ANDRÉ LUIZ LINS DE AQUINO

REDUÇÃO DE DADOS EM REDES DE SENSORES SEM FIO BASEADA EM STREAM DE DADOS

Belo Horizonte

22 Fevereiro de 2008

ANDRÉ LUIZ LINS DE AQUINO

REDUÇÃO DE DADOS EM REDES DE SENSORES SEM FIO BASEADA EM STREAM DE DADOS

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

Orientador: Prof. Dr. Antônio Otávio Fernandes

UNIVERSIDADE FEDERAL DE MINAS GERAIS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Belo Horizonte

22 Fevereiro de 2008

TERMO DE APROVAÇÃO

ANDRÉ LUIZ LINS DE AQUINO

REDUÇÃO DE DADOS EM REDES DE SENSORES SEM FIO BASEADA EM STREAM DE DADOS

Tese defendida e aprovada pela banca examinadora constituída pelos(as) Senhores(as):

Prof. Dr. Antônio Otávio Fernandes – Orientador
Universidade Federal de Minas Gerais

Prof. Dr. Djamel Fawzi Hadj Sadok
Universidade Federal de Pernambuco

Profa. Dra. Luciana Salete Buriol
Universidade Federal do Rio Grande do Sul

Dr. Eduardo Freire Nakamura
Fundação Centro de Análise, Pesquisa e Inovação Tecnológica

Prof. Dr. Antônio Alfredo Ferreira Loureiro
Universidade Federal de Minas Gerais

Prof. Dr. Claudionor José Nunes Coelho Jr.
Universidade Federal de Minas Gerais

Belo Horizonte, 22 Fevereiro de 2008

AGRADECIMENTOS

A Deus,

A mim,

Aos meus pais,

Ao meu amor e

A todos os meus companheiros nessa jornada.

Os homens conseguiriam muito mais coisas se julgassem menos coisas impossíveis.

"*OGK – Only God Knows.*" (−−L)

RESUMO

O mundo ao nosso redor possui uma variedade de fenômenos que podem ser descritos por algumas grandezas como temperatura, pressão e umidade, que podem ser monitorados por dispositivos com poder de sensoriamento, processamento e comunicação. O conjunto desses dispositivos, trabalhando de forma cooperativa, é conhecido como rede de sensores sem fio. Cada um desses dispositivos, chamados nó sensor, tem a capacidade de monitorar um ou mais fenômenos e reportá-los, através de uma comunicação sem fio, para um nó especial chamado de sorvedouro.

Essas redes, devido às características da aplicação, possuem restrições de energia, tempo de resposta e largura de banda. Especificamente no que diz respeito à largura de banda, enviar grandes quantidades de dados pode ser problemático pela quantidade de nós que acessarão o meio, causando atraso demasiado no tempo de resposta e, assim, invalidando os dados. Devido a essas restrições, é necessário adotar-se alguma estratégia para o tratamento dos dados a fim de reduzir ou selecionar apenas os dados mais relevantes para a aplicação.

Os fenômenos monitorados geram dados com algumas características (*online*, impreciso, com ruído e de tamanho moderado, i.e., grandes o suficiente para não poderem ser processados facilmente), que nos leva a defini-los como *stream* de dados. Para tal tipo de dados, encontramos algumas técnicas, como amostragem, histograma, janela deslizante e rascunho, que nos permitem efetuar o processamento e a redução do conjunto de grandezas que representam os fenômenos monitorados, de tal forma que os gastos na rede possam ser reduzidos.

Com isso, o problema geral tratado no nosso trabalho é efetuar a redução de dados em redes de sensores sem fio baseada nas técnicas de *stream* de dados de tal forma que seja possível economizar os recursos da rede sem comprometer a representatividade dos fenômenos monitorados. Como solução é proposta uma arquitetura para redução nas aplicações gerais, que possui uma API de redução baseada nas técnicas de *stream* de dados. Além disso, utilizamos essa arquitetura para modelar aplicações que necessitam efetuar a redução no momento do sensoriamento, através de um nó agregador, e durante o roteamento.

Os resultados revelam que é possível utilizar a nossa solução para as diferentes aplicações modeladas, uma vez que foi possível economizar recursos da rede sem perder a representatividade dos fenômenos monitorados. Especificamente, quando a arquitetura foi integrada à fase de roteamento em aplicações de tempo real vimos através dos resultados que na maioria dos cenários é possível atender aos prazos exigidos pela aplicação e ainda assim manter a representatividade dos fenômenos monitorados.

Palavras-chave: Redes de sensores sem fio, redução de dados e algoritmos de *stream* de dados.

ABSTRACT

IN the world there are a variety of phenomena, such as temperature, pressure, and humidity, which can be monitored by specific sensor devices with processing and communication power. These devices, working cooperatively, are known as wireless sensor networks. Each sensor node can monitor and report some phenomena to a special node called sink node, using a wireless communication.

Despite their potential applications, wireless sensor networks have particular features imposed by resource restrictions, such as low computational power, reduced bandwidth and especially limited power source. Specifically, when we have a lot of data to be sent, the reduced bandwidth problem is increased since more nodes will try to access the wireless medium generating a packet delay. Thus, some data reduction is necessary where only the data relevant to the application is used.

In wireless sensor networks, the monitored phenomena have data stream characteristics (online, imprecise, with noise, and of moderate size). To process a data stream there are some techniques such as sampling, histogram, sliding windows, and sketch. This techniques allow the data processing and reducing where the network requirements, like energy consumption and packet delay.

Thus, the general problem treated here is the stream-based data reduction in wireless sensor networks so the network resources are saved, but at the same time we are interested to have a minimum data quality that represents the monitored phenomena. Our solution, proposed for this problem is a generic architecture that can be applied to general applications. This architecture has an API that allows to apply data reduction techniques to stream-based applications in wireless sensor networks. We use this architecture to model some applications that need to reduce the data at the sensor nodes, cluster heads, or routing nodes.

The results show that it is possible to use our solution in general applications, leading to reduction in both energy consumption and packet delay without loosing the data representativeness. Furthermore, when the architecture is integrated to the routing phase on real-time applications the results show that it is possible to achieve the deadline and keep the information quality about the monitored phenomena.

Keywords: Wireless sensor networks, data reducing and data stream algorithms.

LISTA DE ALGORITMOS

1 Pseudo-código do algoritmo de amostragem. 62

2 Pseudo-código do algoritmo de rascunho. 65

3 Pseudo-código do algoritmo de redução para dados multivariados 66

4 Pseudo-código do algoritmo da fase de encaminhamento. 69

LISTA DE FIGURAS

1 Tipos de redes sem fio. 32

2 Estrutura do nó sensor com os quatro componentes principais e os três componentes opcionais. 32

3 Estrutura de uma rede de sensores considerando não só o nó sensor mas também os demais elementos básicos. 33

4 Técnica de janela deslizante para o tratamento do *stream* de dados. . . 40

5 Técnica de amostragem para o tratamento do *stream* de dados. 40

6 Informações extraídas do *stream* de dados pelos algoritmos de rascunho e histograma. 41

7 Função da distribuição acumulada para 256 valores. 44

8 Representação de um sistema de uma rede de sensores onde é mostrado o comportamento ideal ($\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow D^*$), sensoriado ($\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow D$) e reduzido ($\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow \mathscr{V}' \rightarrow D'$). 49

9 Onde utilizar a redução de dados. 53

10 Arquitetura OGK. 58

11 Passos utilizados para o processamento do *stream* no algoritmo OGK-*amostragem*. 61

12 Passos utilizados para o processamento do *stream* no algoritmo OGK-*rascunho*. 64

13 Passos utilizados para o processamento do *stream* no algoritmo OGK-*multivar*. 66

14 Passos utilizados para o processamento do *stream* no algoritmo OGK-*oráculo*. 68

15 Avaliação do comportamento da rede, considerando a média de energia consumida na rede ao reduzir dados univariados. 75

16 Avaliação do comportamento da rede, considerando a média do atraso do pacote ao reduzir dados univariados. 77

17 Avaliação do comportamento dos dados reduzidos, considerando o erro médio ao aplicar a regra R'_{dist} sobre os dados univariados. 78

18 Avaliação do comportamento dos dados reduzidos, considerando o erro médio ao aplicar a regra R'_{val} sobre os dados univariados. 79

19 Avaliação do comportamento da rede, considerando a média de energia consumida na rede ao reduzir dados multivariados. 84

20 Avaliação do comportamento da rede, considerando a média do atraso do *stream* ao reduzir dados multivariados. 84

21 Avaliação do comportamento dos dados reduzidos, considerando o erro médio ao avaliar a regra R'_{val} sobre os dados multivariados com diferentes tamanhos do item *stream*. 86

22 Avaliação do comportamento dos dados reduzidos, considerando o erro médio ao aplicar a regra R'_{val} sobre os dados multivariados com diferentes número de sensores. 87

23 Topologias de rede consideradas nas aplicações gerais. 90

24 Avaliação do comportamento da rede, considerando a média da energia consumida na rede ao reduzir dados nos nós líderes. 96

25 Avaliação do comportamento da rede, considerando a média do atraso do pacote ao reduzir dados nos nós líderes. 98

26 Cenário I: Valores mínimos para os prazos exigidos pelas aplicações. . 107

27 Cenário I: Atrasos identificados ao utilizar a metade dos prazos suportados pela rede. 108

28 Cenário I: Atrasos identificados ao utilizar atrasos gerados pelos nós roteadores. 110

29 Cenário II: Valores mínimos para os prazos exigidos pelas aplicações. . 112

30 Cenário II: Atrasos identificados ao utilizar a metade dos prazos suportados pela rede. 113

31 Cenário II: Erros identificados ao utilizar a metade dos prazos suportados pela rede. 114

32 Cenário II: Atrasos identificados ao utilizar atrasos gerados pelos nós roteadores. 115

33 Cenário II: Erros identificados ao utilizar atrasos gerados pelos nós roteadores. 116

LISTA DE TABELAS

1 Parâmetros de simulação para redução de dados univariados. 74

2 Parâmetros de simulação para redução de dados multivariados. 83

3 Erro médio em porcentagem da redução de dados multivariados utilizando a solução OGK. 87

4 Razão de bits transmitidos numa rede com 1024 nós. 93

5 Razão de bits transmitidos numa rede com 160 nós. 93

6 Parâmetros de simulação para redução de dados nas redes hierárquicas. 95

7 Parâmetros de simulação para redução de dados nas aplicações de tempo real. 106

8 Cenário I: Valores mínimos para os prazos exigidos pelas aplicações. . 107

9 Cenário I: Atrasos identificados ao utilizar a metade dos prazos suportados pela rede. 107

10 Cenário I: Razão entre os atrasos identificados e os prazos exigidos pela aplicação. 108

11 Cenário I: Porcentagem dos dados recebidos pelo sorvedouro, ao utilizar a metade dos prazos suportados pela rede. 109

12 Cenário I: Erro do teste KS identificado ao utilizar a metade dos prazos suportados pela rede. 109

13 Cenário I: Erro identificado nos valores dos dados ao utilizar a metade dos prazos suportados pela rede. 109

14 Cenário I: Atrasos identificados ao utilizar atrasos gerados pelos nós roteadores. 110

15 Cenário I: Porcentagem dos dados recebidos pelo sorvedouro ao utilizar atrasos gerados pelos nós roteadores. 111

16 Cenário I: Erro no teste KS identificado ao utilizar atrasos gerados pelos nós roteadores. 111

17 Cenário I: Erro nos valores dos dados identificado ao utilizar atrasos gerados pelos nós roteadores. 111

18 Cenário II: Valores mínimos para os prazos exigidos pelas aplicações. . 112

19 Cenário II: Atrasos identificados ao utilizar a metade dos prazos suportados pela rede. 113

20 Cenário II: Razão entre os atrasos identificados e os prazos exigidos pela aplicação. 114

21 Cenário II: Atrasos identificados ao utilizar atrasos gerados pelos nós roteadores. 115

SUMÁRIO

1 INTRODUÇÃO 27

1.1 Motivação 28

1.2 Descrição do problema, objetivos e contribuições 29

1.3 Organização do trabalho 30

2 FUNDAMENTOS TEÓRICOS E TRABALHOS RELACIONADOS 31

2.1 Redes de sensores sem fio 31

2.2 Algoritmos de *stream* de dados 38

2.3 Análise de componentes principais e estudo da qualidade dos dados reduzidos 41

2.3.1 Análise de componentes principais - PCA 41

2.3.2 Qualidade dos dados reduzidos 43

2.4 Trabalhos relacionados 44

2.4.1 *Stream* de dados em redes de sensores 44

2.4.2 Soluções para o processamento dos dados monitorados em redes de sensores 45

2.4.3 Redes de sensores sem fio hierárquicas 46

2.4.4 Soluções de roteamento em redes de sensores 47

2.4.5 Aplicações de tempo real em redes de sensores 47

3 PROBLEMA DE REDUÇÃO DE DADOS EM REDES DE SENSORES SEM FIO 49

4 ARQUITETURA PARA REDUÇÃO DE DADOS 57

4.1 Arquitetura OGK 57

4.2 Algoritmos de redução 61
4.2.1 OGK-*amostragem* 61
4.2.2 OGK-*rascunho* 64
4.2.3 OGK-*multivar* 65
4.2.4 OGK-*oráculo* 67
4.3 Conclusões parciais 68

5 REDUÇÃO DE DADOS NO SENSORIAMENTO 71
5.1 Redução de dados univariados 71
5.1.1 Avaliação do comportamento da rede 73
5.1.2 Avaliação do comportamento dos dados reduzidos 76
5.1.3 Comportamento da rede vs. comportamento dos dados reduzidos 79
5.2 Redução de dados multivariados 81
5.2.1 Avaliação do comportamento da rede 82
5.2.2 Avaliação do comportamento dos dados reduzidos 85
5.2.3 Comportamento da rede vs. comportamento dos dados reduzidos 87
5.2.4 Conclusões parciais 88

6 REDUÇÃO DE DADOS EM REDES HIERÁRQUICAS 89
6.1 Caracterização da redução em redes hierárquicas 89
6.2 Avaliação do comportamento da rede 93
6.3 Conclusões parciais 98

7 REDUÇÃO DE DADOS EM APLICAÇÕES DE TEMPO REAL 101
7.1 Caracterização da redução em aplicações de tempo real 101
7.2 Avaliação do comportamento da solução OGK em aplicações de tempo real 104
7.2.1 Cenário I – Redes sem tráfego 106

7.2.2 Cenário II – Redes com tráfego 111
7.3 Conclusões parciais 115

8 CONCLUSÃO 117

PUBLICAÇÕES

A seguir, é apresentada a lista de publicações obtidas durante o doutorado. As publicações relacionadas diretamente a esta tese estão marcadas com um asterisco (*).

1. AQUINO, A. L. L.; CABRAL, R. da S.; FERNANDES, A. O. Um algoritmo de redução de dados para aplicações de tempo real em redes de sensores sem fio. In: *26st Brazilian Symposium on Computer Networks (SBRC'08)*. Rio de Janeiro, Brazil: SBC, 2008. (*)

2. AQUINO, A. L. L. et al. Sensor stream reduction for clustered wireless sensor networks. In: *23rd ACM Symposium on Applied Computing 2008 (SAC'08)*. Fortaleza, Brazil: ACM, 2008. p. 2052–2056. (*)

3. ANDRADE, A. V. et al. Analysis of selection and crossover methods used by genetic algorithm-based heuristic to solve the lsp allocation problem in mpls networks under capacity constraints. In: *International Conference on Engineering Optimization (EngOpt'08)*. Rio de Janeiro, Brazil: Springer, 2008. p. 1–15.

4. AQUINO, A. L. L. et al. Data stream based algorithms for wireless sensor network applications. In: *21st IEEE International Conference on Advanced Information Networking and Applications (AINA'07)*. Niagara Falls, Canada: IEEE Computer Society, 2007. p. 869–876. (*)

5. AQUINO, A. L. L. et al. A sampling data stream algorithm for wireless sensor networks. In: *IEEE International Conference on Communications (ICC'07)*. Glasgow, Scotland: IEEE Computer Society, 2007. p. 3207–3212. (*)

6. AQUINO, A. L. L. et al. On the use data reduction algorithms for real-time wireless sensor networks. In: *IEEE Symposium On Computers and Communications (ISCC'07)*. Aveiro, Potugal: IEEE Computer Society, 2007. p. 583–588. (*)

7. FIGUEIREDO, C. M. S. et al. Um esquema de gerenciamento para redes de sensores sem fio auto-organizáveis: Atuando sobre regras locais. In: *25st Brazilian*

Symposium on Computer Networks (SBRC'07). Belém, PA, Brazil: SBC, 2007. p. 1–12.

8. GUIDONI, D. L. et al. Sistemas do tipo eixo-raio aplicados à redes de sensores sem fio modeladas como redes small world. In: *39th Brazilian Symposium on Operational Research (SBPO'07)*. Fortaleza, CE, Brasil: SOBRAPO, 2007. p. 1–12.

9. ARTIGUENAVE, F. et al. The tropical biominer project: Mining old sources for new drugs. *OMICS: A Journal of Integrative Biology*, v. 9, n. 2, p. 30–138, June 2005.

10. MENEZES, G. C. et al. Uma abordagem paralela para os problemas de cobertura e conectividade em redes de sensores sem fio. In: *37th Brazilian Symposium on Operational Research (SBPO'05)*. Gramado, RS, Brasil: SOBRAPO, 2005. p. 1–15.

1 INTRODUÇÃO

"*Dêem-me uma alavanca e um ponto de apoio e eu levantarei o mundo.*" (Arquimedes)

O mundo ao nosso redor possui uma variedade de fenômenos que podem ser descritos por algumas grandezas, como temperatura, pressão e umidade, que podem ser monitorados por dispositivos com poder de sensoriamento, processamento e comunicação. O conjunto desses dispositivos, trabalhando de forma cooperativa, é conhecido como rede de sensores sem fio (ESTRIN et al., 1999; AKYILDIZ et al., 2002; TILAK; ABU-GHAZALEH; HEINZELMAN, 2002; ARAMPATZIS; LYGEROS; MANESIS, 2005). Essas redes podem ter, além de nós sensores, elementos atuadores que interferem no meio monitorado, um ou mais sorvedouros que recebem os dados e os processam e os *gateways* que são responsáveis pela comunicação da rede de sensores com outras redes.

Cada nó sensor tem a capacidade de monitorar um ou mais fenômenos. Os dados que representam esses fenômenos monitorados podem ser classificados como: univariados ou multivariados. Dados univariados representam um único conjunto de valores de um mesmo fenômeno. Por exemplo, os dados monitorados por um nó que possui apenas um sensor de temperatura. Já os dados multivariados representam mais de um conjunto de valores de um mesmo fenômeno ou mais de um fenômeno. Por exemplo, os dados recebidos por um nó responsável por processar os dados monitorados por um conjunto de nós que possuem apenas um sensor de temperatura, ou os dados monitorados por um nó que possui simultaneamente os sensores de temperatura, pressão e umidade.

Os fenômenos monitorados são reportados, através de uma comunicação sem fio *ad-hoc* (ROYER; TOH, 1999), para o sorvedouro. Essa comunicação, devido às características da aplicação, possui restrições de energia, tempo de resposta e largura de banda. Especificamente no que diz respeito à largura de banda, enviar grandes quantidades de dados pode ser problemático pela quantidade de nós que terão que acessar o meio, podendo causar atraso demasiado no tempo de resposta e, assim, invalidando os dados. Além disso, um grande tráfego na rede degrada rapidamente

seu tempo de vida. Devido a essas restrições, é necessário adotar alguma estratégia para o tratamento dos dados a fim de reduzir ou selecionar apenas os dados mais relevantes que representam o fenômeno monitorado. Dentre as diversas abordagens para redução de dados em redes de sensores sem fio pode-se destacar:

- A agregação de dados que efetua a redução dos dados sensoriados seguindo alguma métrica exigida pela aplicação. Tem como objetivo principal diminuir o tráfego na rede independente da qualidade dos dados reduzidos (KRISHANAMACHARI; ESTRIN; WICKER, 2002; ZHU; PAPAVASSILIOU, 2004; SANTINI; ROMER, 2006).

