensor node is active in each operation mode. With the information of these tables, and knowing that sensors work with a voltage of 3 V, we can calculate the power consumption for each mode:

- Mode 1: 28.50 μW;
- Mode 2: 38.72 mW;
- Mode 3: 49.20 mW;
- Mode 4: 52.20 mW;
- Mode 5: 71.70 mW;

- Mode 6: 74.70 mW.

Nevertheless, from a practical point of view, not all operation modes are useful due to limitations in the sensor node. The hardware of these nodes does not have the capability of changing between so many states. Besides, in practice, the state transitions consume energy and time. In general, a deeper sleep state consumes less power and has a longer wake-up time [Sinha and Chandrakasan, 2001]. However, in our model, we do not take into account the overhead of these state transitions.

In order to simplify the model that describes the behavior of the sensor node, we use fewer operation modes, eliminating some states. Observing the power consumption of the six operation modes, we can see that there is only a small difference between the power consumption of modes 3 and 4, and between modes 5 and 6. The difference between these modes are the sensing part that consumes only 1 mA. Therefore, in our energy dissipation model, we consider both modes 3 and 4, and modes 5 and 6, as being the same mode. With these modifications, we simplify the energy dissipation model because fewer transition functions have to be defined. Thus, in the energy dissipation model proposed in this work, the following operation modes are used:

- Mode 1: sensing off, radio off (28.50 μW);
- Mode 2: sensing on, radio off (38.72 mW);
- Mode 3: sensing on, radio receiving (52.20 mW);
- Mode 4: sensing on, radio transmitting (74.70 mW).

These values will be used throughout all simulations.

In the SEDM, in each instant of time, each node is in one of the four operation modes. The transitions between these modes are described by the diagram of Figure 4.1. In this diagram, the operation modes are represented by states 1, 2, 3 and 4. In addition, it was necessary to represent more two states 2' and 3'. The state i' also represents the operation mode i. As example, both states 2 and 2' represent the operation mode 2, the only difference is that when a node goes to state 2, it always starts a timer, while in state 2', it verifies if there is any event for it. Thus, in terms of energy consumption, state i is exactly the same as state i'. The only difference is that the behavior of a node that goes to state *i* is different from the one that goes to state *i*'.

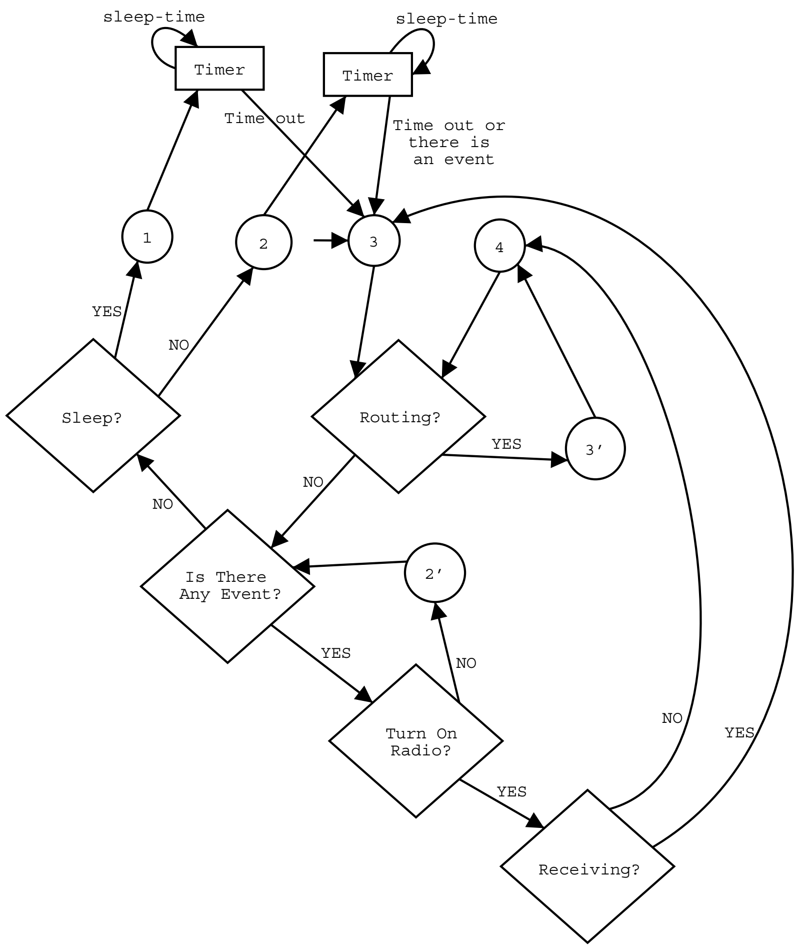


Figure 4.1: Diagram of the State-based Energy Dissipation Model.

In the SEDM, at the beginning of simulation, all nodes go to state 3. When a node is in state 3, its radio is on, and thus it can work as a router, transmitting packets in the direction to the monitoring node. If the node has to act as a router, it goes to state 3, and then to state 4. If the node does not have to work as a router, it verifies if there is any event for it. If there is no event for this node, it verifies if it goes to state 1 or 2. When there is an event for it, the node verifies if it is necessary to turn on its radio. This task is useful to model the cooperation needed in the applications envisioned for sensor networks. If the node has to turn on its radio, another test is executed to see if this node has to turn on its radio in the receiving or in the transmitting mode.

When a node goes to state 1, it will be sleeping for *sleep-time* seconds. During this period, this node will be saving energy but it will not be able to communicate or to sense any event. After *sleep-time* seconds, the node wakes up and goes to state 3 to see if there is any event for it or if there is any node trying to communicate with it. If a node goes to state 2, it will be in this state for *sleep-time* seconds, but unlike in state 1, a node that is in state 2 can see the occurrence of an event because, in this state, the sensing is on. If an event occurs during the *sleep-time* seconds, or the time *sleep-time* ends, the node goes to state 3 to see if there is any node trying to communicate with it.

In the model of Figure 4.1, there are five tests. The answers for these tests are guided by parameters of the model. Bellow, we describe these parameters.

- "Routing?"– When an event ends, all nodes in the straight line between the center of this event and the monitoring node have their *routing-bit* set to one. When a node goes to either state 3 or 4, it checks this *routing-bit* to see if it is necessary to work as a router. The "Routing" test only checks the *routing-bit* to see if it is necessary to turn on the radio in order to route packets to the monitoring node. Using this test, we model the communications needed to route the sensed information to the monitoring node.

- "Is There Any Event?"– When a node is in the operation modes 2, 3 or 4, its sensing is on, and thus it can see if there is an event for it. This test checks it.

- "Sleep?"– This answer is obtained using the parameter *sleep-prob*. With probability *sleep-prob*, the node goes to state 1, and with (1−*sleep-prob*), it goes to state 2. The greater the value of *sleep-prob*, the smaller will be the coverage area. However, more energy will be spent in this case.

- "Turn On Radio?"– This question is answered using the parameter *turn-on-radio-prob*. This parameter models the communication needed to perform cooperative tasks that happen when nodes working in a sensing activity exchange information in order to get a better knowledge about the observed phenomenon.

- "Receiving?"– This question is answered using the parameter *receiving-prob*. When a node has to turn on its radio to perform cooperation, it goes to state 3 with probability *receiving-prob*, and to state 4 with probability (1−*receiving-prob*).

As described above, in this work, we model the transmission of the sensed information to the monitoring node. We simulated this behavior setting to one the *routing-bit* of all nodes in the straight line between the central point of the event and the monitoring node. Thus, when these nodes turn on their radios, they go for a short time to state 3 and after that to state 4. It is clear that this is a really simple model, because it does not deal with the problems faced by the routing algorithms, like retransmissions and collisions. However, this simplistic model can give us a lower bound of the amount of energy spent by any routing algorithm. Accordingly, in practice, whatever routing algorithm will spend more energy than the amount spent by the SEDM.

The state transition described above tries to capture the behavior of a sensor node specially in terms of energy consumption. As there are no real large sensor networks implemented already, we have no information about the real energy dissipation of a sensor node. Although, we believe that, for our purposes, this model can represent the energy drop in an acceptable way.

In this section, we explained the behavior of sensor nodes. However, we still have to model the events in sensor networks. In next two sections, we present the event arrival models and the three models used to simulate the behavior of the events.

4.2.2 Event Arrival Models

In this section, we present the models that describe how the events arrive. We consider that an event occurs when a sensor node picks up a signal with power above a predetermined threshold. An event can be static, such as a localized change in temperature or pressure in an environment monitoring application, or it can propagate, such as signals generated by a moving object in a tracking application.

In the SEDM, we simulate two types of arrival models. In the first one, the event arrival is modeled by a Poisson process with parameter λ. This process is appropriate to model events that happen randomly and independently from each other. In this model, the number of events in each second of simulation is described by the random variable:

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}. \tag{4.1}$$

In the Poisson process, the times between successive events are an independent exponential random variable with rate λ. Thus, the inter-arrival time of the events modeled by a Poisson process is given by:

$$f(x) = \lambda e^{-\lambda x}. \tag{4.2}$$

In order to simulate the inter-arrival time of the Poisson process, we used the information presented in [Ross, 1997, Ross, 1998]. Independent exponential random variables, X_i, with rate λ, are generated using the equation:

$$X_i = -\frac{1}{\lambda}\, lnU_i \tag{4.3}$$

where, U_i represents uniform random variables in the interval $(0, 1)$.

In the second model, the event arrival is modeled by a Pareto distribution. This distribution has a heavy-tailed property that implies that small occurrences are extremely common, whereas large instances represent very few occurrences. When a Pareto distribution is used to simulate the inter-arrival time of the events, they will happen in bursts. This is because most of the inter-arrival time will be small, meaning that we have lots of events. However, the occurrence of large inter-arrival time cannot be neglected, and thus it is possible to have long periods of time without any event. Then, using this distribution, we can model events that arrive in bursts. The Pareto distribution is given by the equation:

$$f(x) = \frac{ak^a}{x^{(a+1)}} \tag{4.4}$$

where a is the shape parameter. A small value of a leads to more bursts in the event arrival. The parameter k represents the initial value and, in our simulations, we consider $k = 1$.

In order to simulate the inter-arrival time of the Pareto distribution, we used the information presented in [Ross, 1997, Ross, 1998]. The inter-arrival time, X_i, with parameter a, is generated using the equation:

$$X_i = \frac{1}{U_i^{\ 1/a}} \tag{4.5}$$

where, U_i represents uniform random variables in the interval $(0, 1)$.

4.2.3 Event Generation Models

In the previous section, we presented the models that describe the event arrival. In this section, we show the models that explain the behavior of the events in the SEDM. When an event arrives, a position (X,Y) is randomly chosen for it, and its behavior is described by one of the following three models:

- Static Event Model: events are static and have a fixed size. This model represents a basic event that is static and has a fixed size. The radius of influence of each event is a random variable uniformly distributed between *event-radius-min* and *event-radius-max*, and all nodes within the circle of influence of an event will be affected by it. This means that the test “Is There any Event?” for these nodes will return true. The duration of each event is uniformly chosen between the values *event-duration-min* and *event-duration-max* seconds.

- Dynamic Event Model: events are dynamic and have a fixed size. Like in the static model, in this model, the radius of influence of each event is a random variable uniformly distributed between *event-radius-min* and *event-radius-max*. However, the duration of each event will not be determined by a random variable like in the static model. In the dynamic model, in each second of simulation, each event will cease with probability *cease-prob*. If the event does not cease, it will move with probability *mobility-prob*, and will stay still with probability (1−*mobility-prob*). If the event has to move, it will choose randomly one of the eight neighboring positions to move to. These neighbor positions are defined in relation to the position (X,Y), that represents the central point of the event, as being the possible combinations: (X±1,Y±1), (X,Y±1) and (X±1,Y).

- Static and Increasing Event Model: events are static and have an increasing size. Each event starts with a minimum size and its area of influence gets larger as the time goes by. In this model, like in the second one, in each second of simulation, each event will cease with probability *cease-prob*. If the event does not cease, it will increase its area of influence with probability *increase-prob*. When an event increases its area, its radio of influence is incremented by one unit of area.

When an event ends, the data have to be propagated to the monitoring node. As described earlier, we simulate this behavior making all nodes in the straight line between the point (X, Y) and the monitoring node go for a short time to state 3 and after to state 4. Thus, in the SEDM, the data is sent to the monitoring node only at the end of each event. In Figure 4.2, we have a scheme of the event generation.

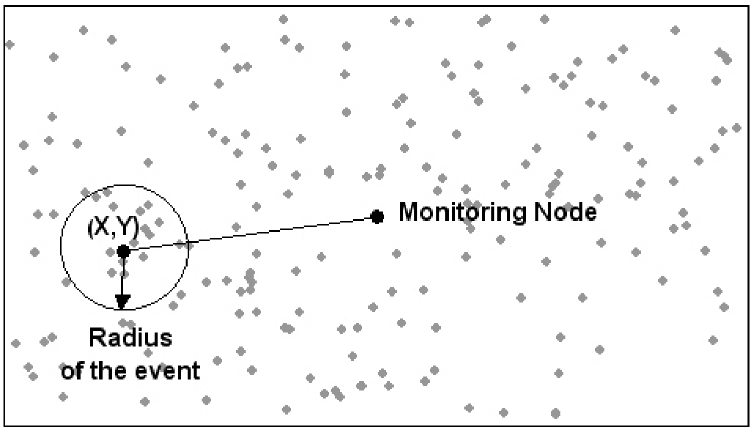


Figure 4.2: Scheme of the event generation.

4.3 Ideal Energy Dissipation Model

In order to evaluate the SEDM, we compare it with an *Ideal Energy Dissipation Model*. In this model, we consider that all nodes have a global knowledge about the network. For example, even if a node is in mode 1, it can see if there is an event for it or if there is any other node trying to communicate with it. In these situations, this node can wake up immediately, without the overhead of waking up frequently to see if there is any task to be done. Therefore, in the Ideal model, all nodes know everything about the network and they can do their best to perform the desired task. This implies that this model represents the lower bound in terms of energy consumption to the type of wireless sensor networks we are considering in this work. Furthermore, this model is able to detect all events that happen in the network.

In Figure 4.3, we have the state transition of the Ideal Energy Dissipation Model. In this model, at the beginning of simulation, all nodes go to state 1. The tests in this diagram are the same of the SEDM model. We can see that the tests "Routing?" and "Is There Any Event?" are reachable from any one of the four states, meaning that all nodes have a global knowledge about the network without the overhead of turning the sensing and the radio on frequently.

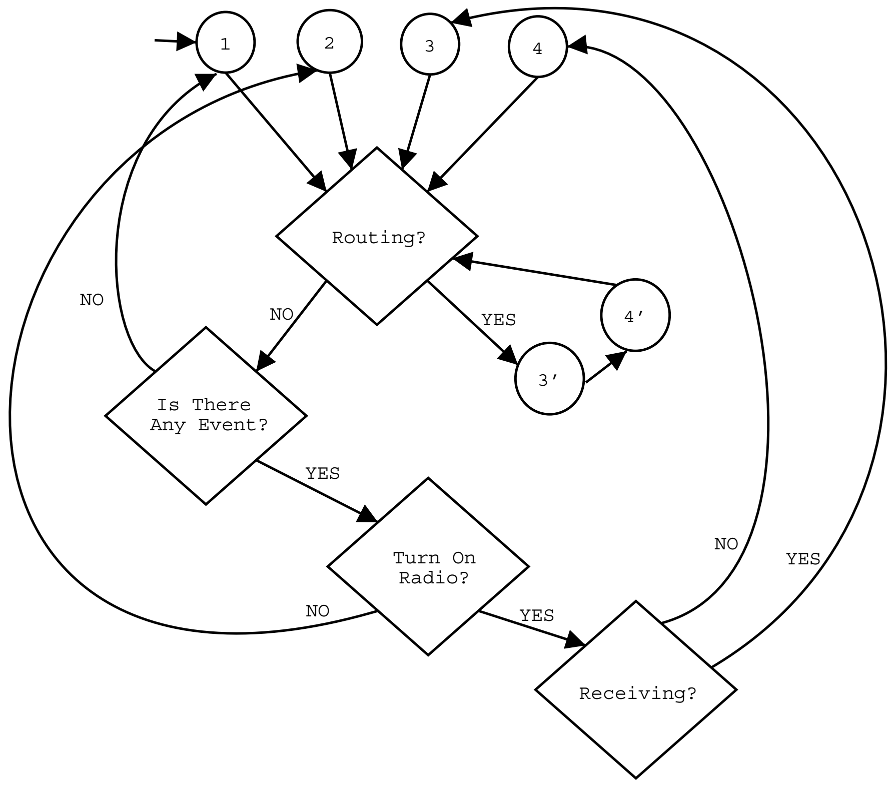


Figure 4.3: Diagram of the Ideal Energy Dissipation Model.

The main difference between the Ideal and the SEDM is the fact that all nodes, in the Ideal model, can do their best because they have a global knowledge about the network. It is important to point out that the Ideal model does not represent the behavior of a real wireless sensor network. In these networks, there is no way to tell a node that there is an event to it if this node has its sensing and its radio off. The behavior of a real sensor network is best represented by the SEDM. Our goal, when presenting the Ideal model, is to have a way of comparing the SEDM with a lower bound in terms of energy consumption. In next section, we present the comparison between these two models.

4.4 Simulation Results

In order to illustrate the behavior of the energy dissipation models, we implemented the SEDM and the Ideal model in the ns-2 simulator [ns2, 2002]. Firstly, in Section 4.4.1, to have a clear

Parameters	**Value**
Number of Nodes	100
Initial Energy	100 J
Communication Range	15 m
Sensor Field Size	50×50 m^2
turn-on-radio-prob	0.4
receiving-prob	0.7
Static Event Model	
event-duration-min	5 s
event-duration-max	50 s
event-radius-min	5 m
event-radius-max	50 m
Dynamic Event Model	
event-radius-min	5 m
event-radius-max	10 m
cease-prob	0.2
mobility-prob	0.5
Static and Increasing Event Model	
cease-prob	0.2
increase-prob	0.8

Table 4.3: Default values used in simulations

idea of the differences between these models, we show their basic operation. Secondly, in Section 4.4.2, we show the amount of residual energy at the end of simulation when using these two models. Finally, in Section 4.4.3, we analyze the performance of the SEDM in terms of non-detected events for a variety of situations.

The numerical values chosen for the base case of our simulations can be seen in Table 4.3. Unless specified otherwise, these values are used as the parameters throughout the remainder of this chapter. The scenario used consists of 100 nodes in a 50×50 m^2 field in which the average degree of each node is 23.6. Moreover, in all simulations, the monitoring node is positioned at the middle of the field at the position $(25, 25)$, all nodes are fixed and can communicate with other nodes within their communication range.

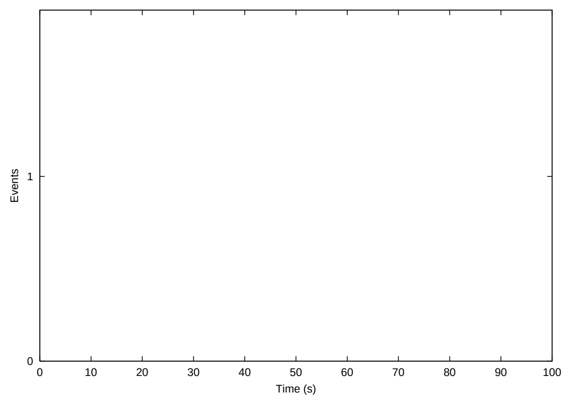
4.4.1 Basic Operation

The goal of this section is to illustrate the basic operation of the SEDM and the Ideal model. To this end, we plot, for the sake of illustration, for one specific node, the interval of time in which there is an event in its area, its states (operation mode) and its energy drop when using the SEDM and the Ideal model during a simulation of 100 seconds. The arrival of the events is modeled by a Poisson process and the following values of λ are used: 0.01, 0.04, 0.1, 0.2, and 0.3. Besides, in all simulations of this section, we used *sleep-prob* = 0.8 and *sleep-time* = 10 *s*.

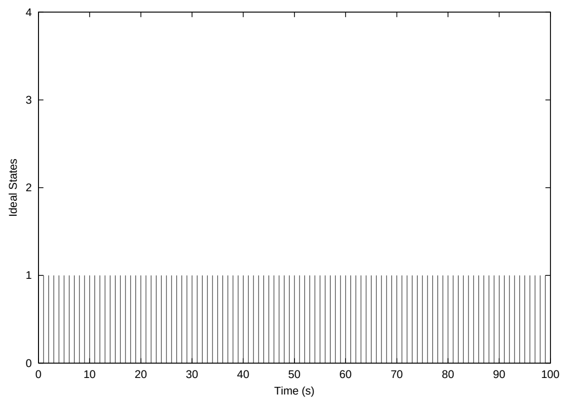
In Figure 4.4-a, we can see that, using $\lambda = 0.01$, no event happens in the area of the node analyzed. In Figure 4.4-b, we plot the states for this node using the Ideal model. We can see that, as this node knows that there is no event for it, it stays in state 1 during all simulation time. On the other hand, we can see in Figure 4.4-c, that, using the SEDM, this node has to wake up frequently to see if there is an event for it. In a real sensor network, we have this overhead, because we consider that a node does not know when an event happens for it, if it keeps its sensing and its radio off. Finally, in Figure 4.4-d, we have the energy drop using these two models. The amount of energy spent using the Ideal model is much smaller than in the SEDM.

Increasing the value of λ for 0.04, we find the results presented in Figure 4.5. In this simulation, we have events for this specific node in the interval from 63 to 99 seconds. When the event starts, in the Ideal model, the node wakes up immediately. Also, using the SEDM, at time 63, the node was in state 2, then this node could see the occurrence of this event immediately. Here, it is important to point out that the amount of energy spent by the two models when the node is working in a sensing event is probabilistically the same. The main difference between these two models, in terms of energy consumption, happens when there is no event and the SEDM has the overhead due to frequently waking up the nodes to see if there is any event for them.

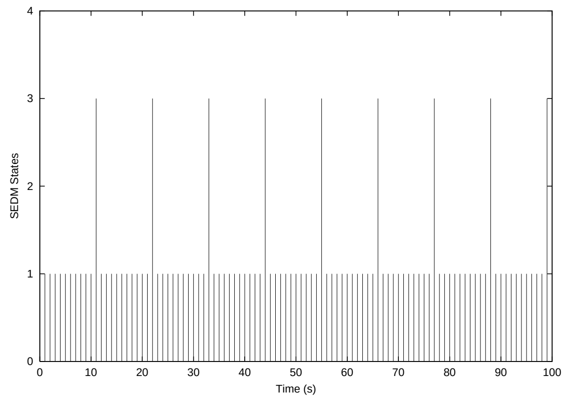
Figure 4.6 shows the results when we change the value of λ to 0.1, and more events are generated in the area of this node. In this simulation, we can see the main difference between the two models. When the event starts, at time 24 seconds, this node, using the SEDM, was in state 1, and thus it did not see the occurrence of this event. It only notices this event at time 33 seconds when it wakes up. In a situation like this, the Ideal model spends more energy than the SEDM. It is important to point out that, using the Ideal model, all nodes inside the area of the influence of the event work on that event during all its duration, and thus all events are detected by all nodes inside its area of influence. Using the SEDM, if all nodes inside the area of influence of an event keep sleeping during its duration, this event will



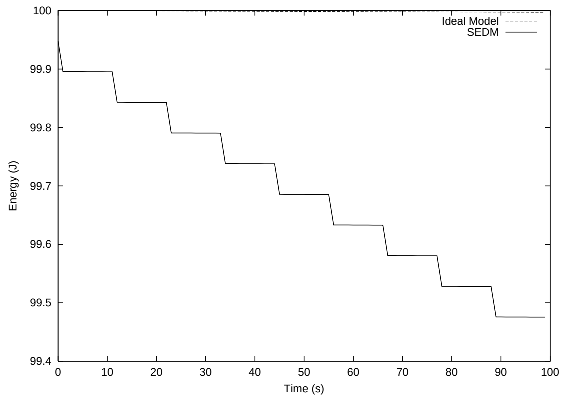
(a) Generation of events.



(b) States transition in a sensor node using the Ideal model.



(c) States transition in a sensor node using the SEDM.



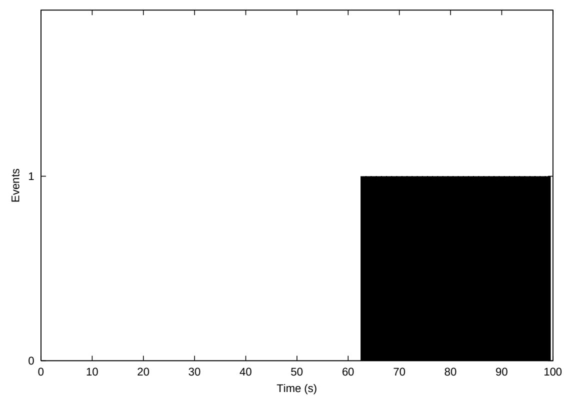
(d) Energy drop in a sensor node using the SEDM and the Ideal Model.

Figure 4.4: Comparison between the SEDM and the Ideal model for $\lambda = 0.01$.

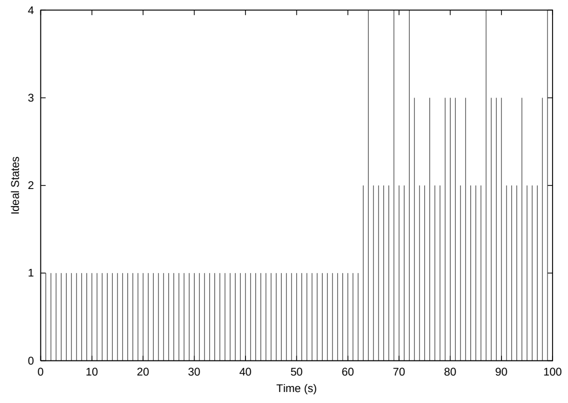
be lost. In Section 4.4.3, we analyze this aspect of the SEDM, evaluating the number of events that are lost.

Next, we executed the same simulation for $\lambda = 0.2$ and the results are depicted in Figure 4.7. Again, in this simulation, we can see that using the SEDM, this node only sees the event that starts at time 52 seconds at time 60 seconds. Because of that, the use of Ideal model makes this node spend more energy than using the SEDM.

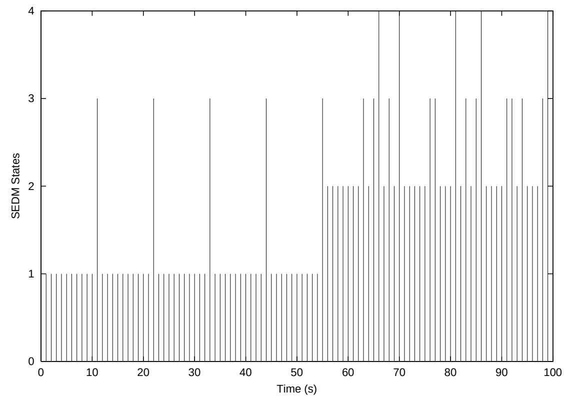
Finally, we show in Figure 4.8, the same simulation described above for $\lambda = 0.3$. Using this value of λ, the events are generated during almost all simulation time and thus the difference between the SEDM and the Ideal model happens only at the beginning of simulation where no event was generated.



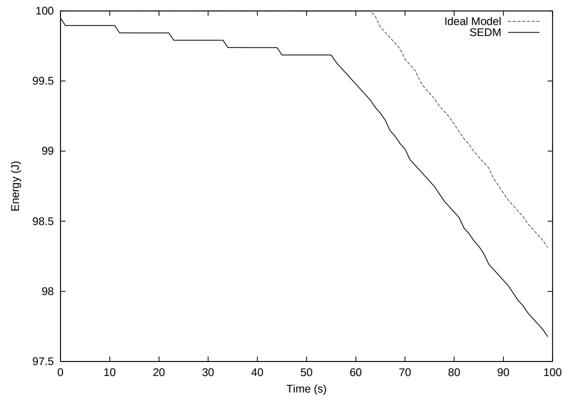
(a) Generation of events.



(b) States transition in a sensor node using the Ideal model.



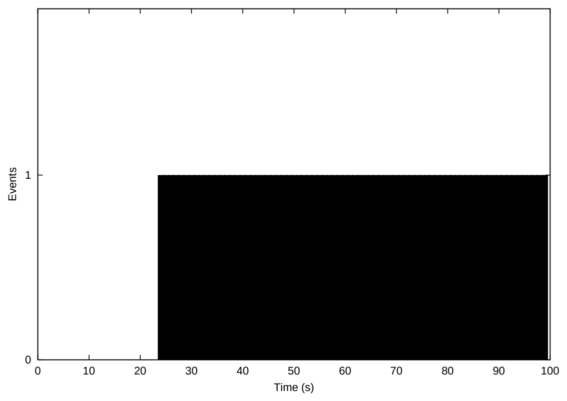
(c) States transition in a sensor node using the SEDM.



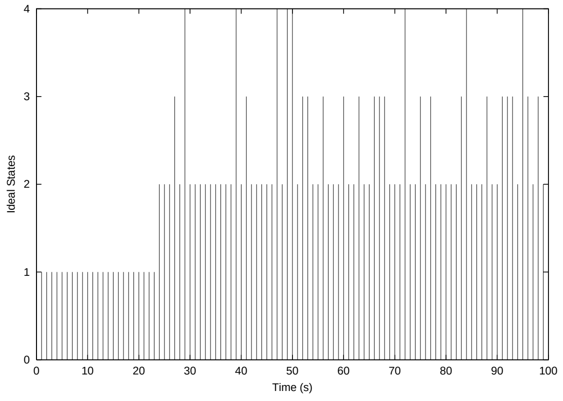
(d) Energy drop in a sensor node using the SEDM and the Ideal Model.

Figure 4.5: Comparison between the SEDM and the Ideal model for $\lambda = 0.04$.

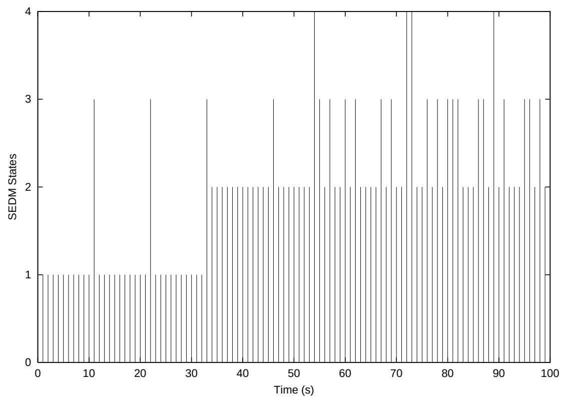
Next, we show the energy drop during a longer period of time. Figure 4.9-a shows the events that happened in the area of one specific node during a simulation of 1000 seconds. In Figure 4.9-b, we have the energy drop using both energy dissipation models. In Figures 4.10 and 4.11, we have the results of the same simulations with the values of $\lambda = 0.04$ and 0.2, respectively. We can see that as we have more events, the energy drop of both models gets more similar. This happens because, the main advantage of the Ideal model is that, as each node has a global knowledge of the network, it can keep itself sleeping whenever there is no event for it. On the other hand, in the SEDM, the nodes have to wake up once in a while to check if is there an event for it. If there are lots of events in one node area, this node will be working in the events during all time in both models.



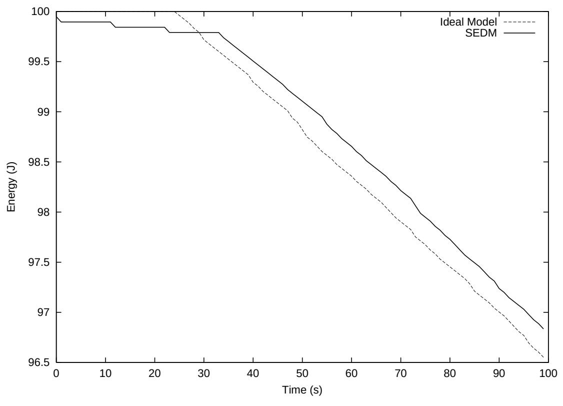
(a) Generation of events.



(b) States transition in a sensor node using the Ideal model.

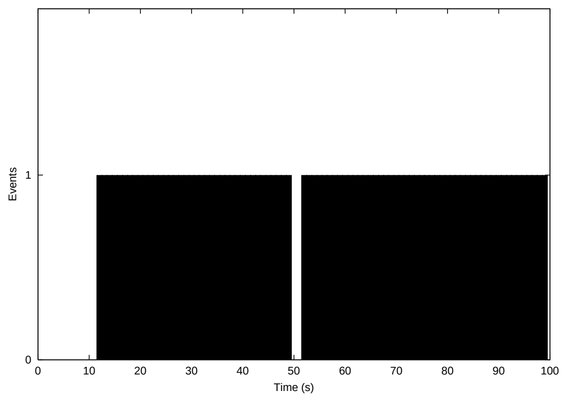


(c) States transition in a sensor node using the SEDM.

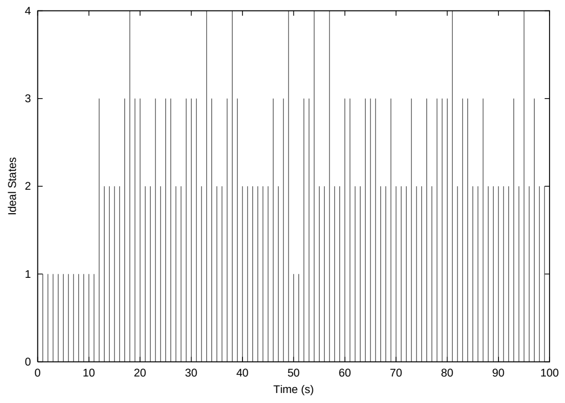


(d) Energy drop in a sensor node using the SEDM and the Ideal Model.

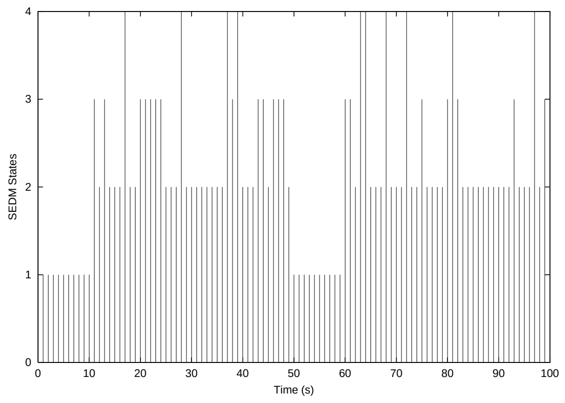
Figure 4.6: Comparison between the SEDM and the Ideal model for $\lambda = 0.1$.



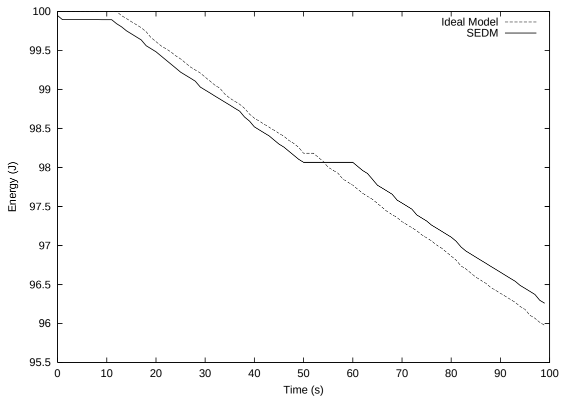
(a) Generation of events.