- A técnica de amostragem adaptativa que, ao longo do tempo de vida da rede, modifica a forma de sensoriamento com o objetivo de propagar apenas a informação mais relevante para a aplicação. Caso os dados possuam características distintas essa técnica apresentará um nível de redução baixo (MARBINI; SACKS, 2003; GANESAN et al., 2004; CHEN; KNOW; CHOI, 2006).

- A redução de dados multivariados que utiliza métodos para estimar o comportamento dos dados multivariados, como sua correlação, permitindo que apenas as diferenças, na correlação dos dados observadas ao longo do tempo, sejam propagadas até o sorvedouro (SEO; KANG; RYU, 2005; LI; ZHANG, 2006).

1.1 Motivação

No contexto de redes de sensores sem fio, existem fenômenos monitorados que geram dados com algumas características que nos leva a defini-los como *stream* de dados (HENZINGER; RAQHAVAN; RAJAGOPALAN, 1998; BABCOCK et al., 2002; ELNAHRAWY, 2003; GOLAB; OZSU, 2003; MUTHUKRISHNAN, 2005). As características gerais de um *stream* de dados tradicional é que ele é obtido de forma *online*, pois é processado no momento da sua recepção; é ilimitado, pois o fenômeno monitorado está constantemente gerando dados; e a ordem de chegada não pode ser controlada. No entanto, ao utilizarmos *stream* de dados para representar os dados de sensoriamento, devemos considerar as diferenças entre *stream de sensoriamento* e o *stream tradicional*. O *stream de sensoriamento* representa um conjunto de amostras de uma determinada população, é impreciso, com ruído e de tamanho moderado. Já o *stream tradicional* possui como entrada a população inteira, os dados são exatos, sem

erros e com tamanho exageradamente grande, i.e., grandes o suficiente para não poderem ser processados (ELNAHRAWY, 2003). Por convenção utilizaremos em nosso texto apenas o termo *stream* de dados para representar os *stream de sensoriamento*.

Para possibilitar a utilização das informações presentes no *stream* de dados, por parte das aplicações, existe uma classe específica de algoritmos, chamada de algoritmos de *stream* de dados. Esses algoritmos podem ser baseados em diferentes técnicas, como por exemplo, amostragem, histograma, janela deslizante e rascunho (MUTHUKRISHNAN, 2005). A aplicação de cada uma dessas técnicas resulta na geração de dados aproximados aos originais, onde a fidelidade dos dados aproximados depende da forma de como os dados são processados.

1.2 Descrição do problema, objetivos e contribuições

O problema geral tratado neste trabalho é efetuar a redução de dados em redes de sensores sem fio. A redução é baseada nas técnicas de *stream* de dados de tal forma que seja possível economizar os recursos da rede sem comprometer a representatividade dos fenômenos monitorados. Com isso, o objetivo principal é mostrar que essas técnicas podem ser aplicadas a reduções em redes de sensores. Para isso, foi proposta uma arquitetura para redução de dados juntamente com uma API de redução que pode ser aplicada a diferentes cenários nessas redes. Além disso, aplicamos a nossa arquitetura em diferentes momentos em que é possível efetuar a redução de dados. Assim, as principais contribuição deste trabalho são:

- Uma arquitetura para redução de dados, chamada OGK – *On a Good Knowledge* (Sobre um bom conhecimento), que utiliza o conhecimento a respeito do *stream* de dados para escolher a solução de redução mais apropriada. Essa arquitetura pode ser utilizada em diversos cenários e aplicações de redes de sensores desde que os dados tenham características de *stream*.

- A disponibilização da API-OGK de redução utilizada para dar suporte a nossa arquitetura e que possibilita a redução de dados nas redes de sensores, desde simples reduções no momento do sensoriamento até reduções habilitadas de forma autônoma pela aplicação.

- A utilização da arquitetura em aplicações que exigem a redução de dados no momento do sensoriamento e através de um nó agregador, onde os algoritmos

disponíveis na API são utilizados para efetuar tal redução. Especificamente, para a redução de dados através de um nó agregador, utilizamos uma formulação matemática para comprovar que a utilização de nós agregadores através de uma rede hierárquica é mais eficiente do que propagar as informações monitoradas através de uma rede plana.

- A utilização da arquitetura OGK, embutida no roteamento, para reduzir os dados quando os prazos das aplicações de tempo real não puderem ser atendidos. Além disso, elaboramos uma formulação matemática para estimar o quanto o dado deve ser reduzido no momento do roteamento.

Os resultados, apresentados ao longo dos próximos capítulos, revelam que é possível utilizar a nossa solução para as diferentes aplicações modeladas, por exemplo, aplicações gerais em redes planas e hierárquicas e cenários com exigências de tempo real. Em todas as aplicações estudadas foi possível economizar recursos da rede (cerca de 90% de economia nos melhores casos) sem perder a representatividade dos fenômenos monitorados (um erro máximo de 20% nos piores casos). Especificamente, quando a arquitetura foi integrada à fase de roteamento em aplicações de tempo real vimos através dos resultados que na maioria dos cenários foi possível atender aos prazos exigidos pela aplicação, uma vez que consideramos aplicações de tempo real *soft*, e ainda assim manter a representatividade dos fenômenos monitorados.

1.3 Organização do trabalho

Este trabalho segue com o capítulo 2 onde discutimos os principais conceitos utilizados e apresentamos alguns trabalhos relacionados. No capítulo 3, apresentamos o problema de redução de dados baseada em *stream* de dados para redes de sensores sem fio. No capítulo 4, mostramos a arquitetura para redução de dados baseada em *stream* de dados e apresentaremos os algoritmos propostos para dar suporte a arquitetura. Nos capítulos 5, 6 e 7, falamos, respectivamente, das considerações e cenários utilizados para o caso de redução no momento do sensoriamento, através de um nó agregador e no momento do roteamento para dar suporte a aplicações de tempo real. No capítulo 8, concluímos o trabalho e apontaremos futuras direções.

2 FUNDAMENTOS TEÓRICOS E TRABALHOS RELACIONADOS

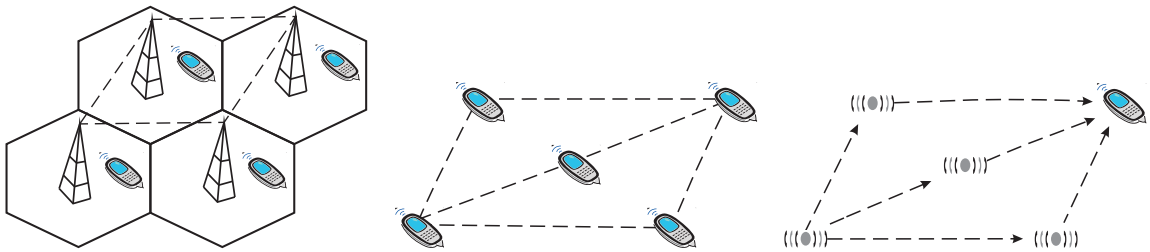
"*A ciência, como um todo, não é nada mais do que um refinamento do pensar diário.*" (Albert Einstein)

ESTE capítulo tem por objetivo apresentar os conceitos básicos necessários para um bom entendimento deste trabalho. Tais conceitos estão relacionados a redes de sensores sem fio, a *stream* de dados, ao método estatístico utilizado na redução de dados multivariados e aos mecanismos para a análise da qualidade dos dados quando reduzidos. Apresentamos apenas os conceitos mais gerais deixando aspectos específicos para serem explicados e referenciados quando necessário. Além disso, ao fim do capítulo apontamos alguns dos trabalhos relacionados.

2.1 Redes de sensores sem fio

Inicialmente, para melhor contextualizar as redes de sensores no ambiente sem fio, consideramos as redes estruturadas e *ad-hoc*. Em relação às redes estruturadas temos que elas possuem nós subordinados a uma estação base responsável pela comunicação entre os elementos da rede (figura 1(a)). Já as redes *ad-hoc* (ROYER; TOH, 1999) não utilizam uma estação base para prover a comunicação entre os elementos da rede, pois a comunicação é feita utilizando os nós que estão entre a origem e o destino (figura 1(b)). Com isso, para as redes de sensores sem fio temos que elas possuem a forma de comunicação como as redes *ad-hoc* com o objetivo de propagar os dados sensoriados para um elemento externo a rede (figura 1(c)).

Com isso, podemos apresentar as redes de sensores como sendo redes formadas por dispositivos compactos e autônomos, chamados de nós sensores, que coletam dados do ambiente e os processam localmente, ou de forma cooperativa entre nós vizinhos. No final, a informação processada pode ser enviada para o usuário. Devido ao seu tamanho os nós sensores possuem uma arquitetura simples e com limitações de processamento e armazenamento sendo formados por quatro componentes básicos:



(a) Rede estruturada. (b) Rede não estruturada. (c) Rede de sensores.

FIGURA 1 – Tipos de redes sem fio.

uma unidade perceptiva que pode possuir alguns sensores e um conversor de sinais analógicos para digitais (ADC); uma unidade de processamento com memória e processador; um transceptor; e uma fonte de energia que geralmente não é renovável. Além disso, de forma opcional podem existir elementos que complementam a estrutura dos sensores, como sistema de localização, mecanismo de mobilidade e gerador de energia. A estrutura básica de um nó sensor com os principais componentes pode ser visto na figura 2.

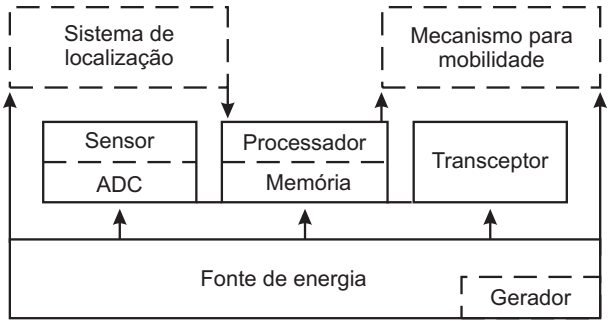


FIGURA 2 – Estrutura do nó sensor com os quatro componentes principais e os três componentes opcionais.

No entanto, uma rede de sensores pode ter outros três elementos básicos: os nós atuadores que possuem a função de atuar ou interferir no meio onde estão inseridos, a fim de corrigir falhas e/ou controlar o objeto monitorado; os sorvedouros ou nós de monitoração que recebem os dados e os processam de forma a extrair alguma informação útil para o usuário; e os nós *gateways* que são responsáveis por prover a comunicação da rede de sensores com outras redes de computadores. Esses três elementos básicos, bem como a estrutura típica de uma rede de sensores, podem ser vistos na figura 3. É importante destacar que esses elementos não precisam ser fisicamente distintos. Por exemplo, o sorvedouro e o *gateway* podem ser o mesmo dispositivo.

Ao observarmos as redes de sensores considerando os seus elementos básicos, a

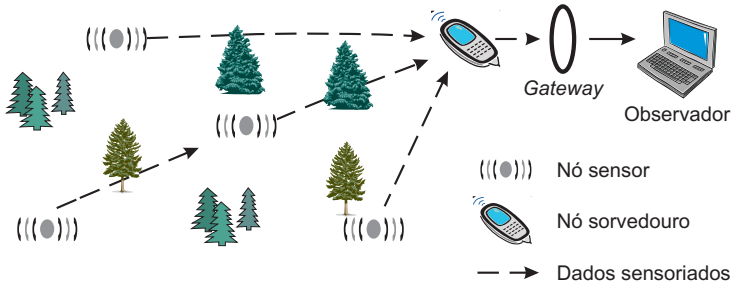


FIGURA 3 – Estrutura de uma rede de sensores considerando não só o nó sensor mas também os demais elementos básicos.

forma com que os nós são dispostos numa área de sensoriamento e a forma com que os fenômenos são monitorados pode-se fazer uma distinção entre os diferentes tipos de redes de sensores existentes. Com isso, as redes de sensores podem ser classificadas como: hierárquica se ela possui agrupamentos de nós, onde existe um líder que representa cada agrupamento, caso contrário a rede é considerada plana; homogênea se os nós possuem a mesma configuração de *hardware*, caso contrário a rede é considerada heterogênea; simétrica se todos os nós possuem o mesmo raio de comunicação, caso contrário a rede é considerada assimétrica; contínua se os dados coletados são enviados continuamente ou programada se os dados são enviados obedecendo a programação previamente estabelecida; dirigida a eventos se a rede envia dados apenas quando ocorre algum evento ou sob demanda quando a rede permite a consulta parcial ou total dos dados aos nós (TILAK; ABU-GHAZALEH; HEINZELMAN, 2002).

Como as redes de sensores possuem capacidade de sensoriamento e processamento distribuído elas podem ser utilizadas em uma grande variedade de aplicações, como por exemplo aplicações médicas, industriais, militares, meio ambiente e agropecuária (ESTRIN et al., 1999; POTTIE; KAISER, 2000; ESTRIN et al., 2001; SHEN; WANG; SUN, 2004; ARAMPATZIS; LYGEROS; MANESIS, 2005; DIAMOND; CERUTI, 2007; FLAMMINI et al., 2007). Essas aplicações podem ter um caráter de monitoramento onde apenas dados do ambiente são coletados ou um caráter de atuação onde ocorre intervenção no meio monitorado (LINS et al., 2003a, 2003b). De forma geral, podemos considerar três níveis de granularidade nas aplicações em redes de sensores:

- As aplicações de sensoriamento que dizem respeito à obtenção, ao processamento e ao tratamento dos dados monitorados antes deles saírem do nó sensor. Nesse caso, para obter um ganho global na rede a aplicação pode processar localmente esses dados.

- Os mecanismos de infraestrutura que são responsáveis por garantir o bom funcionamento da rede para que os dados monitorados possam ser entregues satisfatoriamente ao sorvedouro. Nesse caso, se necessário, a rede deve se autoconfigurar para garantir a qualidade da informação passada para o usuário.

- As aplicações para o usuário onde a rede tem por objetivo prover informações de sensoriamento para algum usuário externo à rede levando em conta suas necessidades. Nesse caso, a aplicação considera todos os recursos da rede para servir ao usuário, inclusive a aplicação de sensoriamento e a infraestrutura.

Com o objetivo de melhor contextualizar os diferentes níveis de granularidade de uma aplicação em redes de sensores, a seguir aprofundaremos um pouco mais a discussão em relação a esses níveis.

As redes de sensores sem fio possuem restrições de recursos que aliadas às necessidades das aplicações tornam o projeto dessas redes complexo. Nesse contexto, existem diversas linhas de pesquisa que tratam problemas relacionados com o projeto dessas redes, como a auto-organização (SCHURGERS et al., 2002; CHEN et al., 2002; FIGUEIREDO et al., 2005) e o gerenciamento de recursos (ZHAO; GOVINDAN; ESTRIN, 2002; RUIZ; NOGUEIRA; LOUREIRO, 2003; ZHAO; GOVINDAN; ESTRIN, 2003; GOUSSEVSKAIA et al., 2005).

Por tratarem de um tipo específico de redes *ad-hoc* e serem utilizadas em ambientes hostis com condições imprevisíveis, as redes de sensores devem ser autoconfiguráveis, adaptáveis e possuir um gerenciamento escalável. Devido às características da aplicação de sensoriamento, as redes de sensores possuem um modelo centrado nos dados (KRISHANAMACHARI; ESTRIN; WICKER, 2002; INTANAGONWIWAT et al., 2003), pois o objetivo dessas redes é levar a informação sensoriada para um ponto fora da rede. Essa característica permite a integração das operações da camada de sensoriamento com a camada de rede, oferecendo soluções mais eficientes.

Fatores relacionados com as características da rede, tipos e configurações dos sensores influenciam diretamente no desenvolvimento das aplicações de sensoriamento. Considerando essas características pode-se classificar as aplicações de sensoriamento em:

- Monitoramento, onde os dados são enviados periodicamente ou em resposta a um evento inesperado. Nesse caso, o nó sensor faz apenas um pré-processa-

mento nos dados deixando as operações mais elaboradas para serem executadas em outros elementos da rede com maior poder de processamento. Esse pré-processamento é necessário, pois o grande volume de dados sensoriados, se enviados sem nenhum tratamento, pode consumir a energia dos nós e comprometer os objetivos da rede.

- Consulta, onde os dados são enviados apenas quando requisitados por algum elemento externo à rede. Nesse caso, o nó sensor deve executar algum processamento sobre os dados de tal forma que apenas o resultado desse processamento seja guardado para ser enviado quando solicitado. Isso ocorre, pois o armazenamento de todos os dados sensoriados pode ser muito caro para o nó, se as consultas não forem freqüentes.

Para o tratamento dos dados, nas aplicações de monitoramento podemos utilizar técnicas como agregação de dados (KRISHANAMACHARI; ESTRIN; WICKER, 2002; ZHAO; GOVINDAN; ESTRIN, 2003; DASGUPTA; KALPAKIS; NAMJOSHI, 2003), fusão de dados (DURRANT-WHYTE, 1988; BROOKS; IYENGAR, 1997; LUO; YIH; SU, 2002; NAKAMURA; LOUREIRO; FRERY, 2007) ou *stream* de dados, como apresentado neste trabalho. Para as aplicações de consultas, a rede é vista como um grande banco de dados onde operações sobre os dados são calculadas internamente na rede (ABADI et al., 2004; MADDEN et al., 2005). Essa é a abordagem tradicional para utilização de *stream* de dados em redes de sensores.

De acordo com Loureiro et al. (2003), as redes de sensores sem fio possuem cinco funcionalidades básicas: o estabelecimento que consiste na configuração inicial da rede; a manutenção que consiste na adaptação da rede às mudanças de configurações que surgem ao longo do tempo; o sensoriamento que trata da coleta de dados sobre o ambiente; o processamento dos dados a serem enviados para o sorvedouro; e a comunicação que é responsável pelo envio desses dados. Discutiremos de forma mais detalhada apenas as funcionalidades de estabelecimento e manutenção, mais especificamente a tarefa de roteamento por estar diretamente relacionada ao mecanismo de infraestrutura explorado neste trabalho.

O estabelecimento de uma rede de sensores basicamente envolve a deposição dos nós na área a ser monitorada e na formação da rede. Essa fase ocorre antes do sensoriamento, e assim, os nós podem realizar tarefas de controle de densidade, formação de agrupamentos e montagem da estrutura de roteamento. Após o estabelecimento da rede é necessário manter a estrutura funcionando eficientemente durante todo o

tempo de vida da rede. Segundo Loureiro et al. (2003): “O objetivo da manutenção é prolongar o tempo de vida da rede, reduzir a imprevisibilidade e atender aos requisitos da aplicação, pois ao longo do tempo alguns nós atingem níveis de energia que podem restringir de forma parcial ou total sua capacidade”. Todas as tarefas realizadas para o estabelecimento da rede devem ser repetidas durante a manutenção, seja periodicamente ou na ocorrência de um determinado evento. Essa decisão dependerá do objetivo da aplicação.

Uma das tarefas que é considerada tanto na fase de estabelecimento como na fase de manutenção é a montagem da estrutura de roteamento. Uma abordagem bastante utilizada em redes de sensores para essa tarefa é o roteamento baseado em árvore cuja montagem consiste em configurar os nós da rede para que eles saibam para qual vizinho enviar suas informações sensoriadas (FIGUEIREDO et al., 2005; NAKAMURA et al., 2005). Basicamente um algoritmo de roteamento baseado em árvore é composto pelas seguintes fases:

- Construção da árvore que é baseada em alguns requisitos de rede ou da aplicação. É construída, via inundação[†] do sorvedouro para os nós. É nesse momento que as informações da aplicação são passadas para os nós sensores.
- Encaminhamento onde os dados sensoriados pelos nós fontes são encaminhados para o sorvedouro. Nessa fase os nós encaminham os dados sensoriados, através da árvore, até o sorvedouro.
- Reconstrução da árvore, em alguns casos, é necessário reconstruir a árvore pois a topologia da rede pode mudar por falha, desligamento ou esgotamento da energia dos nós. A estratégia de reconstrução pode ser feita de forma pró-ativa ou reativa, dependendo do gerenciamento da rede.

Como as redes de sensores são centradas nos dados, possivelmente, a informação presente nos dados é importante nas decisões da camada de roteamento. Caso a rede tenha restrições de energia e atraso, a identificação de dados redundantes na camada de roteamento pode habilitar reduções ou descarte desses dados, ou ainda, caminhos de roteamento alternativos dentro da rede podem ser utilizados para entregar dados com maior prioridade.

As aplicações para o usuário em redes de sensores, normalmente, apenas utilizam a infraestrutura da rede para obter informações do fenômeno monitorado. Contudo

[†]O termo inundação é mais conhecido do inglês *flooding*.

existem aplicações que o simples envio das informações sensoriadas não é suficiente e aspectos relacionados com tempo de resposta são fundamentais (CHAN; KI; NGAN, 2005; LU et al., 2002). Alguns exemplos dessas aplicações com exigência de prazos são: aplicações militares que necessitam efetuar a coleta dos dados e atuação no ambiente monitorado em tempo real; aplicações de segurança que utilizam sensores acústicos e de vídeo para detectar movimentos e soar algum alarme num intervalo de tempo bem pequeno; e aplicações para detecção em tempo real de bio-ataques que utilizam sensores para identificar a presença de elementos biológicos no corpo humano ou no ambiente.

Em sistemas embutidos de tempo real tradicional, o prazo da tarefa é um ponto crítico a ser considerado (tempo real *hard*). Algoritmos de escalonamento são desenvolvidos para reduzir ou evitar a perda dos prazos, seja estatisticamente ou dinamicamente. Em um ambiente dinâmico, o mecanismo de controle de admissão aceitará ou rejeitará a tarefa baseado na restrição de tempo e de outros recursos do sistema. O projeto dessas aplicações, é mais complexo, pois é concebido para ambientes específicos (CHAN; KI; NGAN, 2005). Em redes de sensores é comum haver aplicações de tempo real *soft*, pois o ambiente não é controlado. A aplicação normalmente usa métodos probabilísticos para tratar o dado e não tem confirmação na comunicação. Esses aspectos tornam o uso de tempo real *hard* em redes de sensores bem mais difícil. Por convenção, utilizaremos o termo “aplicações de tempo real” ao invés de tempo real *soft* em redes de sensores.

Considerando as aplicações de tempo real em redes de sensores, utilizar uma solução que garanta a priori o atendimento dos prazos é bem mais difícil, como dito acima, devido as características dessas redes. No entanto, podemos utilizar soluções aproximadas que identificam dentro da rede o momento em que os dados não podem ser entregues a tempo, exigindo que algum processamento nos dados seja feito, de tal forma que alguma informação útil possa chegar para o usuário dentro dos prazos exigidos. No projeto de um sistema de tempo real para rede de sensores, nós devemos conhecer o comportamento do atraso do envio dos dados para cada solução de uma dada aplicação e, com isso, aplicar a melhor solução de processamento dos dados para atender as exigências requeridas.

2.2 Algoritmos de *stream* de dados

Recentemente temos observado um forte crescimento da classe de aplicações *data-intensive*, onde a melhor forma de modelar o dado não é como um dado persistente mas como um *stream* de dados. Alguns exemplos de dados dessas aplicações são: medidas de rede, registros de chamadas telefônicas, páginas *web* visitadas e dados sensoriados. Comparando *stream* de dados com dados convencionais, temos que o dado pertencente ao *stream* chega de forma *online*, o sistema não tem controle na ordem de chegada dos elementos a serem processados, o *stream* é ilimitado e, a partir do momento que ele é processado, ele é descartado e apenas a informação processada é armazenada.

Stream de dados foi definido pela primeira vez por Henzinger, Raqhavan e Rajagopalan (1998) como "uma seqüencia de pontos ordenados $V_1, \ldots, V_n$ que devem ser acessados em ordem e que podem ser lidos uma vez ou um pequeno número de vezes. Cada leitura é chamada de *pass*". Nesse contexto, existem diferentes modelos que descrevem o *stream* de dados. Considere o *stream* de entrada $V_1, V_2, \ldots$ chegando de forma seqüencial, item por item. Esse *stream* descreve o sinal $\mathscr{V}^*$, que é uma função unidimensional $\mathscr{V}^* : [1 \ldots \mathbb{N}] \rightarrow \mathbb{R}$. Os modelos são diferenciados na forma como os V_i's descrevem $\mathscr{V}^*$ (MUTHUKRISHNAN, 2005). Com isso, temos os seguintes modelos:

- *Time series* onde, cada $V_i = \mathscr{V}^*[i]$, ou seja, os elementos são seqüenciais não possuindo relação ente si.
- *Cash register* onde, cada V_i corresponde a um incremento para $\mathscr{V}^*[j]$. Considere $V_i = (j, I_i)$, $I_i \geq 0$ e $\mathscr{V}_i^*[j] = \mathscr{V}_{i-1}^*[j] + I_i$, onde o $\mathscr{V}_i^*$ é o estado do sinal após a medição do i-ésimo termo do *stream*. Em outras palavras, cada evento I_i possui uma relação com um elemento j que reflete no sinal $\mathscr{V}^*$.
- *Turnstile* onde, cada V_i corresponde a uma atualização para $\mathscr{V}^*[j]$. Considere $V_i = (j, Y_i)$ e $\mathscr{V}_i^*[j] = \mathscr{V}_{i-1}^*[j] + Y_i$, onde $\mathscr{V}_i^*$ é o estado do sinal após a medição do i-ésimo termo do *stream* e Y_i pode ser positivo ou negativo. Em outras palavras, cada evento Y_i possui uma relação com um elemento j que reflete no sinal $\mathscr{V}^*$, sendo essa relação de inserção ou remoção.

Através desses modelos, precisamos processar o sinal $\mathscr{V}^*$ em diferentes momentos do *stream*. Normalmente para obtermos informações sobre o *stream*, é feito um processamento através de seleção, agregação, multiplexação ou demultiplexação, freqüência

dos itens, mineração do *stream*, *joins* e consultas em janelas. Para permitir a avaliação de soluções eficientes sobre esses processamentos é necessário considerar as seguintes medidas de desempenho: tempo de processamento por item V_i do *stream*, espaço para o armazenamento do item V_i e tempo computacional para computar as funções sobre $\mathscr{V}^*$. Um fator importante, apenas considerado no contexto de redes de sensores, é a comunicação, ou seja, a quantidade de itens de $\mathscr{V}^*$ que precisam trafegar na rede. Essas métricas devem ser levadas em conta, devido às características do *stream* de dados que exige uma grande capacidade computacional.

De forma geral, alguns princípios podem ser aplicados ao *stream* de dados para obtermos sistemas mais eficientes. Por exemplo, processamento e armazenamento paralelo, amostragem dos dados para controlar a taxa de transmissão ou a utilização de métodos de filtragem e agregação assim que o dado é recebido, deixando cálculos mais complexos no momento que o dado for utilizado, envolvendo menor volume de dados. Nessa direção existem alguns algoritmos de aproximação com o objetivo de reduzir o volume dos dados ao selecionar apenas amostras significativas ou ao realizarem análises dos dados (DATAR; MUTHUKRISHNAN, 2002; MUTHUKRISHNAN, 2005). Dentre os diversos tipos de algoritmos de *stream* de dados destacamos: janela deslizante, amostragem, rascunho e histograma (BABCOCK et al., 2002), sendo esses os algoritmos que nos baseamos para algumas das soluções apresentadas neste trabalho.

Os algoritmos de janela deslizante consistem em manter uma janela de dados mais atual, respeitando um tamanho específico de janela. Esses algoritmos de aproximação têm várias vantagens como, por exemplo, são determinísticos e possuem um fácil entendimento, pois seus métodos de aproximação são claros e o usuário do sistema pode confiar nos dados aproximados produzidos. O mais importante é que esses algoritmos enfatizam os dados mais recentes, onde na maioria das aplicações reais, são mais importantes que os dados antigos. Por exemplo, na figura 4 um item do dado chega a cada intervalo t e expira em um tempo $t+N$, onde N é o tamanho da janela e, conseqüentemente, a quantidade de dados a ser enviada ou processada (BABCOCK et al., 2002).

Os algoritmos de amostragem e rascunho consistem em abandonar a idéia de fazer um processamento em cada elemento do dado que chega, aplicando algum tipo de amostragem e resumo dos dados. Geralmente as amostras utilizadas são suficientes para representar o dado original, como apresentado na figura 5. Os algoritmos de rascunho, por sua vez, utilizam informações dos dados, como mínimo, máximo, média e

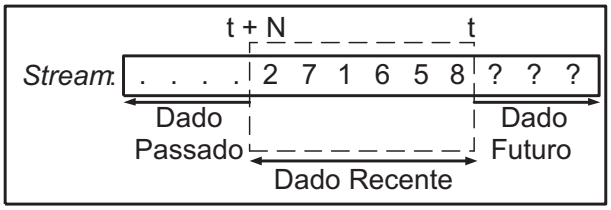


FIGURA 4 – Técnica de janela deslizante para o tratamento do *stream* de dados.

freqüência dos dados, como ilustrado na parte inferior da figura 6. Essas informações são utilizadas para inferir propriedades a respeito do total dos dados, porém a escolha da melhor informação para compor o resumo depende da aplicação (BABCOCK et al., 2002).

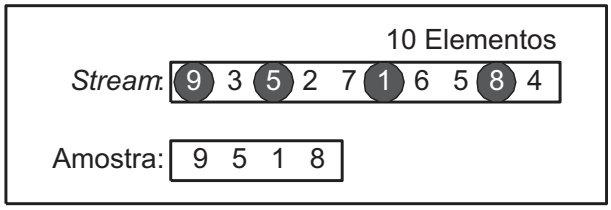


FIGURA 5 – Técnica de amostragem para o tratamento do *stream* de dados.

Algoritmos baseados em histograma são utilizados para capturar uma distribuição do conjunto dos dados, ou seja, os dados são analisados e acumulados em relação ao seu tipo, de tal forma que apenas um dado dessa distribuição seja armazenado. Um histograma é construído através da utilização de uma regra de particionamento na distribuição dos dados formando conjuntos distintos que são as colunas. Nas colunas estão armazenadas as aproximações das freqüências de ocorrência dos valores mais comuns. Sobre cada valor do atributo na coluna é assumido que algum valor entre o menor e o maior elemento pode ocorrer. Na prática as colunas guardam a informação do total de ocorrências, o menor e maior valor ocorrido para cada dimensão e o número de valores distintos (IOANNIDIS; POOSALA, 1999). As informações que compõem um histograma são ilustradas na figura 6. Uma característica importante desses algoritmos é que os histogramas são construídos em $O(n)$, sendo n o número de elementos do *stream*, lendo os dados de forma *online* sem a necessidade de armazená-los.

Como dito anteriormente, existe uma forte relação entre *stream* de dados e redes de sensores. Porém os algoritmos de *stream* tradicionais não podem ser aplicados diretamente em redes de sensores devido às diferenças encontradas entre o *stream* tradicional e o de sensoriamento. Tais diferenças devem ser levadas em consideração

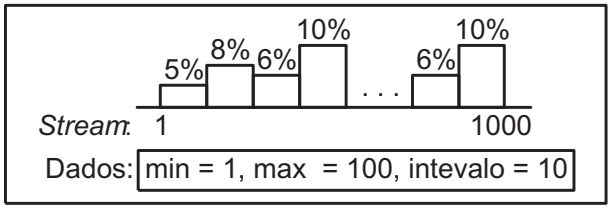


FIGURA 6 – Informações extraídas do *stream* de dados pelos algoritmos de rascunho e histograma.

ao se projetar algoritmos de *stream* de dados para as redes de sensores (ELNAHRAWY, 2003).

2.3 Análise de componentes principais e estudo da qualidade dos dados reduzidos

Neste trabalho utilizamos, para diferentes fins, duas importantes ferramentas estatísticas: análise de componentes principais para auxiliar na redução de dados multivariados; e o teste Kolmogorov-Smirnov para avaliar a qualidade dos dados reduzidos. Ambos os mecanismos serão detalhados a seguir.

2.3.1 Análise de componentes principais - PCA

A transformação de componentes principais[†] (KRZANOWSKI, 1995; JACKSON, 2003), também conhecida como transformação de Karhunen-Loève, é uma das ferramentas mais poderosas para o tratamento de dados multivariados. É uma transformação entre espaços γ-dimensionais, derivada da matriz de covariância dos dados de entrada gerando um novo conjunto de dados, de modo que cada valor resultante é uma combinação linear dos valores originais. O número de componentes principais é igual ao número de dimensões dos dados originais e esses podem ser ordenados de acordo com a sua variância. Com isso, o primeiro e último componentes principais devem ter a maior e a menor variância, respectivamente.

A propriedade mais importante do novo conjunto de dados gerado pelo PCA é que os dados não apresentam correlação (JACKSON, 2003), garantindo dessa forma que não haja redundância entre os dados e que seja obtido um novo conjunto de dados com

[†]Análise de componentes principais é abreviada na literatura como PCA do inglês *Principal Component Analysis*

propriedades para análise multivariada. A transformação de componentes principais pode ser descrita nas seguintes etapas:

1. Calcular Σ, a matriz de covariância dos dados (vamos supor que ela é definida positiva pois estamos tratando de variâncias).
2. Decompor Σ nos autovetores U e autovalores λ. Essa matriz será diagonalizável uma vez que a matriz de covariância é definida positiva (KRZANOWSKI, 1995).
3. Calcular o novo conjunto de dados, multiplicando o valor de cada variável pela matriz dos autovetores.

Os autovalores representam o comprimento dos eixos dos componentes principais do conjunto de dados e são medidos na unidade da variância. Associado a cada autovalor, existe um vetor de módulo unitário chamado autovetor. Os elementos de cada autovetor são fatores de ponderação que definem a contribuição da variável da matriz de dados original para um componente principal, numa combinação linear. Os autovetores representam as direções dos eixos das componentes principais.

O método de componentes principais pode ser formulado da seguinte forma: dada uma matriz de dados originais V, com s variáveis correlacionadas, aplicar PCA consiste em calcular a matriz C, que possui s variáveis não correlacionadas, de forma que cada componente principal será calculado por

$$C_i = u_i'[V - \overline{V}], \tag{2.1}$$

onde para cada $1 \leq i \leq s$, $u_i = (u_{i,1}, \ldots, u_{i,s})$ é o autovetor i da matriz de covariância dos dados V.

Outra propriedade importante do PCA é que a equação (2.1) pode ser invertida restaurando as variáveis originais em função dos componentes principais. Para isso utilizamos

$$V = \overline{V} + U\,C, \tag{2.2}$$

devido a U ser ortonormal (WINTERLE; STEINBRUCH, 1987), temos $U^{-1} = U'$; com isso, dada a matriz C, os dados originais V podem ser unicamente determinados pela equação (2.2).

No contexto do nosso trabalho o PCA é utilizado para classificar os dados multivariados, de tal forma que escolhemos apenas os dados mais correlacionados para propagá-los até o sorvedouro.

2.3.2 Qualidade dos dados reduzidos

Ao efetuarmos a redução dos dados é importante avaliar o quanto o dado reduzido representa o dado original. Nessa direção, duas análises foram realizadas no nosso trabalho: a aproximação entre as distribuições de freqüência dos dados originais e amostrados; e a discrepância entre os valores originais e amostrados.

Para a avaliação da aproximação entre as distribuições de freqüência dos dados originais e amostrados utilizamos o teste de *Kolmogorov-Smirnov* (teste KS) (SIEGEL; CASTELLAN, 1988; RESCHENHOFER, 1997). Esse teste avalia se duas amostras V e V' têm distribuições similares não exigindo que as amostras sigam a distribuição normal, ou seja, caso os valores amostrados sigam outra distribuição este teste também pode ser utilizado. O teste KS é descrito a seguir:

1. Construir a distribuição acumulada Fn dos dois grupos V e V' usando a mesma classe para ambas as distribuições.
2. Determinar as diferenças acumuladas para cada ponto da distribuição e considerar a maior das diferenças (D_{max}).
3. Computar o valor crítico,

   $$D_{crit} = y\sqrt{(|V|+|V'|)/|V||V'|}$$

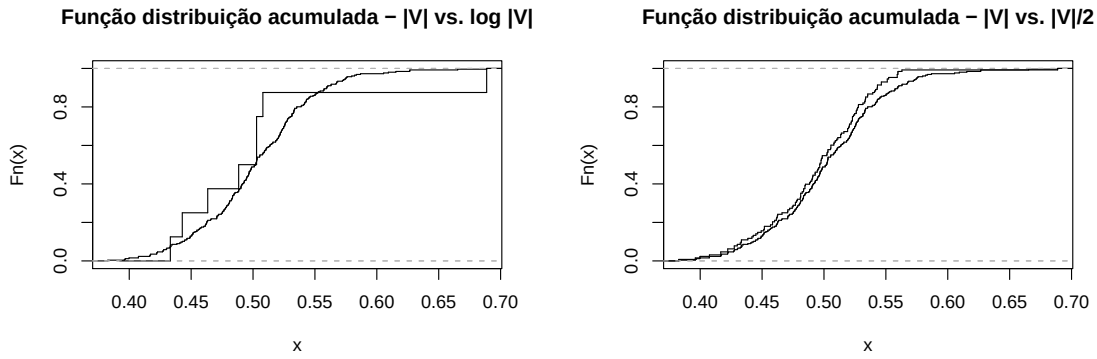
   onde y é um valor tabulado e representa o nível de significância do teste.
4. As amostras seguem a mesma distribuição se

   $$D_{max} \leq D_{crit}. \tag{2.3}$$

Apenas como ilustração, considere a figura 7 que apresenta a comparação entre as distribuições de freqüência acumulada, com $|V| = 256$ e $|V'| = \{\log|V|, |V|/2\}$[†] onde $V' \subset V$. Em ambos os casos, através do teste KS, temos que V' segue a mesma distribuição de V.

Como o teste KS apenas identifica se duas amostras seguem a mesma distribuição, é importante avaliar se os conjuntos V e V' possuem a média de seus valores próximos. Para isso podemos calcular a maior distância entre $\overline{V}$ e os valores do intervalo de confiança $IC = [v_{inf}; v_{sup}]$ de $\overline{V}$. Os passos para essa avaliação são descritos a seguir:

[†] Em todo o trabalho, ao utilizarmos $\log x$, estaremos sempre nos referindo ao logaritmo de x na base dois.



(a) Comparando com log dos dados. (b) Comparando com a metade dos dados.

FIGURA 7 – Função da distribuição acumulada para 256 valores.

1. Obter a média dos valores dos dados reduzidos e originais, que são respectivamente $\overline{V}$ e $\overline{V'}$.

2. Calcular o intervalo de confiança IC com confiança de 95% para $\overline{V'}$.

3. Calcular o valor absoluto da maior diferença entre $\overline{V}$ e IC

$$\varepsilon = \max\{|v_{inf} - \overline{V}|, |v_{sup} - \overline{V}|\}. \tag{2.4}$$

Essas duas análises são utilizadas no nosso trabalho com o objetivo de identificar o comportamento dos fenômenos monitorados e reportados pelos sensores após alguma redução ser efetuada.

2.4 Trabalhos relacionados

Os trabalhos relacionados estão agrupados nos seguintes tópicos: algoritmos de *stream* de dados; *stream* de dados em redes de sensores; soluções alternativas ao processamento do *stream*; soluções de roteamento em redes de sensores; e aplicações de tempo real em redes de sensores.