(b) States transition in a sensor node using the Ideal model.

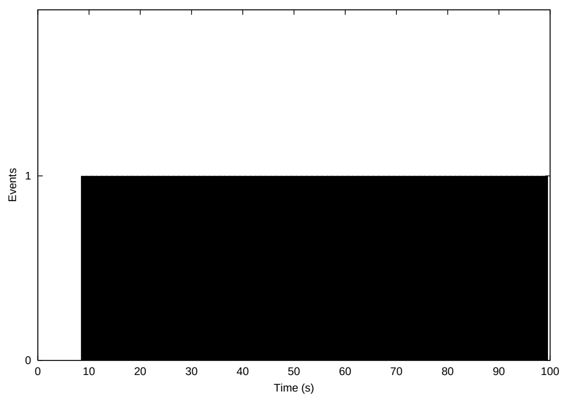


(c) States transition in a sensor node using the SEDM.

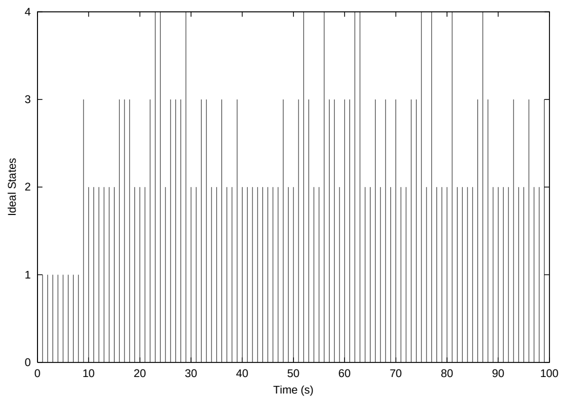


(d) Energy drop in a sensor node using the SEDM and the Ideal Model.

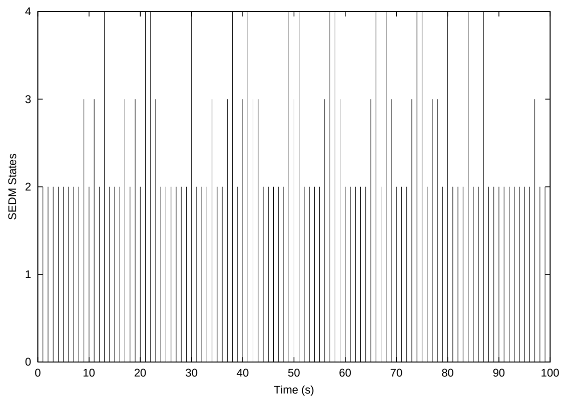
Figure 4.7: Comparison between the SEDM and the Ideal model for $\lambda = 0.2$.



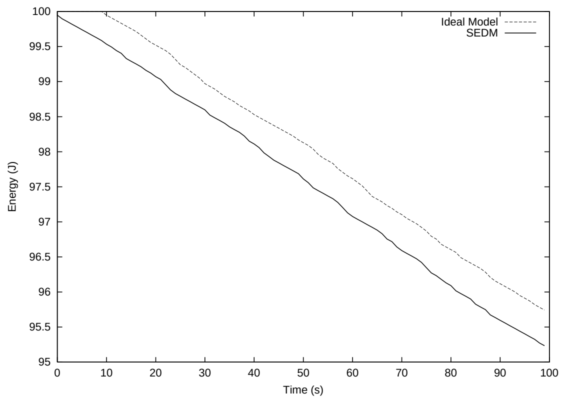
(a) Generation of events.



(b) States transition in a sensor node using the Ideal model.

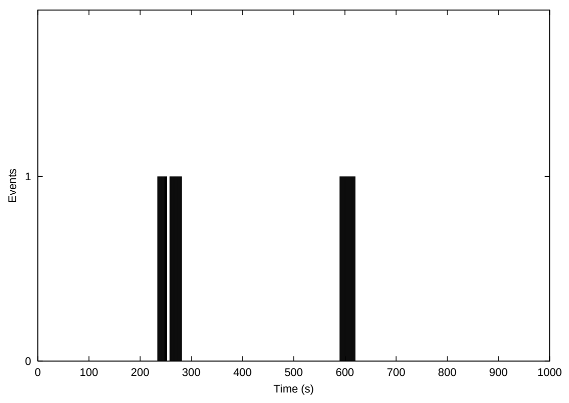


(c) States transition in a sensor node using the SEDM.

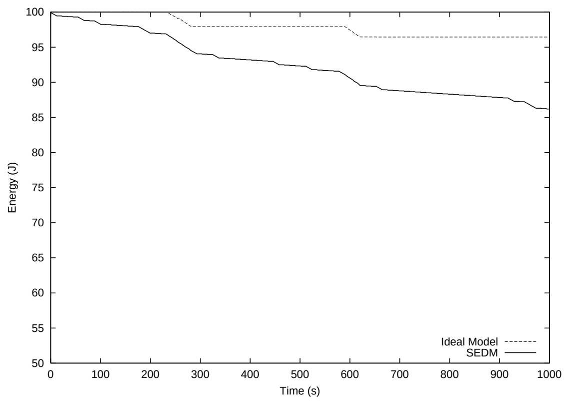


(d) Energy drop in a sensor node using the SEDM and the Ideal Model.

Figure 4.8: Comparison between the SEDM and the Ideal model for $\lambda = 0.3$.

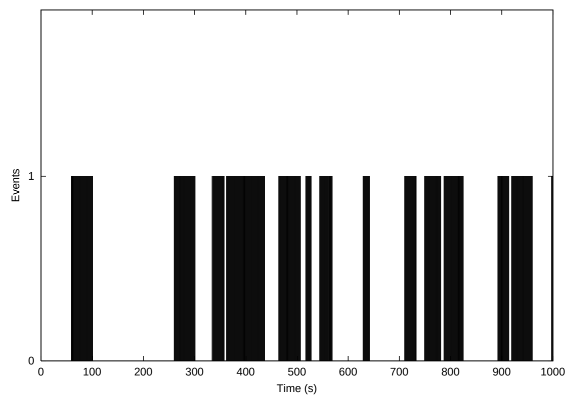


(a) Generation of events.

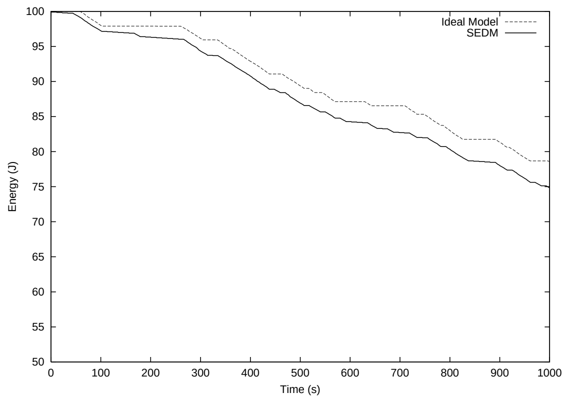


(b) Energy drop in a sensor node using the SEDM and the Ideal Model.

Figure 4.9: Comparison between the SEDM and the Ideal model for $\lambda = 0.01$ during 1000 seconds of simulation.

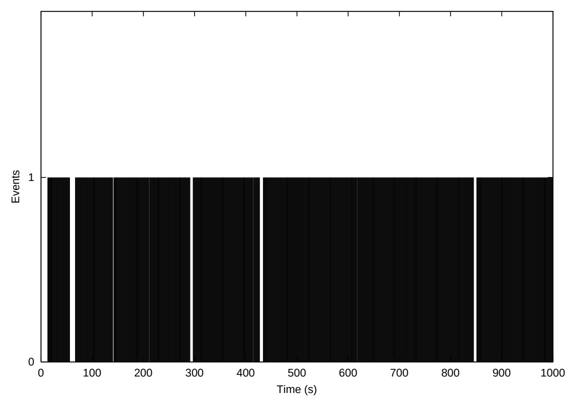


(a) Generation of events.

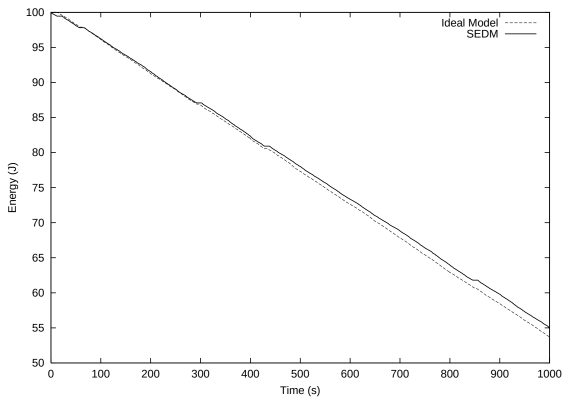


(b) Energy drop in a sensor node using the SEDM and the Ideal Model.

Figure 4.10: Comparison between the SEDM and the Ideal model for $\lambda = 0.04$ during 1000 seconds of simulation.

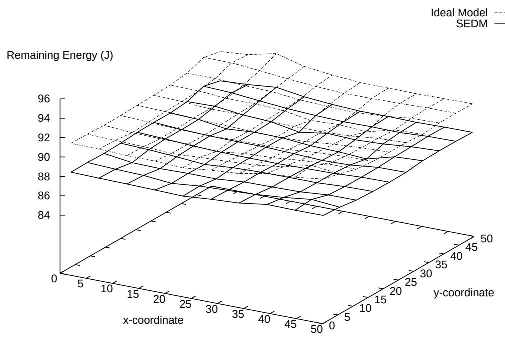


(a) Generation of events.

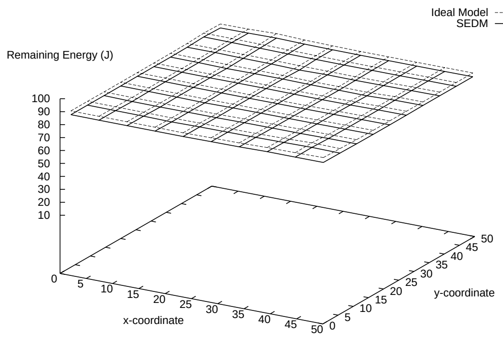


(b) Energy drop in a sensor node using the SEDM and the Ideal Model.

Figure 4.11: Comparison between the SEDM and the Ideal model for $\lambda = 0.2$ during 1000 seconds of simulation.



(a) Residual energy.



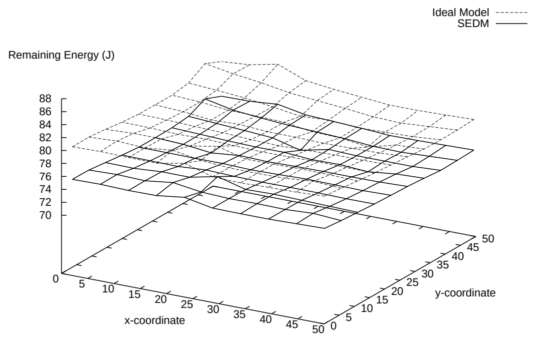
(b) Residual energy in a uniform scale.

Figure 4.12: Final energy map for 500 seconds of simulation, $\lambda = 0.04$.

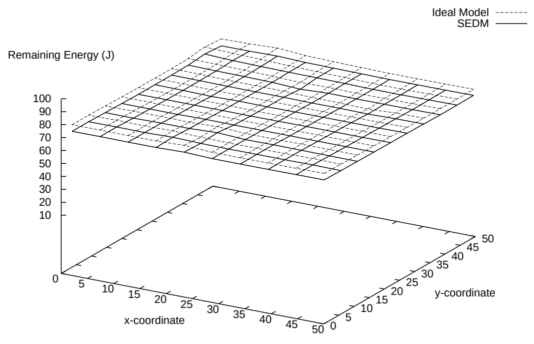
4.4.2 Final Energy Map

In this section, we analyze the final energy map at the end of simulation when using the SEDM and the Ideal model. To this end, we run both models during 500, 1000, 2000 and 3000 seconds of simulation, and plot the amount of remaining energy at each node of the network after this time has elapsed. In all simulations, we use the following values: *sleep-time* = 10 s, *sleep-prob* = 0.8, and $\lambda = 0.04$. The results are depicted in Figures 4.12 to 4.15. In all of these maps, the figure on the left side shows the zoom on the energy map, while the one on the right side shows all maps in the same scale. We can see that the SEDM spends more energy than the Ideal model in all simulations. Besides, the longer the simulation time, the greater the difference between the amount of energy spent by two models.

Our next goal is to show how these final energy maps could be represented using gray level images. We intend to create gray level images that depict the amount of available energy in each part of the network. In these images, light shaded areas represent regions with more remaining energy, and regions short of energy are represented by dark shaded areas. To this end, we generate gray level images of the final energy maps illustrated in Figures 4.12 to 4.15. To represent an energy map as a gray level image, it is necessary to associate an energy value with each point in the sensor field. In this work, we define the energy in each geographical point as the biggest energy value of all sensor nodes that can sense that point. Therefore, the sensing radius is of paramount importance in the construction of the gray level image. In

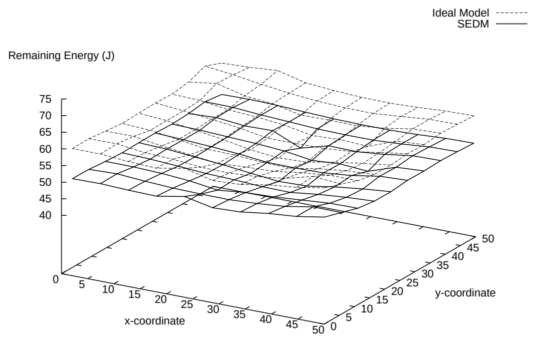


(a) Residual energy.

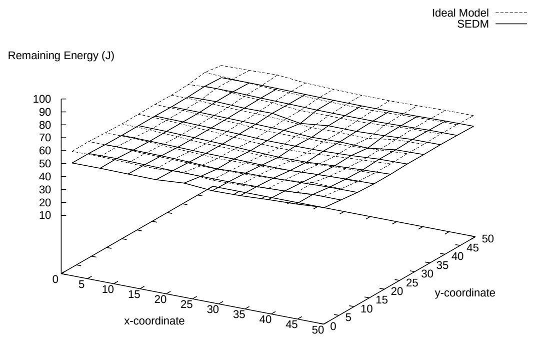


(b) Residual energy in a uniform scale.

Figure 4.13: Final energy map for 1000 seconds of simulation, $\lambda = 0.04$.



(a) Residual energy.

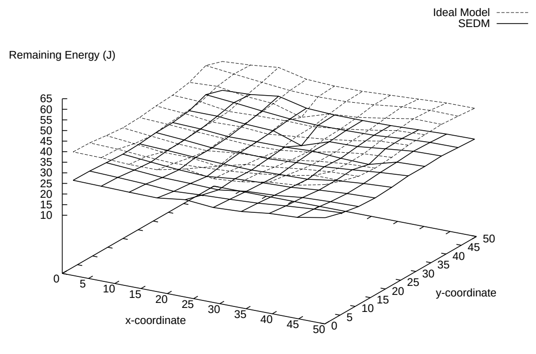


(b) Residual energy in a uniform scale.

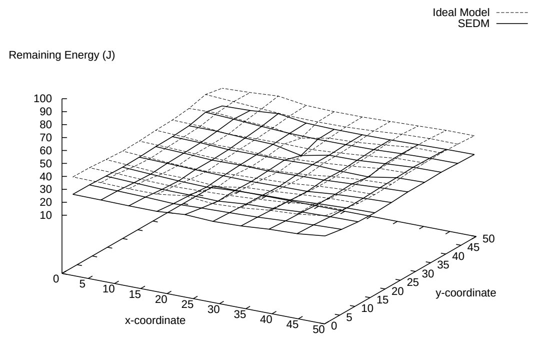
Figure 4.14: Final energy map for 2000 seconds of simulation, $\lambda = 0.04$.

order to generate the pixel value corresponding to each energy value, we use a uniform scale that maps each energy value from 0 to 100 J to a pixel value from 0 to 256. This scale is illustrated in Figure 4.16.

In the scenario used, there are 100 nodes in a sensor field of size 50×50 m^2. In Figure 4.17, we plot the geographical location of these nodes in the field. The gray level images that represent the final energy maps illustrated in Figures 4.12 to 4.15 are shown in Figure 4.18. Thus, in this figure, we have the energy maps when using the Ideal model and the SEDM after 500, 1000, 2000 and 3000 seconds of simulation. In these maps, we used a sensing radius of 5 m. Figures 4.19 and 4.20 show the same results when the sensing radius is 10 and 15 m,



(a) Residual energy.



(b) Residual energy in a uniform scale.

Figure 4.15: Final energy map for 3000 seconds of simulation, $\lambda = 0.04$.

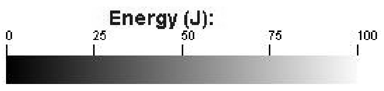


Figure 4.16: Scale used in the gray level image construction.

respectively. We can see that, the smaller the sensing radius, the more uncovered areas exist. Furthermore, it is important to recall that the monitoring node is positioned at the middle of the sensing field, and, because of that, nodes near the center of the field spend more energy than nodes at the corner. This characteristic can be seen in all gray level images.

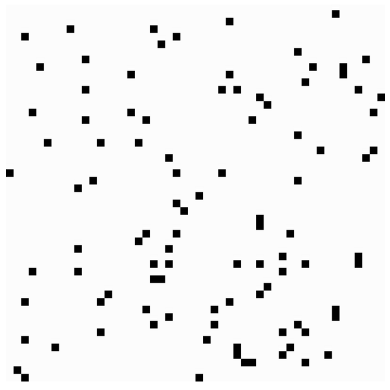
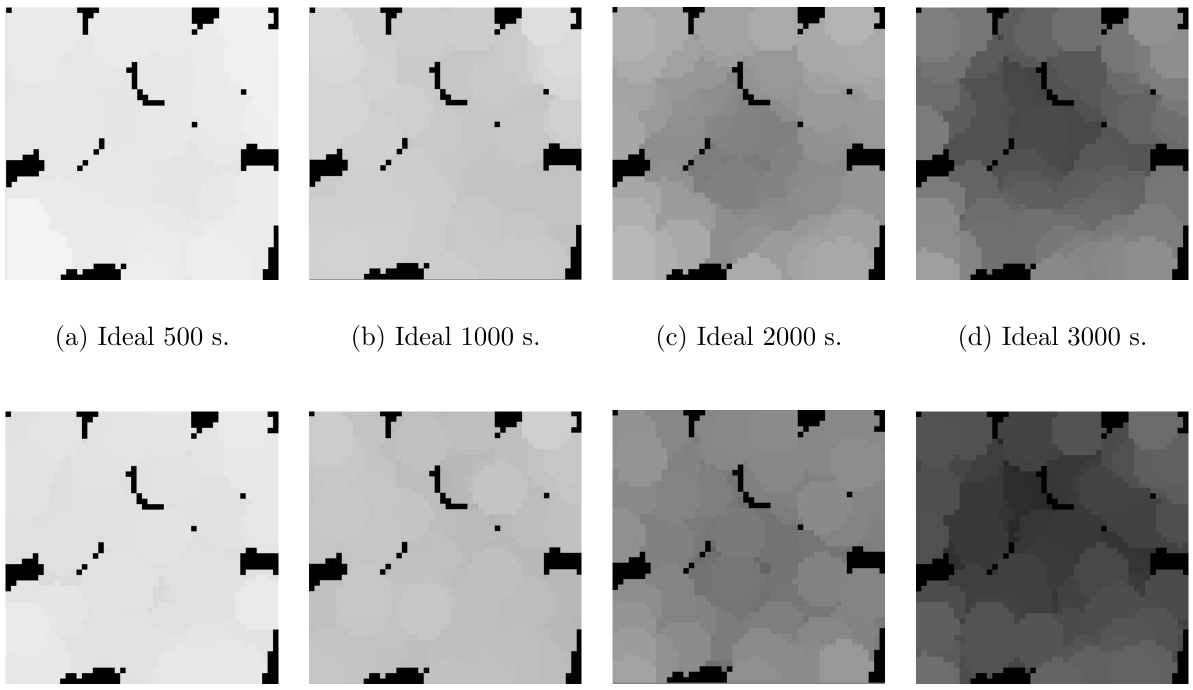


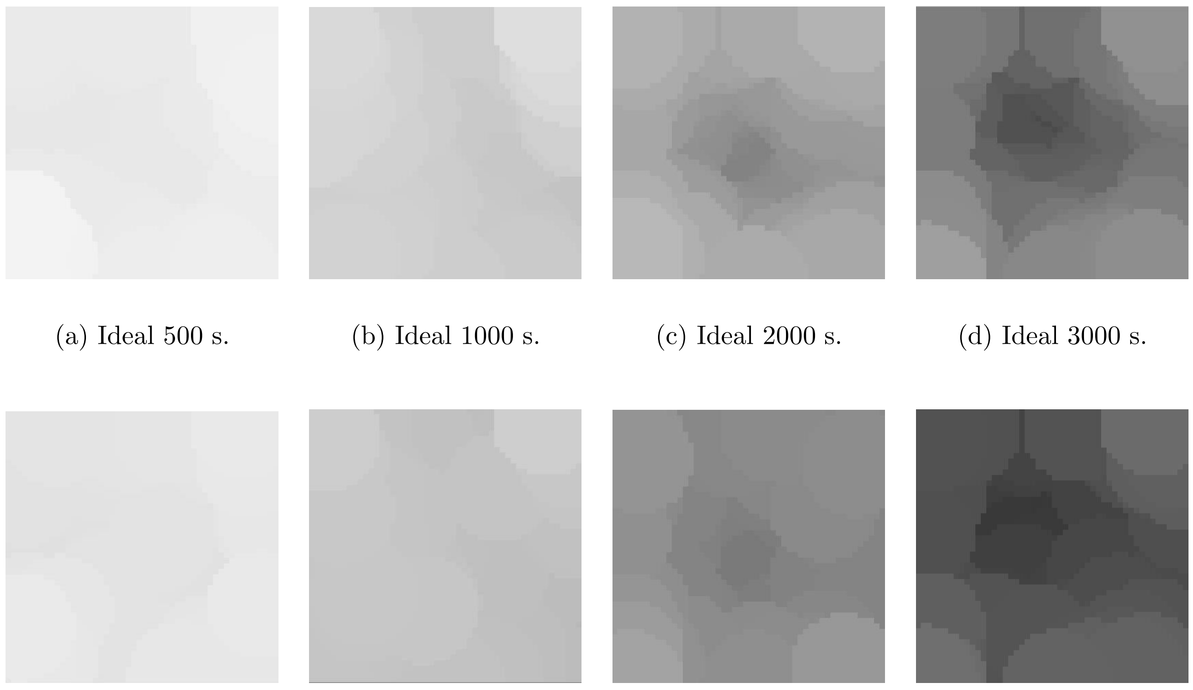
Figure 4.17: Geographical location of nodes in the field of size 50×50 m^2.



(a) Ideal 500 s. (b) Ideal 1000 s. (c) Ideal 2000 s. (d) Ideal 3000 s.

(e) SEDM 500 s. (f) SEDM 1000 s. (g) SEDM 2000 s. (h) SEDM 3000 s.

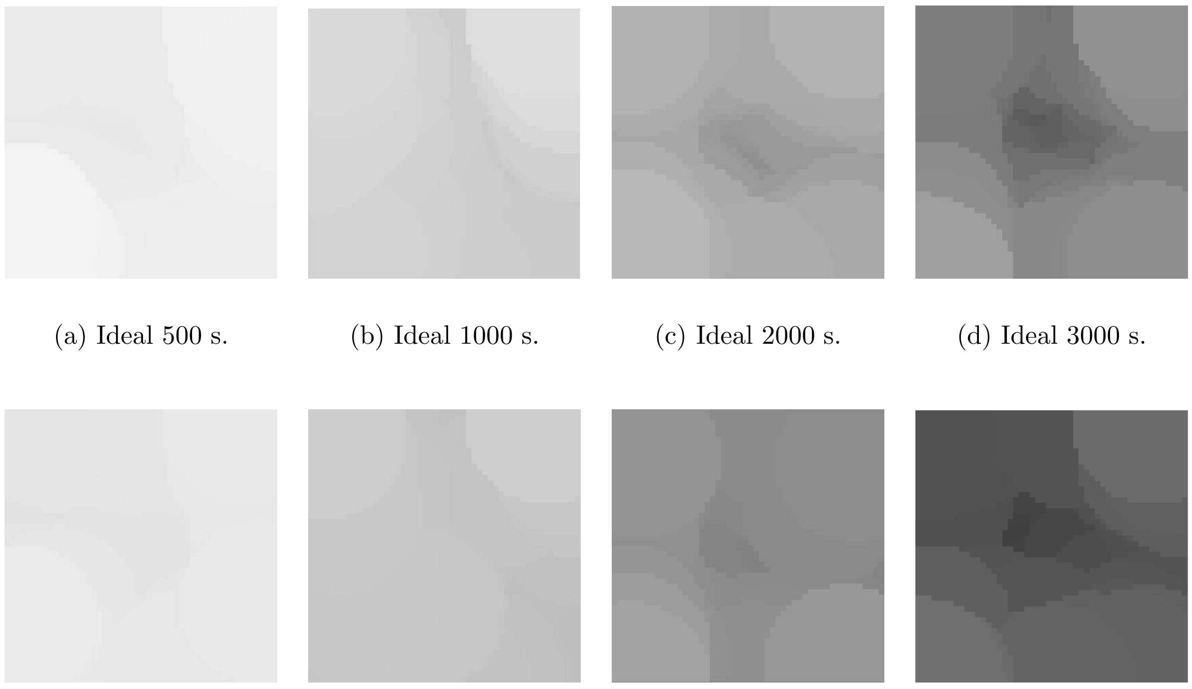
Figure 4.18: Final energy map when the sensing radius is 5 m.



(a) Ideal 500 s. (b) Ideal 1000 s. (c) Ideal 2000 s. (d) Ideal 3000 s.

(e) SEDM 500 s. (f) SEDM 1000 s. (g) SEDM 2000 s. (h) SEDM 3000 s.

Figure 4.19: Final energy map when the sensing radius is 10 m.



(a) Ideal 500 s. (b) Ideal 1000 s. (c) Ideal 2000 s. (d) Ideal 3000 s.

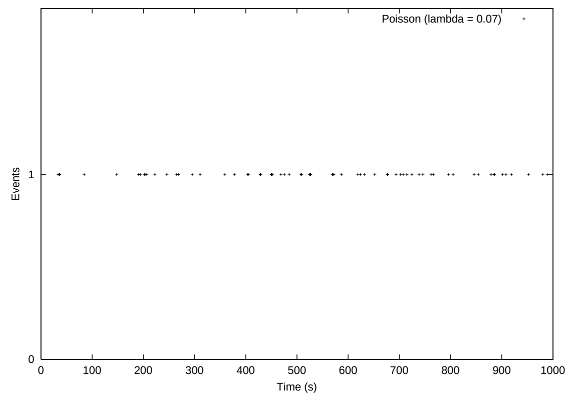
(e) SEDM 500 s. (f) SEDM 1000 s. (g) SEDM 2000 s. (h) SEDM 3000 s.

Figure 4.20: Final energy map when the sensing radius is 15 m.

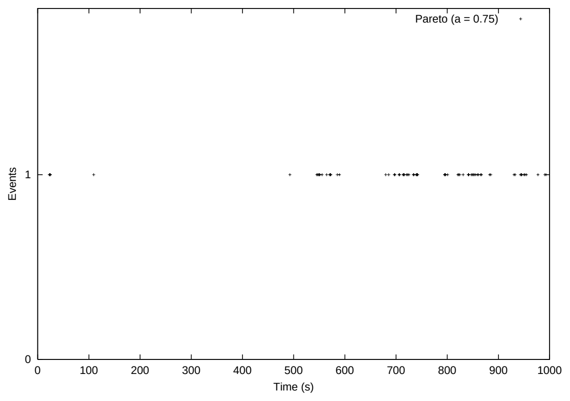
4.4.3 Undetected Events

Our next goal is to analyze the performance of the SEDM in terms of events lost. We consider that an event is lost when no node in its area of influence senses its occurrence. Therefore, if one or more nodes sense the event, we consider that this event was detected. It is important to point out that this kind of analysis is plausible only for the SEDM, since in the Ideal model no event is lost.

To study the number of non-detected events, we run the SEDM using both types of arrival models (Poisson and Pareto); for all three events generation model (Static Event Model, Dynamic Event Model and Static and Increasing Event Model). In order to make a fair comparison between both arrival models, we generated 67 events in both of them. The values that generate this amount of events are $\lambda = 0.07$ and $a = 0.75$. In Figure 4.21-a, we show the instant of time that each event happens using the Poisson process and, in Figure 4.21-b, we have the initial time of events using the Pareto distribution. In these figures, we can see the burst characteristic of the Pareto distribution in opposite to the uniformly distributed characteristic of the Poisson arrival model. The results of all simulations of this section were obtained as an average of 33 runs. In all runs, we kept constant all characteristics of the events (radius of influence, duration, etc.), except their positions. Thus, in all runs, we have the same events arriving at the same time in different positions of the sensing field. This leads to a fair comparison between the models. Also, the results have a 95% confidence level.



(a) Poisson process with $\lambda = 0.07$.



(b) Pareto distribution with $a = 0.75$.

Figure 4.21: Instant of time in which the 67 events were generated.

Next, we illustrate the number of non-detected events when using the three models of event generation, and the two models of event arrival. Firstly, we show the number of non-detected events when we have a static event with a static area of influence. Next, we show the number of non-detected events when we have a moving event with a fixed area of influence. Finally, the results for a static events with an increasing area of influence are shown.

Static Event Model

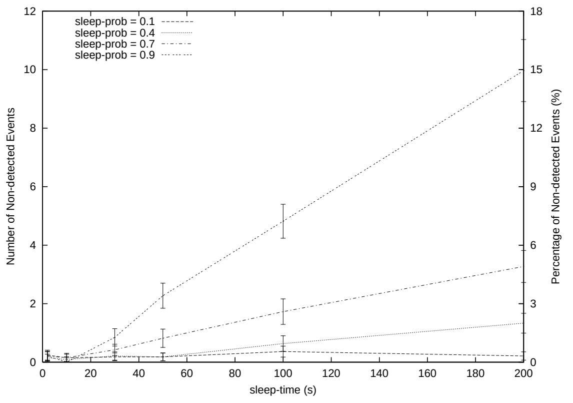
In this section, we changed the values of *sleep-prob* and *sleep-time* in order to examine the behavior of the SEDM in terms events lost. In Figure 4.22-a, we show the number of non-detected events when using the Poisson arrival model. The left axis of these figures show the number of non-detected events whereas, in the right one, this value is showed as a percentage value in relation to the total amount of generated events. In Figure 4.22-c, we have the total amount of energy spent by all nodes in the network for the simulation showed in Figure 4.22-a. We can see that as the value of *sleep-prob* gets larger, more events are lost, and less energy is spent by the network.

In Figure 4.22-b, we have the number of non-detected events when using the Pareto distribution to model the arrival of the events, and, in Figure 4.22-d, we show the total amount of energy spent by all nodes in the network. We can observe that the total amount of energy spent by the two arrival models is basically the same.

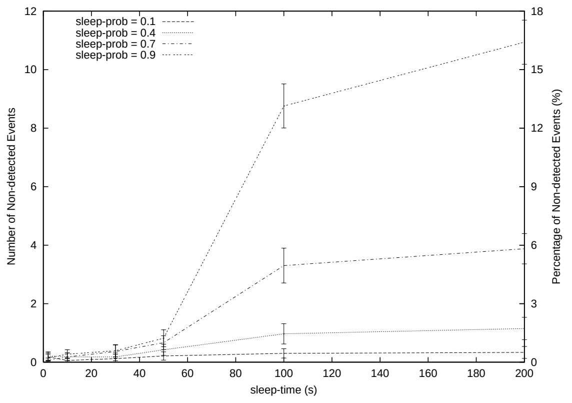
Analyzing the behavior of the SEDM, we can say that, in general, the burst arrival is more difficult to detect than the Poisson process. This is true because, when the events are distributed uniformly in time, it is higher the probability of having a node with its sensing on to detect an event. For that reason, in almost all simulations, the number of lost events using the Pareto distribution is higher than when using the Poisson process. However, in simulations showed in this chapter, this difference is small due to the small time interval analyzed.

Dynamic Event Model

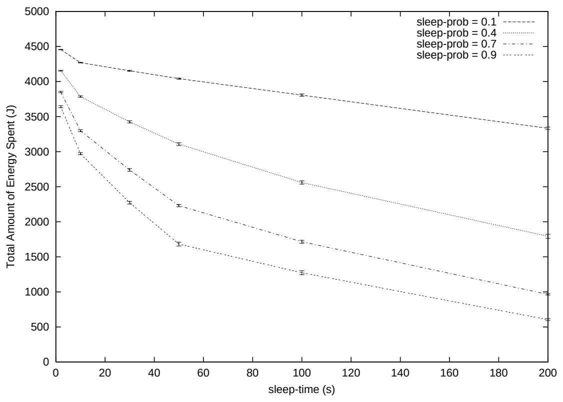
In this section, we consider that events are dynamic and have a fixed size. We can see in Figure 4.23 the number of non-detected events when using Poisson and Pareto, and the respective amount of energy spent. It is possible to observe that, in this model, there are more events lost than in the model presented in the last section. This can be explained by the default



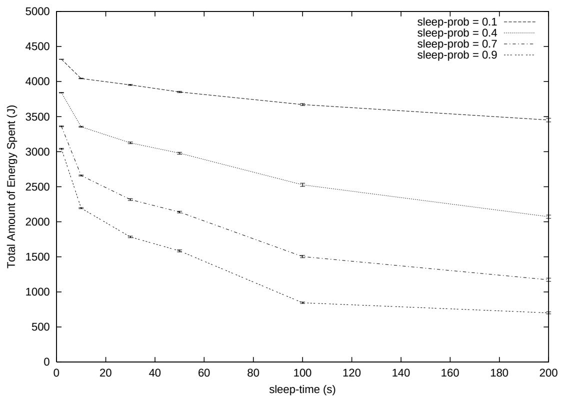
(a) Percentage of non-detected events when using the Poisson process.



(b) Percentage of non-detected events when using the Pareto distribution.



(c) Total amount of energy spent when using the Poisson process.



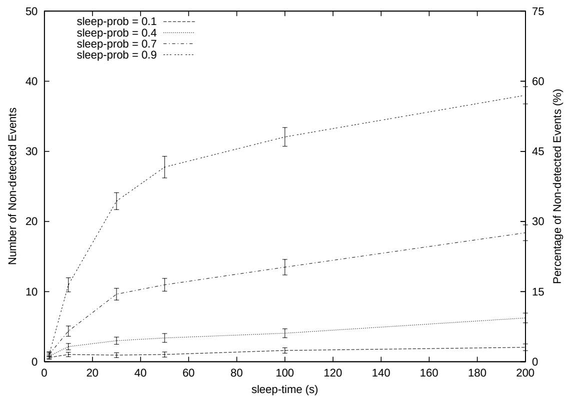
(d) Total amount of energy spent when using the Pareto distribution.

Figure 4.22: Non-detected events in the Static Events Model.

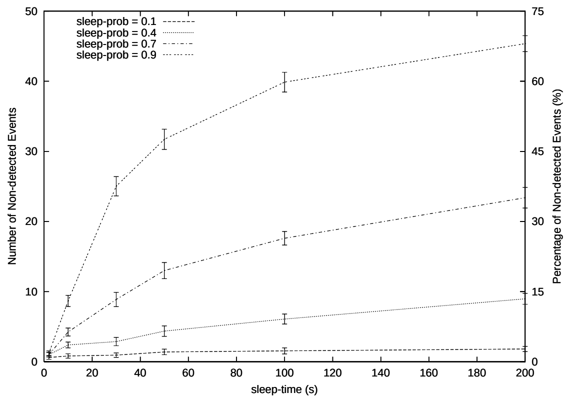
parameters of Table 4.3. The radius of influence of the static model is uniformly distributed between 2 and 50 m, while the radius of the dynamic model is uniformly distributed between 3 and 10 m. Thus, in general, the events of the static model are bigger than the events of the dynamic model. Besides, the fact that the events are static also makes them easier to be detected than the one of the dynamic model. However, it is important to point out that this result depends on the parameters of the event generation model.

Static and Increasing Event Model

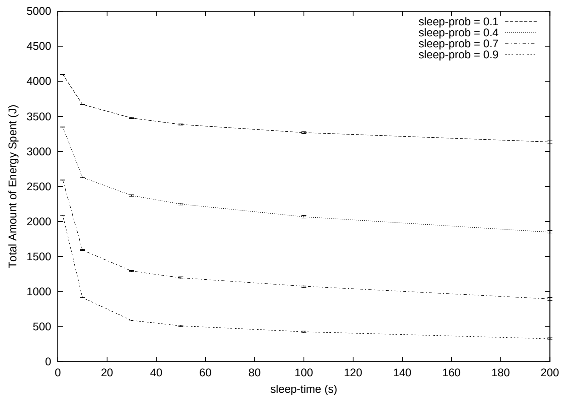
The static and increasing events model are examined in this section. Figure 4.24 shows the results for this model. We can see that these results are very similar to the one of the dynamic



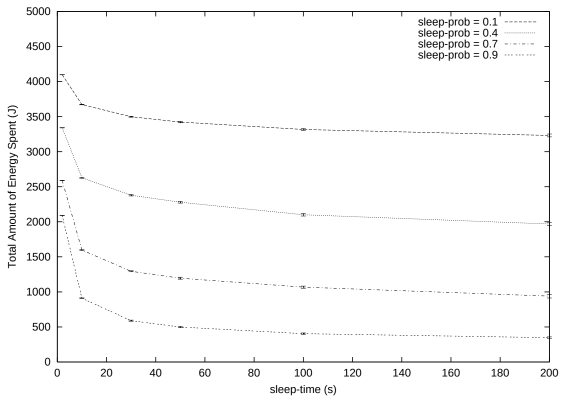
(a) Percentage of non-detected events when using the Poisson process.



(b) Percentage of non-detected events when using the Pareto distribution.



(c) Total amount of energy spent when using the Poisson process.



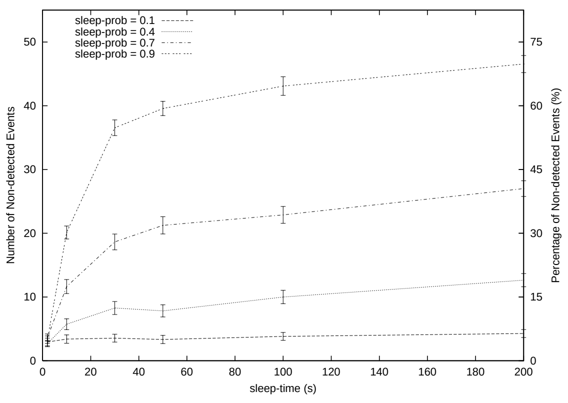
(d) Total amount of energy spent when using the Pareto distribution.

Figure 4.23: Non-detected events in the Dynamic Event Model.

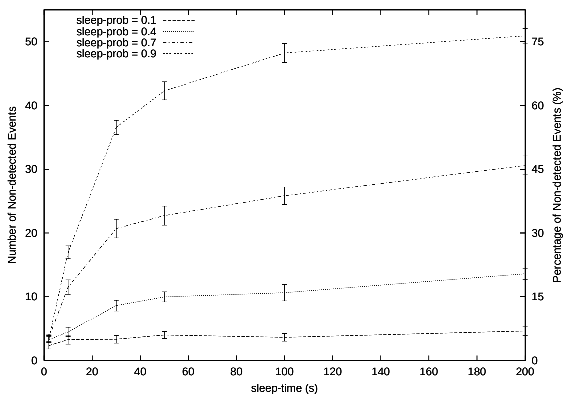
model. In addition, as in the previous simulations, in terms of events lost, the number of non-detected events using the Pareto distribution is slightly higher than when using the Poisson process.

4.5 Concluding Remarks

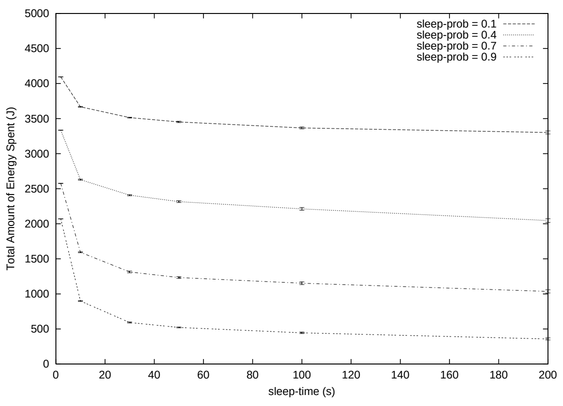
In this chapter, we proposed a State-based Energy Dissipation Model (SEDM) that models the behavior of a sensor node in terms of energy consumption. Using this model, more realistic results can be obtained when evaluating solutions to wireless sensor networks that take into account energy aspects of sensor devices.



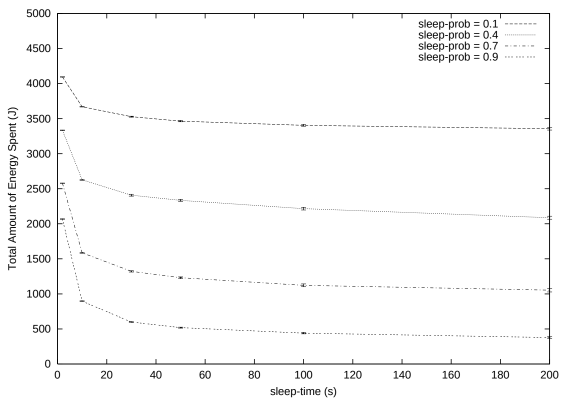
(a) Percentage of non-detected events when using the Poisson process.



(b) Percentage of non-detected events when using the Pareto distribution.



(c) Total amount of energy spent when using the Poisson process.



(d) Total amount of energy spent when using the Pareto distribution.

Figure 4.24: Non-detected events in the Static and Increasing Event Model.

Wireless sensor networks are application-specific in a way that they will be designed specially to the sensing task at hand. Consequently, any model for these networks must be adjustable to work in various applications. The SEDM presents this characteristic since it can be easily adapted in a variety of situations. Other models that represent the event behavior can easily be added in the SEDM. Furthermore, if it is necessary to analyze other sensor nodes, we just have to change the power consumption of operation modes. As an example, if we use the WINS node [Pottie and Kaiser, 2000b], the power consumption of each operation mode should be [Raghunathan et al., 2002]:

- Mode 1: 360.0 mW;
- Mode 2: 416.3 mW;

- Mode 3: 751.6 mW;
- Mode 4: 771.1 mW.

Chapter 5

Prediction-based Energy Map

"I have seen the future and it is very much like the present, only longer."

Kehlog Albran, The Profit.

"Prediction is very difficult, especially if it's about the future."

Nils Bohr, Nobel laureate in Physics.

5.1 Introduction

As described earlier, the knowledge of the amount of available energy in each part of the network is an important information for sensor networks. A naive solution to construct the energy map is to program each node to send periodically its energy level to the monitoring node. As a sensor network may have lots of nodes with limited resources, the amount of energy spent by this approach is prohibitive. For that reason, better energy-efficient techniques have to be designed to gather the information about the available energy in each part of a sensor network.

In this chapter, we discuss the possibilities of constructing the energy map using prediction-based approaches. Basically, each node sends to the monitoring node the parameters of the model that describes its energy drop and the monitoring node uses this information to update locally the information about the available energy in each node. The motivation that guided us to this work is that if a node is able to predict the amount of energy it will spend, it can send this information to the monitoring node and no more energy information will be sent during the period that the model describes satisfactorily the energy dissipation. Thus, if a node can efficiently predict the amount of energy it will dissipate in the future time, we can save energy in the process of constructing the energy map of a sensor network.

In order to predict the dissipated energy, we have studied two models. The first one is a probabilistic model based on a Markov chain, that is described in details in Section 5.2. Section 5.3 describes a statistical model in which the energy level is represented by a time series and the ARIMA (Autoregressive Integrated Moving Average) model [Box and Jenkins, 1976] is used to make the predictions. In Section 5.4, the two models to construct the energy map are evaluated and compared with the naive approach. In Section 5.5, we analyze the possibilities of using sampling techniques to construct the energy map. Finally, Section 5.6 discusses the role of the energy map in the protocol design of wireless sensor networks and, in Section 5.7, we make the final considerations about the use of prediction-based techniques to construct the energy map.

5.2 Probabilistic Model

In this section, we claim that each sensor node can be modeled by a Markov chain[1]. In this case, the operation modes of the node are represented by the states of a Markov chain and the random variables represent the probability of being in each state in a certain time. Then, if a sensor node has M operation modes, it is modeled by a Markov chain with M states.

Using this model, in each node, we have a sequence of random variables, $X_0, X_1, X_2, ...$, that represents the states of this node during the time. It is important to point out that because we have a sequence of random variables for each node, different nodes can be in different modes at the same time. The set of possible values of these random variables is exactly the possible operation modes. Then, if $X_n = i$, we say that the sensor node is in operation mode i at time-step[2] n. At each time the node is in state i, there is some fixed probability, P_{ij}, that it will next be in state j. This probability can be represented by [Ross, 1998]:

[1]A Markov chain is a stochastic process in which the random variable depends on its preceding and is conditionally independent of all the other preceding random variables [Cover and Thomas, 1991].

[2]A time-step is a small amount of time. We suppose that all state transitions occur at the beginning of a time-step.

$$P_{ij} = P\{X_{m+1} = j | X_m = i\} \tag{5.1}$$

P_{ij} represents the probability that a node in operation mode i will enter in mode j at the next transition. We can also define the two-step transition probability, $P_{ij}^{(2)}$, that a node presently in state i will be in state j after two additional transitions. That is,

$$P_{ij}^{(2)} = P\{X_{m+2} = j | X_m = i\} \tag{5.2}$$

The $P_{ij}^{(2)}$ can be computed from the P_{ij} as follows:

$$P_{ij}^{(2)} = \sum_{k=1}^{M} P_{ik} P_{kj} \tag{5.3}$$

The n-stage transition probabilities, denoted as $P_{ij}^{(n)}$, are named *The Chapman-Kolmogorov equations* and are defined as [Ross, 1998]:

$$P_{ij}^{(n)} = \sum_{k=1}^{M} P_{ik}^{(r)} P_{kj}^{(n-r)} \tag{5.4}$$

for any value of $0 < r < n$.

Another way of representing the probabilities of a Markov chain is by a matrix P with M rows and M columns named *transition probability matrix*, where the element of row i column j represents the probability $P_{i,j}$.

$$\mathrm{P} = \begin{Vmatrix} P_{11} & P_{12} & ... & P_{1M} \\ P_{21} & P_{22} & ... & P_{2M} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ P_{M1} & P_{M2} & ... & P_{MM} \end{Vmatrix}$$

The transition probability matrix represents the behavior of the sensor node. With the knowledge of the transition probability matrix and the value of X_0 (initial state of each node), it is possible to compute some probabilities of interest, and to estimate some information of the network. Below, we have some example of information whose answers could be estimated by the model proposed in this section.

1. For how many time-steps will a node be in a state s in the next T time-steps? Suppose that now the node is in state i ($X_0 = i$).

   As $P_{is}^{(t)}$ represents the probability of a node currently in state i will be in state s after t transitions. Then, for any state s, the number of time-steps a node will stay in the state s can be calculated by:

$$\sum_{t=1}^{T} P_{is}^{(t)} \tag{5.5}$$

2. How many state transitions will occur in the next T times?

   If i is the current state, we will have in the first time-step an average of $(1 - P_{ii})$ state transitions and an average of $\sum_{k=1}^{M} P_{ik}(1 - P_{kk})$ in the second time-step. If we extend this equation for any T time-steps, we have that the average number of state transitions can be calculated by:

$$(1 - P_{ii}) + \sum_{t=1}^{T-1} \sum_{k=1}^{M} P_{ik}^{(t)}(1 - P_{kk}) \tag{5.6}$$

3. When will the next state transition occur? Suppose that a node is now in state i.

   We know that there is a probability of $(1 - P_{ii})$ of having a state transition in the first time-step. The probability of the next state transition happens in the second time-step is $P_{ii} \times (1 - P_{ii})$. Thus, the expected value of when the next state transition will occur is:

$$\sum_{t=1}^{T}((\prod_{k=1}^{t-1} Pii) \times (1 - P_{ii})) \times t \tag{5.7}$$

4. Considering that a node is currently in state i, what is the probability that this node will not enter in state s in the next T time-steps?

   As $P_{is}^{(t)}$ represents the probability that a system will enter in state s after t transitions. Then, the probability of the system will not enter in state s is:

$$\prod_{t=1}^{T}(1 - P_{is}^{(t)}) \tag{5.8}$$

In this work, we will use the transition probability matrix to estimate the amount of energy a sensor node will spend in the next T time-steps. Considering that E_s is the amount of energy dissipated by a node that remains one time-step in state s, that the expected number of time-steps the node will remain in state s during the next T time-steps can be calculated by equation (5.5), and also that the node is currently in state i, the expected amount of energy spent in the next T times, E^T, is:

$$E^T = \sum_{s=1}^{M}(\sum_{t=1}^{T} P_{is}^{(t)}) \times E_s \tag{5.9}$$

Using the value E^T, each node can calculate its energy dissipation rate (ΔE) for the next T time-steps. Each node then sends its available energy and its ΔE to the monitoring node. The monitoring node can maintain an estimation for the dissipated energy in each node by decreasing the value ΔE periodically for the amount of remaining energy of each node. The better the estimation the node can do, the fewer the number of messages necessary to obtain the energy information and, consequently, the fewer the amount of energy spent in the process of getting the energy map. In summary, the predictability will be more accurate if the transition probability matrix represents the real behavior of the sensor node.

In this work, the transition probability matrix is calculated using the past history of each node. Then, each node locally constructs its own matrix based only on its past history. In this case, P_{ij} will be the number of times the node was in state i and went to state j divided by the total number of time-steps the node was in state i. With this matrix, each node uses equation (5.9) to find its energy dissipation rate. If each node can predict efficiently its energy dissipation rate, this approach can save energy compared with the naive solution, because no more energy information packet has to be sent while the energy dissipation rate describes satisfactorily the energy drop in this node.

5.3 Statistical Model

In this section, we present the statistical model used to forecast the energy level in sensor nodes. In this model, we represent the energy drop of a sensor node as a time series. A time series is a set of observations x_t, each one being recorded at a specific time t [Brockwell and Davis, 2002]. A discrete-time series is one in which the set of times at which observations are made is a discrete set. Continuous-time series are obtained when observations are recorded continuously over some time interval. There are two main goals of time series analysis [StatSoft, 2002]: identifying the nature of the phenomenon represented by the sequence of observations, and forecasting (predicting future values of the time series variable). In this work, we are interested in using the time series analysis to forecast future values of the available energy in a sensor node. We will use the discrete-time series in such a way that each node will verify its energy level in a discrete time interval.

Figure 5.1 shows an example of a time series that represents the energy drop in a sensor node. We can observe that the time series, which represents the energy drop, has a clear decreasing trend[3] (we suppose that the battery is not recharged) and no seasonality[4]. The decreasing trend will also imply in a decreasing mean and then the time series of the energy drop will also be a nonstationary time series[5].

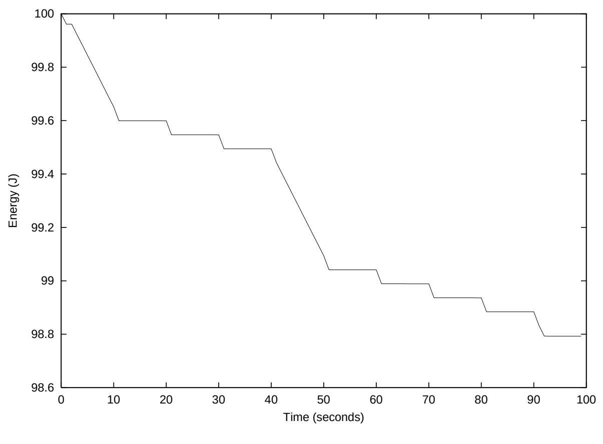


Figure 5.1: Example of a time series that represents the energy drop of a sensor node.

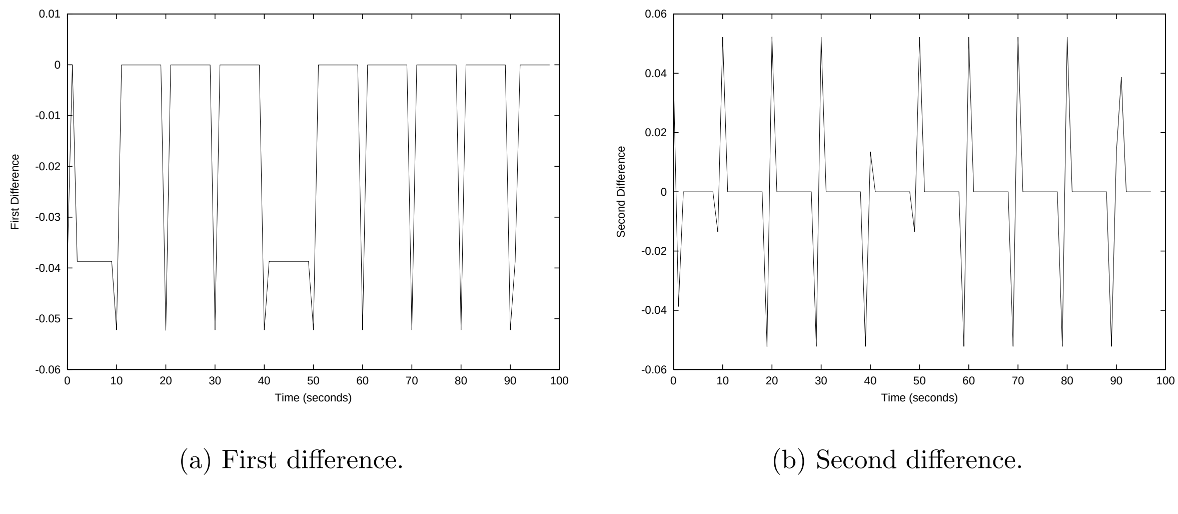
[3]Trend refers to a gradual, long-term movement in the data.

[4]Seasonality refers to periodic fluctuations that are generally related to weather factors or to human-made factors such as holidays and vacations.

[5]A stationary time series is one whose statistical properties such as mean, variance, and autocorrelation, are all constant over time.

We will use the ARIMA (Autoregressive Integrated Moving Average) model to predict future values of the time series. The ARIMA models were proposed by Box and Jenkins [Box and Jenkins, 1976] and they consist of a systematic methodology for identifying and estimating models that could incorporate both autoregressive and moving average approaches. This makes ARIMA a powerful and general class of models [Nist, 2002]. The "Integrated" part of the model is due to the differencing step necessary to make the series stationary.

The first step in developing an ARIMA model is to determine if the series is stationary. A stationarized series is relatively easy to predict: you simply predict that its statistical properties will be the same in the future as they have been in the past. Another reason for trying to stationarize a time series is to be able to obtain meaningful sample statistics such as means, variances, and correlations with other variables. Such statistics are useful as descriptors of future behavior only if the series is stationary. When the original series is not stationary, we need to difference it to achieve stationarity. Given the series S_t, the differenced series is a new series, $X_t = S_t - S_{t-1}$. The differenced data contain one less point than the original one. Although one can difference the data more than once, a small number of differences is usually sufficient to obtain a stationary time series [Nist, 2002]. The number of differencing applied in the original series is represented by the parameter d. Figures 5.2-a and 5.2-b show the first and the second difference of the time series of Figure 5.1.



(a) First difference. (b) Second difference.

Figure 5.2: Difference of the time series of Figure 5.1.

The next step in the construction of the ARIMA model is to identify the autoregressive (AR) terms. An AR model is simply a linear regression of the current value against one or more prior values of the series. The value of p is called the order of the AR model. An AR model of order p can be summarized by the equation:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + Z_t \tag{5.10}$$

where X_t is the time series, $\phi_1, \phi_2, ..., \phi_p$ are the autoregressive model parameters, and Z_t, represents normally distributed random errors. Then, in an AR model, each observation is made up of a random error component and a linear combination of prior observations. AR models can be analyzed by standard linear least squares techniques [Nist, 2002].

After defining the differencing and the autoregressive parameters, we have to identify the moving average (MA) terms. A MA model is essentially a linear regression of the current value of the series against the random shocks of one or more prior values of the series [Nist, 2002]. The random shocks at each point are assumed to come from the same distribution, typically a normal distribution, with constant location and scale. The distinction in this model is that these random shocks are propagated to future values of the time series. A MA model of order q is represented by the equation:

$$X_t = Z_t + \theta_1 Z_{t-1} + \theta_2 Z_{t-2} + ... + \theta_q Z_{t-q} \tag{5.11}$$

where X_t is the time series, $\theta_1, \theta_2, ..., \theta_q$ are the moving average model parameters and Z_t represents random shocks to the series. Then, in a MA model, each observation is made up of a random error component and a linear combination of prior random shocks. Fitting the MA estimates is more complicated that with AR models and iterative non-linear fitting procedures need to be used in place of linear least squares [Nist, 2002]. MA models also have a less obvious interpretation than AR models.

In the ARIMA, the AR and MA models are combined in a systematic methodology for identifying and estimating models that could incorporate both approaches. A time series S_t can be represented by an ARIMA(p,d,q) model if, after differencing this series d times, we find a stationary time series X_t, such that for every t:

$$X_t = \phi_1 X_{t-1} + ... + \phi_p X_{t-p} + Z_t + \theta_1 Z_{t-1} + ... + \theta_q Z_{t-q} \tag{5.12}$$

where the terms in the equation have the same meaning as given for the AR and MA models.

In order to use the ARIMA model we have to identify the values of p (order of the autoregressive model), d (number of differencing required to achieve stationarity), and q (order of the moving average model). The primary tools for estimating the parameters are the autocorrelation function (ACF) and the partial autocorrelation function (PACF). See Appendix A for details about these functions.

After identifying the parameters, we have to estimate the values of the coefficients of the autoregressive and moving average models. The calculation of these parameters is a complicated non-linear estimation. In [Nist, 2002] and [StatSoft, 2002], the authors describe some techniques to help in the process of estimating the parameters.

When the identification and estimation steps are completed, we can use the equation (5.12) to predict the value of the time series at time t. Before generating the forecasts, the series needs to be integrated so that the forecasts are expressed in values compatible with the input data.

In the implementation of the ARIMA model, we have to identify the parameters p, d, q, and to estimate the coefficients of the AR and MA models. The first step in fitting an ARIMA model is the determination of the order of differencing needed to stationarize the series (parameter d). Normally, the correct number of differencing is the lowest order of differencing that yields a time series which fluctuates around a well-defined mean value and whose autocorrelation function plot decays fairly rapidly to zero, either from above or below [Fuqua, 2002]. If the series still exhibits a long-term trend, lack of tendency to return to its mean value, or if its autocorrelations are positive out to a high number of lags, it needs a higher order of differencing. In general, the optimal order of differencing is often the one at which the standard deviation is lowest [Fuqua, 2002]. In addition, if the lag 1 autocorrelation is -0.5 or more negative, the series may be over-differenced. In our simulation, we chose the smallest value of d that produces the lowest standard deviation in such a way that the lag 1 autocorrelation is not more negative than -0.5. The number of AR and MA terms was found using the autocorrelation and partial autocorrelation functions. The lag at which the partial autocorrelation function cuts off indicates the number of AR terms, and the number of MA terms is determined by the lag at which the autocorrelation function cuts off. The values of the coefficients of the AR and MA models were calculated based on a CSS-ML (minimize conditional sum-of-squares and maximum likelihood) method implemented in [R-Project, 2002].

5.4 Simulation Results

In this section, we present simulation results of the prediction-based approaches to construct the energy map and the naive solution. In Section 5.4.1, we illustrate the basic operation of the analyzed approaches. In Section 5.4.2, we analyze the performance of the approaches when the number of events in the network is changed. Finally, in Section 5.4.3, we show the results when we change the accuracy in which the energy maps are constructed.

5.4.1 Basic Operation

In order to analyze the performance of the proposed schemes, we implemented the prediction-based energy maps in the ns-2 simulator [ns2, 2002]. The approaches implemented were the Markov, in which each node sends periodically to the monitoring node its available energy, and its predicted energy consumption rate; and the ARIMA, in which each node sends to the monitoring node its available energy and the parameters of this model. These approaches are compared with the naive one in which each node sends periodically to the monitoring node only its available energy.

In our simulations, we use the energy dissipation model, presented in Chapter 4, to describe the behavior of sensor nodes and, consequently, to simulate their energy dissipation. Therefore, each node has four operation modes: mode 1 (sensing off, radio off), mode 2 (sensing on, radio off), mode 3 (sensing on, radio receiving), and mode 4 (sensing on, radio transmitting). In addition, the events were generated using the static event model and, unless stated otherwise, their arrival is modeled by a Poisson process. All results correspond to an average of the values for 30 different runs and the confidence interval showed correspond to a 95% confidence level.

The accuracy required or the maximum error acceptable in the energy map is controlled by the parameter *threshold*. If we define a *threshold* of 3%, a node will send another energy information to the monitoring node only when the error between the energy value predicted by the monitoring node and the correct value is greater than 3%. Each node can locally determine this error by just keeping the parameters of the last prediction sent to the monitoring node.

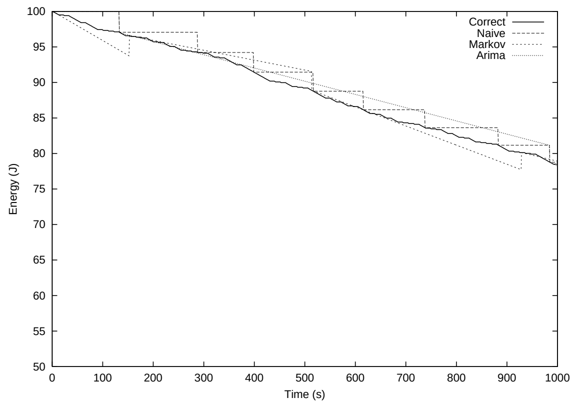
Parameters	**Value**
Number of Nodes	100
Initial Energy	100 J
Communication Range	15 m
Sensor Field Size	50×50 m^2
Threshold	3%
Static Event Model	
event-duration-min	5 sec
event-duration-max	50 sec
event-radius-min	5 m
event-radius-max	15 m

Table 5.1: Default values used in simulations.

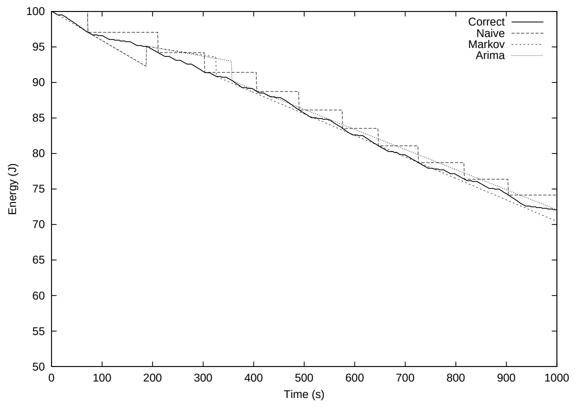
Then, adjusting the value of the *threshold*, we can control the precision at which the energy maps are constructed.

The numerical values chosen for the base case of our simulations can be seen in Table 5.1. Unless specified otherwise, these values are used as the parameters in all simulations throughout the remainder of this chapter. The scenario used consists of 100 nodes in a 50×50 m^2 field in which the average degree of each node is 23.6. Moreover, the monitoring node is positioned at the center of the field at position $(25, 25)$, all nodes are immobile, and can communicate with other nodes within their communication range. We assume that the monitoring node knows the initial energy at each sensor. Before the nodes send their first energy information packet, the monitoring node assumes that their power consumption is the average of the power consumption of all states. At the beginning of all simulations, the monitoring node assumes that all nodes have 100 J and are spending energy at the rate of 41.41 mW.

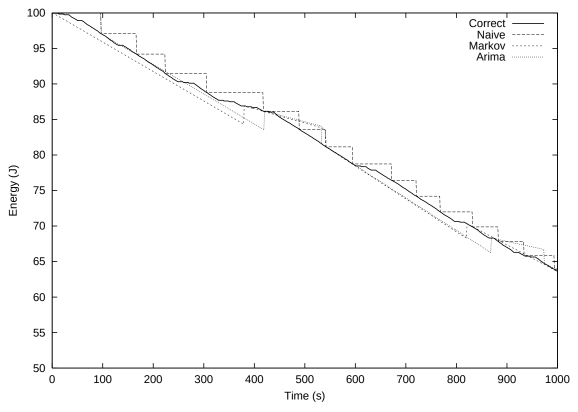
In Figure 5.3-a, we plot the correct value of the available energy in a sensor node and the values found using the naive, Markov and ARIMA models during a simulation of 1000 seconds, when the events arrival is modeled by a Poisson process, and the value of λ is 0.001. This figure shows that making the prediction using the Markov model, during the 1000 seconds of simulation, this specific node had to send three energy information packets (at times 153, 514 and 929 seconds) in order to keep its energy information in the monitoring node with an error no greater than 3% (value of the parameter *threshold*). Using the ARIMA model, it was necessary to send two energy packets (at times 133 and 985 seconds), and using the naive approach the node sent eight packets (at times 133, 288, 399, 517, 616, 738, 883 and 985 seconds). Figures 5.3-b, 5.3-c and 5.3-d show what happens in the same sensor node when we change the number of events in the network. In Figure 5.4, we plot the number of energy information packets this node had to send in simulations of Figure 5.3. We can see that the number of packets sent when using the prediction-based model is lesser than when using the naive approach.



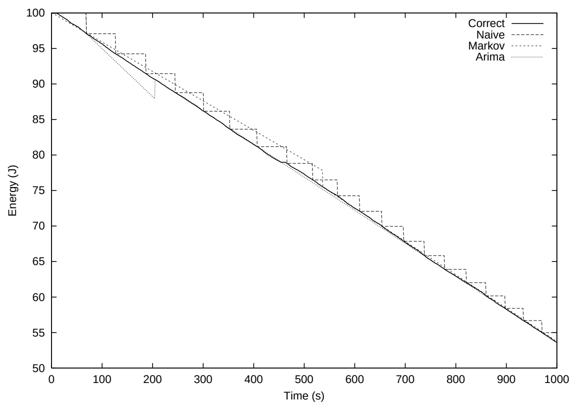
(a) $\lambda = 0.001$.



(b) $\lambda = 0.1$.



(c) $\lambda = 0.5$.



(d) $\lambda = 0.9$.

Figure 5.3: The correct available energy in a sensor node and the values found using the naive, Markov and ARIMA models for different values of λ.

5.4.2 Changing the Number of Events

The goal of this section is to analyze the performance of the energy map construction when we change the number of events in the network. Firstly, we use the Poisson process to model

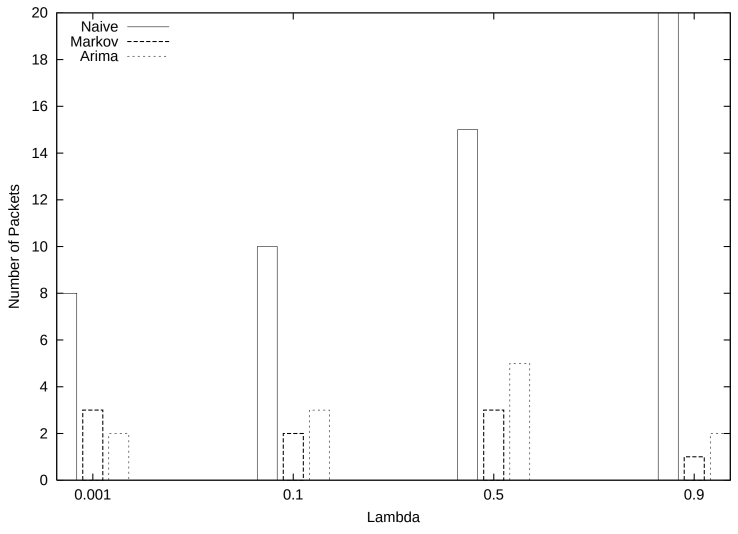
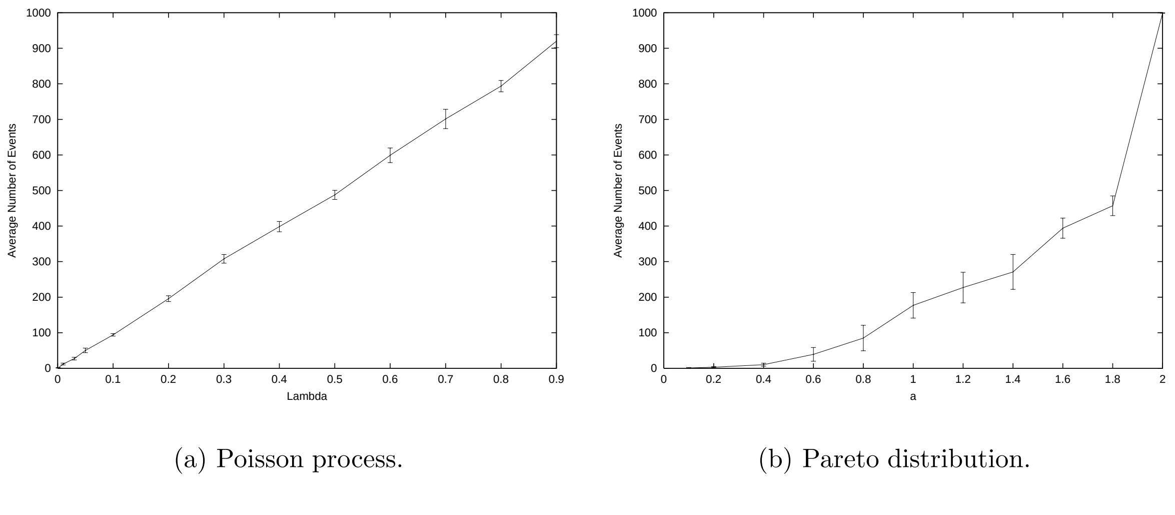


Figure 5.4: Number of energy information packets the node of simulations of Figure 5.3 had to send.

the event arrival, and the value of parameter λ is changed. Secondly, the Pareto distribution is used, and the parameter a is modified. In Figure 5.5, we show the average number of events generated when the parameters λ and a are changed.