2.4.1 *Stream* de dados em redes de sensores

Considerando apenas os estudos relacionados aos algoritmos de *stream* de dados, esses estão quase sempre voltados a estabelecer limites inferiores para a classe de algoritmos onde as principais métricas analisadas são complexidade de tempo

e espaço (ALON; MATIAS; SZEGEDY, 1996; HENZINGER; RAQHAVAN; RAJAGOPALAN, 1998; IOANNIDIS; POOSALA, 1999; DATAR et al., 2002; CORMODE et al., 2003; GUHA et al., 2003; MUTHUKRISHNAN, 2005; AL-KATEB; LEE; WANG, 2007; LIAN; CHEN, 2008; ALTIPARMAK; TUNCEL; FERHATOSMANOGLU, 2008). Existem propostas que apresentam aplicações de *stream* de dados para resolver problemas específicos modelados usando algoritmos de *stream* de dados (INDYK, 1999; BAR-YOSSEFF; KUMAR; SIVAKUMAR, 2002; CHARIKAR; CHEN; FARACH-COLTON, 2002; DATAR; MUTHUKRISHNAN, 2002; BURIOL et al., 2005, 2006b, 2006a; AKCAN; BRONNIMANN, 2007; NASRAOUI et al., 2008; CAMMERT et al., 2008). No entanto, ao considerarmos a utilização dos algoritmos de *stream* de dados em redes de sensores, em alguns casos, é feita uma abstração da rede, ao utilizar uma camada de *software* chamada de *Data Stream Management System* (DSMS). O problema a ser resolvido com essas aplicações é de como responder às consultas efetuadas pelo usuário (BABCOCK et al., 2002; GEHRKE; MADDEN, 2004; ABADI et al., 2004; MADDEN et al., 2005; ROHM; SCHOLZ; GABER, 2007; XU; TANG; LEE, 2008). Algumas propostas usam o *stream* para extrair informações de gerenciamento da rede de sensores, tais como energia, agrupamento e localização de nós (BABU; SUBRAMANIAN; WIDOM, 2001; LEDLIE; NG; HOLLAND, 2005; PHUNG; GABER; ROHM, 2007). O que diferencia esses trabalhos do nosso é que além de estarmos utilizando os algoritmos de *stream* de dados como parte integrante das redes de sensores, utilizamos as informações extraídas do *stream* para auxiliar a infraestrutura da rede, no nosso caso o roteamento, o que não é considerado nesses trabalhos.

2.4.2 Soluções para o processamento dos dados monitorados em redes de sensores

Para o problema de redução de dados e manutenção de qualidade do dado em redes de sensores existem trabalhos que propõem a utilização de amostragem adaptativa, ou seja, à medida que o fenômeno monitorado se modifica, a forma de amostragem se adapta para obter dados mais precisos. Além disso, existem trabalhos que observam os dados objetivando identificar dados correlacionados e eliminar redundância (MARBINI; SACKS, 2003; JAIN; CHANG, 2004; GANESAN et al., 2004; WILLETT; MARTIN; NOWAK, 2004; CHEN; KNOW; CHOI, 2006; ALIPPI et al., 2007; GEDIK; LIU; YU, 2007; YUEN; LIANG; LI, 2008). Também existem trabalhos que fazem fusão, compressão, correlação, redução de dados ou agregação, normalmente baseados na correlação das informações sensoriadas e com o objetivo de economi-

zar recursos da rede, como energia, tempo de resposta e perda de pacotes (KRISHNAMACHARI; ESTRIN; WICKER, 2002; DASGUPTA; KALPAKIS; NAMJOSHI, 2003; ZHAO; GOVINDAN; ESTRIN, 2003; ZHU; PAPAVASSILIOU, 2004; SANTINI; ROMER, 2006; BROWN; SREENAN, 2007; KIM; PARK; CHO, 2007; NAKAMURA; LOUREIRO; FRERY, 2007; ZHENG; BARTON, 2007; GUITTON; SKORDYLIS; TRIGONI, 2007; YUEN; LIANG; LI, 2008; YU; KRISHNAMACHARI; PRASANNA, 2008). Além disso, existem técnicas para redução de dados multivariados que utilizam métodos para estimar o comportamento do dado a ser sensoriado e envia apenas as diferenças observadas ao longo do tempo (SEO; KANG; RYU, 2005; LI; ZHANG, 2006; SCHIZAS; GIANNAKIS; LUO, 2007; CVEJIC; BULL; CANAGARAJAH, 2007). O que diferencia esses trabalhos do nosso é que essas propostas, não são baseadas nas técnicas de *stream* de dados, apesar de não terem disponíveis todos os dados para o seu processamento.

2.4.3 Redes de sensores sem fio hierárquicas

Existem várias soluções para os diversos problemas de agrupamentos em redes de sensores que podem ser vistos em diversos *surveys* da área (POTTIE; KAISER, 2000; AKYILDIZ et al., 2002; LOUREIRO et al., 2003). No entanto, de forma geral as soluções estão relacionadas a melhorar o desempenho da rede através da utilização de agrupamentos ou encontrar a melhor maneira de montar o agrupamento de acordo com o interesse da aplicação (KRISHNA et al., 1997; BASAGNI, 1999; BANERJEE; KHULLER, 2001; KARAATA, 2006; VLAJIC; XIA, 2006; CHANG; LIN; CHEN, 2006; YADAV; YADAV; VARMA, 2007; SOLTAN; MALEKI; PEDRAM, 2007; MUDUNDI; ALI, 2007; REIS et al., 2007; LIAN; NAIK; AGNEW, 2007). Além disso, existem algumas soluções que utilizam agrupamentos na rede com o objetivo de atender às necessidades das aplicações fazendo redução, agregação e o processamento dos dados (HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2000; PHAM; KIM; MOH, 2004; LEE; CHUNG, 2005; CHEN; LIESTMAN; LIU, 2006; GAO et al., 2007; LIU; WU; PEI, 2007). O que diferencia esses trabalhos do nosso é que essas propostas não são baseadas nas técnicas de *stream* de dados para efetuar a redução dos dados através do nó líder e, com isso, eliminar redundância entre os dados de uma mesma região.

2.4.4 Soluções de roteamento em redes de sensores

Várias soluções para os problemas de roteamento em redes de sensores podem ser vistos em alguns *surveys* da área (ESTRIN et al., 1999; ROYER; TOH, 1999; ESTRIN et al., 2001; LUO; LIU; DAS, 2007). Enfatizamos apenas às soluções relacionadas com roteamento em árvores pois foi a técnica abordada neste trabalho. Sabemos que as soluções, para o problema de roteamento, podem ser reativas ou pró-ativas, onde as árvores são montadas levando em consideração aspectos da rede como vizinho mais próximo ou vizinho com mais energia residual (HEINZELMAN; KULIK; BALAKRISHNAN, 1999; SOHRABI et al., 2000; INTANAGONWIWAT et al., 2003; HEIDEMANN; SILVA; ESTRIN, 2003; FIGUEIREDO et al., 2005; VASS; VIDACS, 2007; ZHANG; MA; YANG, 2008). Como as redes de sensores são centradas nos dados, é importante que as soluções de roteamento levem em consideração aspectos dos dados. Porém boa parte dessas soluções desconsideram o dado sensoriado para montar a árvore de roteamento e nenhuma delas consideram técnicas de *stream* de dados. O que diferencia esses trabalhos do nosso é que utilizamos as informações contidas no *stream* para tomar decisões na camada de roteamento como o objetivo de reduzir os dados.

2.4.5 Aplicações de tempo real em redes de sensores

As pesquisas relacionadas com aplicações de tempo real em redes de sensores, no geral, estão voltadas a arquiteturas e modelos matemáticos para aplicações gerais (LU et al., 2002; HE et al., 2003; LI; SHENOY; RAMAMRITHAM, 2004; CHAN; KI; NGAN, 2005; ZHOU; XIONG; LIN, 2007; AFONSO et al., 2007). No entanto, é possível encontrar algumas soluções, de roteamento e/ou aplicação, direcionadas para aplicações específicas (PENG et al., 2007; PAN et al., 2007; LI; GU; ZHAO, 2007). Essas propostas consideram os prazos e a energia, como principais métricas a serem estudadas. Isso ocorre porque os prazos são importantes em aplicações de tempo real e energia é um recurso não renovável em redes de sensores, necessitando de uma atenção especial. O que diferencia esses trabalhos do nosso é que efetuamos de forma *online* simultâneas reduções permitindo que a rede possa atender aos prazos exigidos pela aplicação, empregando as técnicas baseadas em *stream* de dados.

3 PROBLEMA DE REDUÇÃO DE DADOS EM REDES DE SENSORES SEM FIO

"*Aprender sem pensar é trabalho perdido.*" (Confúcio)

COMO discutido anteriormente, as redes de sensores sem fio consistem de dispositivos de sensoriamento autônomos que trabalham de forma distribuída e cooperativa com o objetivo de monitorar condições físicas ou ambientais, tais como temperatura, som, vibrações, pressão, movimento ou poluição (ROMER; MATTERN, 2004). Tais sistemas físicos ou ambientais podem ser representados pelo diagrama mostrado na figura 8, onde $\mathscr{N}$ denota o ambiente e o processo a ser medido, F é o fenômeno de interesse, com $\mathscr{V}^*$ seu domínio espaço-temporal. Se uma observação foi completada sem problemas, teremos um conjunto de regras (R^*) ideais para tomada de decisões ideais (D^*). De acordo com essas características nós consideramos $\mathscr{V}^*$ o *stream* de dados.

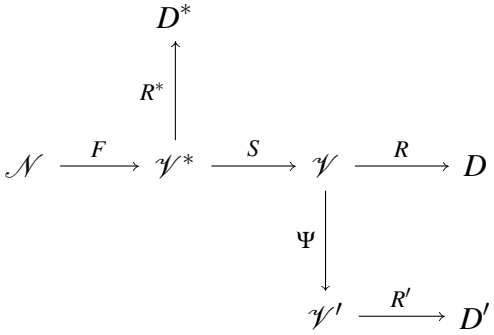


FIGURA 8 – Representação de um sistema de uma rede de sensores onde é mostrado o comportamento ideal ($\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow D^*$), sensoriado ($\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow D$) e reduzido ($\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow \mathscr{V}' \rightarrow D'$).

Ao invés de uma situação ideal, temos um conjunto de s sensores, $S = (S_1, \ldots, S_s)$, monitorando um fenômeno e produzindo conjuntos de amostras no domínio $\mathscr{V}_i$, com $1 \leq i \leq s$; todos os possíveis conjuntos do domínio são denotados por $\mathscr{V} = (\mathscr{V}_1, \ldots, \mathscr{V}_s)$. Usando todas essas informações, podemos conceber um conjunto de regras (R) para prover um conjunto de decisões (D). Portanto, diferentemente de $\mathscr{V}^*$, $\mathscr{V}$ é um *stream*

de sensoriamento.

Para melhor contextualizar os possíveis sistemas que podem ser representados através da figura 8, seguindo o comportamento sensoriado $\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow D$, considere os seguintes exemplos de aplicações:

- O projeto Great Duck Island (MAINWARING et al., 2002) teve por objetivo monitorar o comportamento e o habitat de um pássaro chamado Storm Petrel. Inicialmente, 32 nós sensores foram colocados nas tocas dos pássaros e também próximos a elas. Posteriormente, a rede foi aumentada, ganhando mais nós e estações meteorológicas. Os nós sensores podiam medir a temperatura (t), a umidade do ar (u) e a pressão barométrica (b), entre outras variáveis. As leituras periódicas dos nós eram transmitidas a um *gateway* através da comunicação entre os nós. O *gateway* retransmitia as leituras até a estação base, onde eram armazenadas. Uma réplica do banco de dados era transmitida, via satélite, para um servidor na Universidade de Berkeley a cada 15 minutos. Essa aplicação, possui diferentes comportamentos de sensoriamento que convergem para a seguinte representação:

  $$\mathscr{N} \rightarrow \{\mathscr{V}_t^*, \mathscr{V}_u^*, \mathscr{V}_b^*\} \rightarrow \{\mathscr{V}_t, \mathscr{V}_u, \mathscr{V}_b\}_{15\,min} \rightarrow D,$$

  onde $\mathscr{N}$, D representam, respectivamente, todo o ecossistema e o comportamento do Storm Petrel. A decisão D pode ser, por exemplo, que o pássaro está dormindo, se alimentando, acasalando, colocando ou chocando ovos. Note que essas decisões são tomadas utilizando amostras periódicas do ambiente a cada 15 minutos.

- O projeto ZebraNet (JUANG et al., 2002), utilizou colares sensores equipados com GPS instalados em zebras na reserva de Sweetwaters, Quênia. Do ponto de vista biológico, o objetivo do projeto era monitorar o comportamento noturno dos animais e ainda responder questões acerca de migração e relacionamento inter espécies. Os sensores foram projetados para registrarem e armazenarem a posição (p) dos animais a cada 3 minutos. A cada hora, outros sensores registravam por 3 minutos dados meteorológicos (m), ambientais (a), de iluminação (i), de temperatura (t) e de movimentos corporais (mc). A estação base móvel percorria o campo dos sensores, habitat das zebras, para recolher os dados. Essa aplicação, possuía diferentes comportamentos de sensoriamento que con-

vergiam para a seguinte representação:

$$\mathscr{N} \rightarrow \{\mathscr{V}_p^*, \mathscr{V}_m^*, \mathscr{V}_a^*, \mathscr{V}_i^*, \mathscr{V}_t^*, \mathscr{V}_{mc}^*\} \rightarrow \{\mathscr{V}_p\}_{3\,min} \cup \{\mathscr{V}_m, \mathscr{V}_a, \mathscr{V}_i, \mathscr{V}_t, \mathscr{V}_{mc}\}_{3\,min \subset 1\,h} \rightarrow D,$$

onde $\mathscr{N}$, D representam, respectivamente, todo o ecossistema e o comportamento noturno da zebra. A decisão D pode ser, por exemplo, que a zebra se alimenta mais, procura lugares mais afastados do bando ou acasala várias vezes no período da noite. Note que essas decisões são tomadas utilizando amostras sobre a localização das zebras a cada três minutos e amostras que representam o ambiente local monitorado constantemente durante três minutos a cada hora.

- Existem também as aplicações mais gerais (ARAMPATZIS; LYGEROS; MANESIS, 2005), utilizadas como motivação em diversos trabalhos, como: monitoramento de incêndios, detecção de enchentes, detector de poluição e agricultura de precisão. Nessas aplicações, os nós são distribuídos a priori ou aleatoriamente para recolher informações do ambiente referente a algum evento (e). As informações, no geral, são transmitidas ao sorvedouro utilizando uma comunicação multi-saltos. Logo que um evento anormal é detectado o sorvedouro deve ser notificado. Algum usuário externo à rede, ao observar os dados recolhidos pelo sorvedouro, pode julgar o evento ocorrido como verdadeiro ou não. Nesse caso ele pode atuar na rede ou enviar uma equipe para avaliar a situação no local. Essas aplicações, possuem um comportamento padrão de sensoriamento que pode ser representado pelo diagrama:

$$\mathscr{N} \rightarrow \mathscr{V}_e^* \rightarrow \mathscr{V}_e \rightarrow D,$$

onde $\mathscr{N}$ e D representam, respectivamente, o fenômeno monitorado e a ocorrência de incêndio, no caso da aplicação de detecção de incêndios. Para essas aplicações quanto menor o intervalo de leituras para o conjunto $\mathscr{V}_e$ mais precisa e eficiente pode ser a decisão D.

Devido à complexidade dos sistemas nos quais as redes de sensores atuam, é impraticável utilizar um conjunto de regras gerais R que possam ser aplicadas em qualquer sistema, por exemplo, as regras utilizadas pelo ZebraNet não podem ser as mesmas utilizadas pela aplicação de detecção de incêndio. Tanto as regras como as decisões que devem ser tomadas são específicas para cada aplicação. Por essa razão, para cada aplicação é necessário definir a forma de representação dos valores V do conjunto $\mathscr{V}_i = \{V_1, V_2, \ldots\}$, ou seja, definir como o item *stream* será representado no

sistema. Considerando os exemplos de aplicações apresentados anteriormente podemos considerar os seguintes itens *stream*:

- No projeto Great Duck Island, temos $\mathscr{V} = \{\mathscr{V}_t, \mathscr{V}_u, \mathscr{V}_b\}_{15\,min}$, nesse caso o item V que descreve os três conjuntos $\mathscr{V}_t$, $\mathscr{V}_u$ e $\mathscr{V}_b$ é no formato numérico tal que $V \in \mathbb{R}$, $-100 \leq V \leq 100$. A leitura e o processamento de V ocorre a cada $15\,min$.

- No projeto ZebraNet, temos $\mathscr{V} = \{\mathscr{V}_p\}_{3\,min} \cup \{\mathscr{V}_m, \mathscr{V}_a, \mathscr{V}_i, \mathscr{V}_t, \mathscr{V}_{mc}\}_{3\,min \subset 1\,h}$, nesse caso temos diferentes representações para os itens: o conjunto $\mathscr{V}_p$ é descrito pelo item V no formato (x,y,z) tal que $x,y,z \in \mathbb{R}$. A leitura e o processamento de V ocorre a cada $3\,min$; os conjuntos $\mathscr{V}_m$, $\mathscr{V}_a$ e $\mathscr{V}_t$ são descritos pelo item V que é representado por um conjunto $V = \{v_1, \ldots, v_{3a}\}$, onde a é o número de leituras do ambiente por minuto e v é um valor no formato numérico tal que $v \in \mathbb{R}$, $-100 \leq v \leq 100$. A leitura e o processamento de V ocorre a cada hora; e os conjuntos $\mathscr{V}_i$ e $\mathscr{V}_{mc}$ são descritos pelo item V que é representado por um conjunto $V = \{v_1, \ldots, v_{3a}\}$, onde v é no formato binário tal que $v \in \{0,1\}$. A leitura e o processamento de V ocorre a cada hora.

- No caso das aplicações gerais, temos $\mathscr{V} = \mathscr{V}_e$ que é descrito pelo item V representado por um conjunto $V = \{v_1, \ldots, v_n\}$, onde n é o número de leituras do ambiente e v é um valor no formato numérico tal que $v \in \mathbb{R}$, $-100 \leq v \leq 100$. A leitura e o processamento de V ocorre quando n itens forem obtidos.

Com o objetivo de cobrir um maior número de aplicações e apresentar soluções mais abrangentes, neste trabalho consideramos apenas as aplicações gerais. Logo os sistemas que estaremos tratando seguem o comportamento de sensoriamento

$$\mathscr{N} \rightarrow \mathscr{V}_e^* \rightarrow \mathscr{V}_e \rightarrow D,$$

onde o conjunto $\mathscr{V}_e$ é descrito pelo item V que é representado por um conjunto $V = \{v_1, \ldots, v_n\}$, onde n é o número de leituras do ambiente e v é um valor no formato numérico tal que $v \in \mathbb{R}$, $0 \leq v \leq 1$[†] seguindo uma distribuição normal com os parâmetros $\mu = 0.5$ e $\sigma = 0.1$, a escolha da distribuição normal foi feita arbitrariamente. A leitura e o processamento de V ocorre quando n itens forem obtidos. Por convenção, utilizaremos no decorrer do trabalho $\mathscr{V}$ para representar o *stream* $\mathscr{V}_e$ nas aplicações gerais.

[†]Utilizamos por convenção valores entre 0, 1 apenas para facilitar o processamento dos dados via simulação. No entanto a utilização de outro intervalo de valores, não compromete a qualidade da solução.