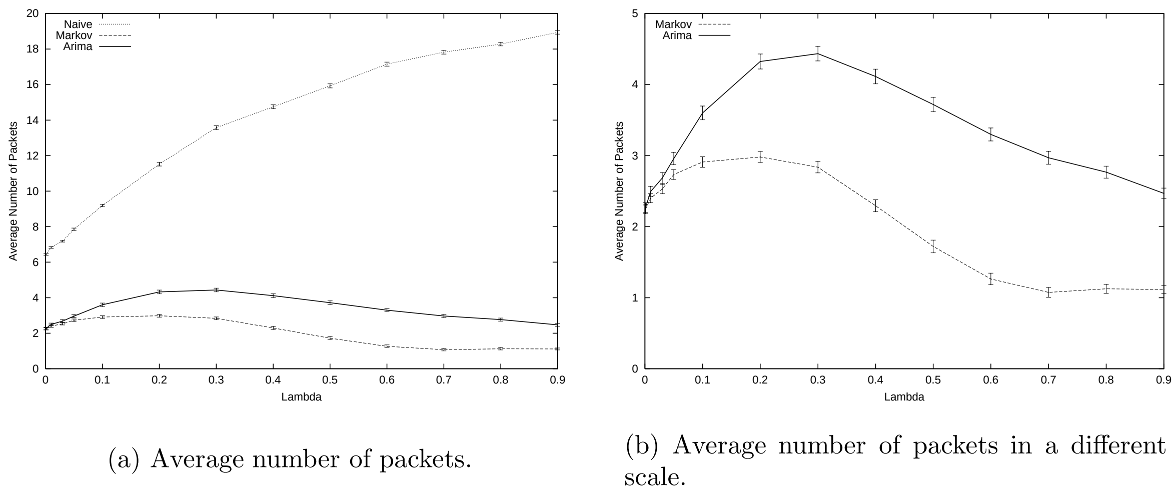


(a) Poisson process.

(b) Pareto distribution.

Figure 5.5: Average number of events when the parameters λ and a are changed.

Using the Poisson process to describe the event arrival, we executed the three approaches in the same scenario described above, during 1000 seconds of simulation. Figure 5.6-a shows the average number of energy information packets that each node had to send to the monitoring node, in order to construct an energy map with an error no greater than 3%. In Figure 5.6-b,



(a) Average number of packets.

(b) Average number of packets in a different scale.

Figure 5.6: Average number of packets for different values of λ, *threshold* $= 3\%$.

we plot the same graph of Figure 5.6-a in a different scale in order to provide a better way of comparing the Markov with the ARIMA model. We can see that for all values of λ, the naive spends more energy information packets than the prediction-based approaches. Also, for small values of λ, the power of making prediction of the Markov is very similar to the ARIMA model. However, when the value of λ increases, the Markov is better than the ARIMA. In addition, when the network becomes more active, the difference between the number of packets required by the naive and by the prediction-based approaches is bigger, meaning that the Markov and the ARIMA models are more scalable in relation to the number of events in the network than the naive solution.

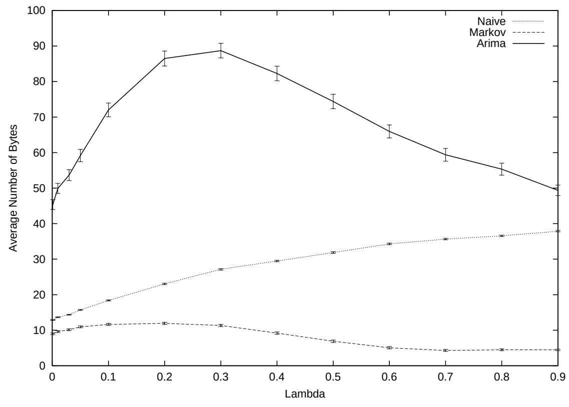
Nevertheless, the graph of Figure 5.6 is not a fair way of comparing the three approaches because when a node, running the naive algorithm, has to send an energy information packet, the size of the extra information required is only 2 bytes (its available energy). In the Markov algorithm, the overhead is of 4 bytes (its available energy and its current power consumption), and, in the ARIMA model, the overhead is about 20 bytes (with the parameters p, d, q and the coefficients of the AR and MA models). In order to perform a fair comparison between the three approaches, we have to analyze the average number of bytes that each node has to send when running the naive, Markov and ARIMA algorithms. Thus, the metric used to define energy efficiency will be the number of bytes transmitted. Figure 5.7-a compares the average number of bytes that each node had to send to the monitoring node without taking into account the overhead of the packet header. In this situation, we use piggybacking to send the energy information. We can see that the number of bytes that the ARIMA has to send is even larger than the number sent by the naive approach. This is explained by the large amount of parameters the sensor node, using the ARIMA model, has to transmit to the monitoring node.

In Figure 5.7-b, we plot the total number of bytes each node had to send considering that the packet header is of size 30 bytes. In this situation, each time a node has to send its energy information, it will send 32 bytes in the naive algorithm, 34 in the Markov and 50 bytes in the ARIMA. We can see that, in this case, the performance of the ARIMA is closer to the naive, and the Markov is still the best of the three. Figures 5.7-c and 5.7-d show what happens when the packet header is of size 60 and 90 bytes, respectively. In all situations, the Markov approach is still better than the other two for all values of packet size, and also that when the size of the packet increases, the difference between the number of bytes transmitted by the prediction-based approaches and the naive increases.

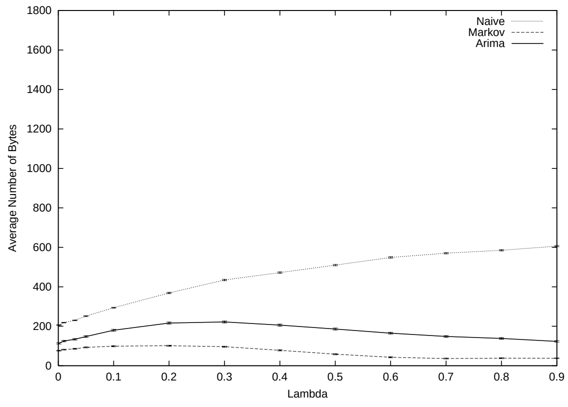
In next simulations, we use the Pareto distribution to describe the behavior of the event arrival in the network. Figure 5.8-a shows the average number of energy information packets each node had to send to the monitoring node, and Figure 5.8-b shows the same information in a different scale. In Figure 5.9, we analyze the number of bytes transmitted by each approach.

We can observe that the results of the Pareto distribution are similar to the Poisson process. Observing these graphs and the average number of events generated in each model (Figure 5.5), we can say that the event arrival model does not influence the performance of the prediction-based energy map construction. In fact, a uniformly distributed event arrival model tends to be slightly easier to be predicted because the future is more likely the present. However, this trend is faintly, and probably will be more observed in long time simulations.

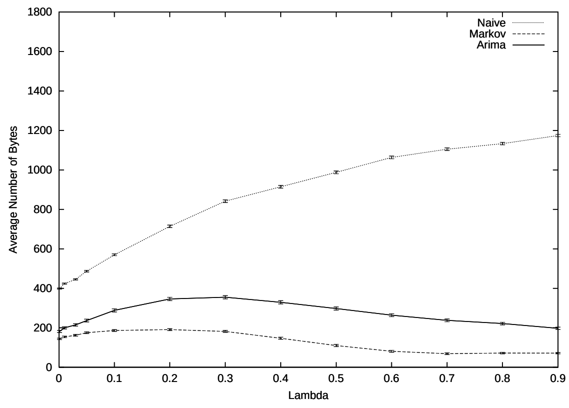
One interesting fact is that both prediction approaches have a better behavior when the number of events is very small or big. The worst case of these approaches happens for medium values of number of events. Using the Poisson model, the worst case for the Markov is $\lambda = 0.2$ and for the ARIMA is $\lambda = 0.3$. In the Pareto distribution, the worst case for both models is $a = 1.4$. This means that the fact of having more events does not make the problem of prediction more difficult. The more difficult situations for the prediction approaches happens when there is a medium number of events. On the other hand, in the naive approach, as more



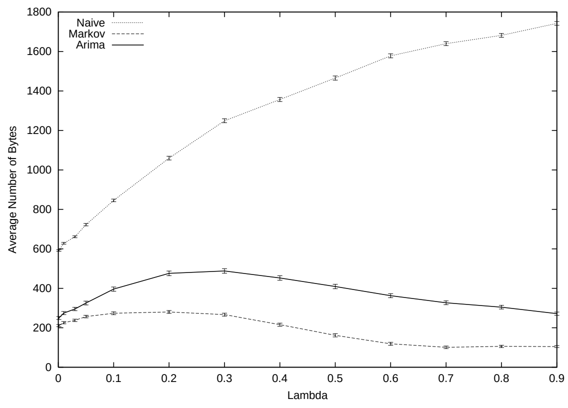
(a) Using piggybacking to send the data..



(b) Packet header of 30 bytes.



(c) Packet header of 60 bytes.

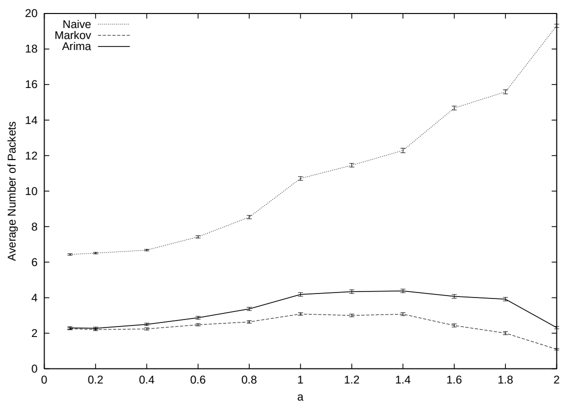


(d) Packet header of 90 bytes.

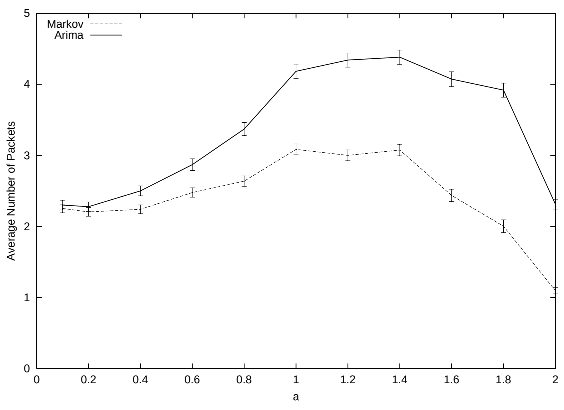
Figure 5.7: Average number of bytes for different values of λ, $\mathit{threshold} = 3\%$.

events happen, more energy will be spent by a node, and more often it will have to send energy information packets to the monitoring node. Thus, the prediction-based approaches scale well when the number of events increases or, the power of making prediction is improved when the activity of the network increases.

As described above, when using the Poisson process, the value $\lambda = 0.2$ represents the worst case for the Markov. As the Markov model provides the best performance, its worst case will be used in next simulations. In the next section, we change the value of *threshold* in a scenario in which the events are generated by a Poisson process using the value $\lambda = 0.2$.



(a) Average number of packets.



(b) Average number of packets in a different scale.

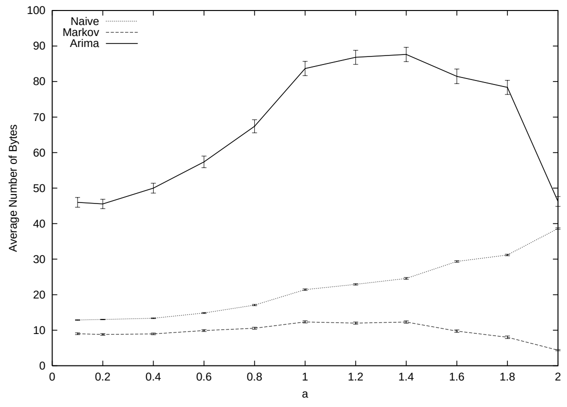
Figure 5.8: Average number of packets for different values of a, *threshold* $= 3\%$.

5.4.3 Changing the Energy Map Precision

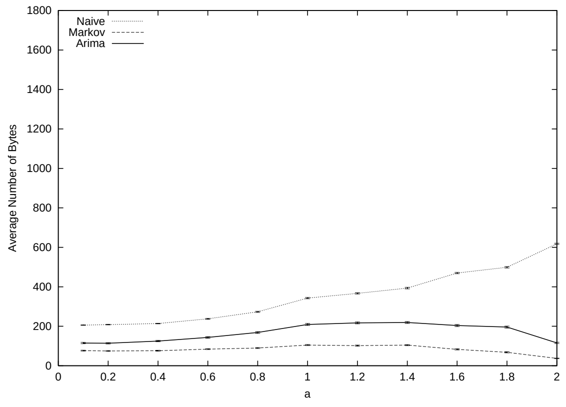
In order to analyze the performance of the approaches in situations where it is necessary an energy map with a very low error (small *threshold*), and also when we can tolerate a greater error (big *threshold*), we changed the value of the parameter *threshold*. We ran the naive, Markov and ARIMA algorithms for 100 nodes in the same scenario described above, using a Poisson process to model the event arrival. In these simulations, we analyze the worst case for the Markov model that is when the value of λ is 0.2.

Figure 5.10 shows the average number of energy information packets that each node had to send to the monitoring node, during a simulation of 1000 seconds, in order to construct an energy map with an error no greater than the corresponding *threshold*. We can see that, except for situations in which a very high precision is required (*threshold* $= 0.1\%$), the Markov approach is better than the other two for all values of *threshold*. The ARIMA is better than the Markov only when the value of the parameter *threshold* is very small. However, even in this situation, when we compare the number of bytes instead of the number of packets, the ARIMA becomes worse than the Markov. This comparison is done in Figure 5.11.

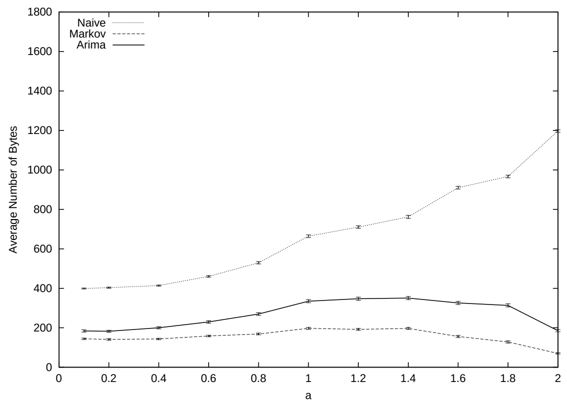
Figure 5.11-a compares the average number of bytes that each node had to send to the monitoring node when piggybacking is used to send the energy information. Figures 5.11-b, 5.11-c and 5.11-d show the average number of bytes when the packet header has 30, 60 and 90 bytes, respectively. We can see that, if we use piggybacking to send the data, the ARIMA



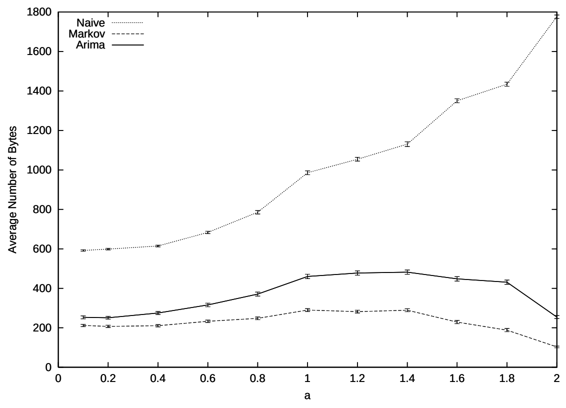
(a) Using piggybacking to send the data.



(b) Packet header of 30 bytes.



(c) Packet header of 60 bytes.

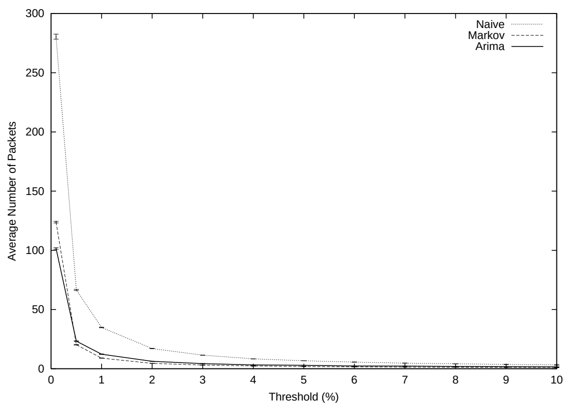


(d) Packet header of 90 bytes.

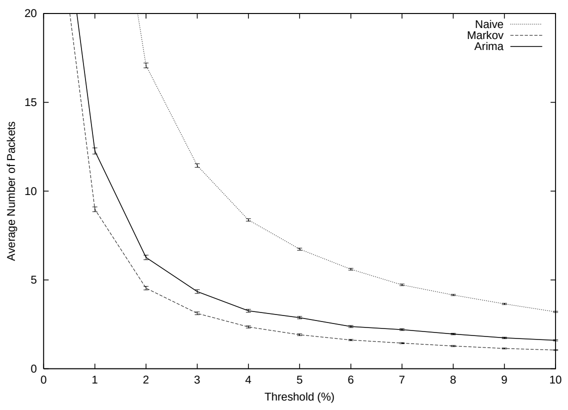
Figure 5.9: Average number of bytes for different values of a, *threshold* $= 3\%$.

is really a bad approach due to the large number of parameters sent from sensors to the monitoring node. When the packet header size is 30 bytes, the performance of the ARIMA is closer to the naive, and the Markov is still the best of the three. Also, as the normal packet size increases, the naive becomes even worse because, in these situations, the overhead of the large amount of information required by the ARIMA has a smaller impact in the total number of bytes sent. Thus, for all values of *threshold* analyzed, the Markov model was more energy-efficient than the other two models, and for sensor networks whose packet header is small, the performance of the ARIMA is very close to the naive approach.

It is important to point out that the results shown in this section represent the worst case for the Markov model. For all other values of λ, the difference, in terms of energy consumption,



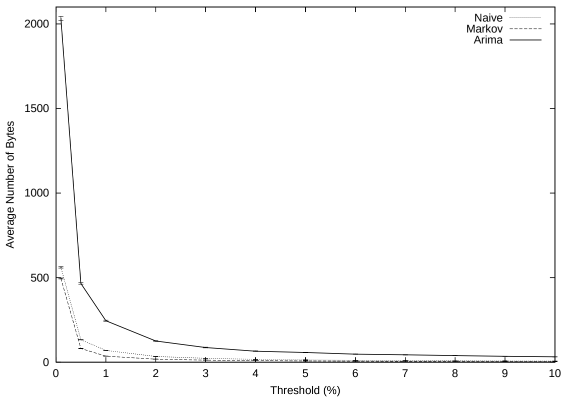
(a) Average number of packets.



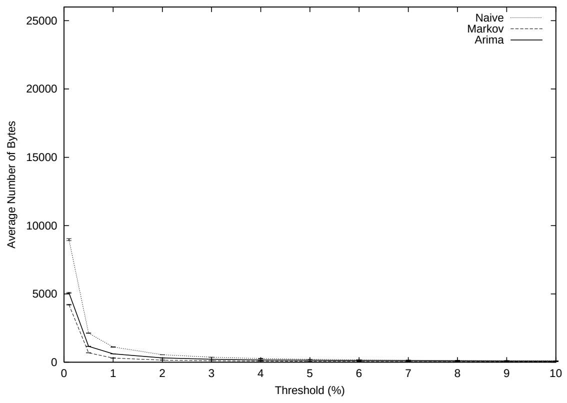
(b) Average number of packets in a different scale.

Figure 5.10: Average number of packets for different values of *threshold*, $\lambda = 0.2$.

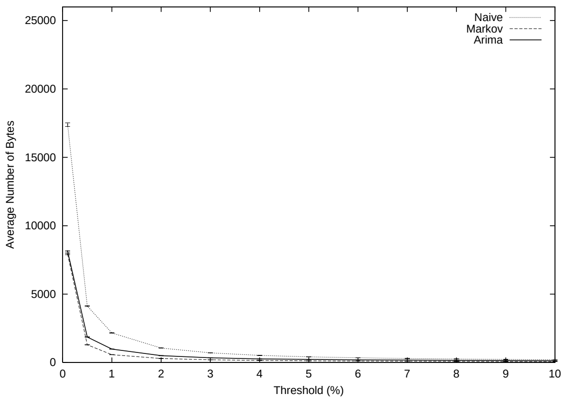
between this model and the other two is even higher.



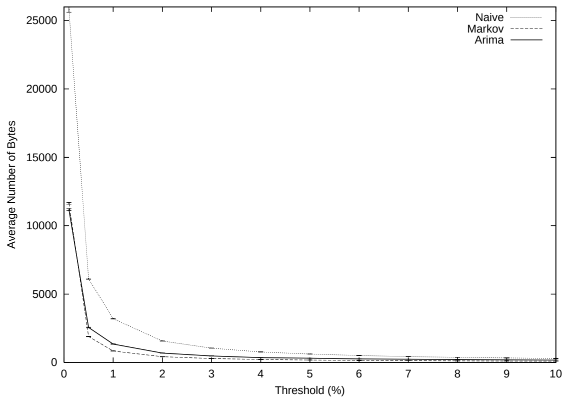
(a) Using piggybacking to send the data.



(b) Packet header of 30 bytes.



(c) Packet header of 60 bytes.



(d) Packet header of 90 bytes.

Figure 5.11: Average number of bytes for different values of *threshold*, $\lambda = 0.2$.

5.5 Energy Map Construction using Sampling Technique

The goal of this section is to analyze the possibilities of using sampling techniques to construct the energy map of a wireless sensor network. In some sensor network applications, neighboring nodes tend to spend their energy similarly. In situations like this, we can use sampling techniques in a way that it is not necessary that all nodes send their energy information to the monitoring node. The energy dissipation rate of a node that did not send its energy information packet is estimated using the information received from its neighboring nodes. Simulation results compare the performance of a sampling approach with the Markov model presented in Section 5.2. Results show that the use of sampling techniques produce more constant error curves, and also that these approaches can reduce the number of energy information packets needed to construct the energy map.

This section is organized in the following way. In Section 5.5.1, we present the sampling model used to determine when each node will send its energy information packet, and also how the energy consumption rate of a node that did not send its energy information is estimated by the monitoring node. In Section 5.5.2, we present simulation results that compare the sampling technique proposed in this section with the original Markov model.

5.5.1 Sampling Model

In this section, we propose an extension in the prediction-based energy map that uses a sampling technique. In the original Markov model, when the error between the energy in a sensor node and the correspondent value in the monitoring node is greater than the parameter *threshold*, another energy information packet is always sent to the monitoring node. We intend to use the sampling model in such a way that, when the error reaches the value of *threshold*, another energy information packet is sent with probability p. Using this assumption, we can consider the original Markov a special case of the sampling model in which the value of p is always 1.

It is important to point out that the choice of the parameter p has to be done locally without any communication between sensor nodes. The value of p can be defined statically

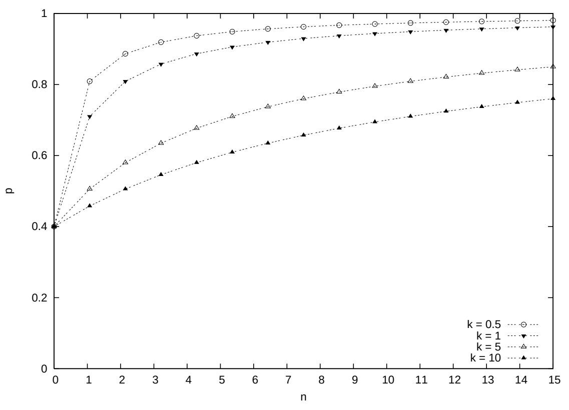


Figure 5.12: Probability p for $d = 0.4$.

or dynamically. In both cases, a constant d, that represents the sampling degree, is defined. This constant determines the initial value of probability p. In the static sampling, p is always equals to d during all simulation. On the other hand, in the dynamic one, its value increases whenever the error reaches the *threshold* and no energy packet is sent. Thus, the larger the error, the bigger the value of p and, consequently, the bigger the probability of a node to send its energy information packet. To this end, we define the value of probability p according to equation (5.13).

$$p = d + (1 - d) \times \left(1 - \frac{k}{k + n}\right) \tag{5.13}$$

where k determines the speed that probability p reaches one. For small values of k, p reaches asymptotically one faster. Furthermore, n is the number of times the error reached the *threshold*. Figure 5.12 illustrates the value of p for different values of k when $d = 0.4$.

The sampling technique described in equation (5.13) diminishes the number of packets used in the map construction, and, consequently, it increases the error in the map. In order to minimize the increase in the error curve, the monitoring node has to estimate the energy consumption rate of nodes that did not send their energy information packet. We suppose that a node and its neighbors spend energy in a similar way. When the monitoring node receives an energy packet, it uses interpolation to update the energy consumption rate of its neighboring nodes of the received packet. This update takes into account the last consumption rate sent by the node, named c_{node}, and the average energy consumption rate of its neighboring nodes that sent their energy information packet after this node sending the packet, named

$c_{neighbors}$. The trade-off between this two pieces of information is defined by equation (5.14), that determines the weight of the energy consumption rate of the neighboring nodes. In this equation, $n_{interpolations}$ represents the number of interpolations executed for the node.

$$p_{neighbors} = (1 - d) + d \times \left(1 - \frac{k}{k + n_{interpolations}}\right) \tag{5.14}$$

Therefore, when the monitoring node receives an energy information packet, it updates the consumption rate of all neighboring nodes of this packet. This new consumption rate, named $c_{estimated}$, is defined by equation (5.15).

$$c_{estimated} = c_{neighbors} \times p_{neighbors} + c_{node} \times (1 - p_{neighbors}) \tag{5.15}$$

The goal of equation (5.15) is to update the node consumption rate with a more recent information received from its neighbors. The use of the value $n_{interpolations}$ in equation (5.14) is justified because, as the node information is estimated several times in the monitoring node, the last energy packet information values loose significance. Consequently, the more recently the energy packet is, the more expressive its value in the map. The value of $p_{neighbors}$ depends also on the sampling degree. The smaller this value is, the greater the value of $p_{neighbors}$. It is important to point out that equation (5.14) is only used in the dynamic approach. In the static model, the same equation is $p_{neighbors} = (1 - d)$.

5.5.2 Simulation Results

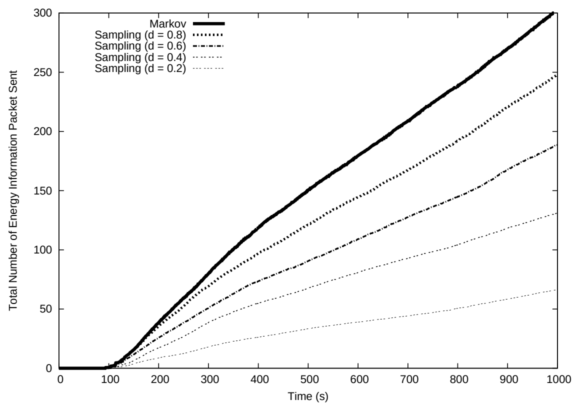
In order to compare the sampling technique with the original Markov model, we implemented the energy map construction using sampling in the ns-2 simulator [ns2, 2002]. Unless specified otherwise, the default values used in simulations of this section are the same defined in Section 5.4.1.

We analyzed the sampling technique using the static and the dynamic event generation models defined in Section 4.2.3. In both models, the event arrival is simulated using a Poisson process with $\lambda = 0.2$.

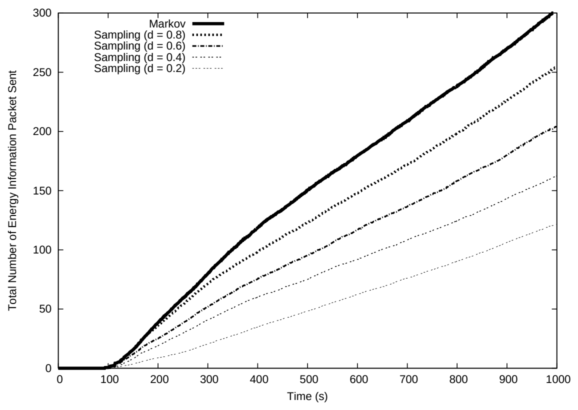
Simulation Results for the Static Event Model

In this section, we analyze the performance of the sampling technique using the static event model. The event radius is defined by a random variable uniformly distributed between 5 and 15 m, and its duration time is uniformly distributed between 5 and 50 s.

Our goal, in the first simulation, is to analyze the total number of energy information packets sent using the original Markov and the sampling technique for the following values of d: 0.2, 0.4, 0.6 and 0.8. Figure 5.13 shows these results for the static and dynamic sampling models. As it was expected, in all simulations, the total number of energy information packets sent by sampling techniques was less than the amount sent by the Markov. We can observe that, the smaller the sampling degree is, the lower the total number of energy packets sent. In the static approach, this value is probabilistically equals to the sampling degree multiplied by the total number of energy packets sent by Markov. On the other hand, in the dynamic approach, the number of packets sent is greater than this multiplication, because the probability p of sending a packet increases whenever the node reaches the value of *threshold*, and its energy information is not sent. The speed of this increase is determined by the value of k. In all simulations, we use $k = 1$. Table 5.2 shows the number of energy information packets sent at the end of simulation for both models.



(a) Static Sampling.



(b) Dynamic Sampling.

Figure 5.13: Number of energy packets sent using the static and dynamic sampling approaches.

Next, we plot the average error in percentage during a simulation of 1000 seconds. The results are shown in Figure 5.14. Analyzing these figures, we verify that the Markov has

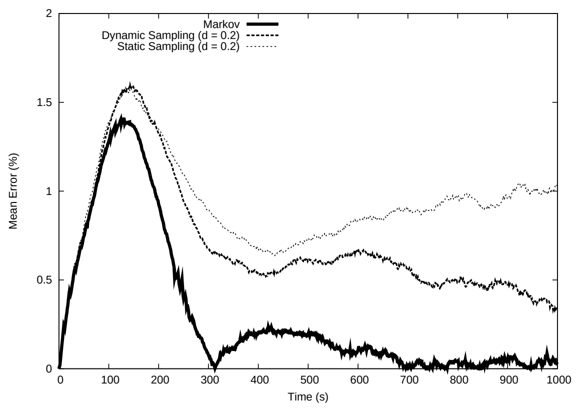
Model	**Static Approach**	**Dynamic Approach**
Sampling ($d = 0.2$)	66	122
Sampling ($d = 0.4$)	132	162
Sampling ($d = 0.6$)	189	204
Sampling ($d = 0.8$)	248	255
Markov	303	

Table 5.2: Total number of energy packets sent using the static and dynamic sampling approaches.

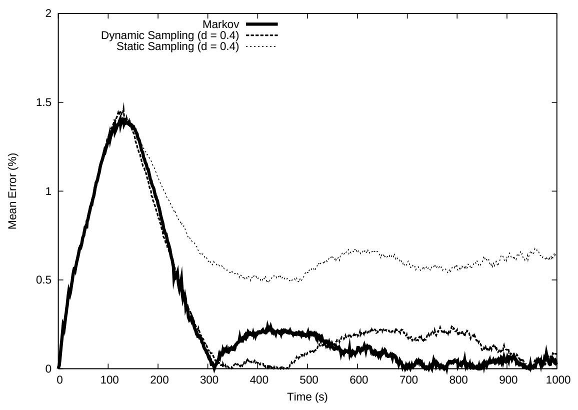
the smallest error, followed by the dynamic and static approaches, respectively. This is due to the total number of energy information packets sent in each approach. Other interesting information obtained when comparing Figures 5.13 with 5.14 is that the dynamic sampling model has a better performance than the static one. As an example, the dynamic model using $d = 0.2$ sends 122 packets, while the static using $d = 0.4$ sends 132. However, the errors of both approaches are very similar. When we compare the dynamic sampling approach using $d = 0.4$ with the static one using $d = 0.6$, we verify that the first one sends less energy packets, and has smaller errors. We can observe the same result when we compare the dynamic using $d = 0.6$ with the static using $d = 0.8$. The advantage of the dynamic model is due to the increase in the p probability when a node does not send an energy packet.

Our next goal is to compare sampling techniques with and without the interpolation defined in equation (5.15). In Figure 5.15, we verify that the greater the sampling degree, the smaller the advantage of using the interpolation phase. This result was expected because when the sampling degree increases, the number of energy information packets received in the monitoring node also increases. As the interpolation does not have any influence in the sampling phase, the number of energy information packets is exactly the same in both curves of the same figure. It is important to point out that, in our simulation model, failures are not considered. Consequently, in models where failures are considered, the interpolation phase can improve the map quality because lost information can be estimated from information of neighboring nodes.

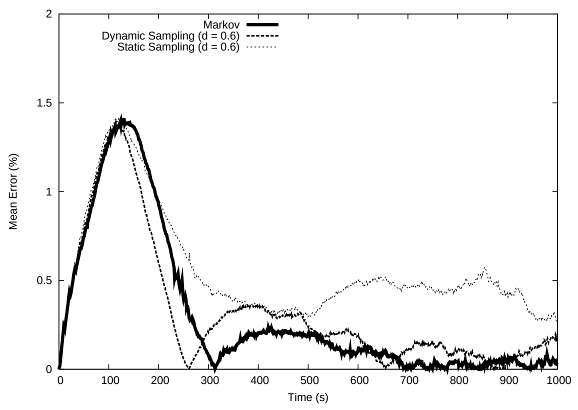
As observed in Section 5.4.3, one way to diminish the number of energy information packets needed to construct the map is to increase the value of the parameter *threshold*. Our next goal is to compare the Markov using a large value of *threshold* with sampling techniques. When we increase the value of the parameter *threshold* in the Markov model, all nodes send



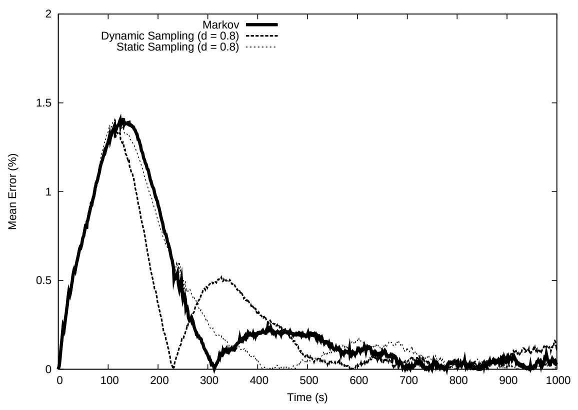
(a) $d = 0.2$



(b) $d = 0.4$



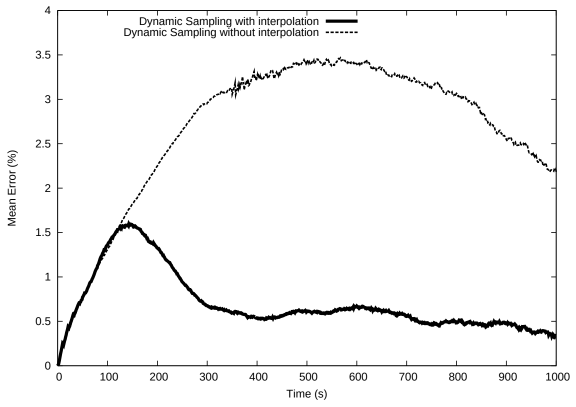
(c) $d = 0.6$



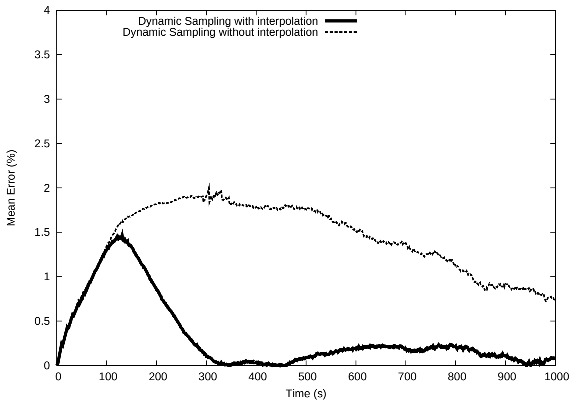
(d) $d = 0.8$

Figure 5.14: Error in percentage using the static and dynamic sampling approaches.

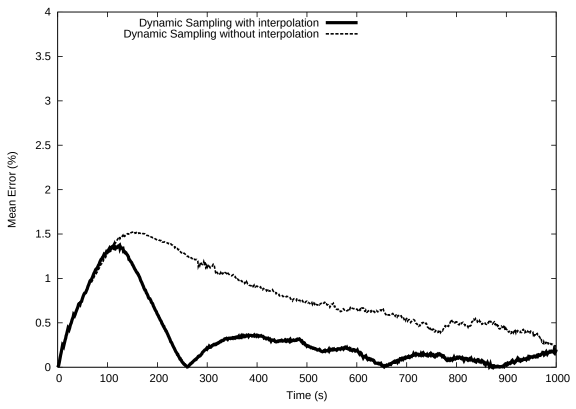
their energy information less frequently. On the other hand, in sampling models, few nodes send their energy information more frequently. Figure 5.16 compares these two approaches. In Figures 5.16-a and 5.16-b, we compare the Markov using *threshold* $= 5\%$ with the sampling model using *threshold* $= 3\%$ and $d = 0.5$ and 0.6. In Figures 5.16-c and 5.16-d, we show the Markov with *threshold* $= 7\%$ and the sampling with *threshold* $= 3\%$ and $d = 0.3$ and 0.4. Figures 5.16-a and 5.16-c show the total number of energy packets sent during all simulation time, and Figures 5.16-b and 5.16-d show the mean error in percentage. In these simulations, we note that, for similar number of energy packets, the error of the sampling model is smaller than the one of the Markov. Analyzing Figures 5.16-a and 5.16-c, we note that, at the beginning of simulation, the number of packets sent by Markov is smaller, and, consequently,



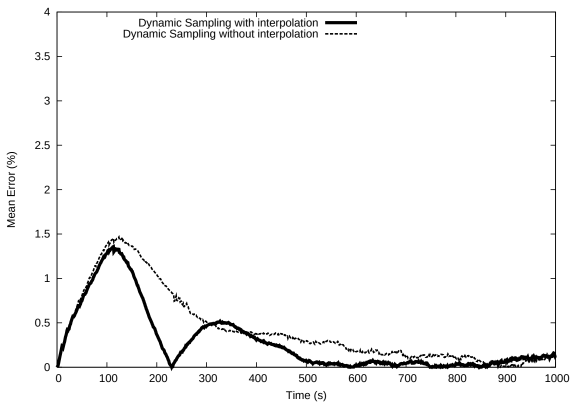
(a) $d = 0.2$



(b) $d = 0.4$



(c) $d = 0.6$



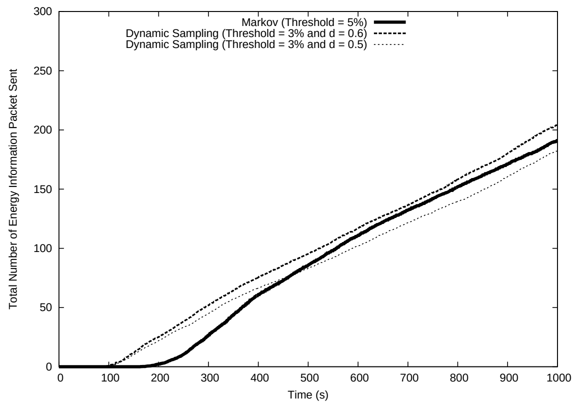
(d) $d = 0.8$

Figure 5.15: The influence of the interpolation phase in sampling techniques.

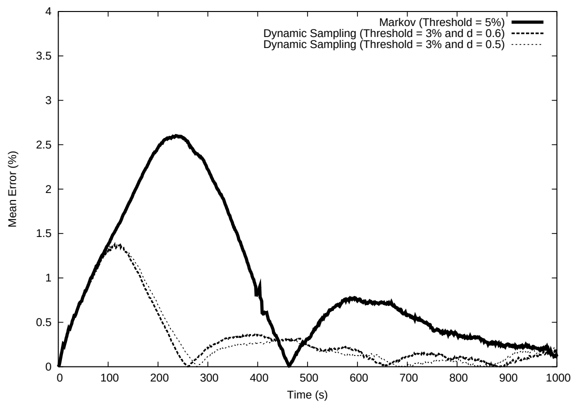
its error is bigger than the error of sampling models. This happens because the big value of *threshold* delays the sending of the energy packets in the Makov model. Therefore, the sampling model produces more constant error curves than the original Markov. This is the greatest advantage of sampling models over the original Markov.

Simulation Results for the Dynamic Event Model

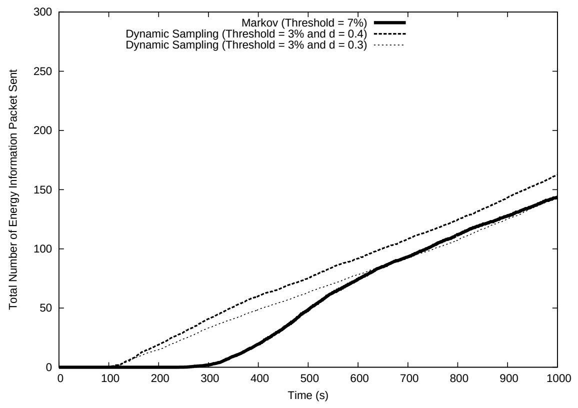
In this section, we analyze the performance of the sampling model and the original Markov when the network events are mobile and have a fixed size. The event radius is defined by a random variable uniformly distributed between 5 and 15 m. In simulations of this section, we use *cease-prob* $= 0.2$ and *mobility-prob* $= 0.5$.



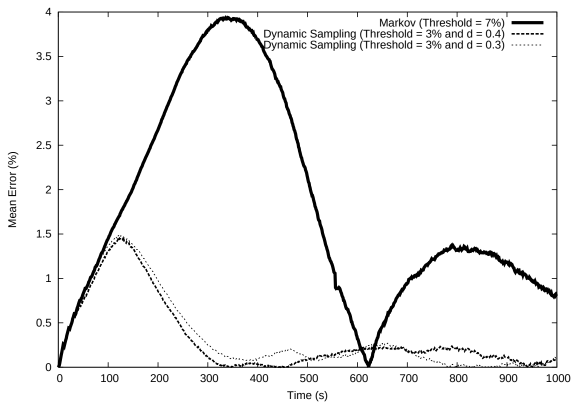
(a) Number of energy packets



(b) Average Percentage Error.



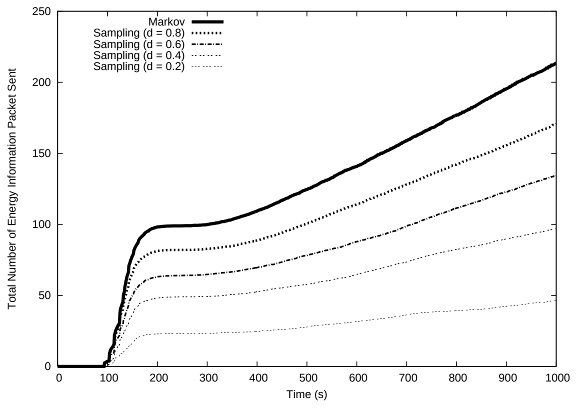
(c) Number of energy packets



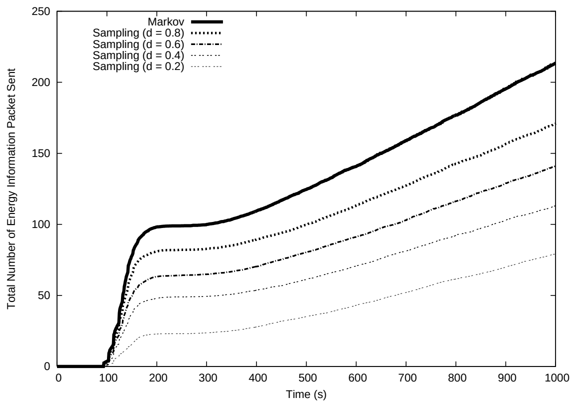
(d) Average Percentage Error.

Figure 5.16: Comparison between sampling models with the original Markov using a great value of *threshold*.

In Figure 5.17, we show the number of energy information packets sent by each approach in each second of simulation. Table 5.2 shows the number of energy information packets sent at the end of simulation for both models. Figure 5.18 shows the correspondent error curves. We can see that the Markov error is smaller than the error of sampling approaches. We verify, in Figures 5.17 and 5.18, that, the greater the sampling degree, the more energy packets are sent, and, the lower the error in percentage. We can also compare the performance of the dynamic sampling with the static one. The dynamic model, using $d = 0.4$, sends 113 packets, and the static, using $d = 0.6$, sends 134, but the error of the dynamic sampling is smaller than the error of the static model. Thus, the performance of the dynamic sampling model is better than static one. Finally, in Figure 5.19, we illustrate the influence of using interpolation in the second event generation model. We can see that the interpolation phase improves the results mainly for small values of the degree of sampling. In fact, the use of sampling models depends very much on the development of good interpolation algorithms. The sampling technique improves the results only in situations in which the monitoring node can estimate the available energy of nodes that did not send their energy information packets.



(a) Static Sampling.



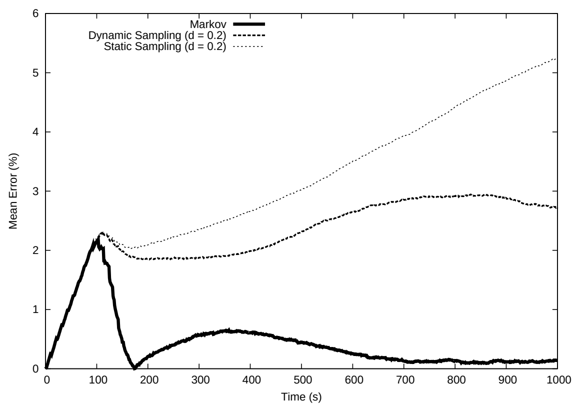
(b) Dynamic Sampling.

Figure 5.17: Number of energy packets sent using the static and dynamic sampling approaches.

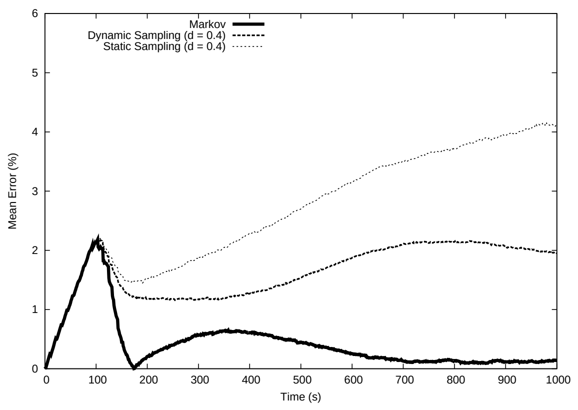
Model	**Static Approach**	**Dynamic Approach**
Sampling ($d = 0.2$)	47	79
Sampling ($d = 0.4$)	97	113
Sampling ($d = 0.6$)	134	141
Sampling ($d = 0.8$)	171	172
Markov	214	

Table 5.3: Total number of energy packets sent using the static and dynamic sampling approaches.

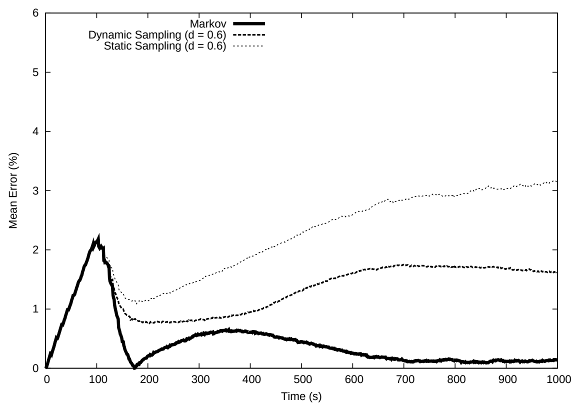
Next, we compare the sampling model with the original Markov using a large value of *threshold*. In Figures 5.20-a and 5.20-b, we compare the Markov using *threshold* = 5% with the sampling model using *threshold* = 3% and $d = 0.7$ and 0.6. In Figures 5.20-c and 5.20-d, the compared values are Markov using *threshold* = 7% and sampling using *threshold* = 3% and $d = 0.5$ and 0.4. Analyzing these results, we verify that the sampling approaches send less energy packets, and the Markov errors are smaller. In the other work, for the mobile events model analyzed, other interpolation techniques should be investigated. However, even for this event generation model, the sampling model produces more constant error curves.



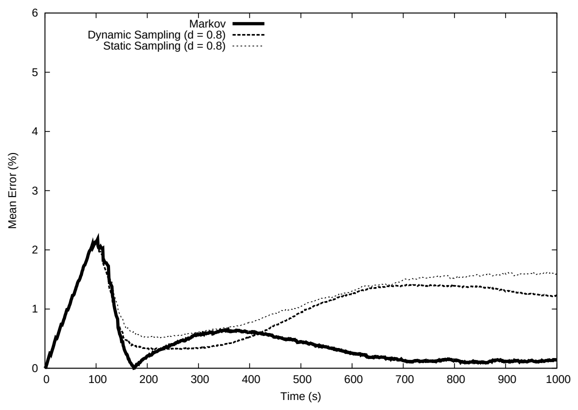
(a) $d = 0.2$



(b) $d = 0.4$



(c) $d = 0.6$

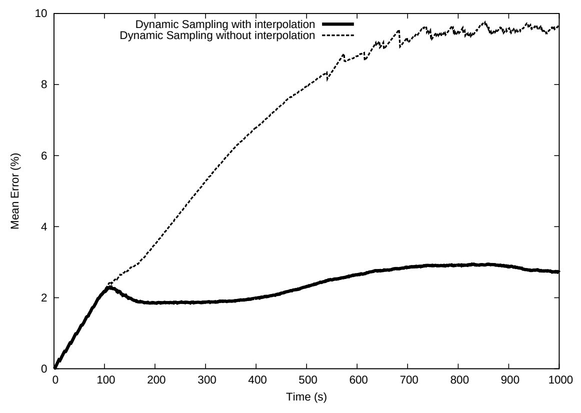


(d) $d = 0.8$

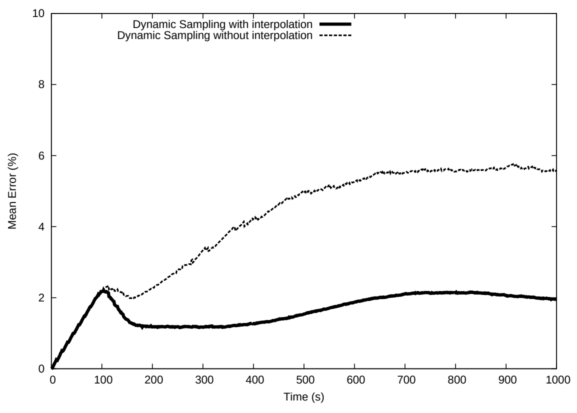
Figure 5.18: Error in percentage using the static and dynamic sampling approaches.

5.6 The Role of the Energy Map in the Protocol Design of Wireless Sensor Networks

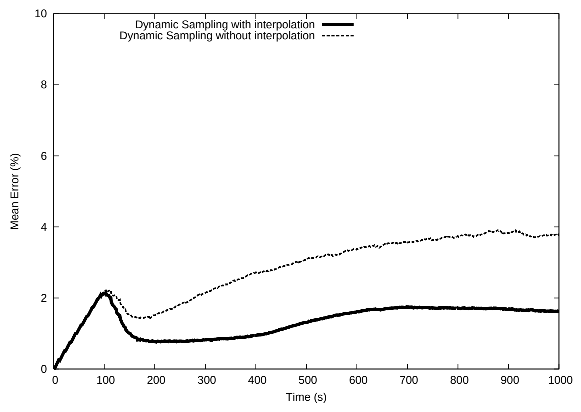
In the protocol area, the term protocol engineering was coined to denote the protocol development cycle [Liu, 1989, Piatkowski, 1981]. This area includes disciplines such as formal methods, software and knowledge-based engineering principles and basically follows the traditional software life cycle. Protocol engineering has been a very active research area during the last two decades where the fundamentals for traditional computer networks were defined and protocol design became a more systematic activity [Holzmann, 1991, Holzmann, 1992, King, 1991]. However, with the advent of wireless sensor networks, new protocol engineering principles need



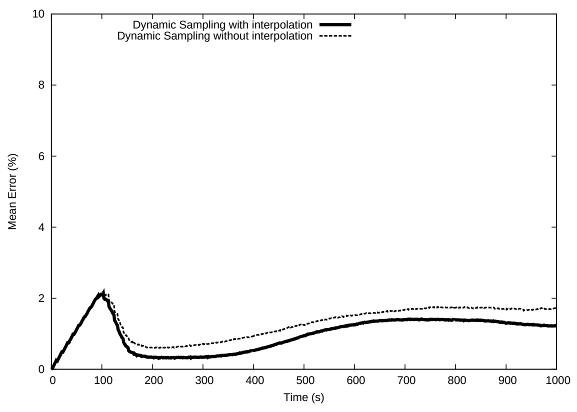
(a) $d = 0.2$



(b) $d = 0.4$



(c) $d = 0.6$



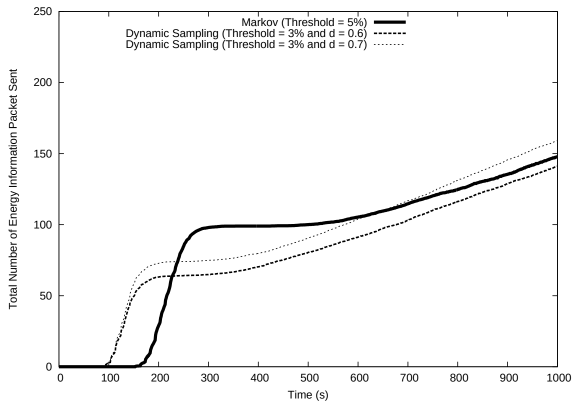
(d) $d = 0.8$

Figure 5.19: The influence of the interpolation phase in sampling techniques.

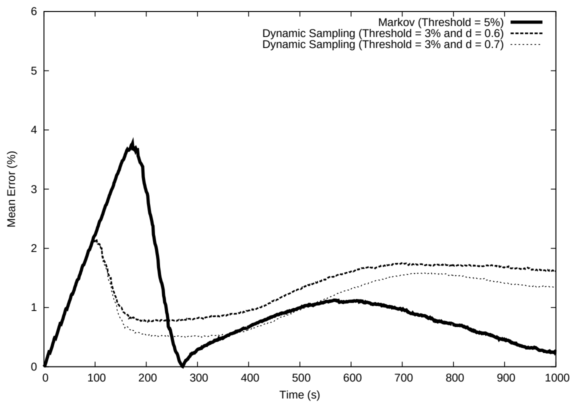
to be established.

The key challenge in the design of a wireless sensor network is maximizing its lifetime. From the point of view of protocol design, a protocol architecture for these networks should consider a power management plane as depicted in Figure 5.21. Protocols for wireless sensor networks must be energy-efficient in order to make better use of the limited energy supply of sensor nodes.

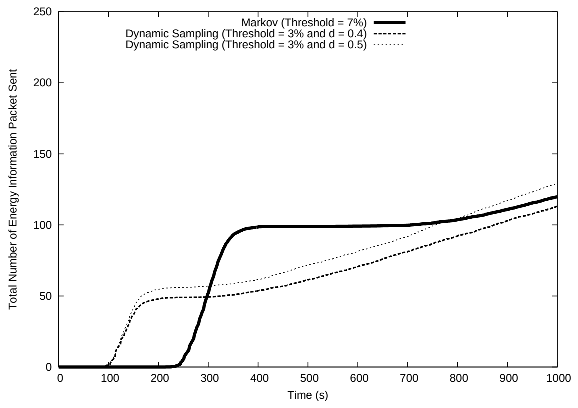
Most of the protocols proposed for wireless sensor networks, which take into account the available energy in a sensor node, use the information available locally when performing a given task. For instance, some protocols [Intanagonwiwat et al., 2000, Woo and Culler, 2001, Ye et al., 2002] try to reduce the energy consumption in order to be suitable for this new kind



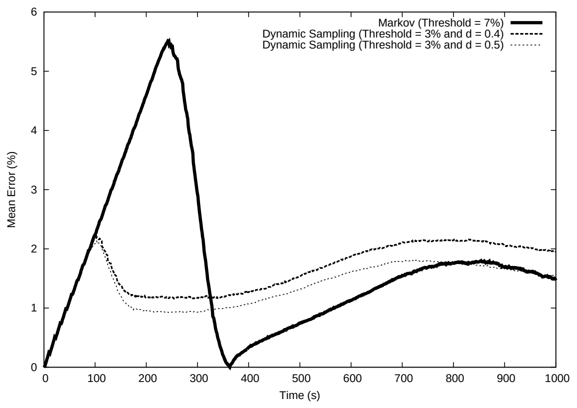
(a) Number of energy packets



(b) Average Percentage Error.



(c) Number of energy packets



(d) Average Percentage Error.

Figure 5.20: Comparison between sampling models with the original Markov using a great value of *threshold*.

of network. Other protocols [Heinzelman et al., 1999, Kulik et al., 1999, Yu et al., 2001] use the amount of available energy in the node when they make a decision. In many cases, to consider only the amount of the available energy in a node may either be sufficient or lead to an acceptable solution. Even in these cases, it would be interesting to evaluate whether an energy map could provide a better solution.

There are fundamental problems in wireless sensor networks, such as routing, that can benefit from having the energy map of the entire network. A routing algorithm can make a better use of the energy reserves if it chooses routes that use nodes with more residual energy. The protocol proposed in [Niculescu and Nath, 2003] is an example of a routing protocol that could take advantage of the energy map. In that work, the Trajectory Based Forwarding

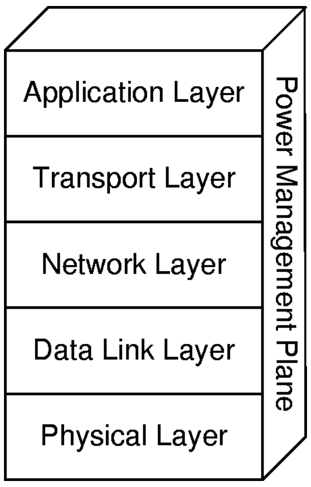


Figure 5.21: Protocol architecture for a wireless sensor network with a power management plane.

protocol that is a new forwarding algorithm suitable for routing packets along a predefined curve is described. The idea is to embed the trajectory in each packet, and let the intermediate nodes make the forwarding decisions based on their distances from the desired trajectory. If this protocol had the information about the energy map, the trajectory could be planned in order to pass through regions with more energy, thus preserving or avoiding regions of the network with small reserves. Again, the goal here is to make better use of the energy reserves to increase the lifetime of the network.

We support the idea that the energy map of a wireless sensor network should be used as a new protocol engineering principle when designing new protocols for this kind of network. If this is the case, the design of a new protocol for a wireless sensor network can specify, given a particular scenario in the network, the best action to be taken to improve its energy efficiency. Therefore, an energy map can be the basis for the entire design trajectory including all functionalities to be included in the wireless sensor network. The effectiveness of having an application running in a wireless sensor network will depend on the success in obtaining an energy map. Note that the applicability of this map is not restricted to a particular aspect of the application, but to all activities present in the network since all of them need energy to be carried out.

5.7 Concluding Remarks

The use of prediction techniques is very common in many research areas such as meteorology [Calvo et al., 2000], stock market [Ge, 1998] and biology [G.Pollastri et al., 2001]. In computer networks, prediction algorithms have been used, for instance, to predict network traffic [Sang and qi Li, 2000]. The ability to predict traffic patterns within a network is one of the fundamental requirements of network management.

Wireless sensor nodes tend to have very strict hardware resources. Thus, a prediction algorithm for these networks is required to be simple. This simplicity implies that the processing time in estimating the future energy consumption rate and the number of parameters that have to be sent to the monitoring node cannot pose a heavy burden on the sensor node. Another characteristic that we pursue when choosing a prediction algorithm, is that all computation is done locally. Each node should make its own prediction based only on its past behavior and no communication between neighboring nodes is required.

Our goal when choosing the Markov chain is to have a very simple prediction algorithm based on states, like the energy dissipation model presented in this work. The main idea is that transitions between states will happen in the future in the same way they happened in the past. As example, if in 30% of the time when a node was in operation mode 1, it went to operation mode 2, it means that when this node will be in state 1, it will go to state 2 with probability 0.3. This prediction technique has two main advantages for wireless sensor networks:

- The computation of the prediction is simple and it is done locally, since each node computes its power consumption by only keeping track of its past state transitions.
- It is suitable for wireless sensor networks since, in these networks, the node has to turn off parts that are not been used to save energy. Thus, nodes can be modeled by states of operation.

We chose the ARIMA model in order to have a more sophisticated technique to be contrasted with the Markov chain. Our goal was to compare a technique to make predictions based on time series with the very simple one based on states. The results showed that ARIMA is not suitable for wireless sensor networks due to its complexity in terms of the number of communications.

In this chapter, we showed that the use of prediction is a good approach to construct the energy map of a wireless sensor network. Due to the nondeterministic characteristic of sensor networks, it is better to perform predictions that are simple both in terms of the computation required to find the parameters of the prediction model, and mainly in terms of the number of parameters that have to be sent to the monitoring node. Thus, in the construction of prediction-based energy maps, it is probably better to use simple models instead of sophisticated predictions that demand a lot of communication between sensors and the monitoring node.

In this chapter, we also analyzed the use of a sampling technique to reduce the number of packets needed to construct the map. The basic supposition is that neighboring nodes tend to spend their energy similarly. Thus, the information sent by a node can be used to update the energy consumption rate of its neighboring nodes. Results show that the sampling model can diminish the number of energy packets sent, and that its most important advantage is to produce more constant error curves.

The energy map construction, presented in this chapter, was designed taking into account the parameter *threshold*. We defined the precision in which the map should be constructed, and no restriction concerning the amount of energy that could be spent in this construction was defined. In the next chapter, we will look at the energy map construction in a different way. A restrictive energy map construction will be presented in a way that, instead of defining the accuracy of the map, we define the maximum amount of resources that can be spent in its construction.

Chapter 6

Finite Energy Budget

6.1 Introduction

The large use of wireless sensor networks depends on the design of a scalable and low-cost sensor network architecture. Furthermore, the design must consider the energy conservation a fundamental issue and devise mechanisms for extending the network lifetime. This scenario leads us naturally to the following problem: what is the best performance a protocol can achieve given that it can spend only a finite amount of energy? Using this idea, we can associate a finite energy budget for each network activity, and ask this activity to achieve its best performance using only its budget. This is a new way of dealing with network related problems, and should be considered a new paradigm to design algorithms for networks that are battery powered, specially for wireless sensor networks.

The finite energy budget paradigm is highly applicable in the construction of the energy map of a wireless sensor network. It is worthless if we construct the best energy map spending all available energy. In this chapter, we extend the prediction-based approaches in order to define a way of constructing the energy map in situations where a finite energy budget is defined. Our goal is to achieve the performance limits in the construction of the energy map under the constraint that each node can spend only a certain amount of energy in this construction.

The remainder of this chapter is organized in the following way. In Section 6.2, we show how the finite energy budget model is applied in the energy map construction. In Section 6.3, we present simulation results for this model. In Section 6.4, we expand the basic finite budget model, presenting an adaptive process to build the energy map. Section 6.5 shows simulation results for the adaptive energy map construction. Finally, in Section 6.6, we present the final comments about using this new paradigm to construct the energy map of wireless sensor networks.

6.2 Non-Adaptive Energy Map Construction under a Finite Energy Budget

In the last chapter, we presented prediction-based approaches to construct the energy map for wireless sensor networks. In those approaches, the parameter *threshold* defines the accuracy in which the energy map is constructed. However, due to the paramount importance of the energy conservation in wireless sensor networks, it is highly desirable to define the amount of energy we can spend in the energy map construction, thus leaving the remaining energy to be used by other network activities. In this section, we show how the finite energy budget paradigm can be applied in the construction of the energy map of a wireless sensor network.

In the energy map construction, the term finite energy budget means that each node can spend a certain amount of energy in the process of constructing the energy map of the entire network. This amount of energy can be represented by the number of bytes each node can send with energy information to the monitoring node. Knowing the size of this information, we can transform the number of bytes into the number of packets each node can send with energy information. In this work, the number of packets is used as the metric for energy budget, and we deal with it as the maximum number of times each node can send its energy information to the monitoring node. Our goal is to construct the best energy map under the constraint that each node can send no more than a certain number of packets with energy information.

Ideally, a solution that approaches the performance limits in the construction of the energy map should keep the error almost constant during all time, and the budget should finish exactly at the end of the network lifetime. To achieve this goal, we have to decide when is the best time for each node to send its energy information packet under a finite energy budget. In this section, we propose a way of making this decision.

The moment to send the energy information packet is decided locally by each node without exchanging information with its neighbors. Thus, in each second of simulation, each node should decide if it will send another energy information packet or not. We propose that this decision should be taken according to a certain probability p. In that case, the value of p determines the frequency in which nodes will send their energy information and, thus, the amount of energy spent in the process of constructing the energy map. The value of p depends on the following parameters: the number of packets a node still can spend, the error between the predicted and the correct energy value, and its estimated lifetime. This last supposition is reasonable because given the restriction of the available energy in the sensor network, we can expect to design a wireless sensor network to work for a period of time that can be defined during its design phase.

In order to find the best way to determine the value of the probability p for each node, we start defining the sending of an energy information as a binomial distribution in which the event is the sending of an energy packet and each second of simulation is an experiment. In each second of simulation, we have a probability p that the event occurs (the node sends the energy packet) and a probability $(1 - p)$ that the event does not occur (the node does not send the energy packet). Then, we have to find the correct value of p that maximizes the probability that each node sends the number of packets determined by its budget. We call T_{total} the estimated lifetime, and T_{now} the current time. In addition, P_{total} is the number of energy information packets each node can send, and P_{used} the number of packets the node has already used. Thus, a node still can send $(P_{total} - P_{used})$ packets in the remaining $(T_{total} - T_{now})$ seconds. To find the probability of sending a packet in each second of simulation, we have to maximize the probability of happening $(P_{total} - P_{used})$ events in $(T_{total} - T_{now})$ experiments. If p is the probability of the occurrence of an event, we have that:

$$P(X = P_{total} - P_{used}) = \binom{T_{total} - T_{now}}{P_{total} - P_{used}} p^{(P_{total} - P_{used})} (1 - p)^{(T_{total} - T_{now}) - (P_{total} - P_{used})} \tag{6.1}$$

To find the best value of p in a way that a node will send $(P_{total} - P_{used})$ packets in $(T_{total} - T_{now})$ seconds, we have to maximize the function of equation (6.1). The value of p

that maximizes this function is:

$$p = \frac{P_{total} - P_{used}}{T_{total} - T_{now}} \tag{6.2}$$

Using equation (6.2), each node can determine the probability in which it will send an energy packet in each second of simulation. This equation defines the value of p only in terms of the energy budget and the estimated lifetime.

Nevertheless, each node also knows the error between the energy value predicted by the monitoring node and the correct one. It can locally determine this value by just keeping the parameters of the last prediction sent to the monitoring node. Thus, each node keeps track of the error of its energy value. We use the percentage error because the impact of the difference between the correct and the predicted value depends on the available energy at the node. For example, if a node has 100 J of energy and we predict that it has 99 J is better than when we make the same error of 1 J when the total energy available in a node is only 2 J. Because of that, considering the percentage error is more suitable than using its absolute value. Therefore, in our approach, the error is always considered as a percentage value in relation to the correct available energy.

We intend to use the information of the error to change the value of the probability p in such a way that, when the error is small, we should decrease the value of p in order to postpone the sending of an energy packet. With this behavior we can save energy to be spent when the error is larger. On the other hand, when the error is large, we should increase the value of the probability in order to force the node to send a new energy information. This behavior can be obtained with the aid of the function:

$$f(x) = \left(1 - \frac{1}{c^x}\right); \qquad c > 1 \tag{6.3}$$

In Figure 6.1, we plot this function. We can see that this function tends to one when the value of x increases. The speed of this trend depends on the value of constant c. This constant has to be greater than one, and the larger its value, the faster $f(x)$ tends to one. This function will be useful to make the value of the probability p to adapt according to the error in the energy information.