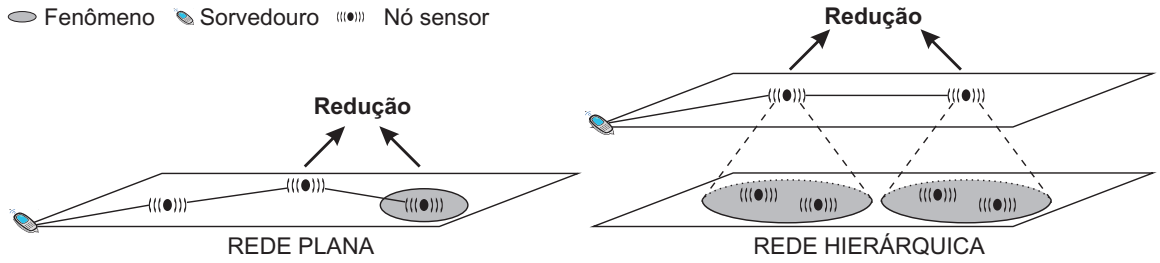


FIGURA 9 – Onde utilizar a redução de dados.

Em todos os casos apresentados anteriormente, utilizar todo o conjunto $\mathscr{V}$ pode ser muito oneroso em termos de energia, largura de banda, recursos computacionais e, especialmente, no tempo de entrega das mensagens gerando atrasos para as aplicações. Uma vez que existe redundância nos dados da maioria das aplicações, a redução dos dados pode não degradar a informação sensoriada. Na figura 8, as técnicas de redução de dados nos sensores são denotadas por Ψ, e a redução dos dados do domínio $\mathscr{V}$ é denotada por $\mathscr{V}'$. Com isso, as novas regras que usam $\mathscr{V}'$ são denotadas por R', e elas conduzem a um conjunto de decisões D'. Motivado pela necessidade de efetuarmos uma redução Ψ no conjunto $\mathscr{V}$, o problema geral abordado neste trabalho pode ser definido como segue:

Definição 1. *(Problema) Dado que as aplicações gerais nas redes de sensores que necessitam reduzir os dados, queremos utilizar técnicas de stream de dados para efetuar a redução Ψ de tal forma que seja possível reduzir gastos de energia e atraso de pacotes na rede. Além disso, o novo conjunto de decisões D' tomadas após a redução Ψ deve ser equivalente às decisões D que seriam tomadas pela aplicação sem a redução.*

Antes de apresentarmos algumas hipóteses para o nosso problema, devemos considerar quais situações, numa rede de sensores, podem nos levar a utilizar alguma redução Ψ. Como apresentado na figura 9 temos nas aplicações gerais a redução de dados em três casos: no momento do sensoriamento, através de um nó agregador e durante o roteamento. A redução no momento do sensoriamento pode ocorrer caso a aplicação geral necessite de muitas leituras do ambiente em um curto período de tempo, o que torna necessário o processamento do item $V = \{v_1, \ldots, v_n\}$ através da função de redução Ψ gerando um conjunto reduzido V'. Para essa redução devemos garantir que as decisões D' tomadas pela aplicação não sejam comprometidas.

A redução através de um nó agregador ocorre quando a aplicação geral necessita agrupar os nós com o objetivo de identificar a ocorrência de eventos numa região

específica. Cada agrupamento $S_A = (S_1, \ldots, S_{s_a})$ possui s_a sensores que reportam para o nó agregador n leituras do ambiente. Com isso o nó agregador será responsável por processar o *stream* $\mathscr{V}$ descrito pelo item $V = \{v_1, \ldots, v_{s_a \times n}\}$ onde $s_a \times n$ é o número de leituras de seu agrupamento. Como para a redução no momento do sensoriamento, o item $V = \{v_1, \ldots, v_{s_a \times n}\}$ é processado através da função de redução Ψ gerando um conjunto reduzido V'. Mais uma vez, para essa redução devemos garantir que as decisões D' tomadas pela aplicação não sejam comprometidas.

A redução durante o roteamento ocorre quando a aplicação geral necessita de muitas leituras do ambiente em um curto período de tempo e possui exigências para a entrega dos dados, como prazos e consumo de energia reduzido. Nessas aplicações, devido a dinâmica da rede, as exigências podem não ser cumpridas e as decisões sobre os dados precisam ser tomadas ao longo do roteamento. Com isso, o item V gerado pelos nós sensores pode ser reduzido nos nós roteadores permitindo que as exigências sejam cumpridas e o máximo de dados possa ser entregue.

Com o objetivo de cobrir um maior número de aplicações gerais e apresentar soluções mais abrangentes para os casos de redução apresentados anteriormente, no nosso trabalho definimos para o comportamento dos dados reduzidos $\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow \mathscr{V}' \rightarrow D'$ as seguintes regras gerais:

- R'_{dst} que tem como objetivo identificar se os itens V e V' seguem a mesma distribuição. Para processar essa regra é utilizado o teste de Kolmogorov-Smirnov, descrito anteriormente (equação 2.3). Essa regra nos ajuda a tomar a decisão D'_{dst} que corresponde a confirmar ou rejeitar a questão: *V e V' seguem a mesma distribuição de probabilidade?*

- R'_{val} que tem como objetivo avaliar a diferença entre os valores que compõem o item V e V'. Para processar essa regra utilizamos o erro ε, mostrado anteriormente (equação 2.4). Essa regra nos ajuda a tomar a decisão D'_{val} que corresponde a confirmar ou rejeitar a questão: *A média dos valores do item V' está próxima da média dos valores do item V?*

- R'_{ord} que tem como objetivo avaliar se os valores do item V' mantêm a mesma ordem de chegada do item V original. Para essa regra basta identificar se a seqüência dos valores em V' é preservada. Essa regra nos ajuda a tomar a decisão D'_{ord} que corresponde a aceitar ou rejeitar a questão: *A ordem de chegada dos valores do item V é preservada quando ele é reduzido?*

Considerando os diferentes casos de redução para o problema apresentado na definição 1, temos como hipóteses:

Hipótese 1. *Considerando as aplicações gerais em redes de sensores, é possível projetá-las através de uma solução genérica de tal forma que seja possível aplicar a redução de dados Ψ, baseada em stream de dados, no momento do sensoriamento, através de um nó agregador e durante o roteamento.*

Hipótese 2. *Ao utilizar a redução de dados Ψ, baseada em stream de dados, no momento do sensoriamento em um rede plana, é possível economizar gastos relacionados à energia e atrasos sendo possível tomar as decisões D' para as aplicações gerais.*

Hipótese 3. *Ao utilizar a redução de dados Ψ, baseada em stream de dados, através de um nó agregador em um rede hierárquica, é possível economizar gastos relacionados à energia e atrasos sendo possível tomar as decisões D' para as aplicações gerais. Além disso, a redução em um elemento agregador é mais eficiente em relação aos gastos na rede quando comparado com a redução para as mesmas aplicações em uma rede plana.*

Hipótese 4. *Ao utilizar a redução de dados Ψ, baseada em stream de dados, durante o roteamento em uma rede plana nas aplicações gerais de tempo real é possível alcançar os prazos determinados pela aplicação e tomar as decisões D'. Além disso, é possível determinar de forma online o tamanho da redução de tal forma que as exigências de prazos sejam cumpridas enviando o máximo de dados possíveis.*

Ao longo deste trabalho tem-se como objetivo confirmar as hipóteses apresentadas acima. No próximo capítulo apresentaremos a arquitetura proposta para redução de dados utilizada como solução para o problema apresentado na definição 1 e para aceitar a hipótese 1. Além disso, nos demais capítulos a arquitetura será utilizada em diversos cenários e experimentos realizados para aceitar ou recusar as hipóteses 2 a 4 apresentadas anteriormente.

4 ARQUITETURA PARA REDUÇÃO DE DADOS

"*O pensamento é uma idéia em trânsito.*" (Pitágoras)

ESTE capítulo apresenta a arquitetura, chamada OGK, que é direcionada para aplicações de redução de dados baseadas em técnicas de *stream* de dados. Essa arquitetura sugere que seja utilizado o conhecimento a respeito do *stream* de sensoriamento[†], de tal forma que a melhor solução de redução seja aplicada nas aplicações gerais em redes de sensores sem fio. Essa arquitetura é utilizada para auxiliar nas soluções provenientes do problema relacionado à redução de dados em redes de sensores baseadas nas técnicas de *stream* de dados, de acordo com a definição 1 (pg. 53), e através dela mostraremos que é possível aceitar a hipótese 1 (pg. 55) que diz respeito ao projeto de uma solução genérica onde é possível aplicar a redução de dados Ψ.

4.1 Arquitetura OGK

A arquitetura OGK, ilustrada na figura 10, é voltada para as aplicações gerais que seguem o comportamento de redução

$$\mathscr{N} \xrightarrow{F} \mathscr{V}^* \xrightarrow{S} \mathscr{V} \xrightarrow{\Psi} \mathscr{V}',$$

onde $\Psi = \psi_1 \circ \psi_2 \circ \ldots \circ \psi_n$, é a função de redução composta por n fases ψ. O item V que descreve o stream $\mathscr{V}$ é passado para a função de redução, através do conjunto S de sensores presentes no próprio nó ou em outros nós, representados na figura 10 respectivamente por *Fenômeno* e *Roteamento*.

Para representar os diferentes casos para a leitura do item V na arquitetura OGK, utilizamos os componentes *Sensor*, *Interface de rede*, *Receptor* e *Transmissor*. O *Sensor*

[†]A premissa de utilizar o conhecimento a respeito do *stream* deu origem a sigla OGK que vem do inglês *On a Good Knowledge*.

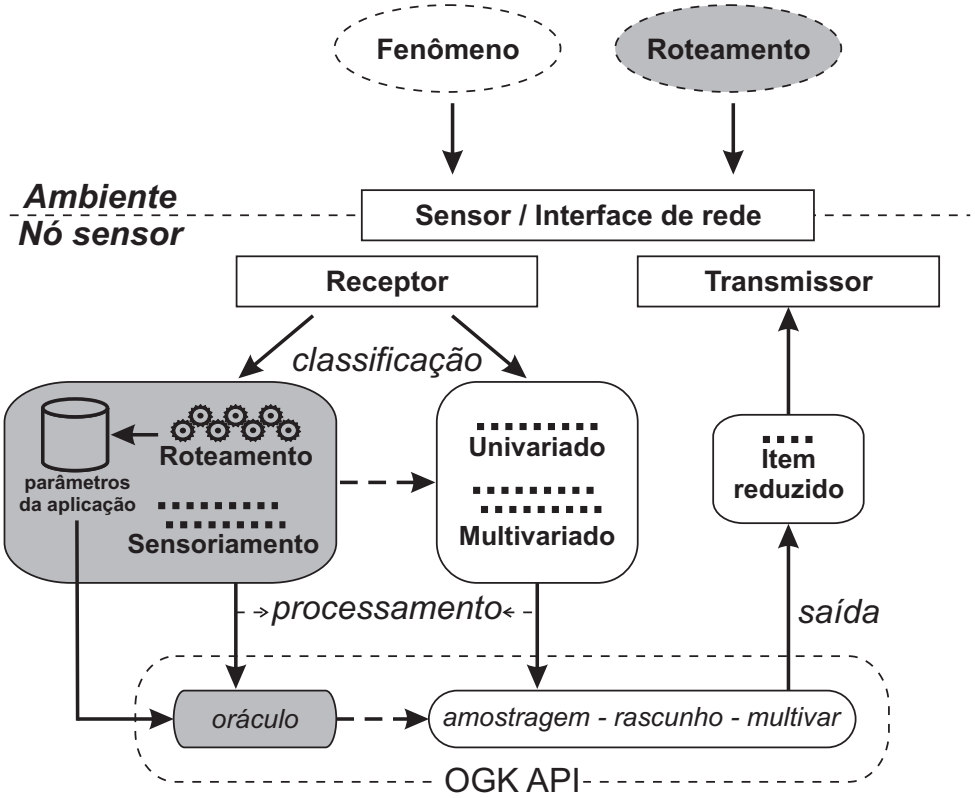


FIGURA 10 – Arquitetura OGK.

é o componente de *hardware* que representa os s sensores, $S = (S_1, \ldots, S_s)$, monitorando um fenômeno e produzindo conjuntos de amostras no domínio $\mathscr{V}_i$, com $1 \leq i \leq s$; onde todos os possíveis conjuntos do domínio são denotados por $\mathscr{V} = (\mathscr{V}_1, \ldots, \mathscr{V}_s)$. Já a *Interface de rede* é um componente de *hardware* que executa as tarefas de envio e de recebimento do item. Antes de ser enviado, o item V é fragmentado em $V = \{V^1, \ldots, V^{n_f}\}$, onde n_f é o número do fragmento. Todos V^j, com $1 < j \leq n_f$, podem encapsular algumas informações da aplicação. Esses fragmentos são enviados e recebidos através da *Interface de rede*. Os componentes de *software* que podem ser encontrados fora da camada de aplicação são: o *Receptor* que recebe o item V e o direciona para os componentes de classificação presentes nas aplicações de redução; e o *Transmissor* que, se necessário, agrega as informações da aplicação e envia o item *stream* reduzido.

Uma vez que o item V tenha chegado na camada de aplicação, alguma redução Ψ é aplicada. Para compor Ψ, a arquitetura considera três funções básicas, apresentadas na figura 10 como *classificação* (ψ_c), *processamento* (ψ_p) e *saída* (ψ_s). A função ψ_c é responsável por identificar se o item V é proveniente de um *Fenômeno* ou do *Roteamento*, pois para cada origem do item temos um tratamento diferenciado. Por exemplo, um item originado do *Roteamento* pode chegar à aplicação em pequenos

fragmentos exigindo um tratamento diferenciado, tal como, esperar que todos ou parte dos fragmentos cheguem.

Após o item ser identificado, através da função ψ_c, ele é passado para a função ψ_p que efetua a redução, de acordo com a necessidade da aplicação. Por fim, o item reduzido é passado para a função ψ_s que, se necessário, agrega as informações da aplicação ao item V e o passa para o componente *Transmissor*. Com isso, temos na arquitetura OGK, a função geral de redução

$$\Psi = \psi_c \circ \psi_p \circ \psi_s.$$

É fácil identificar que devemos possuir tratamentos distintos para o item V. Com isso, temos as funções Ψ_F e Ψ_R, que correspondem, respectivamente, às funções de redução para as aplicações que utilizam itens originados do *Fenômeno* e do *Roteamento*. Inicialmente considere Ψ_F onde as funções que a compõem são detalhadas a seguir:

- *Classificação* (ψ_c): A primeira decisão a ser tomada ao se projetar uma aplicação de redução é: *Qual é o tipo do item stream presente na nossa solução?* Na nossa arquitetura os dados das aplicações gerais são representados por um *stream Univariado* ou *Multivariado*. Com isso, essa questão pode ser respondida a priori pelo projetista da aplicação, onde o tipo do item *stream* identificado é utilizado por todo o tempo de vida da aplicação. Ou pode ser respondida dentro da rede através de algum mecanismo de inferência que identifique o tipo do item *stream*.

- *Processamento* (ψ_p): Uma vez que o tipo do item *stream* é definido precisamos responder a questão: *Qual é o mecanismo de redução mais apropriado para reduzirmos o item stream de forma eficiente em relação aos requisitos da rede e da aplicação?* O item *stream* pode ser processado pela API-OGK, que é composta por algoritmos que permitem a redução de itens univariados ou multivariados. Cada algoritmo tem parâmetros específicos que podem ser definidos a priori ou de forma *online*. Como na função anterior, a escolha e configuração do algoritmo que pode ser utilizado é determinada pelo projetista ou implementada utilizando algum mecanismo de inferência "inteligente", como a escolha, por parte dos nós, da quantidade de dados a serem reduzidos.

- *Saída* (ψ_s): Após reduzir o item *stream* o novo dado deve ser enviado até o

sorvedouro de forma *ad-hoc*. No entanto algumas informações da aplicação podem ser necessárias pelos nós que irão rotear o novo item. É através dessa função que essas informações são agregadas aos itens e em seguida passadas para o componente *Transmissor*.

Agora considere Ψ_R onde as funções que a compõem são detalhadas a seguir:

- *Classificação* (ψ_c): As aplicações que suportam esse tipo de redução possuem simultaneamente itens provenientes do *Roteamento* e do *Fenômeno*. Os itens originados do *Fenômeno*, considerados aqui como itens de *Sensoriamento*, são passados diretamente para o próximo passo de classificação (unvariado ou multivariado). No entanto, os itens provenientes do *Roteamento* chegam fragmentados no formato $V = \{V^1, \ldots, V^{n_f}\}$, os nf fragmentos devem ser aguardados para assim serem passados para o próximo passo da classificação. Em ambos os casos, ao chegar no passo de classificação seguinte, a aplicação se comporta como na redução para os dados do tipo *Fenômeno*. Uma vez que os dados são de *Roteamento*, algumas informações da aplicação, agregadas aos fragmentos, são lidas e armazenadas no banco de dados de *parâmetros da aplicação*, para serem utilizadas no momento da redução. Um ponto importante é que essa função (ψ_c) não pode ser determinada a priori, e se utilizada, deve ser implementada e executada dentro da rede.

- *Processamento* (ψ_p): Uma vez que o tipo do item *stream* é definido e as informações das aplicações são obtidas, é necessário identificar qual a melhor solução de redução para a aplicação corrente. Essa decisão é tomada baseada nos requisitos da aplicação e da rede e as informações necessárias para essa decisão devem estar presentes no banco de dados de *parâmetros da aplicação*. Essa função é implementada na API-OGK. Basicamente utilizando os parâmetros da aplicação o oráculo escolhe e configura o melhor algoritmo de redução disponível na API.

- *Dados de saída* (ψ_s): Assim como na função de redução Ψ_F, as informações da aplicação são agregadas aos dados reduzidos e em seguida passadas para o componente *Transmissor*.

Como observado acima a função ψ_p, referente ao processamento do item *stream*, é composta por um conjunto de algoritmos presentes na API-OGK. Tais algoritmos serão detalhados e apresentados na próxima seção.

4.2 Algoritmos de redução

Nesta seção discutiremos o projeto e a implementação dos algoritmos de redução de dados baseados nas técnicas de *stream* de dados que fazem parte da API-OGK e são utilizados pela função ψ_p, referente ao processamento do item *stream*. É importante destacar que devido às exigências das aplicações e às limitações encontradas nos nós sensores, os algoritmos nesses dispositivos devem ser simples e eficientes em relação ao processamento, espaço e comunicação.

4.2.1 OGK-*amostragem*

O algoritmo OGK-*amostragem*, como o nome sugere, é baseado na técnica de *stream* de dados baseada em amostragem. De forma geral, a técnica de amostragem processa cada item *stream*, aplicando algum tipo de seleção dos dados mais significativos. A idéia principal é obter amostras suficientes para podermos representar o fenômeno monitorado. O nosso objetivo, em utilizar essa técnica, é maximizar a relação qualidade dos dados vs. manutenção da rede. Na nossa implementação, o tamanho do conjunto de amostras pode ser regulado de forma *online*, porém ele precisa ser representativo de tal forma que possamos tomar decisões coerentes nas aplicações (D').

O algoritmo OGK-*amostragem* possui duas variantes: *aleatório* que faz a escolha das amostras de forma aleatória; e *central* que escolhe os elementos centrais das colunas do histograma gerado a partir dos dados originais. Considerando V o item *stream* de entrada, o algoritmo OGK-*amostragem* segue os seguintes passos (figura 11):

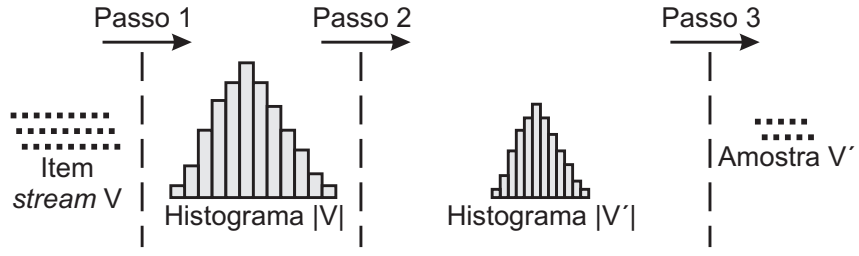


FIGURA 11 – Passos utilizados para o processamento do *stream* no algoritmo OGK-*amostragem*.