The desired adaptation is obtained by equation (6.4). In that equation, we redefine the

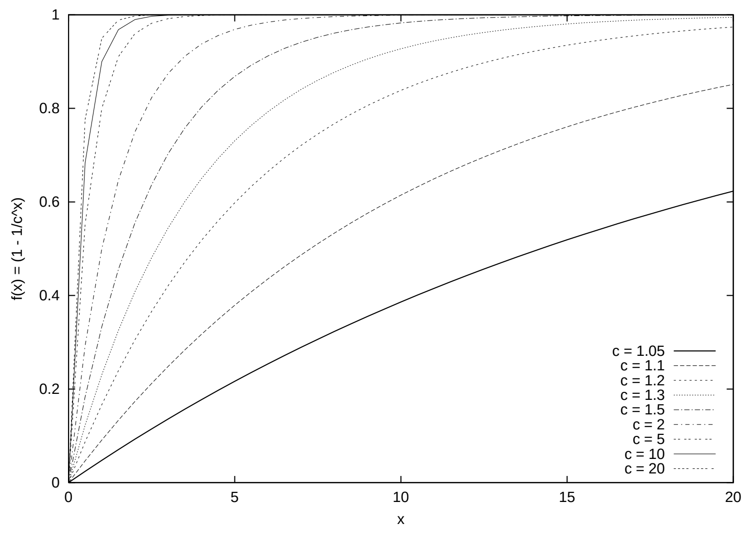


Figure 6.1: Function defined by equation (6.3) for different values of c.

value of p and call it p'. Then, in each second of simulation, each node will send another energy information packet with probability p'. In the first part of equation (6.4), the value of p is multiplied by the function $(1 - \frac{1}{c^{error}})$, and p is decreased when the error is small, and it is almost unchanged when the error increases. However, when the error gets larger, the value of the probability of sending a packet should increase and becomes larger than p. This behavior is obtained using the second part of equation (6.4), in which the value of $(1 - p)$ is multiplied by the same function but with different parameters. The expression $max(0, error - k)$ is different from zero only when the error is larger than k. Using the value of c and k, we can control the shape of the curve that represents the probability of sending a packet.

$$p' = p \times \left(1 - \frac{1}{c^{error}}\right) + (1 - p) \times \left(1 - \frac{1}{c^{max(0, error-k)}}\right) \tag{6.4}$$

Before using equation (6.4), we have to decide on the value of k. This value determines when the second curve will start. For example, if we want that the second curve starts when the curve $(1 - \frac{1}{c^{error}})$ is 0.99 (this means that the value of p' is 99% of p), we make $(1 - \frac{1}{c^k}) = 0.99$ and thus we find $k = \log_c 100$. If we want that the second curve starts when the first is 0.999, we make $k = \log_c 1000$. In Figure 6.2, we plot the curve of p' when $p = 0.5$ and $c = 2$ for different values of k. We can see that, when we increase the value of k, we are postponing the appearance of the second slope and delaying the increase of the value of p'.

In Figure 6.3, we plot the curve p' when the value of p is 0.5 and $k = \log_c 1000$ for different

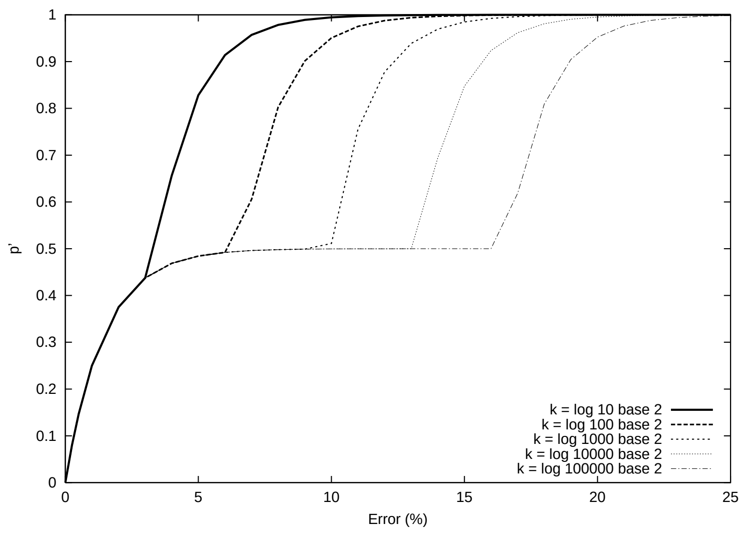


Figure 6.2: Function p' for $p = 0.5$, $c = 2$ and different values of k.

values of c. We can see that the larger the value of c, the faster the value of p' will be one. Thus, if a node has only a small number of packets to spend in the construction of the energy map, it should use a small value of c. On the other hand, if a node can spend a lot of packets with energy information, it should use a larger c.

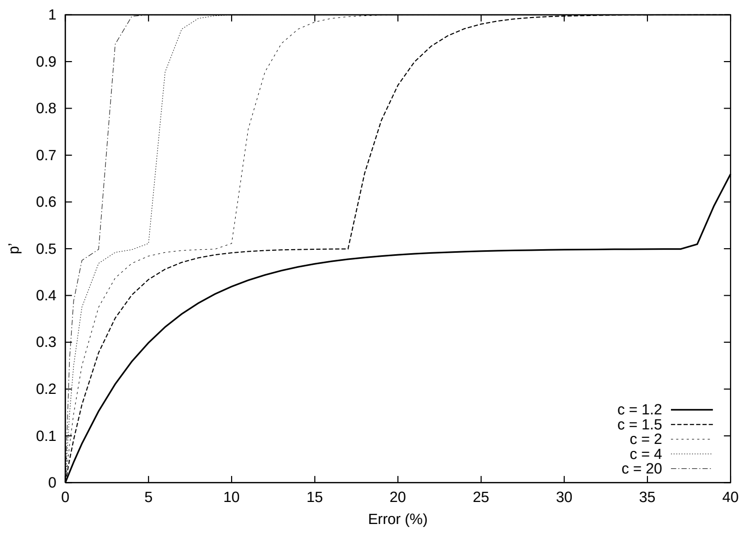


Figure 6.3: Function p' for $p = 0.5$, $k = \log_c 1000$ and different values of c.

In equation (6.4), we used the same value of c in both parts. Nevertheless, using different values for these constants, as shown in equation (6.5), we have more flexibility to control the behavior of the function p'. In Figure 6.4, we plot equation (6.5) using different values for constants c_1 and c_2. Basically, using different values for these constants and changing the value of k, we can change the shape of the function p' and the way the energy budget is spent.

$$p' = p \times \left(1 - \frac{1}{{c_1}^{error}}\right) + (1-p) \times \left(1 - \frac{1}{{c_2}^{max(0,error-k)}}\right) \quad (6.5)$$

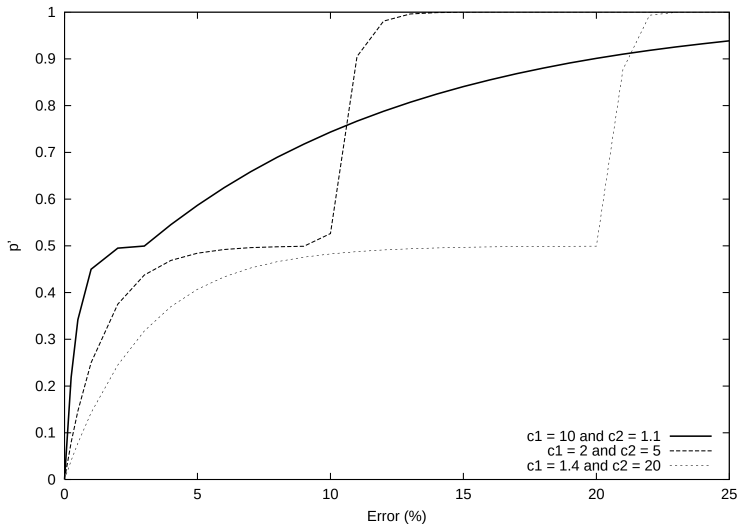


Figure 6.4: Function p' for $p = 0.5$, $k = \log_c 1000$ and different values for c_1 and c_2.

6.3 Simulation Results for the Non-Adaptive Energy Map Construction

In order to analyze the performance of the finite energy budget scheme, we implemented the construction of the energy map in the ns-2 simulator [ns2, 2002]. In our simulations, we use the State-based Energy Dissipation Model, presented in Chapter 4, to describe the behavior of sensor nodes and to simulate their energy dissipation. The events were generated using the static event model and their arrival is modeled by a Poisson process.

The numerical values chosen for the base case of our simulations can be seen in Table 6.1. Unless specified otherwise, these values are used as the parameters in all simulations throughout the remainder of this chapter . Moreover, the monitoring node is positioned at the center of the field at position $(25, 25)$, all nodes are immobile, and can communicate with other nodes within their communication range. We assume that the monitoring node knows the initial energy at each sensor. Also, before the nodes send their first energy information packet, the

Parameters	**Value**
Number of Nodes	100
Initial Energy	100 J
Communication Range	15 m
Sensor Field Size	50×50 m^2
k	$\log_c 1000$
λ	0.1
Static Event Model	
event-duration-min	5 sec
event-duration-max	50 sec
event-radius-min	5 m
event-radius-max	15 m

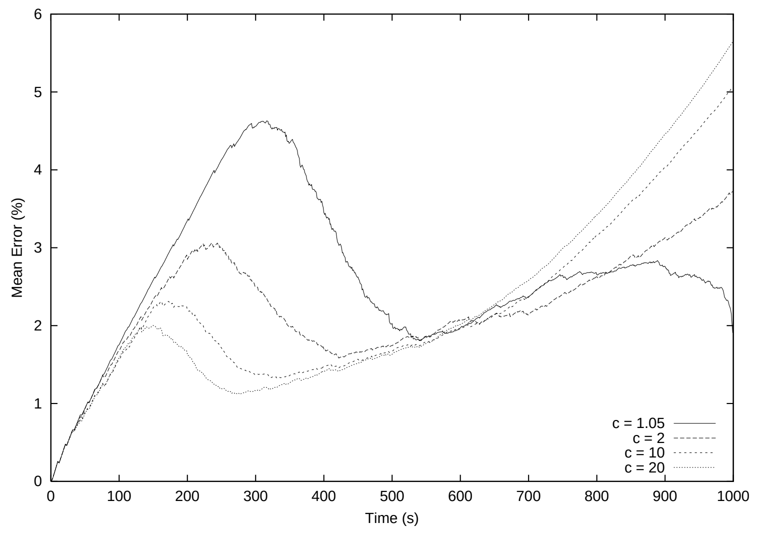
Table 6.1: Default values used in simulations.

monitoring node supposes that their power consumption is the average of the power consumption of all states. Thus, at the beginning of all simulations, the monitoring node assumes that all nodes have 100 J, and are spending energy at the rate of 41.41 mW. Besides, the results showed in all simulations correspond to an average of these values for 30 different runs.

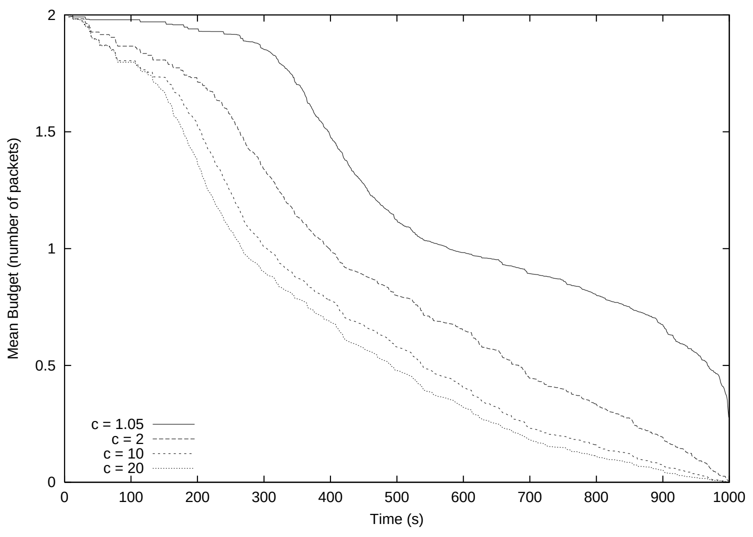
In the finite energy budget, presented in the last section, the choice of the best value for the constant c is of fundamental importance to determine the behavior of probability p', and consequently the way each node will spend its budget. In order to analyze the influence of this constant for different values of budget, we ran the prediction-based energy map for 100 nodes in the same scenario described above. Figure 6.5-a shows the average percentage error when each node can send only 2 energy information packets to the monitoring node, and, in Figure 6.5-b, we have the mean budget for each value of c. We can see that the larger the value of c, the faster the nodes use their budget. For example, for $c = 20$, the budget is used fast and, at the end of simulation, almost all nodes have already spent their budget, increasing the mean error. On the other hand, for $c = 1.05$, all nodes spend their budget slowly.

Using the constant $c = 1.05$ and a budget size of 2 packets/node, the error starts larger and goes down after the start up period, keeping almost constant until the end of simulation. This behavior can be explained by equation (6.2). As the denominator represents the simulation time in seconds, and the numerator the budget size per node, the denominator range value is larger than the numerator one. Therefore, at the beginning of simulation, the value of p tends to be decreased due to the large value in the denominator. This collateral damage is good, because it is better to make an error at the beginning of simulation when nodes have more energy than at the end, when the energy becomes even scarcer. Then, the shape of the error is good for sensor network applications.

We can see in Figure 6.5-b that, for $c = 1.05$, the nodes do not spend all their budget. In a situation like this, we can increase the value of c in order to use all the available budget to obtain a smaller error.



(a) Mean Error (%).



(b) Mean Budget (number of packets).

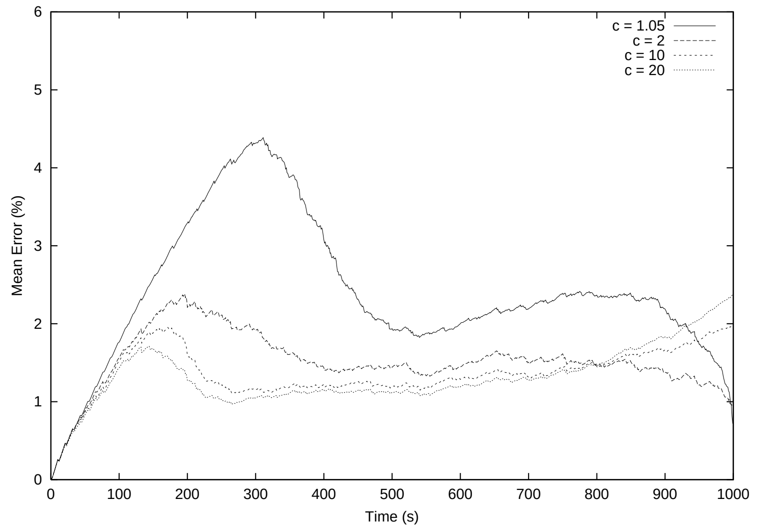
Figure 6.5: Changing the value of c when each node can spend 2 packets with energy information, $k = \log_c 1000$ and $\lambda = 0.1$.

Next, we repeated the simulation above with a budget size of 4 packets/node as presented in Figure 6.6. Using $c = 1.05$, the nodes do not spend all their budget and the error is large. Using $c = 20$, the nodes spend fast their budget and, at the end of simulation, the error gets larger because all budget is used. On the other hand, if we use $c = 2$, we achieve a good performance because the budget is over almost at the end of simulation, and the error keeps almost constant during all the time. This means that the value of $c = 2$ is a good choice when we have a budget size of 4 packets/node.

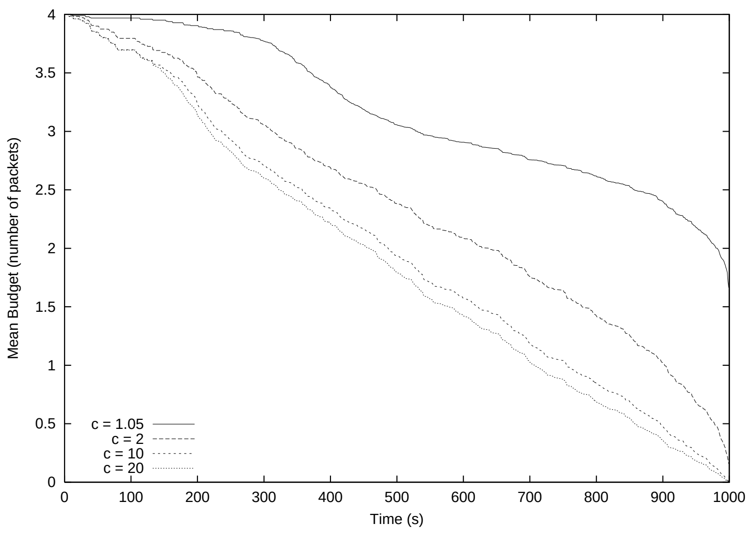
When the budget size is increased, we should use a larger constant. This can be seen in Figure 6.7, where a budget size of 8 packets/node is used. In this case, the best performance was achieved using $c = 10$ or $c = 20$. We can also observe that for $c = 1.05$, the budget is not appropriately used and, at the end of simulation, the nodes spend more of their budget and the error curve goes down. Even though, for this constant, only a small amount of the available budget is used.

Using a budget size of 16 packets/node, the best constant value is $c = 20$, as illustrated in Figure 6.8. We can see that using $c = 1.05$ is not a good option when we have a large budget to spend. Also, the decrease in the percentage error is not linear to the increase in the budget size. When we double the budget size, only a small decrease in the mean error is obtained.

Observing the slope of the budget curve, we can see that equation (6.4) provides a small adaptation during the simulation. For all values of c, when time passes, the slope of the budget curve is changed in order to adjust to the remainder budget and simulation time. Nevertheless, this adaptation is not enough to achieve our goal in the energy map construction: use all the available budget keeping the percentage error almost constant. The performance of using equation (6.4) to decide when sending the energy packet is highly dependable on the right choice of the constant c. If it is used a wrong value of c, the budget cannot be completely used or it can be used too fast increasing the percentage error. Therefore, in order to achieve the desired behavior, it is necessary to find a way of choosing the value of constant c automatically. This is the topic of the next section.



(a) Mean Error (%).



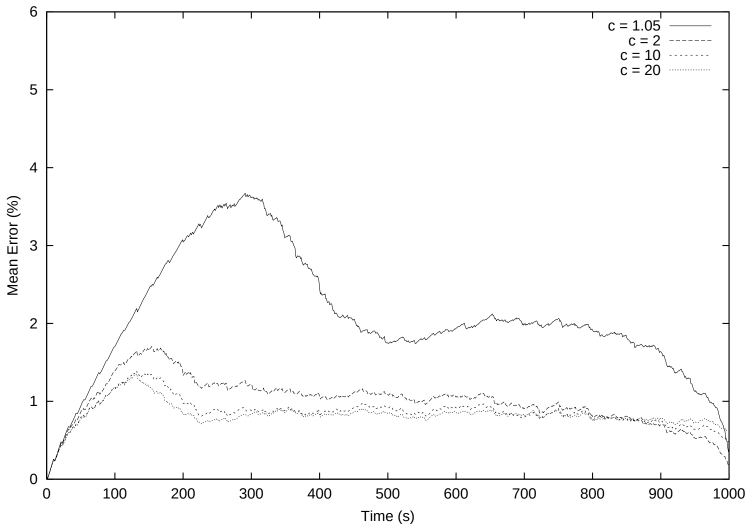
(b) Mean Budget (number of packets).

Figure 6.6: Changing the value of c when each node can spend 4 packets with energy information, $k = \log_c 1000$ and $\lambda = 0.1$.

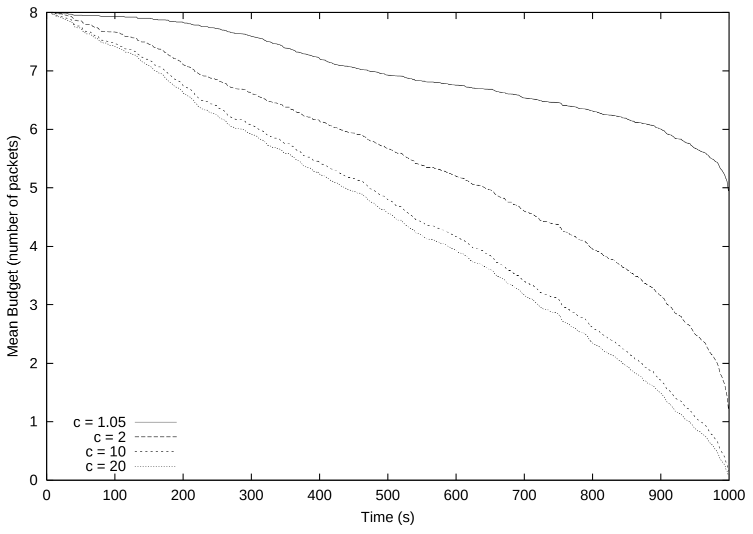
6.4 Adaptive Energy Map Construction under a Finite Energy Budget

6.4.1 Changing the Value of c

In the previous section, we saw that the performance of the finite energy budget approach is highly dependable on the value of constant c. Now, we present a way of changing the value



(a) Mean Error (%).

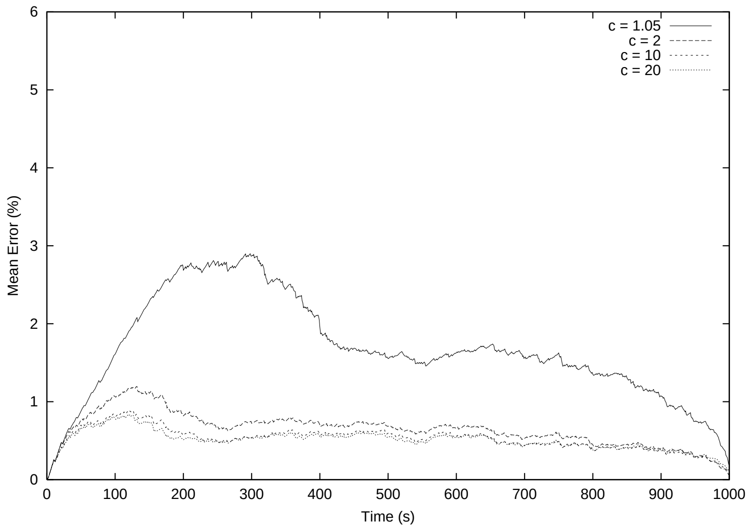


(b) Mean Budget (number of packets).

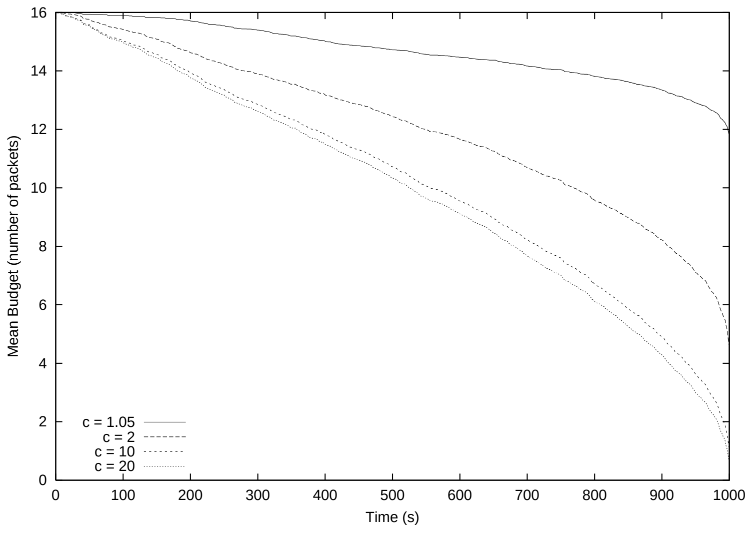
Figure 6.7: hanging the value of c when each node can spend 8 packets with energy information, $k = \log_c 1000$ and $\lambda = 0.1$.

of this constant automatically such that when we have a large budget, a big value of c is used and, when a small budget is available, its value should be small.

The key information that guide us to choose the right value of c is the budget curve. We start the adaptive process analyzing the budget curve periodically. As example, for each 3% of the simulation time, we apply a linear regression in the budget curve in order to predict when



(a) Mean Error (%).



(b) Mean Budget (number of packets).

Figure 6.8: hanging the value of c when each node can spend 16 packets with energy information, $k = \log_c 1000$ and $\lambda = 0.1$.

it would be the end of budget. Using this information, we can make the following observations:

1. If the predicted end of budget is beyond the end of simulation, the value of c must be increased, otherwise, it must be decreased;

2. In order to have a conservative behavior, at the beginning of simulation, we should increase less and decrease more the value of c, depending on the case. On the other

hand, at the end of simulation, we should increase more and decrease less its value. This behavior is justified because it is more difficult to keep the percentage error constant at the end of simulation than at the beginning since, at the end, the available energy is lesser.

3. When the value of c is small, we should increase less its value. This is true because for small c, small changes in its value produce more distant curves.

The three remarks described above can be seen as the requirements that an adaptive process must follow. Next, we describe how we deal with each one of these requirements.

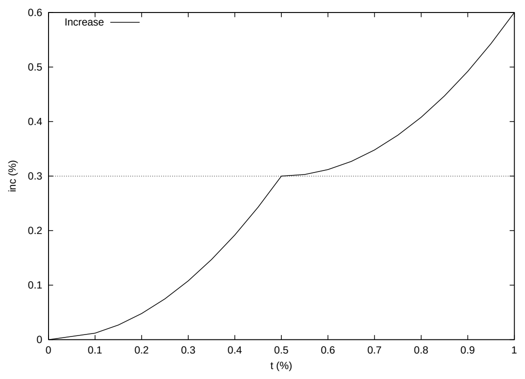
In order to achieve the first requirement, we calculate a value of *dif* as being the amount in percentage that we should increase or decrease in the predicted end of budget in order to make this value equals to the end of simulation time. If $dif > 0$, we increase the value of c, otherwise we decrease the value of c. The amount of increasing and decreasing is defined taking into account the second requirement. Knowing that t is the percentage of simulation time already done. The percentage of increasing is given by the function:

$$inc = \begin{cases} \frac{dif}{0.25}\, t^2 & \text{if t} \leq 0.5, \\ dif + \frac{dif}{0.25}(t-0.5)^2 & \text{if t} > 0.5. \end{cases} \tag{6.6}$$

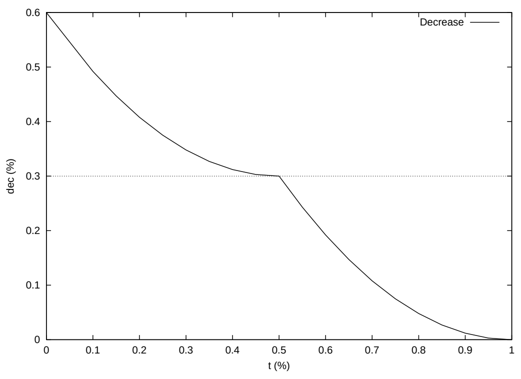
The amount of decreasing is set according to the following equation:

$$dec = \begin{cases} \frac{dif}{0.25}\, t^2 - \frac{dif}{0.25}\, t + 2\ dif & \text{if t} \leq 0.5, \\ \frac{dif}{0.25}\, t^2 - 8\ dif\, t + \frac{dif}{0.25} & \text{if t} > 0.5. \end{cases} \tag{6.7}$$

Figure 6.9 shows these functions when $|dif| = 0.3$. This means that the predicted end of budget should be increased or decreased in 30% in order to achieve the end of simulation. We can see that using this function, at the beginning of simulation, we increase less the value of c and decrease more. On the other hand, at the end of simulation, we increase more and decrese less its value. This approach provides a conservative behavior since it will try to save budget at the beginning of simulation to be used at the end. This is a good approach because, at the end of simulation, the available energy is smaller, and thus it is more difficult to keep the percentage error constant.



(a) Increase function when $dif = 0.3$.



(b) Decrease function when $dif = -0.3$.

Figure 6.9: Increase and decrease function.

In other to understand the third requirement, we have to notice that the smaller the value of c, more distant will be the curves when we make a change in its value. In Figure 6.10, we can see that the distance between the curves $c = 1.05$ and $c = 1.05 + 10\% = 1.155$ is bigger than the distance between the curves $c = 2$ and $c = 2 + 10\% = 2.2$. This means that when we are working with a small c, the error is bigger, and a small increase in its value will change p' to a curve where the probability of sending a packet for the current error is almost 1. In situations like this, the new value of c will make all nodes send an energy information packet at the same time, ending the budget. In addition, small values of c happens in situations in which a small budget is assigned to the energy map construction, and, in these cases, an error in the decision of when to send an energy information packet cannot be undo.

To illustrate the problem with small values of c, we run a 2000 second simulation with a budget size of 1 packet per node. Figure 6.11-a shows the value of c in each second of simulation and, in Figure 6.11-b, we plot the error in the energy map. The simulation starts with $c = 1.01$ and, at time 200 seconds, the adaptive process decides to increase its value to $c = 1.028358$. This small change makes all nodes send their energy packet at the same time, ending the budget. Besides, in this case, the error cannot be undo because the budget associated with each node is only 1 packet. Because of that, no more energy information packet can be sent and the error curve goes up.

In order to solve the problem of increasing and decreasing the value of c when it is small, we made another modification in the value of inc for small values of c. If the value of c is smaller

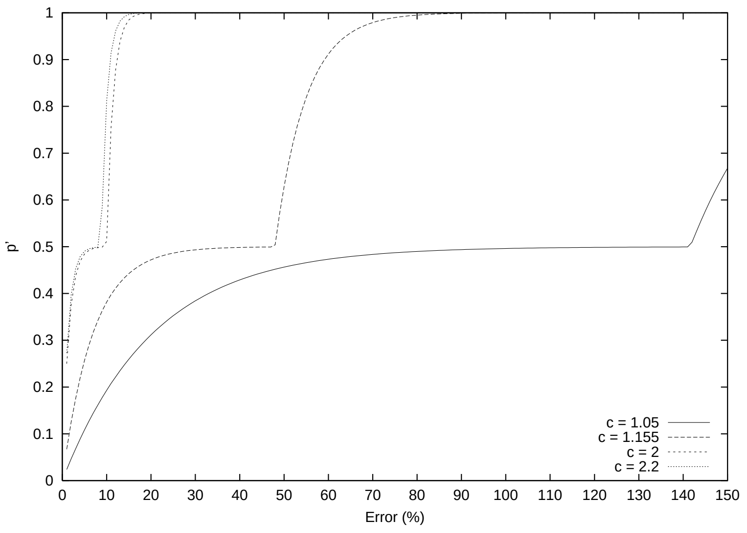


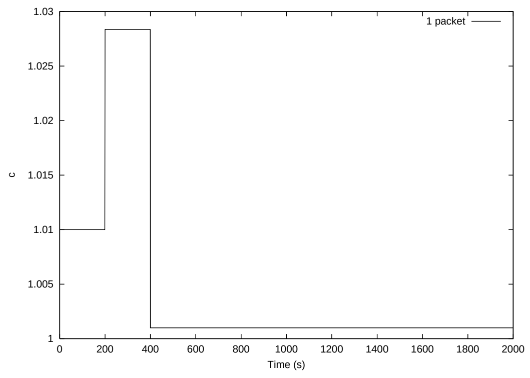
Figure 6.10: Increasing the value of c in 10%.

than 1.001, we make $inc = \frac{inc}{100}$. If the value of c is between 1.001 and 1.01, we make $inc = \frac{inc}{10}$. The algorithm presented in Figure 6.12 summarizes all the adaptive process described above.

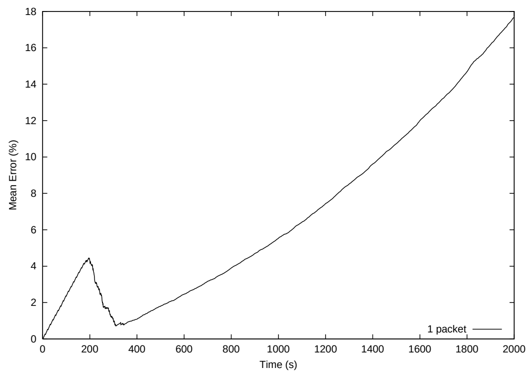
6.4.2 Choosing the Initial Value of c

In the last section, we saw how the adaptation in the value of c is done. However, we still have the problem of choosing the initial value of c. If the budget size is small, we cannot initialize c with a big value because, when the adaptive process realizes that the value of c is big, it can be very late to make a good use the of available budget. On the other hand, if we start the value of c with a value smaller than the right one, the error can increase too much until the adaptive process increases c to the right value. Therefore, in order to see the influence of the choice of the initial value of c, we run the adaptive process in the same scenario described above for three different values of budget: 1, 4 and 10. In all simulations, the budget curve is verified for each 3% of simulation time.

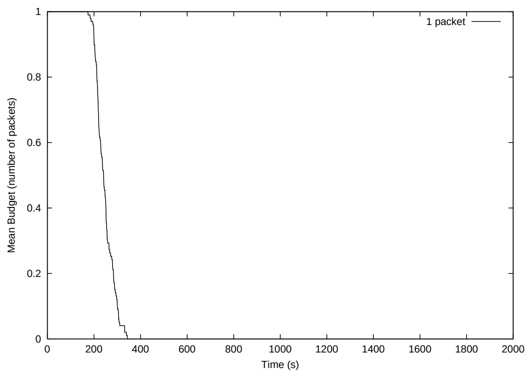
Figure 6.13 illustrates what happens when each node can send only 1 packet to the monitoring node. In Figure 6.13-a, the value of c is plotted in each second of simulation. Figure 6.13-b shows the percentage error, and, Figure 6.13-c plots the available budget. We can see that when each node can send only 1 packet to the monitoring node, the right value of c is a little bit greater than 1. Also, in this scenario, the choice of the initial value of c does not make a great difference. The wrong choice of initializing the value of c with a big value, $c = 4$, is



(a) Value of constant c.



(b) Mean Error (%).



(c) Mean Budget (number of packets).

Figure 6.11: The problem of using a small budget in a 2000 second simulation when the budget size is 1 packet per node, $k = \log_c 1000$ and $\lambda = 0.1$.

undone very fast at the beginning of simulation. This can be explained by the characteristics of the adaptive process. The value of c is decreased more at the beginning of simulation,

```
**Adaptive Algorithm:**
**Input**:
   simulationTime                              {total amount of simulation time}
   t                                   {percentage of simulation time already gone}
   c                                                          {current value of c}
begin
   endOfBudget ← LinearRegression(budgetCurve);
   dif ← (endOfBudget − simulationTime) / endOfBudget;
   if (dif > 0) then begin
                  if (t ≤ 0.5) then
                       inc ← (dif × $t^2$) / 0.25
                  else inc ← dif + (dif × $(t - 0.5)^2$) / 0.25;
                  if (c ≤ 1.001) then
                       inc ← inc / 100;
                  else if (c ≤ 1.01) then
                            inc ← inc / 10;
                  c ← c + c × inc;
               end
          else begin
                  if (t ≤ 0.5) then
                       dec ← (dif × $t^2$) / 0.25 − (dif × t) / 0.25 + 2 × dif;
                  else dec ← (dif × $t^2$) / 0.25 − 8 × dif × t + dif / 0.25;
                  c ← c + c × dec;
               end;
end {Adaptive Algorithm}.
```

Figure 6.12: Adaptive Algorithm

adjusting the wrong value of c to its right value very fast. It is important to point out that the speed of the adaptive process is influenced by the verification period, that has to be small. For example, if we check the budget curve in each 50% of the simulation time, and use the initial value of c as being 4 for this scenario, all the available budget should be finished at the first verification.

In Figure 6.14, we have the results using a budget of size 4. The wrong choice of $c = 1.001$ is not undone very fast due to the conservative behavior of increasing less the value of c at the beginning. Thus, when we under estimate the value of c, the adaptive process is not so efficient in leading the value of c to its correct value. In such situation, the error gets bigger until the adaptive process increases the value of c.

Figure 6.15 shows the results for a budget of size 10 packets. As in the previous scenario, when the initial value of the constant c is smaller than the correct value, the error gets bigger

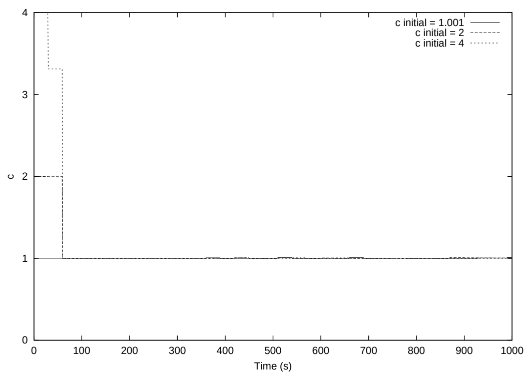
$\frac{budget}{simulationTime}$	Initial Value of c
≤ 0.001	1.001
> 0.001 and ≤ 0.002	1.01
> 0.002 and ≤ 0.004	1.5
> 0.004 and ≤ 0.008	2
> 0.008	4

Table 6.2: Initial value of constant c.

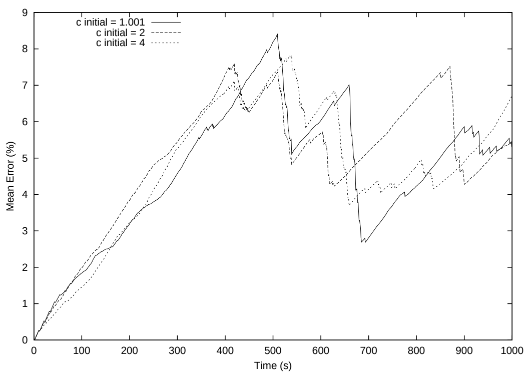
and the assignment of the right value of c is delayed.

Due to the conservative restriction of increasing less the value of c and decreasing more at the beginning of simulation, the over estimation in the value of c is better supported than the under estimation. Even though, it is better to find a way of choosing a good initial value of c in order to minimize these problems.

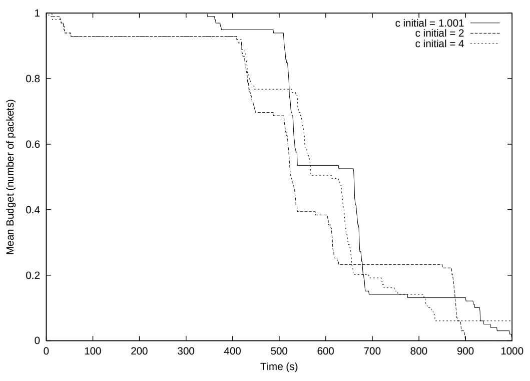
In the following simulations, we choose five default values to be the initial value of c. This choice depends on the value of budget and on the simulation time. The value of c is proportional to $\frac{budget}{simulationTime}$. Table 6.2 shows the chosen initial values.



(a) Value of constant c.

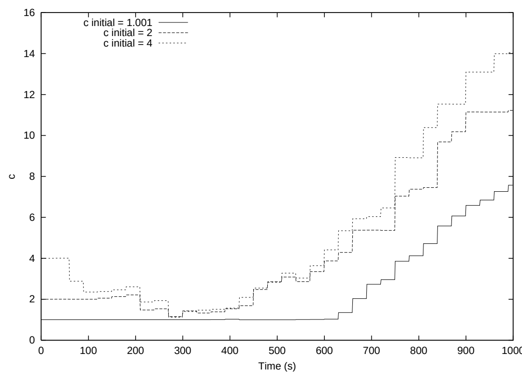


(b) Mean Error (%).

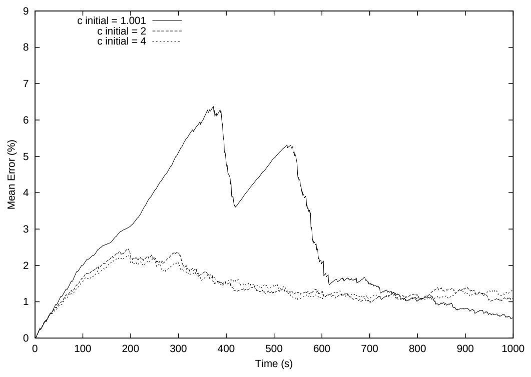


(c) Mean Budget (number of packets).

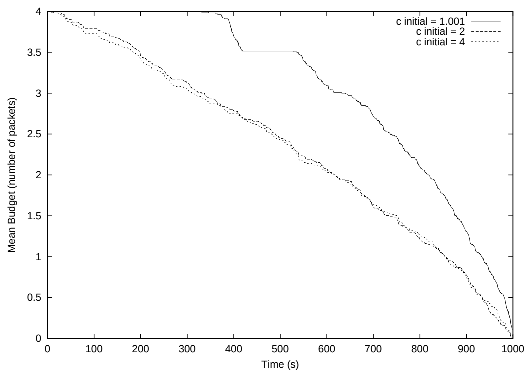
Figure 6.13: Changing the initial value of c when the budget size is 1 packet, $k = \log_c 1000$ and $\lambda = 0.1$.



(a) Value of constant c.

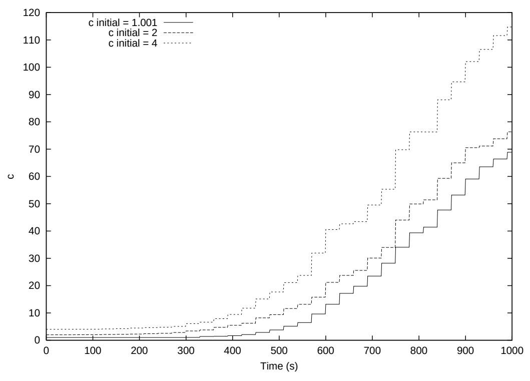


(b) Mean Error (%).

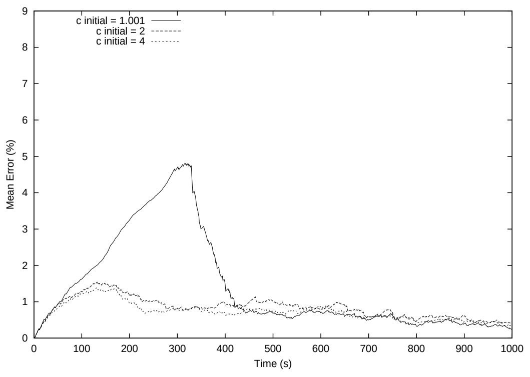


(c) Mean Budget (number of packets).

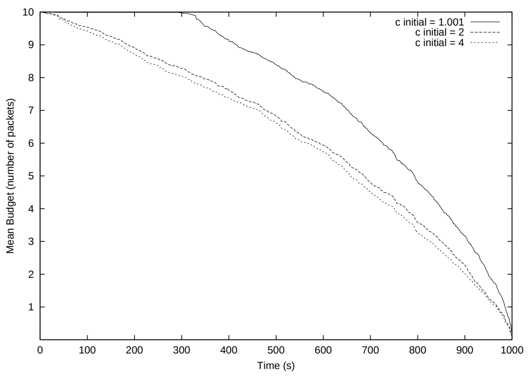
Figure 6.14: Changing the initial value of c when the budget size is 4 packets, $k = \log_c 1000$ and $\lambda = 0.1$.



(a) Value of constant c.



(b) Mean Error (%).



(c) Mean Budget (number of packets).

Figure 6.15: Changing the initial value of c when the budget size is 10 packets, $k = \log_c 1000$ and $\lambda = 0.1$.

6.5 Simulation Results for the Adaptive Energy Map Construction

The goal of this section is to analyze the adaptive process when the initial value described in the last section is used. To this end, we change the simulation time and the size of the budget, and plot the error and the available budget in each second of simulation.

Figure 6.16 shows the results for a simulation of 500 seconds using 1, 3, 5 and 7 packets. For all sizes of budget, the error is almost constant during all the simulation time, and the budget finishes at the end of simulation, meaning that the adaptive process achieved a good performance in these scenarios.

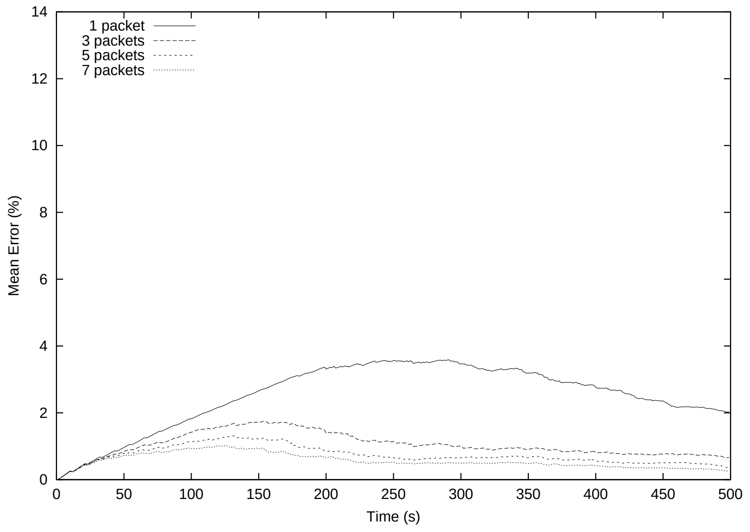
If we take a look at Figure 6.16-b, it is possible to see that the budget curves are above the straight line defined by equation (6.8). We can see this in Figure 6.17 in which this equation is plotted together with the curve of budget of size 7 of Figure 6.16-b. The fact that the budget curve is above this function means that the adaptive process is achieving a conservative behavior of saving budget at the beginning of simulation to be spent at the end.

$$y = -\frac{budget}{simulationTime}\, x + budget \qquad (6.8)$$

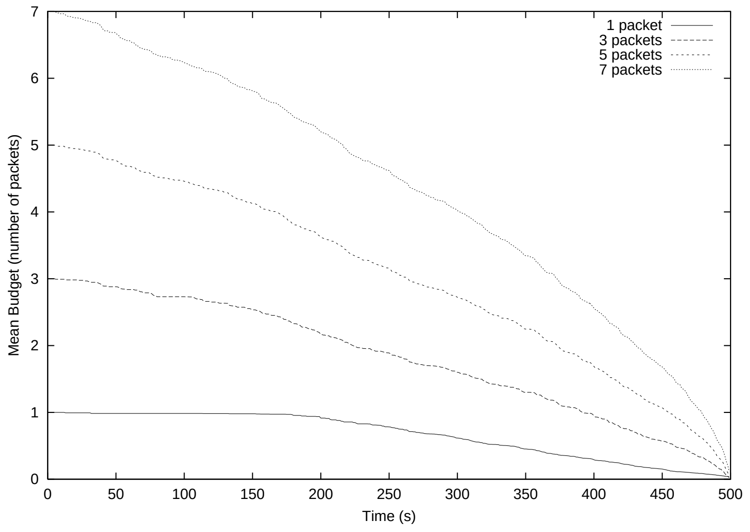
Figure 6.18 shows the results for 1000 second simulation. In this case, the error curve is not so constant like in the previous case, but the budget curve is still well used during all simulation.

Figure 6.19 illustrates the results for a simulation period of 1500 seconds. We can see that even when each node can use only 1 packet during all simulation, the use of the budget is distributed during all the simulation time. The same can be seen, in Figure 6.20, in which a simulation period of 2000 seconds is analyzed. In all cases, the budget is used uniformly during the simulation time in order to keep the error almost constant.

Next, we analyzed the influence of changing the parameters of the model that describes the events arrival: parameter λ of the Poisson process and a of the Pareto distribution. As explained in the previous chapters, changing the value of these parameters makes only a small difference in the prediction-based energy map construction. As these changes do not influence too much the energy map construction under a finite energy budget, the results for these



(a) Mean Error (%).



(b) Mean Budget (number of packets).

Figure 6.16: Changing the budget size in a 500 second simulation, $k = \log_c 1000$ and $\lambda = 0.1$.

simulations are shown in Appendix B.

6.6 Concluding Remarks

It is important to point out that the adaptive process proposed in this chapter works fine for the sensor network analyzed in this work. Other kinds of sensor networks must be carefully

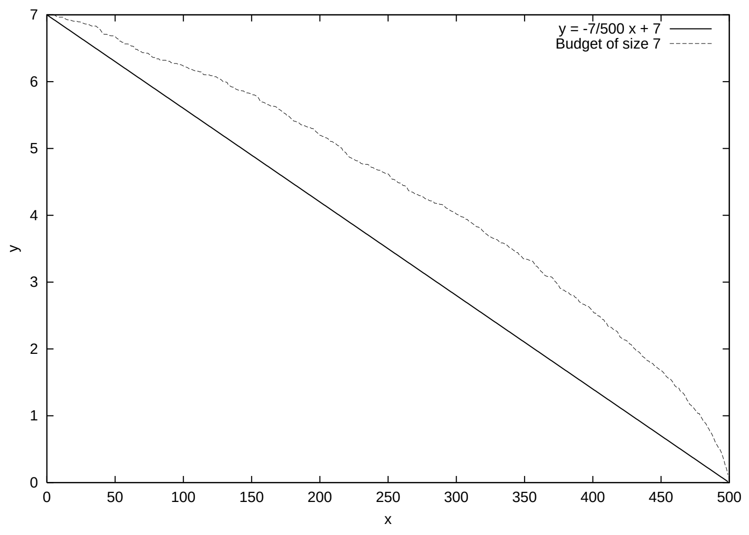
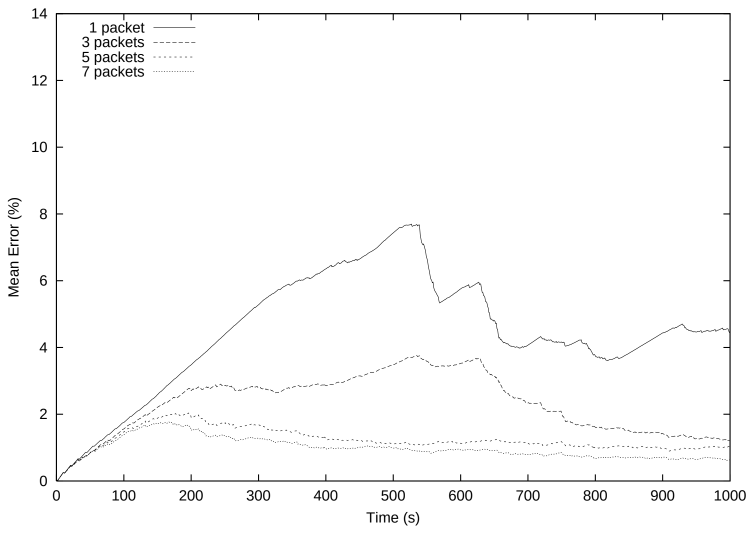
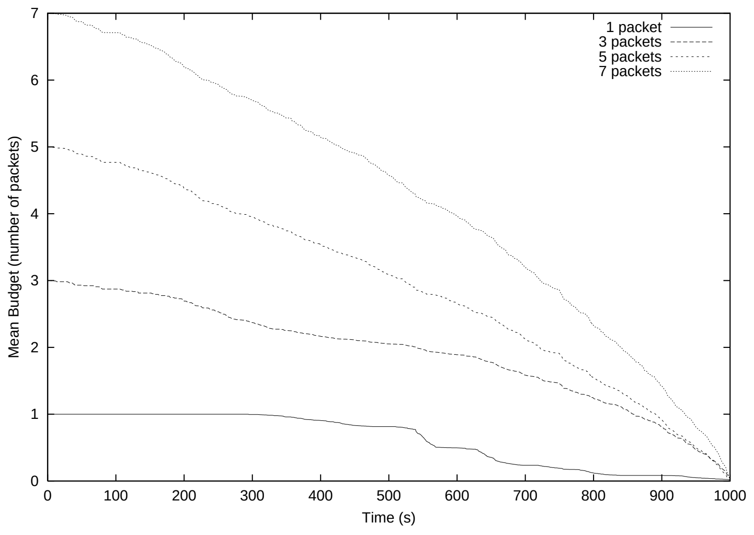


Figure 6.17: Function defined by equation (6.8) and the budget of size 7 of Figure 6.16.

studied. As example, the adaptive process is highly dependable on the amount of initial energy. The bigger this value, the more difficult to keep the percentage error constant. In other words, more budget will be necessary at the end of simulation to keep the percentage error constant, meaning that the budget curve should be more above the straight line $y = -\frac{budget}{simulationTime}\ x + budget$ than it was in simulations presented in this work. However, we support the idea that the three requirements described in Section 6.4 must be achieved by the adaptive processes for all kinds of wireless sensor networks. Consequently, the adaptive process should be guided by the three requirements, but it should also take into account the characteristics of sensor networks.

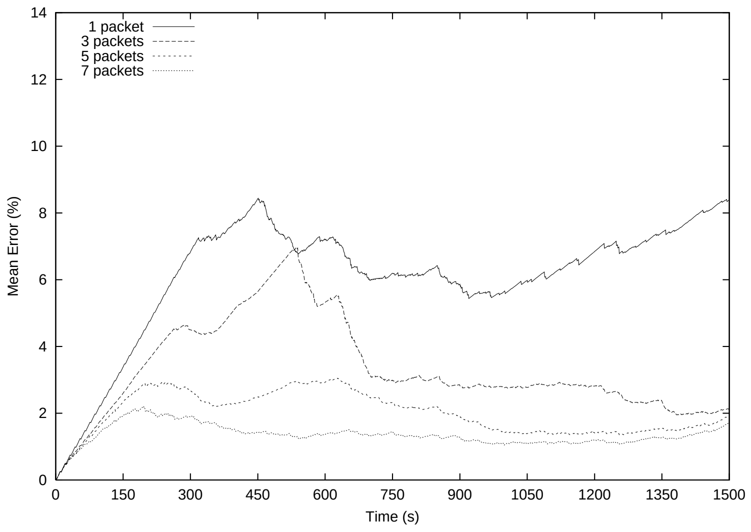


(a) Mean Error (%).

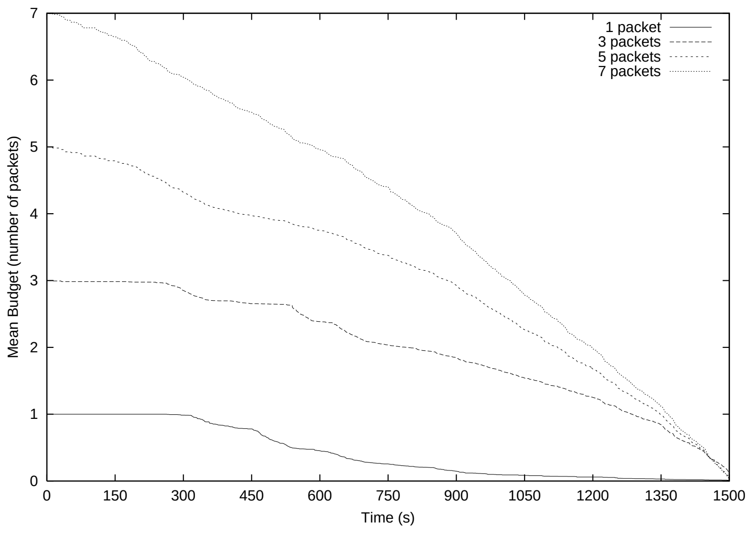


(b) Mean Budget (number of packets).

Figure 6.18: Changing the budget size in a 1000 second simulation, $k = \log_c 1000$ and $\lambda = 0.1$.

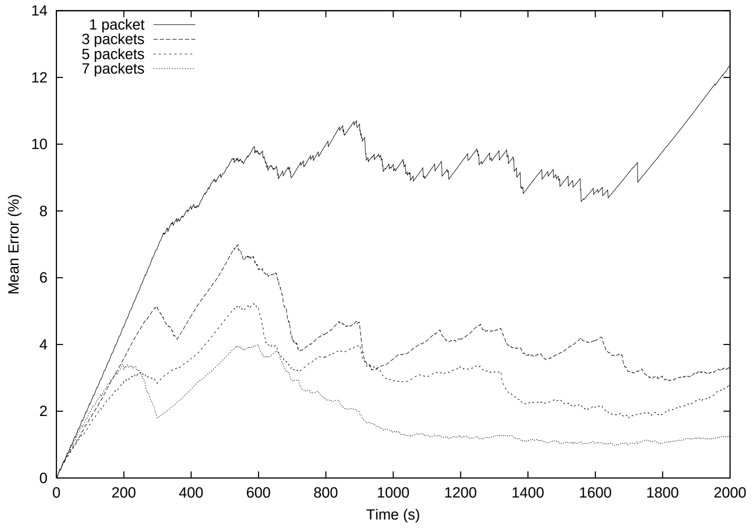


(a) Mean Error (%).

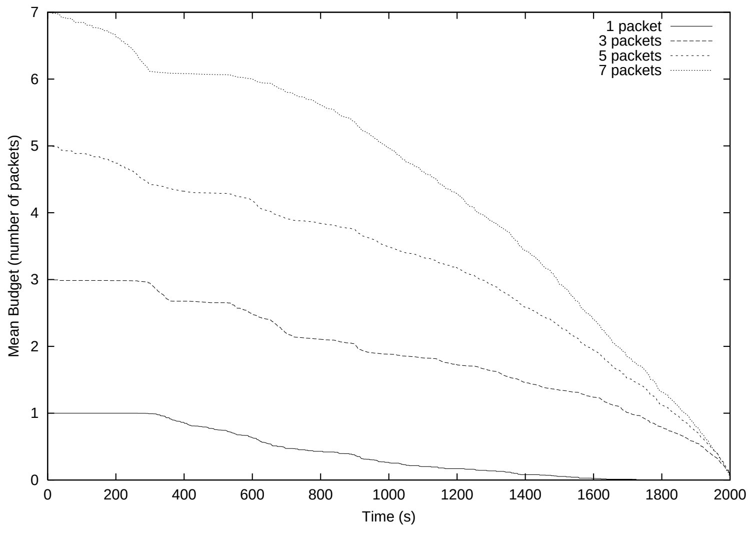


(b) Mean Budget (number of packets).

Figure 6.19: Changing the budget size in a 1500 second simulation, $k = \log_c 1000$ and $\lambda = 0.1$.



(a) Mean Error (%).



(b) Mean Budget (number of packets).

Figure 6.20: Changing the budget size in a 2000 second simulation, $k = \log_c 1000$ and $\lambda = 0.1$.

Chapter 7

Conclusions and Future Directions

7.1 Summary of Accomplished Work

In this work, we extended the state of the art in three main directions. Firstly, we proposed a State-based Energy Dissipation Model that represents the behavior of a sensor node in terms of energy consumption. Secondly, we proposed the use of prediction algorithms to construct the energy map of wireless sensor networks. We showed how sensor nodes can predict their energy consumption in order to construct a non-restrictive energy map. Our third contribution is the proposal of a new paradigm, named finite energy budget, to design solutions for wireless sensor networks. We also showed how this new paradigm can be applied in the construction of a restrictive energy map.

The main goal of this work was to construct the energy map of a wireless sensor network. In order to perform the simulation analysis, we presented a State-based Energy Dissipation Model that characterizes more realistically the behavior of a sensor network in terms of energy consumption. We also compared the SEDM with the Ideal model that represents the lower bound in terms of energy consumption. Using the SEDM, we evaluated the construction of prediction-based energy maps. In prediction-based energy maps, each node tries to estimate the amount of energy it will spend in near future, and it sends this information, along with its available energy, to the monitoring node. Initially, the energy map was constructed using a non-restrictive approach in such a way that no restriction regarding the amount of energy spent in its construction was defined. In this approach, the construction was performed according to a parameter that defines the accuracy in which the map should be constructed. Simulations were conducted in order to compare the performance of prediction-based approaches with a naive one, in which only the available energy is sent to the monitoring node. Results indicated that prediction-based approaches are more energy-efficient than the naive solution, and also that these approaches are more scalable with respect to the number of sensing events.

Using the non-restrictive energy map construction, we also evaluated the possibilities of using sampling techniques. The basic supposition, when using sampling, is that sensors in close proximity are likely to have correlated energy dissipation rate. Therefore, the energy information sent by a node can be used to update the energy consumption rate of its neighboring nodes. Results show that the best advantage of the sampling technique is to produce more constant error curves. However, it is important to point out that the use of sampling techniques is appropriate only in situations where neighboring nodes spend energy similarly. As an example, if sensor nodes are organized in classes with different energy consumption rates, the sampling model, as presented in this work, is not suitable.

We also investigated the possibility of constructing the energy map under a restrictive energy model. We presented a new model for constructing the energy map of wireless sensor networks under a finite energy budget. The energy budget was used in the context of defining the maximum number of packets each node could send with energy information to the monitoring node. We proposed a model to represent the probability in which a node sends an energy information packet, and an approach to adjust this probability in order to construct the best energy map under the given energy constraint. Simulation results indicate that we can approach the performance limits using the proposed finite energy budget model.

As discussed in this work, prediction-based techniques are a good approach to construct the energy map of wireless sensor networks. The approaches studied, with and without energy restriction, represent two different ways of dealing with this construction. Due to the paramount importance of energy conservation, we believe that the restrictive energy map construction is more suitable for most sensor network applications. Furthermore, we support the idea that the finite energy budget model, proposed in this work, should be extended to other network activities.

7.2 Future Directions

The work presented in this thesis can be extended in a variety of directions. The first one is to improve the SEDM in order to produce more precise models. In the SEDM, sensor nodes are modeled as event-driven dynamic systems in such a way that only the discrete events are considered. We intend to extend the SEDM proposing a hybrid model in which both continuous and discrete dynamics are modeled. We believe that representing the sensor nodes as hybrid systems can produce more accurate energy dissipation models.

As discussed in Section 5.6, we support the idea that the energy map should be used as a new protocol engineering principle when designing new protocols for wireless sensor networks. Using this idea, it would be interesting if the energy map was used in the Trajectory Based Forwarding protocol proposed in [Niculescu and Nath, 2003]. The information presented in the energy map could be used in order to plan the trajectory according to energy reserves, preserving or avoiding regions with small energy reserves.

Another facet of the energy map construction that we intend to investigate is its temporal aspect. The energy information that arrives at the monitoring node represents a past state. This information corresponds to the instant of time when its packet left the sensor node. Thus, the longer this information takes to arrive at the monitoring node, the more outdated it will be. More precise maps can be constructed if we take into account the temporal aspect of the energy information. Thus, we can construct more accurate energy maps if we estimate the time elapsed since the energy packet left the sensor node. As an example, we can make this estimation considering the geographical distance between nodes and the monitoring node or the number of hops traveled by the energy information packets.

In all energy map constructions presented in this work, the map of the entire network was constructed in the monitoring node. However, in some situations, it is enough to know the energy information of a neighboring region. An energy map that gives information about a region surrounding the node is named localized energy map. This localized energy information can be useful to improve the energy efficiency of other algorithms such as the routing protocols. The construction of localized energy map is a promising extension of this work.

In all simulation models used in this work, failures are not considered. The interpolation phase used in the sampling technique can be used in situations where failures can happen, improving the map quality through the estimation of lost information.

When we design a wireless sensor network or any other network that is battery powered, it would be interesting to determine the energy budget associated with each network activity. This is a new paradigm in the design of wireless sensor networks. The investigation of the use of this new paradigm in other network activities is a good extension of this work. As example, we can design mechanisms that disseminate information to the maximum number of nodes under the constraint that they can use only a determined amount of energy. Another possibility is the design of network management functions to achieve their best performance using only a finite amount of energy.

Bibliography

[Asada et al., 1998] Asada, G., Dong, T., Lin, F., Pottie, G., Kaiser, W., and Marcy, H. (1998). Wireless integrated network sensors: Low power systems on a chip. In *European Solid State Circuits Conference*, The Hague, Netherlands.

[Basagni et al., 1998] Basagni, S., Chlamtac, I., and Syrotiuk, V. R. (1998). A distance routing effect algorithm for mobility (DREAM). In *Mobicom'98*, pages 76–84, Dallas Texas.

[Bhatnagar et al., 2001] Bhatnagar, S., Deb, B., and Nath, B. (2001). Service differentiation in sensor networks. In *The Fourth International Symposium on Wireless Personal Multimedia Communications.*

[Box and Jenkins, 1976] Box, G. E. P. and Jenkins, G. M. (1976). *Time series analysis: forecasting and control.* San Francisco: Holden-Day.

[Braginsky and Estrin, 2002] Braginsky, D. and Estrin, D. (2002). Rumor routing algorithm for sensor networks. In *The First ACM International Workshop on Wireless Sensor Networks and Applications - WSNA 2002*, Westin Peachtree Plaza, Atlanta, GA, USA.

[Brockwell and Davis, 2002] Brockwell, P. J. and Davis, R. A. (2002). *Introduction to time series and forecasting.* New York : Springer, 2nd edition.

[Bult et al., 1996] Bult, K., Burstein, A., Chang, D., Dong, M., Fielding, M., Kruglick, E., Ho, J., Lin, F., Lin, T., Kaiser, W., Marcy, H., Mukai, R., Nelson, P., Newburg, F., Pister, K., Pottie, G., Sanchez, H., Sohrabi, K., Stafsudd, O., Tan, K., Yung, G., Xue, S., and Yao, J. (1996). Low power systems for wireless microsensors. In *Proceedings of the 1996 International Symposium on Low Power Electronics and Design*, pages 17–22, Monterey, CA, USA.

[Bulusu et al., 2001a] Bulusu, N., Estrin, D., Girod, L., and Heidemann, J. (2001a). Scalable coordination for wireless sensor networks: Self-configuring localization systems. In *International Symposium on Communication Theory and Applications - ISCTA*, Ambleside, Lake District, UK.

[Bulusu et al., 2000] Bulusu, N., Heidemann, J., and Estrin, D. (2000). GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications*, 7:28–34.

[Bulusu et al., 2001b] Bulusu, N., Heidemann, J., and Estrin, D. (2001b). Adaptive beacon placement. In *Proceedings of the 21st International Conference on Distributed Computing Systems - ICDCS-21*, pages 489–498, Phoenix, Arizona, USA.

[Calvo et al., 2000] Calvo, R., Navone, H., and Ceccatto, H. (2000). *Southern Hemisphere Paleo- and Neoclimates: Key Sites, Methods, Data & Models*, chapter Neural network analysis of time series: Applications to climatic data. Springer-Verlag.

[Câmara and Loureiro, 2000] Câmara, D. and Loureiro, A. A. (2000). A GPS/Ant-like routing algorithm for ad hoc networks. In *IEEE Wireless Communications and Networking Conference*, pages 1232–1237, Chicago, IL, USA.

[Cover and Thomas, 1991] Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. John Willey & Sons, INC.

[Deb et al., 2002] Deb, B., Bhatnagar, S., and Nath, B. (2002). A topology discovery algorithm for sensor networks with applications to network management. In *IEEE CAS workshop*.

[Deb et al., 2003] Deb, B., Bhatnagar, S., and Nath, B. (2003). Stream: Sensor topology retrieval at multiple resolutions. *Journal of Telecommunications, Kluwer Publications, Special Issue on Wireless Sensor Networks*.

[Doherty et al., 2001a] Doherty, L., Pister, K. S., and Ghaoui, L. E. (2001a). Convex position estimation in wireless sensor networks. In *Proc. of IEEE Infocom 2001*.

[Doherty et al., 2001b] Doherty, L., Warneke, B., Boser, B., and Pister, K. (2001b). Energy and performance considerations for smart dust. *International Journal of Parallel Distributed Systems and Networks*, 4(3):121–133.

[Elson and Estrin, 2000] Elson, J. and Estrin, D. (2000). An address-free architecture for dynamic sensor networks. Technical Report 00-724, Computer Science Department USC.

[Elson and Estrin, 2001a] Elson, J. and Estrin, D. (2001a). Random, ephemeral transaction identifiers in dynamic sensor networks. In *Proceedings of the 21st International Conference on Distributed Computing Systems - ICDCS-21*, Phoenix, Arizona, USA.

[Elson and Estrin, 2001b] Elson, J. and Estrin, D. (2001b). Time synchronization for wireless sensor networks. In *Proceedings of the 2001 International Parallel and Distributed Processing Symposium - IPDPS, Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Boston MA USA.

[Estrin et al., 2001] Estrin, D., Girod, L., Pottie, G., and Srivastava, M. (2001). Instrumenting the world with wireless sensor networks. In *International Conference on Acoustics, Speech, and Signal Processing - ICASSP*, Salt Lake City, Utah.

[Estrin et al., 1999] Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. (1999). Next century challenges: scalable coordination in sensor networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, Seattle, WA USA.

[Fuqua, 2002] Fuqua (2002). School of business. Forecasting. http://www.duke.edu/ rnau/411home.htm.

[Ganesan et al., 2001] Ganesan, D., Govindan, R., Shenker, S., and Estrin, D. (2001). Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM Mobile Computing and Communications Review*, 5(4).

[Ge, 1998] Ge, X. (1998). Pattern Matching in Financial Time Series Data. http://www.datalab.uci.edu/people/xge/chart/index.html.

[G.Pollastri et al., 2001] G.Pollastri, P.Baldi, P.Fariselli, and R.Casadio (2001). Improved Prediction of the Number of Residue Contacts in Proteins by Recurrent Neural Networks. *Bioinformatics*, 17:S234–S242.

[Heinzelman et al., 2000] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proccedings of the Hawaii Conference on System Sciences*.

[Heinzelman, 2000] Heinzelman, W. B. (2000). *Application-Specific Protocol Architectures for Wireless Networks*. PhD thesis, Massachusetts Institute of Technology.

[Heinzelman et al., 1999] Heinzelman, W. R., Kulik, J., and Balakrishnan, H. (1999). Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185, Seattle, WA USA.

[Hill et al., 2000] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. (2000). System architecture directions for networked sensors. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*.

[Holzmann, 1991] Holzmann, G. (1991). *Design and Validation of Computer Protocols*. Prentice-Hall Software Series. Prentice-Hall.

[Holzmann, 1992] Holzmann, G. (1992). Protocol Design: Redefining the State of the Art. *IEEE Software*, 9(1):17–22.

[Intanagonwiwat et al., 2002] Intanagonwiwat, C., Estrin, D., Govindan, R., and Heidemann, J. (2002). Impact of network density on data aggregation in wireless sensor networks. In *Proceedings of International Conference on Distributed Computing Systems - ICDCS*, Vienna, Austria.

[Intanagonwiwat et al., 2000] Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the sixth annual international conference on Mobile computing and networking*, pages 56–67, Boston, MA USA.

[Kahn et al., 2000] Kahn, J., Katz, R., and Pister, K. (2000). Emerging challenges: Mobile networking for “smart dust”. In *J. Comm. Networks*, pages 188–196.

[Kahn et al., 1999] Kahn, J. M., Katz, R. H., and Pister, K. S. J. (1999). Next century challenges: Mobile networking for smart dust. In *Proceedings of MOBICOM*, pages 271–278, Seattle.

[King, 1991] King, P. (1991). Formalization of Protocol Engineering Concepts. *IEEE Transactions on Computers*, 40(4):387–403.

[Kleinrock and Silvester, 1978] Kleinrock, L. and Silvester, J. (1978). Optimum transmission radii for packet radio networks or why six is a magic number. In *National Telecommunications Conference*, pages 4.3.1–4.3.5.

[Ko and Vaidya, 1998] Ko, Y. B. and Vaidya, N. H. (1998). Location-aided routing (LAR) in mobile ad hoc networks. In *Mobicom'98*, pages 66–75, Dallas Texas.

[Kulik et al., 1999] Kulik, J., Heinzelman, W. R., and Balakrishnan, H. (1999). Negotiation-based protocols for disseminating information in wireless sensor networks. In *ACM/IEEE Int. Conf. on Mobile Computing and Networking*, Seattle, WA.