Passo 1 Construção do histograma a partir do item original V. No momento da construção do histograma o número de classes utilizadas interfere na qualidade do resultado.

Passo 2 (*aleatório*) Criação de um novo histograma com o tamanho $|V'|$, a partir do histograma obtido no passo 1. Esse novo histograma possui a mesma distribuição de freqüência do original e os valores que irão compor suas colunas são escolhidos aleatoriamente no histograma original. Esses valores são utilizados para compor V'.

Passo 2 (*central*) Criação de um novo histograma com o tamanho $|V'|$, a partir do histograma obtido no passo 1. Para criar esse novo histograma, foram escolhidos os elementos centrais de cada classe do histograma original. A amostra resultante V' será representada pelo novo histograma.

Passo 3 Ordenação de V' de acordo com a ordem de chegada de V.

O pseudo-código do algoritmo OGK-*amostragem* pode ser visto no algoritmo 1.

Algoritmo 1: Pseudo-código do algoritmo de amostragem.

Entrada: V – item *stream* original
Entrada: $|V'|$ – tamanho da amostra resultante
Saída: V' – amostra resultante

```
1  início
2      Ordene(V)
3      lg ← "Largura de cada classe do histograma"
4      pr ← 0 {primeiro índice da primeira classe do histograma}
5      n_c ← 0 {número de elementos, por classe do histograma}
6      k ← 0
7      para i ← 0 até |V| − 1 faça
8          se V[i] > V[pr] + lg ou i = |V| − 1 então
9              n'_c ← ⌈n_c |V'|/|V|⌉ {número de elementos por coluna em V'}
10             indice ← "Escolha do índice seguindo o passo 2 da função ψ_p"
11             para j ← 0 até n'_c faça
12                 V'[k] ← V[indice]
13                 k ← k + 1
14                 indice ← "Escolha do índice seguindo o passo 2 da função ψ_p"
15             fim
16             n_c ← 0
17             pr ← i
18         fim
19         n_c ← n_c + 1
20     fim
21     Ordene(V') {de acordo com a ordem original}
22 fim
```

No algoritmo 1 temos duas possibilidades para a execução das linhas 10 e 14, que representam a escolha das amostras que irão compor o item V' de saída. No caso da escolha aleatória, temos para ambas as linhas

$$indice \leftarrow Aleatoria(pr, pr + n'_c),$$

onde a função $Aleatoria$, retorna algum número inteiro entre $[pr, pr+n'_c]$. Já no caso da escolha pelos elementos centrais da coluna do histograma, temos na linha 10

$$indice \leftarrow pr + \lceil (n_c - n'_c)/2 \rceil$$

e na linha 14 temos $indice \leftarrow indice + 1$. Fazendo a análise de complexidade do algoritmo 1 temos:

Linha 2 Executa em $O(|V| \log |V|)$;

Linhas 3–6 Correspondem à inicialização das variáveis.

Linhas 11–15 Definem o laço interno que determina o número de elementos de cada classe do histograma da amostra resultante. Considere, H_{cn} o número de classes dos histogramas. O tempo de execução do laço interno é de $O(|V'|)$, onde na linha 11 $n'_c = |V'| \leftrightarrow H_{cn} = 1$, ou seja, teríamos uma única classe no histograma da amostra com $|V'|$ elementos a serem percorridos.

Linhas 7–20 Definem o laço externo onde a entrada dos dados é lida e os elementos da amostra são escolhidos. H_{cn} é o número de classes dos histogramas. Antes da linha 8 ser aceita, executamos n_c vezes o laço externo, o que corresponde a contagem do número de elementos de uma classe do histograma original (linha 19). Após a condição da linha 8 ser aceita, o laço mais interno é executado n'_c vezes. A condição da linha 8 é aceita apenas H_{cn} vezes. Com isso, temos uma execução de $H_{cn}(n_c + n'_c)$ para o laço mais externo. Como $|V| = H_{cn} n_c$ e $|V'| = H_{cn} n'_c$, temos um tempo de execução para o laço externo de $O(|V| + |V'|)$.

Linha 21 Executa em $O(|V'| \log |V'|)$.

Assim, a complexidade do algoritmo OGK-*amostragem* é

$$O(|V| \log |V|) + O(|V| + |V'|) + O(|V'| \log |V'|) = O(|V| \log |V|),$$

já que $|V'| \leq |V|$. A complexidade de espaço é $O(|V| + |V'|) = O(|V|)$ pois armazenamos o item *stream* V e a amostra V' resultante. Já que todo nó envia sua amostra em direção ao sorvedouro, a complexidade de comunicação é $O(|V'|D)$, onde D é a maior rota (em saltos) da rede.

4.2.2 OGK-*rascunho*

O algoritmo OGK-*rascunho* é baseado na técnica de *stream* de dados de rascunho. De forma geral, a técnica de rascunho utiliza informações dos itens *stream*, como mínimo, máximo, média e freqüência para inferir propriedades dos itens. A técnica de rascunho mantém a freqüência dos dados sem perda e utiliza um tamanho fixo de pacote para o envio da informação sensoriada, o que permite economizar recursos da rede. O nosso objetivo, em utilizar essa técnica, é obter a distribuição de freqüência dos dados de tal forma que possamos gerar fora da rede o item *stream* original. Esse algoritmo é direcionado a aplicações cuja decisão D'_{ord}, apresentada no capítulo 3, não é considerada. Considerando V o item *stream* de entrada, o algoritmo OGK-*rascunho* segue os seguintes passos (figura 12):

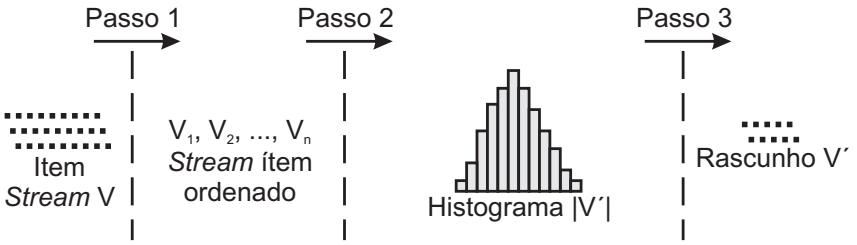


FIGURA 12 – Passos utilizados para o processamento do *stream* no algoritmo OGK-*rascunho*.

Passo 1 Ordenação dos dados e identificação dos valores mínimo e máximo dentro do item *stream* V.

Passo 2 Construção dos dados de saída apenas com as freqüências do histograma.

Passo 3 Montagem do rascunho V' com os dados de saída obtidos no passo 2 e as informações a respeito do histograma obtidas no passo 1.

O pseudo-código do algoritmo OGK-*rascunho* pode ser visto no algoritmo 2 e sua análise de complexidade é apresentada a seguir:

Linha 2 Executa em $O(|V| \log |V|)$.

Linhas 3–7 Correspondem à inicialização das variáveis.

Linhas 8–16 Definem o laço para construção do histograma, executado em $O(|V|)$.

Algoritmo 2: Pseudo-código do algoritmo de rascunho.

Entrada: V – item *stream* original
Saída: V' – rascunho resultante

```
1  início
2      Ordene(V)
3      lg ← "Largura de cada classe do histograma"
4      |V'| ← ⌈(V[|V|] − V[0])/lg⌉
5      pr ← V[0] {primeiro elemento da classe do histograma}
6      c ← 0 {contador}
7      indice ← 0
8      para i ← 0 até |V| − 1 faça
9          se V[i] > pr + lg ou i = |V| − 1 então
10             v'[indice] ← c
11             indice ← indice + 1
12             c ← 0
13             pr ← V[i]
14         fim
15         c ← c + 1
16     fim
17 fim
```

Assim, a complexidade do algoritmo OGK-*rascunho* é

$$O(|V|\log|V|) + O(|V|) = O(|V|\log|V|).$$

A complexidade de espaço é $O(|V| + |V'|) = O(|V|)$, pois armazenamos o item *stream* V e a amostra V' resultante. Já que todo nó envia sua amostra em direção ao sorvedouro, a complexidade de comunicação é $O(|V'|D)$, onde D é a maior rota (em saltos) da rede.

4.2.3 OGK-*multivar*

O algoritmo OGK-*multivar* utiliza em seu favor a técnica de PCA para classificar dados multivariados de acordo com sua correlação e, com isso, identificar onde podemos fazer uma amostragem mais significativa sem comprometer a semântica dos dados. Como estamos escolhendo um conjunto de dados dentro dos dados originais, o OGK-*multivar* tem como base a técnica de *stream* de dados de amostragem. O nosso objetivo, em utilizar essa técnica, é abstrair redundâncias e detalhes pouco significativos para gerar uma amostra de dados que represente as características do conjunto dos itens originais com a perda mínima de informação. Devido à complexidade desse algoritmo sua utilização é mais aceitável em uma rede hierárquica com nós agregadores com maior poder computacional. Considerando o item *stream* de entrada $V = \{V_1, \ldots, V_s\}$, onde s são os diferentes sensores. O algoritmo OGK-*multivar*

segue os seguintes passos (figura 13):

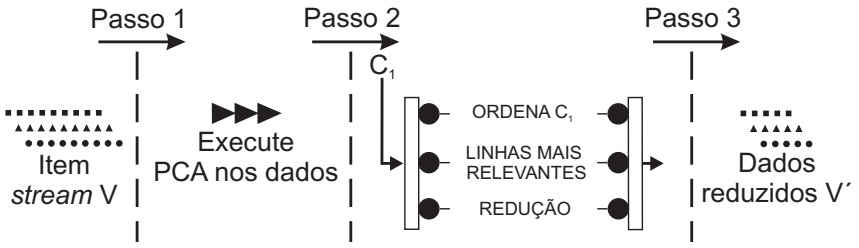


FIGURA 13 – Passos utilizados para o processamento do *stream* no algoritmo OGK-*multivar*.

Passo 1 Cálculo dos componentes principais C do item V.

Passo 2 Seleção do primeiro componente C_1. As posições dos maiores valores em C_1 são usadas para referenciar as posições das linhas em V que irão compor o dado reduzido V'. Os maiores valores em C_1 representam os dados mais correlacionados de V.

Passo 3 Montagem do dado de saída V', que contém as linhas mais significativas de V selecionadas no passo anterior.

O pseudo-código do algoritmo OGK-*multivar* pode ser visto no algoritmo 3 e sua análise de complexidade é apresentada a seguir:

Algoritmo 3: Pseudo-código do algoritmo de redução para dados multivariados

Entrada: V – item *stream* original
Entrada: $|V'_s|$ – tamanho da amostra resultante dos s sensores
Saída: V' – rascunho resultante

```
1 início
2     C ← calculaPca(V)
3     I ← ordena(C[1])
4     I ← ordena(I, |V'_s|)
5     para i ← 1 até |V'_s| faça
6         V'[i] ← V[I[i]]
7     fim
8 fim
```

Linha 2 Efetua o cálculo dos componentes principais, através do $calculaPca()$. A ordem de complexidade do cálculo do PCA, de forma geral, pode ser estimada em $O(m^2m' + m^2n)$, onde m se refere à dimensão dos dados originais†, m' é

†No nosso caso a dimensão dos dados corresponde ao número de sensores s.

a dimensão dos dados reduzidos e n o tamanho dos dados originais. Entretanto, se $m > m'$ e $m > n$, a ordem de complexidade pode ser estimada em $O(m^2)$ (SHARMA; PALIWAL, 2007). Como no nosso caso $m = m' = s$, $m < n$ e $n = |V_s|$, temos $O(s^3 + s^2|V_s|)$.

Linha 3 Ordenação do vetor com primeiro componente $C[1]$, onde os índices I dos valores mais correlacionados são obtidos. A ordem de complexidade da ordenação é $O(|V_s| \log |V_s|)$, uma vez que $|C[1]| = |V_s|$.

Linha 4 Ordenação do vetor I considerando apenas os primeiros índices $|V'_s|$. Isso é necessário para se manter a ordem de chegada dos elementos escolhidos para V'. A ordem de complexidade da ordenação é $O(|V'_s| \log |V'_s|)$.

Linhas 5–7 Montagem dos dados de saída, cuja ordem de complexidade é de $O(|V'_s|)$.

Assim, a complexidade do algoritmo OGK-*multivar* é

$$O(s^3 + s^2|V_s|) + O(|V_s| \log |V_s|) + O(|V'_s| \log |V'_s|) + O(|V'_s|) = O(s^3 + s^2|V_s|).$$

Para a complexidade de espaço considere as matrizes V, V', C e Σ, que correspondem respectivamente, ao item *stream* de entrada, amostra de saída, componentes principais e a matriz de covariância. A complexidade de espaço é dada por $3\,O(s|V_s|) + O(s|V'_s|) = O(s|V_s|)$. Uma vez que cada nó fonte envia V' até o sorvedouro, a complexidade de comunicação é $O(s|V'_s|D)$, onde D é a maior rota na rede.

4.2.4 OGK-*oráculo*

O algoritmo OGK-*oráculo*, na atual versão da API-OGK, é utilizado apenas nas funções de redução que consideram dados recebidos do roteamento e é responsável por identificar qual a melhor solução de redução para a aplicação corrente, ou seja, ele escolhe e configura os algoritmos de redução para que eles possam ser utilizados pela aplicação. Para fazer essa escolha, utilizamos informações sobre os requisitos da aplicação e da rede presentes no banco de dados de *parâmetros da aplicação*.

Existem diferentes possibilidades para a implementação do oráculo. No entanto, para os cenários que utilizamos, o OGK-*oráculo* é integrado à fase de encaminhamento do algoritmo de roteamento baseado em árvore proposto por Nakamura et al. (2005). Considerando o item V fragmentado em $V = \{V^1, \ldots, V^{n_f}\}$, onde n_f é o número do

fragmento, temos como entrada o fragmento V^j, com $1 < j \leq n_f$. O algoritmo integrado na fase de encaminhamento do roteamento segue os seguintes passos (figura 14):

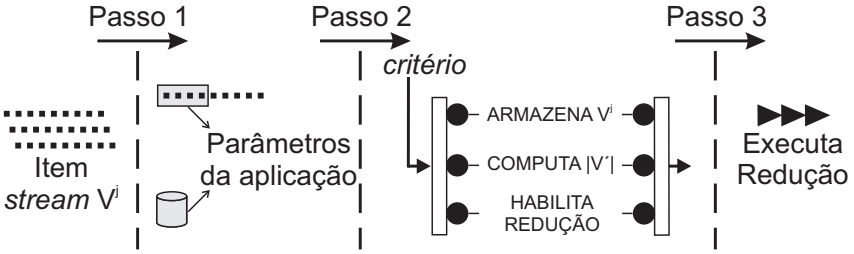


FIGURA 14 – Passos utilizados para o processamento do *stream* no algoritmo OGK-*oráculo*.

Passo 1 Recebe o fragmento V^j para rotear.

Passo 2 Quando V^1 é recebido, os critérios de redução de dados são verificados. Se satisfeito, V^1 é armazenado e $\{V^2, \ldots, V^{n_f}\}$ são recebidos e armazenados. Caso contrário, todos os fragmentos são re-encaminhados para o sorvedouro.

Passo 3 $|V'|$ é computado baseado nos *Parâmetros da aplicação* e a redução é ativada após todos os fragmentos chegarem.

Passo 4 Redução e encaminhamento do dado reduzido seguindo um dos algoritmos de redução apresentados anteriormente. Depois da redução todo o item *stream* V é descartado.

O pseudo-código do algoritmo OGK-*oráculo* pode ser visto no algoritmo 4. O tempo de execução do oráculo está relacionado ao algoritmo de redução utilizado.

4.3 Conclusões parciais

Este capítulo apresentou a arquitetura OGK utilizada para as aplicações de redução de dados baseadas em técnicas de *stream* de dados. Essa arquitetura utiliza o conhecimento a respeito do *stream* de sensoriamento escolhendo a melhor solução de redução para as aplicações gerais em redes de sensores sem fio. Como pode ser observado, através da proposta de utilização da arquitetura OGK para projetar soluções de redução de dados em aplicações gerais de redes de sensores, do ponto de vista conceitual, é possível aceitar a hipótese 1 (pg. 55). Essa hipótese diz respeito ao projeto de uma solução genérica onde é possível aplicar a redução de dados Ψ e está

Algoritmo 4: Pseudo-código do algoritmo da fase de encaminhamento.

Entrada: V^j – fragmento do item *stream* recebido

```
1  início
2      "Obtenha de V^j as informações do fragmento"
3      se j = 1 então
4          "O critério é calculado"
5          se critério aceito então
6              "Ativar o armazenamento do item V"
7              |V'| ← "Cálculo do tamanho dos dados de saída"
8          fim
9      fim
10     se Armazenamento ativo então
11         "Armazene V^j"
12         se j = n_f então
13             V' ← "Execute um OGK-algoritmo de redução em V com o tamanho de |V'|"
14             "Envie V'"
15             "Descarte V"
16         fim
17     senão
18         "Encaminhe V^j"
19     fim
20 fim
```

diretamente relacionada ao problema apresentado na definição 1 (pg. 53) relacionado a redução de dados em redes de sensores baseada nas técnicas de *stream* de dados.

Uma análise empírica da utilização da arquitetura para aplicações gerais será feita nos capítulos seguintes. Especificamente, no próximo capítulo trataremos do problema de redução de dados no momento do sensoriamento.

5 REDUÇÃO DE DADOS NO SENSORIAMENTO

"*Se enxerguei mais longe foi porque me apoiei nos ombros de gigantes.*" (Isaac Newton)

ESTE capítulo apresenta um estudo, baseado em simulação, a respeito da redução de dados baseada nas técnicas de *stream* no momento do sensoriamento. Nesse estudo consideramos diferentes cenários onde os dados que representam os fenômenos monitorados pelos nós sensores podem ser univariados ou multivariados.

Os resultados são utilizados para aceitar a hipótese 2 (pg. 55), que diz respeito à economia de recursos da rede sem perder a qualidade nos dados através da redução de dados em redes planas. Além disso, o estudo apresentado sobre a arquitetura é reforçado, com o objetivo de podermos aceitar a hipótese 1 (pg. 55) que diz respeito ao projeto de uma solução genérica onde é possível aplicar à redução de dados Ψ. Ambas hipóteses são derivadas do problema relacionado à redução de dados em redes de sensores sem fio baseada nas técnicas de *stream* de dados apresentado na definição 1 (pg. 53).

5.1 Redução de dados univariados

Para efetuar a redução de dados univariados consideramos as aplicações gerais com o seguinte comportamento de sensoriamento

$$\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow D.$$

Nessas aplicações quanto mais leituras tivermos no conjunto $\mathscr{V}$ mais precisa e eficiente pode ser a decisão D. Alguns exemplos de aplicações gerais que utilizam dados univariados são: monitoramento de incêndios, detecção de enchentes, detector de poluição e agricultura de precisão. De uma forma geral, nesse tipo de aplicação os nós são distribuídos de forma aleatória e os eventos monitorados podem ser enviados para o sorvedouro periodicamente.

Como apresentado anteriormente para cada tipo de aplicação devemos definir como o item *stream* deve ser representado, ou seja, como representar os valores do conjunto $\mathscr{V}_i = \{V_1, V_2, \ldots\}$. Sendo assim, para as nossas aplicações gerais com dados univariados temos o *stream* $\mathscr{V} = \mathscr{V}_e$ que é descrito pelo item V representado por um conjunto $V = \{v_1, \ldots, v_n\}$, onde n é o número de leituras do ambiente e v é no formato numérico tal que $v \in \mathbb{R}$, $0 \leq v \leq 1$ e segue uma distribuição normal com os parâmetros $\mu = 0.5$ e $\sigma = 0.1$, tais parâmetros foram escolhidos de forma arbitrária. A leitura e o processamento de V ocorre quando n itens forem obtidos. O tamanho máximo do pacote suportado pela aplicação numa rede de sensores pode ser definido por pct_t, com isso, temos $|V'| > pct_t \rightarrow V' = \{V'^1, \ldots, V'^{n_f}\}$, onde n_f é o número de fragmentos. Por convenção, utilizamos em todo o trabalho $pct_t = 20$.

As aplicações gerais onde os item são dados univariados, modeladas utilizando a arquitetura OGK, possuem uma função de redução Ψ_F composta pelas seguintes funções:

- *Classificação* (ψ_c): Como apresentado acima utilizamos a priori dados *Univariados* para representar o *stream* $\mathscr{V}^*$.
- *Processamento* (ψ_p): Utilizamos diretamente sobre o item *stream* os algoritmos definidos na API para dados univariados (OGK-*amostragem*$_{aleatório}$ e OGK-*rascunho*). Esses algoritmos, bem como seus parâmetros, foram definidos a priori. Para o OGK-*amostragem*$_{aleatório}$ consideramos diferentes tamanhos para o conjunto de amostras $|V'| = \{|V|/2, \log|V|\}$ e para o OGK-*rascunho*, fixamos o tamanho do rascunho $|V'| = pct_t$, por ser o tamanho máximo suportado para um pacote de dados.
- *Dados de saída* (ψ_s): O item reduzido V', sem informações adicionais é enviado até o sorvedouro de forma *ad-hoc*. Nos casos onde $|V'| > pct_t$ os V'^j, com $1 < j < n_f$, serão enviados seqüencialmente.

As simulações realizadas, para aceitar a hipótese 2 (pg. 55), são baseadas nas seguintes considerações:

- A nossa avaliação foi feita na ferramenta de simulação NS-2 (*Network Simulator 2*) versão 2.32[†]. Cada cenário simulado foi executado em 33 diferentes topolo-

[†]Informações a respeito da ferramenta de simulação NS podem ser encontradas no sítio: `http://nsnam.isi.edu/nsnam/index.php/Main_Page`, visitado em janeiro de 2008.

gias aleatórias. Ao fim, para cada cenário, apresentamos os resultados utilizando a média das 33 execuções com intervalo de confiança de 95%.

- Utilizamos um algoritmo de roteamento baseado em árvore chamado EF-Tree (NAKAMURA et al., 2005). A densidade da rede é mantida constante e todos os nós têm a mesma configuração de *hardware*. A árvore é construída apenas uma vez na fase de estabelecimento da rede, pois o consumo para montagem da árvore pode interferir nós resultados.

- Os parâmetros de configuração variados são: número de nós na rede, tamanho do item *stream* em bytes e número de nós monitorando o ambiente. Especificamente para o OGK-$amostragem_{aleatório}$ analisamos o comportamento da rede e a qualidade dos dados (decisões D') para cada parâmetro de configuração utilizando um conjunto de amostras de tamanhos $|V|/2$ e $\log|V|$. Alguns parâmetros importantes utilizados nas simulações são apresentados na tabela 1[†]. Nós variamos o tamanho da rede para manter a densidade da rede constante em $8,48$. Para isso, consideramos a densidade da rede com a relação $\pi r^2 |S|/A$, onde r é o alcance do rádio, $|S|$ é o número de sensores e A é a área da rede. O tamanho da fila suportada por cada nó varia com o tamanho do *stream* para que não haja descarte de pacotes, i.e., cada nó suporta no máximo o tamanho do item *stream* utilizado pela aplicação. O tempo de simulação é 5000s, onde os 1000s iniciais são utilizados para montagem e configuração da rede e da estrutura de roteamento, os 1000s finais são utilizados para permitir que a rede escoe os pacotes restantes na rede. O tráfego real de dados dura 3000s onde 50 *streams* são lançados na rede um a cada 60s como determinado pelo período do *stream*. O alcance do rádio e a largura de banda seguem a especificação do MicaZ[‡]. Por fim, a energia inicial utilizada é 1000, com isso os nós nunca terão suas reservas de energia esgotadas.

5.1.1 Avaliação do comportamento da rede

Inicialmente considere a avaliação do comportamento da rede ao utilizarmos a função de redução Ψ_F como definida anteriormente. Para essa avaliação identificamos o consumo total de energia na rede e a média do atraso, para entregar um pacote partindo dos nós que estão monitorando o ambiente até o sorvedouro. Além disso, utilizamos

[†]Os parâmetros utilizados levam em conta a arquitetura do Mica2.

[‡]`http://www.xbow.com/`

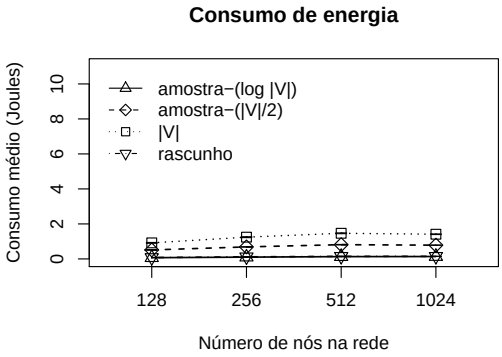
TABELA 1 – Parâmetros de simulação para redução de dados univariados.

Parâmetro	**Valor**
Tamanho da rede	varia com a densidade
Tamanho da fila	varia com $\|V\|$
Tempo de simulação (s)	5000
Tráfego (s)	[1000, 4000]
Período do *stream* (s)	60
Alcance do rádio (m)	50
Largura de banda (kbps)	250
Energia inicial (Joules)	1000
Localização do sorvedouro	coordenadas (0, 0)

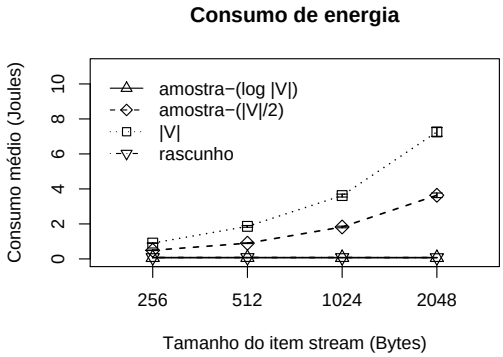
para comparação dos resultados, o comportamento da rede sem utilizar nenhuma redução, ou seja, amostras de tamanho $|V|$. As curvas nas figuras representam os resultados para a utilização dos algoritmo OGK-*rascunho* e OGK-*amostragem*$_{aleatório}$ com $|V'| \in \{\log|V|, |V|/2\}$ e o comportamento da rede sem utilizar redução ($|V|$).

A figura 15 mostra no eixo-y o consumo médio de energia em Joules para a avaliação do comportamento da rede, onde variamos no eixo-x o número de nós na rede em $128, 256, 512$ e 1024, o tamanho do item *stream* em $256, 512, 1024$ e 2048 bytes e o número de nós monitorando o ambiente em $1, 5, 10$ e 20. Para todos os cenários, quando não variados, fixamos os respectivos parâmetros em 128, 256 e 1. Analisando a figura 15 de forma geral, em todos os casos observamos que quando $|V'|$ diminui o consumo de energia também diminui, onde a amostra-($\log|V|$) e a solução de rascunho apresentam o melhor desempenho. Isso ocorre porque para ambos os casos enviamos para o sorvedouro apenas um pacote com no máximo 20 leituras do ambiente ($|V'| = 20$), o que corresponde ao pct_t considerado.

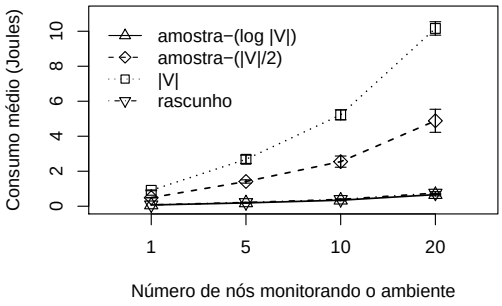
Analisando os resultados separadamente, quando o número de nós na rede varia (figura 15(a)) a energia consumida não varia muito. Isso ocorre porque temos apenas um nó gerando dado, o tamanho do *stream* é constante e a densidade é mantida quando aumentamos o tamanho da rede. O pequeno aumento observado no consumo de energia quando aumentamos o número de nós na rede ocorre devido à densidade constante, pois o caminho que o pacote leva para chegar até o sorvedouro torna-se um pouco maior à medida que aumentamos o número de nós, também aumentando o consumo de energia. A amostra-($\log|V|$) e o rascunho tiveram o menor impacto sobre o consumo de energia porque os nós nesses casos enviaram apenas um pacote para cada item *stream* processado. A amostra-($\log|V|$) terá para cada tamanho de item *stream* ($256, 512, 1024$ e 2048), respectivamente, amostras de 8, 9, 10 e 11 que são menores que o tamanho de pacote ($pct_t = 20$) utilizado. O rascunho, para todos os casos, terá um item reduzido constante de tamanho pct_t. Já para os outros resultados



(a) Diferentes tamanhos de rede.



(b) Diferentes tamanhos do item *stream*.



(c) Diferentes números de fontes.

FIGURA 15 – Avaliação do comportamento da rede, considerando a média de energia consumida na rede ao reduzir dados univariados.

o item *stream* resultante foi fragmentado gerando mais pacotes e, conseqüentemente, mais tráfego e consumo de energia na rede.

Considerando a variação do tamanho do item *stream* (figura 15(b)), a amostra-($\log|V|$) e o rascunho tiveram o melhor desempenho em todos os casos. Apesar de aumentarmos o tamanho da amostra, a energia consumida se manteve quase constante. No caso da amostra-($\log|V|$), isso ocorre porque o tamanho do pacote é incrementado apenas de um elemento quando aumentamos o tamanho do item e no caso do rascunho o tamanho do pacote é sempre constante. Os outros resultados (amostra-($|V|/2$) e $|V|$) tiveram um desempenho pior, pois o tamanho dos pacotes aumenta proporcionalmente ao tamanho do item e mais pacotes são lançados na rede devido à fragmentação.

Quando o número de nós monitorando o ambiente é variado (figura 15(c)), mais uma vez, a amostra-($\log|V|$) e o rascunho obtiveram um melhor desempenho em todos os casos. Isso ocorre porque à medida que aumentamos o número de nós monitorando

o ambiente mais pacotes estão trafegando na rede. Cada nó fonte que usa amostra-($\log|V|$) ou rascunho precisa apenas de um pacote para enviar suas informações para o sorvedouro, pois nesse caso a informação não excede o tamanho máximo do pacote. Nos outros resultados (amostra-($|V|/2$) e $|V|$), cada nó fonte gera mais de um pacote por monitoramento. Logo quando os dados de todos os sensores são propagados até o sorvedouro a rede fica sobrecarregada, causando mais consumo de energia.

Em relação ao atraso nos pacotes, temos a figura 16 que mostra no eixo-y o atraso médio em segundos e no eixo-x a variação do número de nós na rede em $128, 256, 512$ e 1024, o tamanho do item *stream* em $256, 512, 1024$ e 2048 bytes e o número de nós monitorando o ambiente em $1, 5, 10$ e 20. Para todos os cenários, quando não variado, fixamos os respectivos parâmetros em 128, 256 e 1. Como para o consumo de energia, podemos observar que quando o $|V'|$ diminui temos um menor atraso, pela mesma razão do consumo de energia. Os mesmos efeitos do consumo de energia para a variação do número de nós na rede, tamanho do item *stream* em bytes e número de nós monitorando o ambiente são observados no atraso dos pacotes (figuras 16(a)–16(c)). Mais uma vez, em todos os casos a amostra-($\log|V|$) e o rascunho tiveram o melhor desempenho.

5.1.2 Avaliação do comportamento dos dados reduzidos

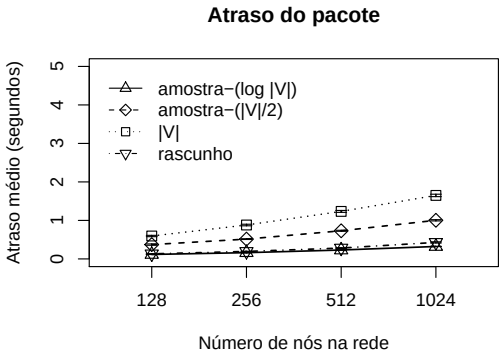
Para a avaliação do comportamento dos dados reduzidos devemos considerar as regras R' presentes no comportamento da redução

$$\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \xrightarrow{\Psi_F} \mathscr{V}' \xrightarrow{R'} D',$$

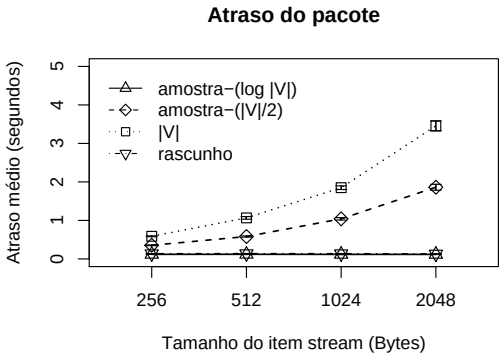
que são R'_{dist}, R'_{val} e R'_{ord} que verificam, respectivamente, se os itens V e V' seguem a mesma distribuição através do teste KS, se a diferença do valor da média de V' quando comparados à média de V são próximas, e se a ordem de chegada dos valores de V' é preservada. Essas regras são utilizadas para tomar as decisões D' nas aplicações gerais. Para essa avaliação, no caso do algoritmo OGK-*amostragem*$_{aleatório}$, utilizamos as R'_{dist}, R'_{val} e R'_{ord}, e no caso do algoritmo OGK-*rascunho* o R'_{ord}. Além disso, as curvas nas figuras representam os resultados para a utilização do algoritmo OGK-*amostragem*$_{aleatório}$ com $|V'| \in \{\log|V|, |V|/2\}$ e o comportamento dos dados sem utilizar redução ($|V|$).

A figura 17 mostra no eixo-y o erro médio em porcentagem[†] ao utilizarmos a regra R'_{dist}

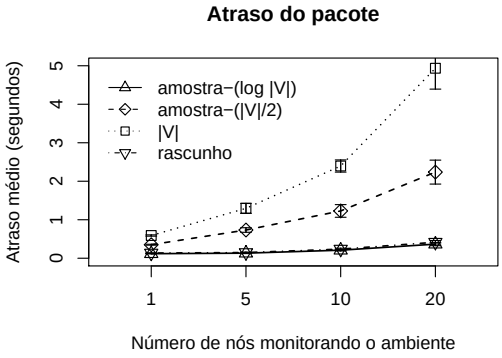
[†]Como o item *stream* V possui valores $v \in \mathbb{R}$, $0 \leq v \leq 1$, podemos considerar o erro em porcentagem.



(a) Diferentes tamanhos de rede.



(b) Diferentes tamanhos do item *stream*.



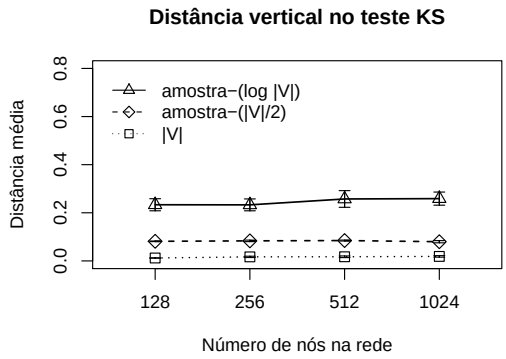
(c) Diferentes números de fontes.

FIGURA 16 – Avaliação do comportamento da rede, considerando a média do atraso do pacote ao reduzir dados univariados.

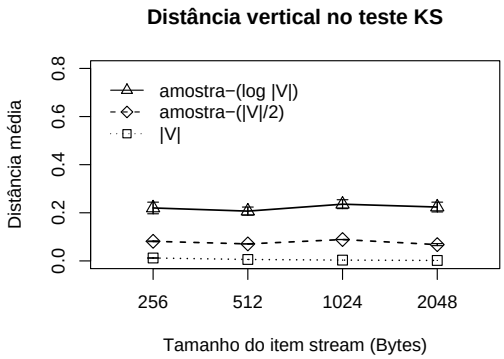
para a avaliação do comportamento dos dados, onde variamos no eixo-x o número de nós na rede em $128, 256, 512$ e 1024, o tamanho do item *stream* em $256, 512, 1024$ e 2048 bytes e o número de nós monitorando o ambiente em $1, 5, 10$ e 20. Para todos os cenários, quando não variado, fixamos os respectivos parâmetros em 128, 256 e 1.

De forma geral, os valores encontrados para o erro referente à distância vertical do teste KS foram de 20% para a amostra-$(\log|V|)$ e 10% para a amostra-$(|V|/2)$. Em todos os casos, o erro é constante porque a perda de pacotes na rede é muito pequena, ou seja, tudo que é enviado chega ao sorvedouro. O maior erro ocorre quando utilizamos a amostra-$(\log|V|)$, porém a similaridade entre os dados ainda é preservada, ou seja, em todos os casos a regra R'_{dist} nos leva a tomar a decisão D'_{dist} de forma correta.

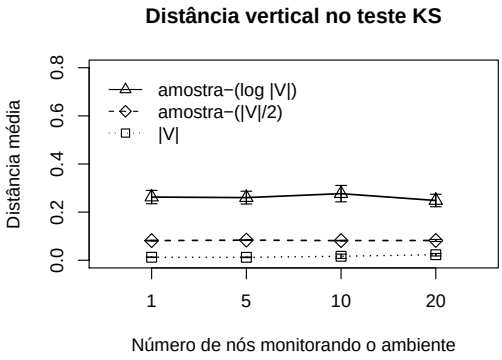
A figura 18 mostra no eixo-y o erro médio em porcentagem ao utilizarmos a regra R'_{val} para a avaliação do comportamento dos dados, onde variamos no eixo-x o número de nós na rede em $128, 256, 512$ e 1024, o tamanho do item *stream* em $256, 512, 1024$ e 2048 bytes e o número de nós monitorando o ambiente em $1, 5, 10$ e 20. Para todos os



(a) Diferentes tamanhos de rede.



(b) Diferentes tamanhos do item *stream*.



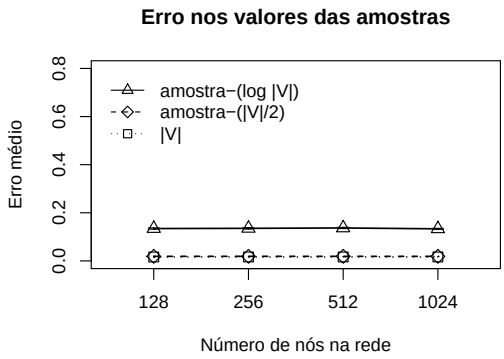
(c) Diferentes números de fontes.

FIGURA 17 – Avaliação do comportamento dos dados reduzidos, considerando o erro médio ao aplicar a regra R'_{dist} sobre os dados univariados.

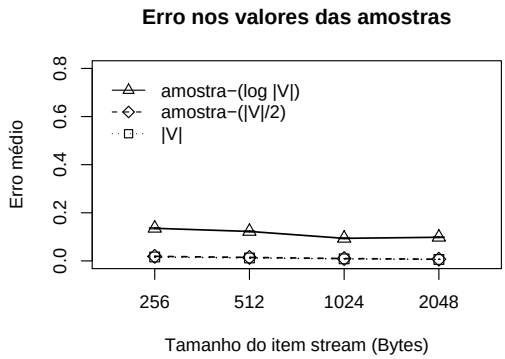
cenários, quando não variado, fixamos os respectivos parâmetros em 128, 256 e 1.