[Li et al., 2003] Li, X., Wan, P., Wang, Y., and Frieder, O. (2003). Coverage in wireless ad-hoc sensor networks. *IEEE Transactions on Computers*, 52(6).

[Liu, 1989] Liu, M. (1989). Protocol Engineering. In Yovits, M. C., editor, *Advances in Computers*, volume 29, pages 79–195. Academic Press, San Diego, CA, USA.

[Loureiro et al., 2003] Loureiro, A. A. F., Nogueira, J. M. S., Ruiz, L. B., Mini, R. A. F., Nakamura, E. F., and Figueiredo, C. M. S. (2003). Mini curso: Redes de sensores sem fio. In *21^o Simpósio Brasileiro de Redes de Computadores - SBRC*, Natal, RN, Brasil.

[Manet, 2002] Manet (2002). Mobile ad-hoc networks - manet. URL:.

[Meguerdichian et al., 2001] Meguerdichian, S., Koushanfar, F., Qu, G., and Potkonjak, M. (2001). Exposure in wireless ad-hoc sensor networks. In *The seventh annual international conference on Mobile computing and networking 2001*, pages 139–150, Rome, Italy.

[Min et al., 2001] Min, R., Bhardwaj, M., Cho, S.-H., Sinha, A., Shih, E., Wang, A., and Chandrakasan, A. (2001). Low-power wireless sensor networks. In *VLSI Design*.

[Niculescu and Badrinath, 2001] Niculescu, D. and Badrinath, B. R. (2001). Ad hoc positioning system (APS). In *Symposium on Ad-Hoc Wireless Networks, GlobeCom*, San Antonio, Texas.

[Niculescu and Nath, 2003] Niculescu, D. and Nath, B. (2003). Trajectory-Based Forwarding and its Applications. In *Proceedings of the Ninth Annual International Conference on Mobile Computing and Networking*, San Diego, California.

[Nist, 2002] Nist (2002). Nist/sematech - e-handbook of statistical methods. http://www.itl.nist.gov/div898/handbook.

[ns2, 2002] ns2 (2002). The network simulator. http://www.isi.edu/nsnam/ns/index.html.

[Park et al., 2000] Park, S., Savvides, A., and Srivastava, M. B. (2000). SensorSim: a simulation framework for sensor networks. In *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 104–111, Boston, MA USA.

[Piatkowski, 1981] Piatkowski, T. (1981). An Engineering Discipline for Distributed Protocol Systems. In Rayner, D. and Hale, R., editors, *Proc. IFIP Workshop on Protocol Testing—Towards Proof?, Volume 1: Specification and Validation*, pages 177–215, Teddington, Middlesex, U.K.

[Pottie and Kaiser, 2000a] Pottie, G. and Kaiser, W. (2000a). Embedding the internet wireless integrated network sensors. In *Communications of the ACM*, volume 43, pages 51–58.

[Pottie and Kaiser, 2000b] Pottie, G. and Kaiser, W. (2000b). Wireless integrated network sensors. In *Communications of the ACM*, volume 43, pages 551–8.

[R-Project, 2002] R-Project (2002). The R project for statistical computing. http://www.r-project.org/.

[Rabaey et al., 2000] Rabaey, J. M., Ammer, M. J., da Silva Jr., J. L., Patel., D., and Roundy, S. (2000). Picoradio supports ad hoc ultra-low power wireless networking. *IEEE Computer*, 33(7).

[Raghunathan et al., 2002] Raghunathan, V., Schurgers, C., Park, S., and Srivastavar, M. (2002). Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50.

[Ross, 1998] Ross, S. (1998). *A First Course in Probability.* Prentice Hall, fifth edition.

[Ross, 1997] Ross, S. M. (1997). *Simulation - Statistical Modeling and Decision Science.* Academic Press, second edition.

[Royer and Toh, 1999] Royer, E. M. and Toh, C.-K. (1999). A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, pages 46–55.

[Sang and qi Li, 2000] Sang, A. and qi Li, S. (2000). A Predictability Analysis of Network Traffic. In *INFOCOM*, pages 342–351.

[Savvides et al., 2001] Savvides, A., Han, C.-C., and Strivastava, M. B. (2001). Dynamic fine-grained localization in ad-hoc networks of sensors. In *The seventh annual international conference on Mobile computing and networking 2001*, pages 166–179, Rome, Italy.

[Schwiebert et al., 2001] Schwiebert, L., Gupta, S. K., and Weinmann, J. (2001). Research challenges in wireless networks of biomedical sensors. In *The seventh annual international conference on Mobile computing and networking 2001*, pages 151–165, Rome, Italy.

[Servetto and Barrenechea, 2002] Servetto, S. and Barrenechea, G. (2002). Constrained random walks on random graphs: Routing algorithms for large scale wireless sensor networks. In *The First ACM International Workshop on Wireless Sensor Networks and Applications - WSNA*, Atlanta, Georgia, USA.

[Shih et al., 2001] Shih, E., Cho, S.-H., Ickes, N., Min, R., Sinha, A., Wang, A., and Chandrakasan, A. (2001). Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *The seventh annual international conference on Mobile computing and networking 2001*, pages 272–287, Rome, Italy.

[Sinha and Chandrakasan, 2001] Sinha, A. and Chandrakasan, A. (2001). Dynamic power management in wireless sensor networks. *IEEE Design & Test of Computers*, 18(2).

[Sohrabi et al., 2000] Sohrabi, K., Gao, J., Ailawadhi, V., and Pottie, G. (2000). Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7:16–27.

[Srivastava et al., 2001] Srivastava, M., Muntz, R., and Potkonjak, M. (2001). Smart kindergarten: Sensor-based wireless networks for smart developmental problem-solving environments. In *The seventh annual international conference on Mobile computing and networking 2001*, pages 132–138, Rome, Italy.

[StatSoft, 2002] StatSoft (2002). Inc. (2002). Electronic Statistics Textbook. Tulsa, OK: StatSoft. http://www.statsoft.com/textbook/stathome.html.

[Tang and Baker, 2000] Tang, D. and Baker, M. (2000). Analysis of a local-area wireless network. In *Mobicom 2000*, Boston MA USA.

[Tilak et al., 2002] Tilak, S., Abu-Ghazaleh, N. B., and Heinzelman, W. (2002). A taxonomy of wireless micro-sensor network models. *Mobile Computing and Communication Review*, 6(2).

[Woo and Culler, 2001] Woo, A. and Culler, D. E. (2001). A transmission control scheme for media access in sensor networks. In *The seventh annual international conference on Mobile computing and networking 2001*, pages 221–235, Rome, Italy.

[Xu et al., 2000] Xu, Y., Heidemann, J., and Estrin, D. (2000). Adaptive energy-conserving routing for multihop ad hoc networks. Research Report 527, USC/Information Sciences Institute.

[Ye et al., 2002] Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies - INFOCOM*, New York, NY, USA.

[Yu et al., 2001] Yu, Y., Govindan, R., and Estrin, D. (2001). Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Department Technical Report.

[Zhao et al., 2002] Zhao, Y. J., Govindan, R., and Estrin, D. (2002). Residual energy scans for monitoring wireless sensor networks. In *IEEE Wireless Communications and Networking Conference - WCNC'02*, Orlando, FL, USA.

Appendix A

ACF and PACF

A.1 Autocorrelation Functions

Autocorrelation functions are a commonly-used tool for checking randomness in a data set or the degree of dependence in the data. If the data are random, the autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero. Let $x_1, ..., x_n$ be observations of a time series. The sample mean of $x_1, ..., x_n$ is:

$$\bar{x} = \frac{1}{x} \sum_{t=1}^{n} x_t. \tag{A.1}$$

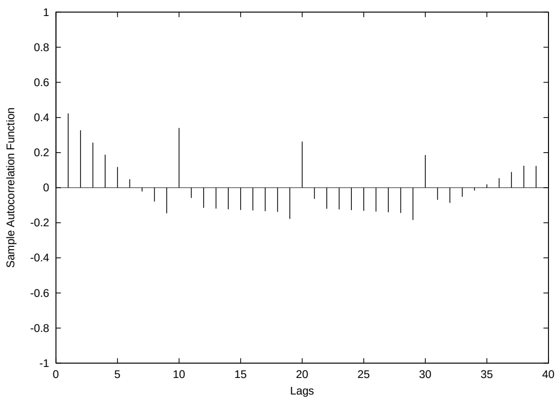
The sample autocovariance function in the lag h is defined as:

$$\hat{\gamma}(h) = n^{-1} \sum_{t=1}^{n-|h|} (x_{t+|h|} - \bar{x})(x_t - \bar{x}), \quad -n < h < n. \tag{A.2}$$

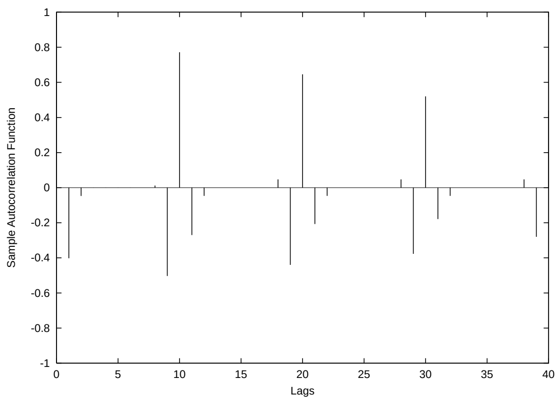
The sample autocorrelation function in the lag h is:

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}, \quad -n < h < n. \tag{A.3}$$

Figures A.1-a and A.1-b show the autocorrelation functions of the graphs of Figure 5.2-a and 5.2-b respectively.



(a) Autocorrelation function of the first difference.



(b) Autocorrelation function of the second difference.

Figure A.1: Autocorrelation functions of the time series of Figure 5.2.

A.2 Partial Autocorrelation Functions

The partial autocorrelation at lag h is the autocorrelation between X_t and X_{t-h} that is not accounted for by lags 1 through $h-1$. For any set of observations $x_1, ..., x_n$, the sample partial autocorrelation function, $\hat{\alpha}(h)$, is given by

$$\hat{\alpha}(0) = 1 \tag{A.4}$$

and

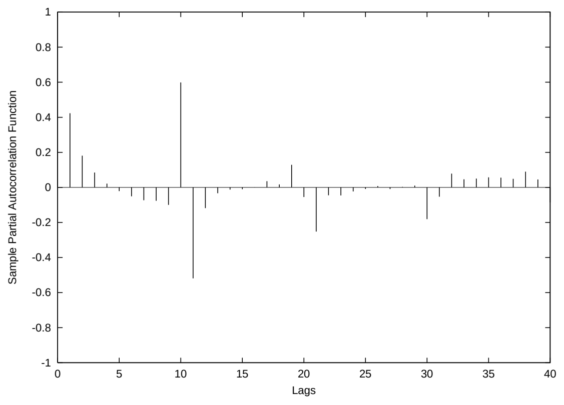
$$\hat{\alpha}(h) = \phi_{hh}, \qquad h \geq 1, \tag{A.5}$$

where ϕ_{hh} is the last component of

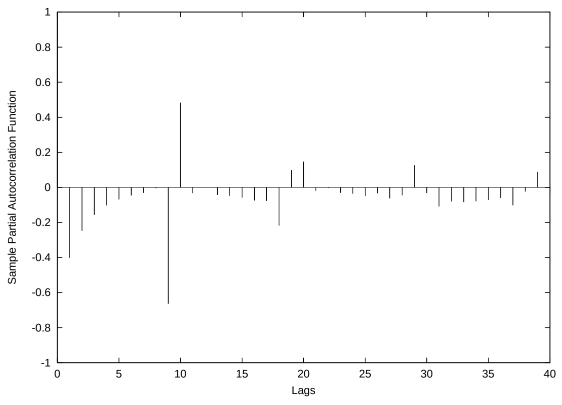
$$\phi_h = \Gamma_h^{-1} \Upsilon_h, \tag{A.6}$$

where $\Gamma_h = [\hat{\gamma}(i-j)]_{i,j=1}^{h}$ and $\Upsilon_h = [\hat{\gamma}(1), \hat{\gamma}(2), ..., \hat{\gamma}(h)]'$.

Figures A.1-a and A.1-b show the sample partial autocorrelation function of the graphs of Figure 5.2-a and 5.2-b respectively.



(a) Sample partial autocorrelation function of the first difference.



(b) Sample partial autocorrelation function of the second difference.

Figure A.2: Sample partial autocorrelation functions of the time series of Figure 5.2.

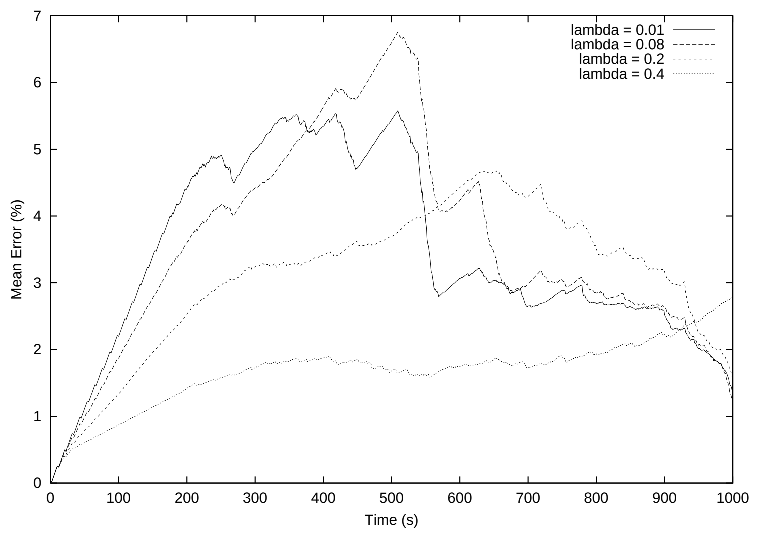
Appendix B

Changing the Number of Events in the Finite Energy Budget

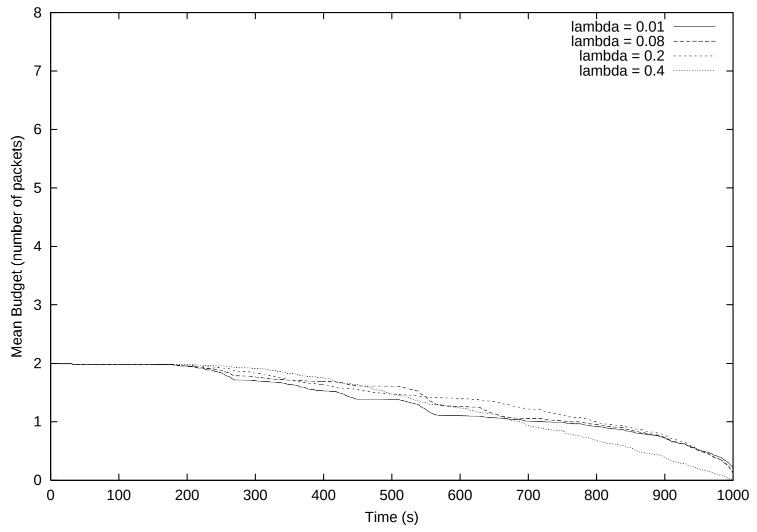
In this appendix, we analyze the influence of changing the parameters of the model that describes the events arrival, parameter λ of the Poisson process and a of the Pareto distribution, in the finite energy budget model.

B.1 Poisson Arrival Model

Figures B.1 to B.4 illustrate the results when we change the parameter λ in situations in which each sensor node can spend 2, 4, 6 and 8 packets to send its energy information, respectively. As we can see, changes in the value of λ make only a small difference in the prediction-based energy map construction under a finite energy budget.

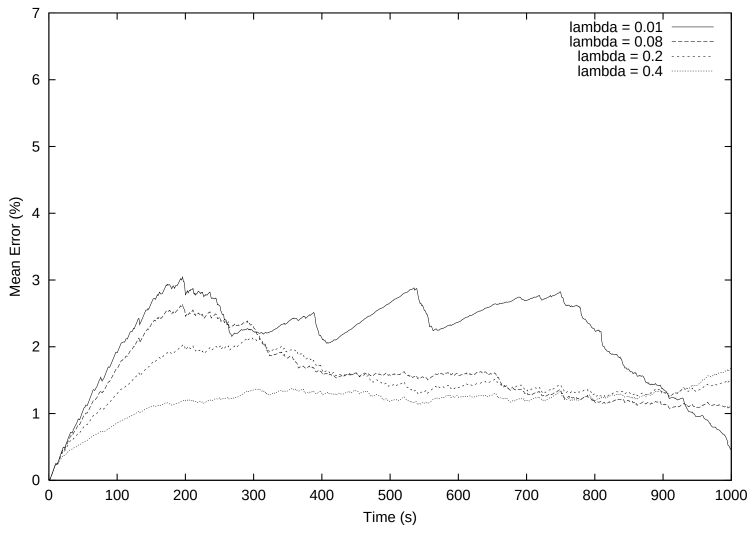


(a) Mean Error (%).

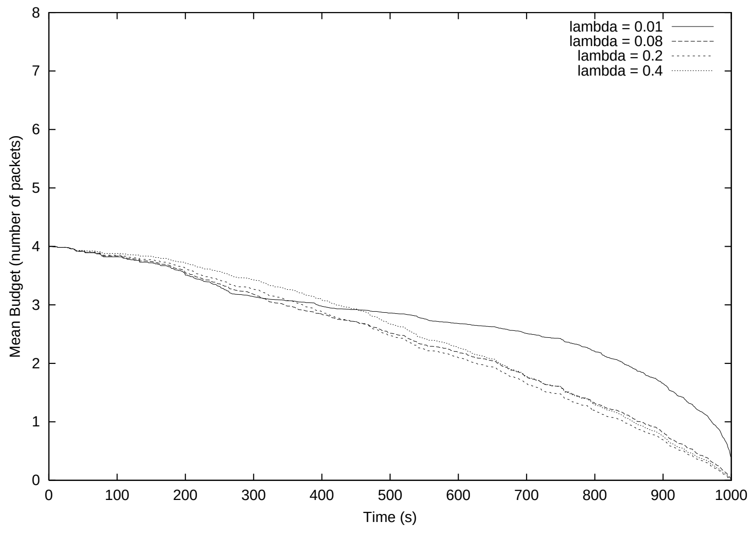


(b) Mean Budget (number of packets).

Figure B.1: Changing the value of λ when the number of packets is 2 in a 1000 second simulation, $k = \log_c 1000$.

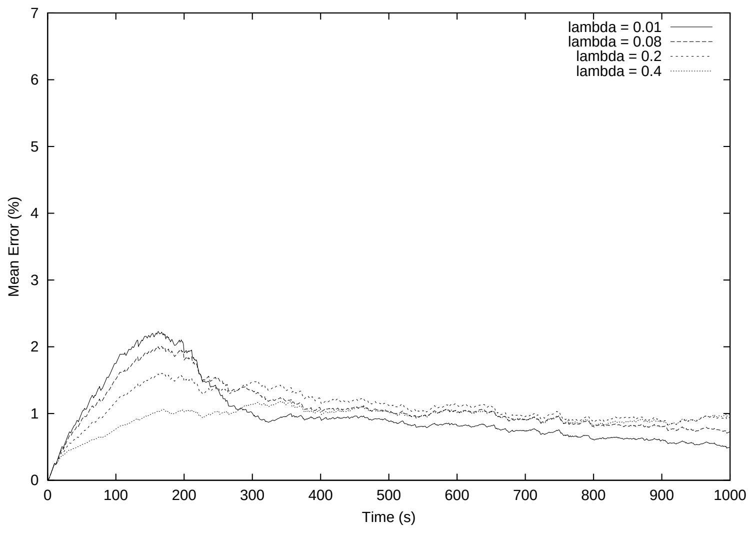


(a) Mean Error (%).

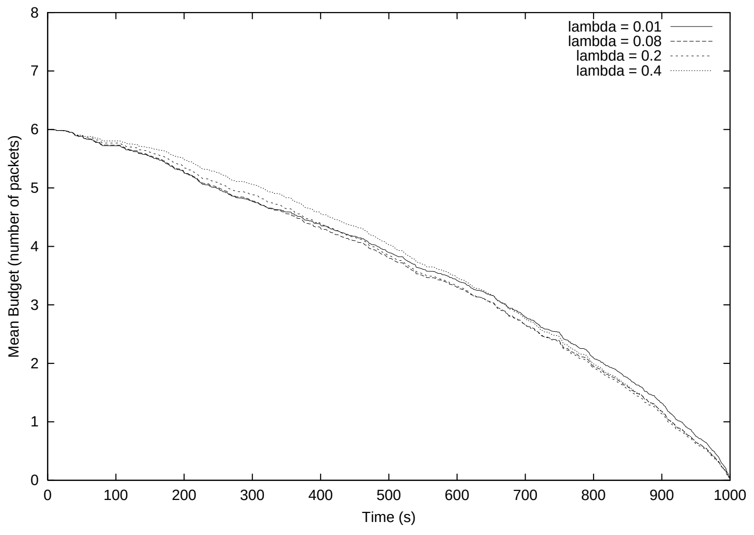


(b) Mean Budget (number of packets).

Figure B.2: Changing the value of λ when the number of packets is 4 in a 1000 second simulation, $k = \log_c 1000$.

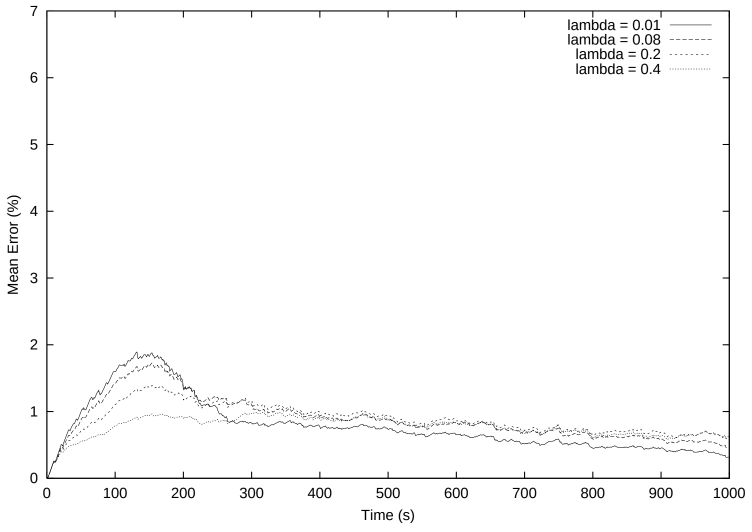


(a) Mean Error (%).

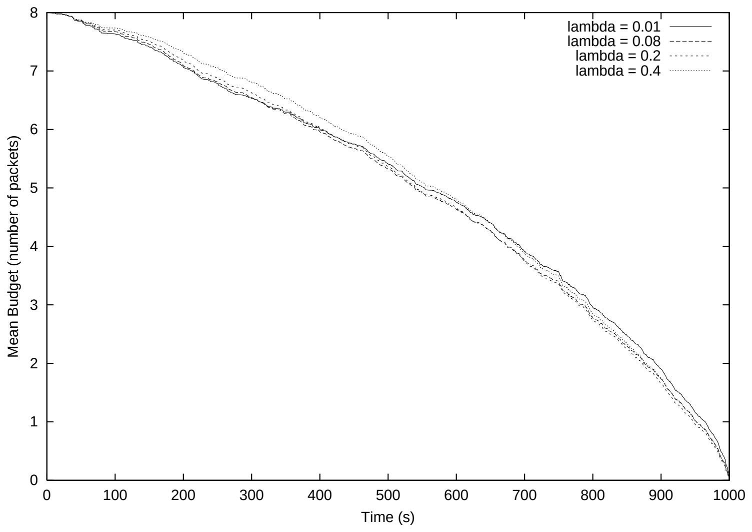


(b) Mean Budget (number of packets).

Figure B.3: Changing the value of λ when the number of packets is 6 in a 1000 second simulation, $k = \log_c 1000$.



(a) Mean Error (%).

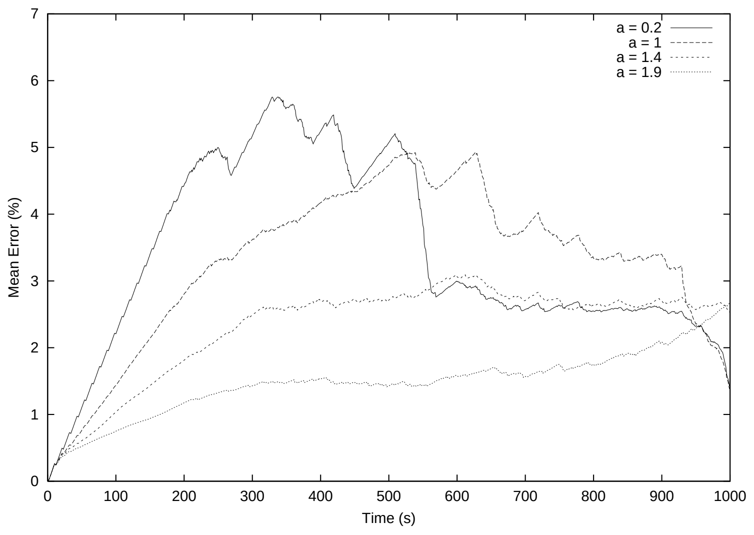


(b) Mean Budget (number of packets).

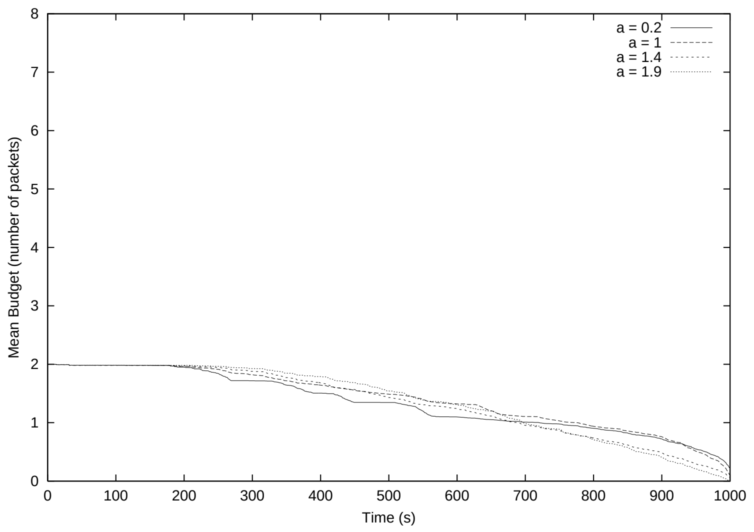
Figure B.4: Changing the value of λ when the number of packets is 8 in a 1000 second simulation, $k = \log_c 1000$.

B.2 Pareto Arrival Model

Figures B.5 to B.8 show the same results when we change the parameter a of the Pareto distribution. As in the Poisson process, changes in the value of a do not influence the energy map construction.

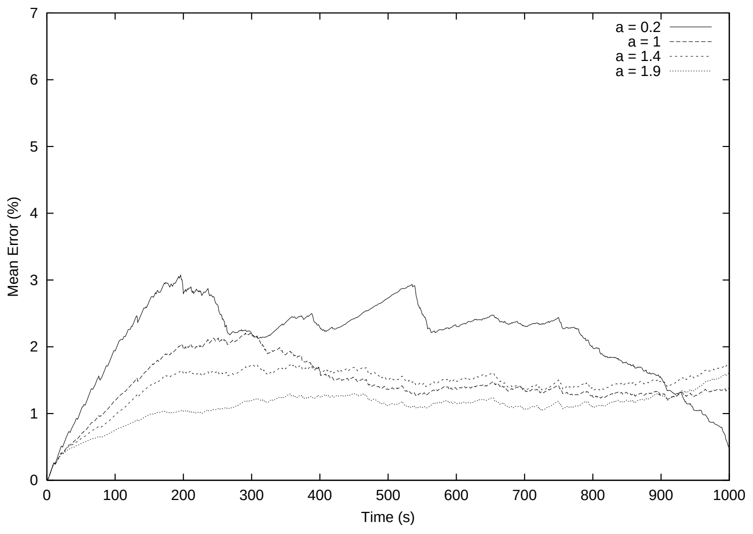


(a) Mean Error (%).

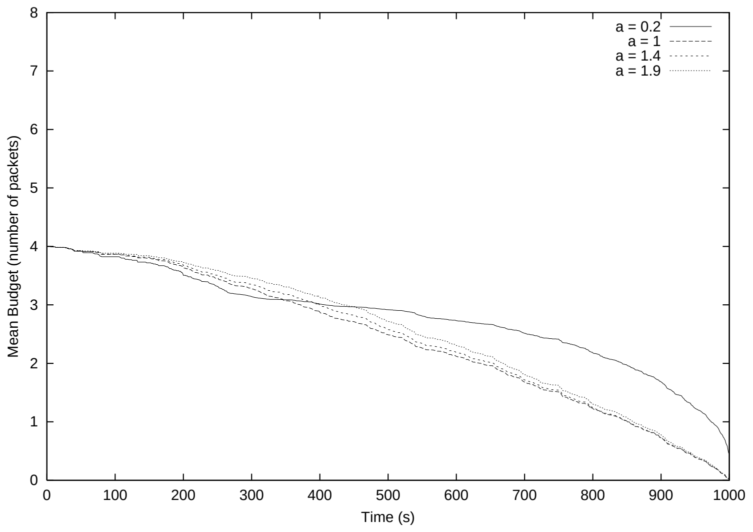


(b) Mean Budget (number of packets).

Figure B.5: Changing the value of a when the number of packets is 2 in a 1000 second simulation, $k = \log_c 1000$.

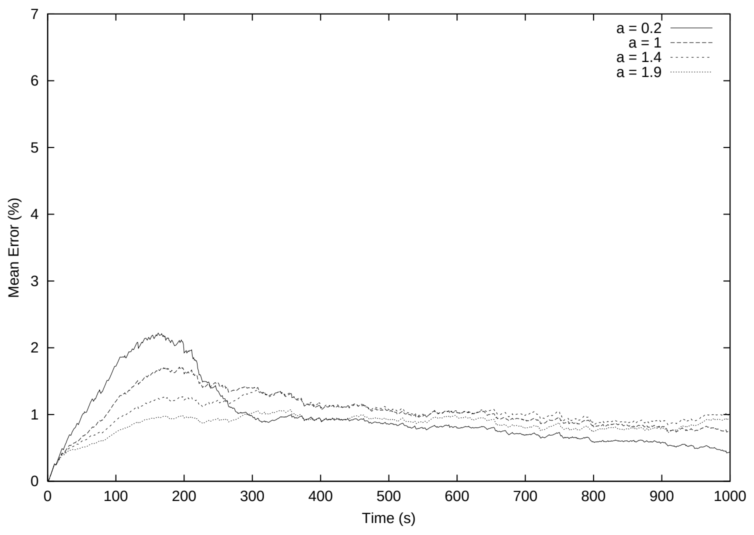


(a) Mean Error (%).

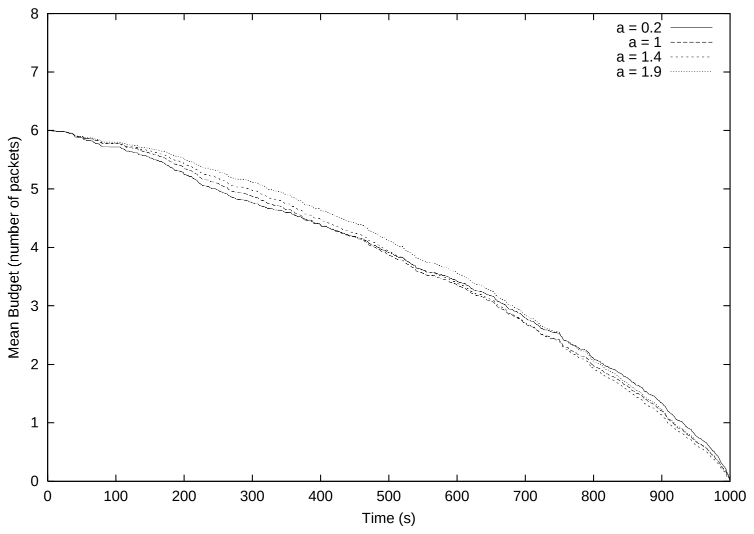


(b) Mean Budget (number of packets).

Figure B.6: Changing the value of a when the number of packets is 4 in a 1000 second simulation, $k = \log_c 1000$.

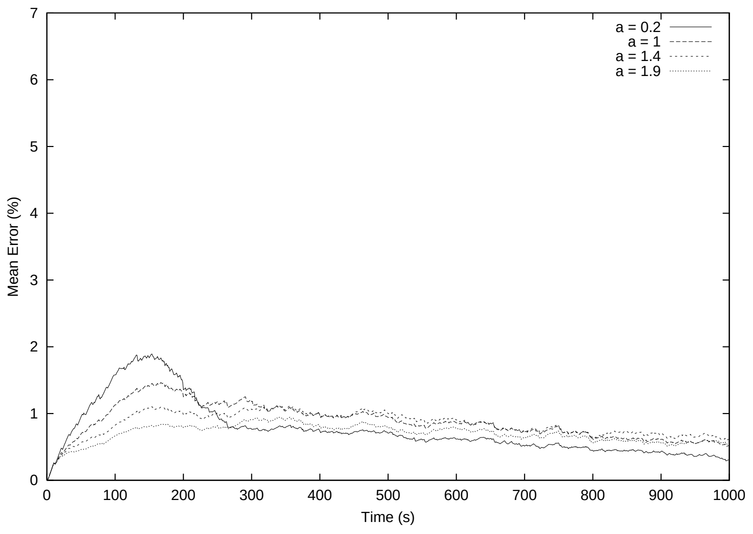


(a) Mean Error (%).

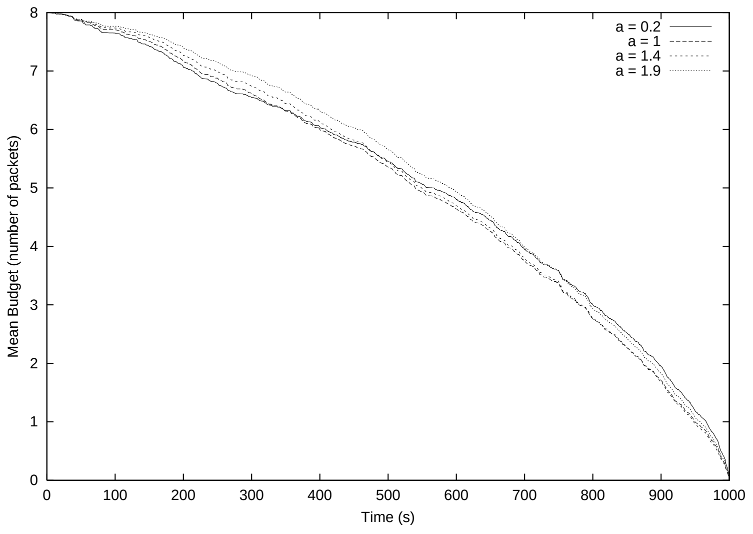


(b) Mean Budget (number of packets).

Figure B.7: Changing the value of a when the number of packets is 6 in a 1000 second simulation, $k = \log_c 1000$.



(a) Mean Error (%).



(b) Mean Budget (number of packets).

Figure B.8: Changing the value of a when the number of packets is 8 in a 1000 second simulation, $k = \log_c 1000$.