De forma geral, os valores encontrados para o erro referente aos valores das amostras foram de 10% para a amostra-($\log|V|$) e praticamente 0% para a amostra-($|V|/2$). Como na avaliação da regra R'_{dist}, em todos os casos o erro é constante porque a perda de pacotes na rede é muito pequena, ou seja, tudo que é enviado chega ao sorvedouro. Com isso, se quisermos manter o máximo da qualidade do dado considerando apenas a regra R'_{val} podemos utilizar a amostra-$|V|/2$, ou seja, a regra R'_{val} nos leva a tomar a decisão D'_{val} de forma correta.

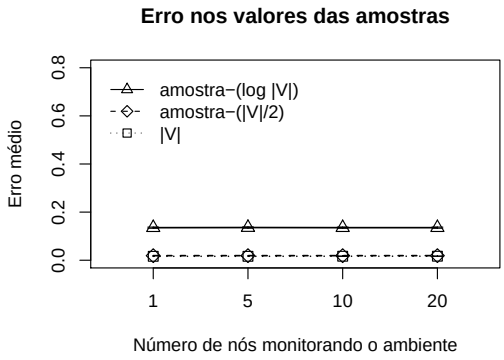
Ao considerarmos a regra R_{ord}, devemos observar apenas o projeto dos algoritmos. No caso do OGK-*amostragem*$_{aleatório}$, é feita uma ordenação do item resultante V', com o objetivo de preservar a ordem de chegada dos elementos escolhidos para compor a amostra. Logo esse algoritmo, quando aplicado às soluções de redução, permite que a decisão D'_{ord} seja tomada de forma correta. Já no caso do algoritmo OGK-*rascunho*, apenas a informação de como V é distribuído é obtida. Os valores dos dados não



(a) Diferentes tamanhos de rede.



(b) Diferentes tamanhos do item *stream*.



(c) Diferentes números de fontes.

FIGURA 18 – Avaliação do comportamento dos dados reduzidos, considerando o erro médio ao aplicar a regra R'_{val} sobre os dados univariados.

são enviados, mas podem ser gerados “artificialmente” no sorvedouro. No entanto, essa geração não garante que os valores sejam na mesma ordem que os originais. Com isso, quando aplicamos o algoritmo OGK-*rascunho* como solução de redução a decisão D'_{ord} não pode ser tomada corretamente.

5.1.3 Comportamento da rede vs. comportamento dos dados reduzidos

Quando analisamos a qualidade dos dados e o comportamento da rede, temos as seguintes conclusões:

- O OGK-*rascunho* quando utilizado apresenta um baixo consumo de energia e atraso pois envia apenas um pacote para o sorvedouro. Uma vez que os dados podem ser gerados “artificialmente” no sorvedouro, a qualidade dos dados não é afetada em relação às regras R'_{dist} e R'_{val}. O problema é a ordem de chegada

dos dados que é perdida, não permitindo aplicar a regra R_{ord}. Porém, a perda na seqüência é aceitável em aplicações onde as restrições da rede são maiores.

- O OGK-*amostragem*$_{aleatório}$, quando utilizado com amostra-($\log|V|$), possui um baixo consumo de energia e atraso pois, para os cenários avaliados, apenas um pacote é enviado para o sorvedouro. Entretanto, a qualidade dos dados é bastante afetada nas regras R'_{dist} e R'_{val} com erros, respectivamente, de 20% e 10%. Porém, assim como no OGK-*rascunho*, essa perda na qualidade pode ser aceitável em aplicações onde as restrições da rede são maiores.

- O OGK-*amostragem*$_{aleatório}$ com amostra-($|V|/2$) pode ser utilizado nos casos onde a prioridade da aplicação de sensoriamento é reduzir o erro quando aplicamos a regra R'_{val} (no nosso caso perto de zero) ou nos cenários descritos pela figura 15(a), em que o tamanho do item *stream* e o número de nós monitorando o ambiente não variam.

- Não é interessante usar nossos algoritmos, ou seja, enviar todo o item V, quando a prioridade da aplicação geral seja reduzir o erro quando aplicamos a regra R'_{dist} ou em casos que não existam fortes restrições de rede.

- Finalmente, *quando usar OGK-amostragem*$_{aleatório}$ *ou OGK-rascunho?* Se a ordem dos dados for importante, podemos utilizar amostragem. Nesse caso, devemos sempre analisar os requisitos da aplicação para decidir qual o melhor tamanho do conjunto de amostras. Caso a seqüência não seja importante podemos utilizar o rascunho pois ele sempre terá melhor desempenho de rede e ainda manterá a qualidade dos dados quando aplicarmos as regras R'_{dist} e R'_{val}.

Os resultados mostram que a solução OGK pode ser utilizada no problema relacionado à redução de dados em redes de sensores baseada nas técnicas de *stream* de dados apresentado na definição 1 (pg. 53). Além disso, através da concepção da arquitetura OGK como solução para a redução de dados em redes planas e dos resultados observados é possível confirmar a hipótese 2 (pg. 55) que diz respeito à economia de recursos da rede sem perder a qualidade nos dados através da redução de dados em redes planas.

5.2 Redução de dados multivariados

Para a redução de dados multivariados também iremos consider as aplicações gerais com o seguinte comportamento de sensoriamento

$$\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow D,$$

onde $\mathscr{V} = \{\mathscr{V}_1, \ldots, \mathscr{V}_s\}$ e s representa os diferentes sensores. Nessas aplicações quanto mais leituras tivermos no conjunto $\mathscr{V}$, mais precisa e eficiente pode ser a decisão D. Um exemplo de aplicação geral que pode ser considerada e que utiliza dados multivariados, é o monitoramento de incêndios onde a temperatura, pressão e umidade são utilizados. De uma forma geral, nesse tipo de aplicação os nós são distribuídos de forma aleatória e os eventos monitorados podem ser enviados para o sorvedouro periodicamente.

Para as nossas aplicações gerais com dados multivariados temos o conjunto $\mathscr{V} = \{\mathscr{V}_1, \ldots, \mathscr{V}_s\}$, onde cada $\mathscr{V}_s$ é descrito pelo item V representado por um conjunto $V = \{v_1, \ldots, v_n\}$, onde n é o número de leituras do ambiente e v está no formato numérico tal que $v \in \mathbb{R}$, $0 \leq v \leq 1$ e segue uma distribuição normal com os parâmetros $\mu = 0.5$ e $\sigma = 0.1$, tais parâmetros foram escolhidos de forma arbitrária. Com isso, o item V pode ser representado como uma matriz $V_{s,n}$, com s colunas e n linhas. A leitura e o processamento de V ocorre quando n itens de todos os s sensores forem lidos do ambiente.

As aplicações gerais com dados multivariados, modeladas utilizando a arquitetura OGK, possuem uma função de redução Ψ_F composta pelas seguintes funções definidas a seguir:

- *Classificação* (ψ_c): Utilizamos a priori dados multivariados para representar o *stream* $\mathscr{V}^*$.
- *Processamento* (ψ_p): Utilizamos diretamente sobre os itens *stream* o algoritmo definido na API para dados multivariados (OGK-*multivar*) e definimos a priori diferentes tamanhos para o conjunto de amostra $|V'_s| = \{|V_s|/2, \log|V_s|\}$.
- *Dados de saída* (ψ_s): O dado reduzido $V' = \{V'_1, \ldots, V'_s\}$, sem informações adicionais é enviado até o sorvedouro de forma *ad-hoc*. Nos casos onde $|V'| > pct_t$ os V'^j, com $1 < j < n_f$, serão enviados seqüencialmente.

Para aceitar a hipótese 2 (pg. 55), avaliamos o comportamento da rede, via simulação,

e o comportamento dos dados via execução do algoritmo de forma *offline*. Ambas avaliações serão descritas e detalhadas a seguir.

5.2.1 Avaliação do comportamento da rede

As simulações realizadas para a avaliação do comportamento da rede, quando reduzimos dados multivariados, são baseadas nas seguintes considerações:

- A nossa avaliação foi feita na ferramenta de simulação NS-2 (*Network Simulator 2*) versão 2.32. Cada cenário simulado foi executado em 33 diferentes topologias aleatórias. Ao fim, para cada cenário apresentamos os resultados utilizando a média das 33 execuções com intervalo de confiança de 95%.

- Utilizamos um algoritmo de roteamento baseado em árvore chamado EF-Tree. A densidade da rede é mantida constante e todos os nós têm a mesma configuração de *hardware*. A árvore é construída apenas uma vez na fase de estabelecimento da rede, pois o consumo para montagem da árvore pode interferir nos resultados.

- Os parâmetros de configuração variados são: número de nós na rede e tamanho do item *stream* em bytes. Para cada parâmetro de configuração analisamos o comportamento da rede utilizando um conjunto de amostras de tamanhos $|V_s|/2$ e $\log|V_s|$[†]. Alguns parâmetros importantes utilizados nas simulações são apresentados na tabela 2. Nós variamos o tamanho da rede para manter a densidade da rede constante em $8,48$. Para isso, consideramos a densidade da rede com a relação $\pi r^2 |S|/A$, onde r é o alcance do rádio, $|S|$ é o número de sensores e A é a área da rede. O tamanho da fila suportada por cada nó varia com o tamanho do *stream* para que não haja descarte de pacotes, i.e., cada nó suporta no máximo o tamanho do item *stream* utilizado pela aplicação. O tempo de simulação é 1100s, onde os 500s iniciais são utilizados para montagem e configuração da rede e da estrutura de roteamento, os 500s finais são utilizados para permitir que a rede escoe os pacotes restantes na rede. O tráfego real de dados dura 100s onde 10 *streams* são lançados na rede um a cada 10s como determinado pelo período do *stream*. O alcance do rádio e a largura de banda seguem a especificação do MicaZ[‡]. Por fim, a energia inicial utilizada é 100, com isso os nós nunca terão suas reservas de energia esgotadas.

[†]Para simplificar a apresentação dos resultados utilizamos nos gráficos $|V|$ ao invés de $|V_s|$.
[‡]`http://www.xbow.com/`

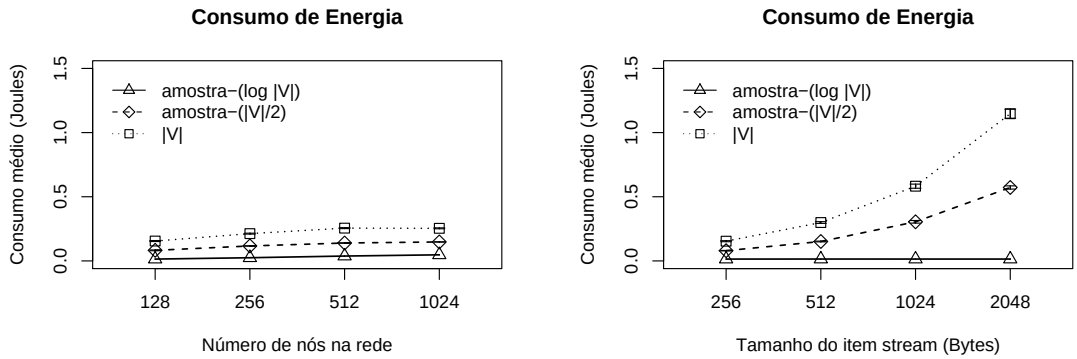
TABELA 2 – Parâmetros de simulação para redução de dados multivariados.

Parâmetro	**Valor**
Tamanho da rede	varia com a densidade
Tamanho da fila	varia com $\|V_s\|$
Tempo de simulação (s)	1100
Tráfego (s)	[500, 600]
Período do *stream* (s)	10
Alcance do rádio (m)	50
Largura de banda (kbps)	250
Energia inicial (Joules)	100
Localização do sorvedouro	coordenadas (0, 0)

Inicialmente considere a avaliação do comportamento da rede ao utilizarmos a função de redução Ψ_F definida acima nas aplicações gerais, onde fixamos o número de sensores $s = 5$ em um único nó monitorando o ambiente. Para esta avaliação identificamos o consumo total de energia na rede e a média do atraso para entregar um item *stream* do nó que está monitorando o ambiente até o sorvedouro. Além disso, utilizamos para comparação dos resultados sem utilizar nenhuma solução de redução ($|V_s|$ amostras). As curvas nas figuras 19 e 20 representam os resultados para a utilização do algoritmo OGK-*multivar* com $|V'| \in \{\log|V|, |V|/2\}$ e o comportamento da rede sem utilizar redução ($|V|$).

A figura 19 mostra no eixo-y o consumo médio de energia em Joules. Para a avaliação do comportamento da rede, variamos no eixo-x o número de nós na rede em 128, 256, 512 e 1024 e o tamanho do item *stream* em 256, 512, 1024 e 2048 bytes. Para todos os cenários, quando não variados, fixamos os respectivos parâmetros em 128 e 256. Analisando de forma geral, em todos os casos quando $|V'|$ diminui o consumo de energia também diminui. Isso ocorre porque na amostra-($\log|V|$) utilizamos menos pacotes, ao fragmentar V', no envio para o sorvedouro.

Analisando os resultados separadamente temos na figura 19(a) que à medida que aumentamos o número de nós o consumo de energia na rede não varia muito. Isso ocorre porque apenas um item com tamanho fixo está trafegando na rede. No entanto, podemos observar um pequeno aumento no consumo para cada $|V'|$. Isso ocorre porque mais nós estão sendo requisitados para o encaminhamento dos pacotes até o sorvedouro, logo mais energia na rede será consumida. Já na figura 19(b) o consumo de energia varia bastante quando aumentamos o tamanho do item *stream*. Isso ocorre porque quanto maior for $|V'|$ mais tráfego é inserido na rede e, conseqüentemente, mais energia será gasta no processamento dos pacotes. Para a amostra-($\log|V|$) não observamos esse aumento, pois o tráfego gerado no incremento do item *stream* é pequeno, por exemplo, para $V_{5,256}$, apenas $5 \times \log 256 = 40$ elementos são enviados

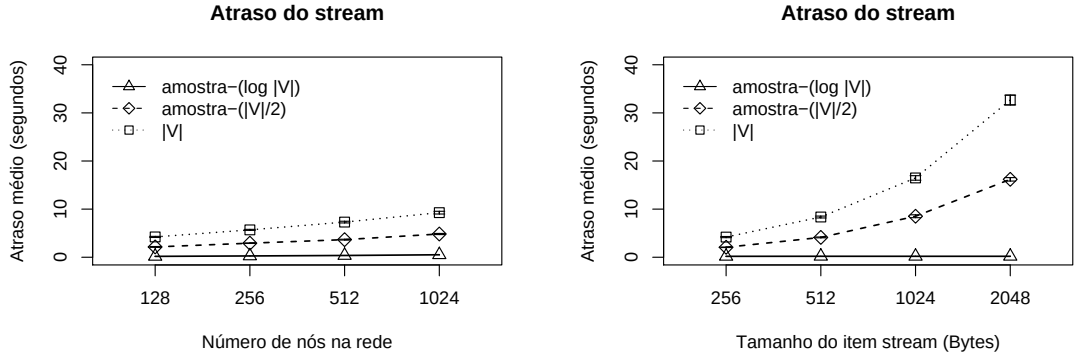


(a) Diferentes tamanhos de rede. (b) Diferentes tamanhos do item *stream*.

FIGURA 19 – Avaliação do comportamento da rede, considerando a média de energia consumida na rede ao reduzir dados multivariados.

em dois pacotes e para $V_{5,2048}$, apenas $5 \times \log 2048 = 55$ elementos são enviados em três pacotes.

A figura 20 mostra no eixo-y o atraso médio, em segundos, para enviar até o sorvedouro o *stream* reduzido. Variamos no eixo-x o número de nós na rede em $128, 256, 512$ e 1024 e o tamanho do item *stream* em $256, 512, 1024$ e 2048 bytes. Para todos os cenários, quando não variados, fixamos os respectivos parâmetros em 128 e 256. Como nos resultados para o consumo de energia, podemos observar que quando o $|V'|$ diminui temos um menor atraso, pela mesma razão do consumo de energia. Os mesmos efeitos observados para o consumo de energia na variação do número de nós na rede e tamanho do item *stream* em bytes são identificados no atraso do *stream* (figuras 20(a) e 20(b)). Mais uma vez, em todos os casos a amostra-($\log |V|$) tem o melhor desempenho.



(a) Diferentes tamanhos de rede. (b) Diferentes tamanhos do item *stream*.

FIGURA 20 – Avaliação do comportamento da rede, considerando a média do atraso do *stream* ao reduzir dados multivariados.

5.2.2 Avaliação do comportamento dos dados reduzidos

Agora considere a avaliação do comportamento dos dados reduzidos ao utilizarmos a função de redução Ψ_F definida acima nas aplicações gerais com dados multivariados. Basicamente iremos observar a regra R'_{val} no comportamento da redução

$$\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow \mathscr{V}' \overset{R'_{val}}{\longrightarrow} D'.$$

sendo que essa regra será aplicada para cada item V'_s originados nos s sensores, ou seja, em $V_1, \ldots, V_s$, o erro quantificado será o maior encontrado entre os s sensores. Essa regra é utilizada para tomar a decisão D'_{val} nas aplicações gerais.

As avaliações realizadas para o comportamento da rede quando reduzimos dados multivariados são baseadas nas seguintes considerações:

- Realizamos a avaliação através de simulações offline e implementamos o algoritmo em C++. Cada cenário simulado foi executado com 33 diferentes entradas geradas de forma aleatória. No fim, para cada cenário, traçamos um gráfico dos valores médios do erro observado nos dados reduzidos com um intervalo de confiança de 95%.

- Variamos os parâmetros tamanho dos dados sensoriados ($|V_s|$) e número de sensores (s) e para cada parâmetro avaliado, aplicamos as reduções $|V_s|/2$, $|V_s|/3$ e $\log|V_s|$[†].

A figura 21 mostra no eixo-y o maior erro médio em porcentagem entre os itens V_s ao utilizarmos a regra R'_{val} para a avaliação do comportamento dos dados, onde variamos no eixo-x o número de dados monitorados por cada sensor em $256, 512, 1024$ e 2048 bytes. Além disso, fixamos o número de sensores em $s = 350$. As curvas nas figuras representam os resultados para a utilização do algoritmo OGK-*multivar* com $|V'| \in \{\log|V|, |V|/3, |V|/2\}$.

Podemos observar que à medida que aumentamos a taxa de redução (amostra-($\log|V|$)), percebe-se que o erro médio também aumenta em 20%, ou seja, a qualidade diminui, porém ainda se consegue um nível de qualidade aceitável para diversas aplicações, o que mostra que em todos os casos a regra R'_{val} nos leva a tomar a decisão D'_{val} de forma correta. Além disso, os resultados evidenciam a escalabilidade de nossa

[†]Vale ressaltar que para simplificar a apresentação dos resultados utilizamos nos gráficos $|V|$ ao invés de $|V_s|$

solução, pois à medida que se aumenta a quantidade de dados ($|V|$), o erro se mantém pequeno e praticamente constante. Isso ocorre pelo fato de sempre mantermos a mesma proporção de dados reduzidos o que nos leva a valores em porcentagem bastante próximos. Uma conclusão parcial que podemos ter ao analisar esse cenário é que se precisarmos manter uma alta qualidade nos dados, não precisamos enviar todos os dados, mas utilizar a amostra-($|V|/3$) que obteve um erro próximo a zero.

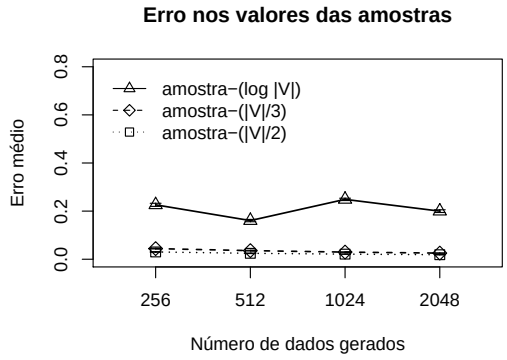


FIGURA 21 – Avaliação do comportamento dos dados reduzidos, considerando o erro médio ao avaliar a regra R'_{val} sobre os dados multivariados com diferentes tamanhos do item *stream*.

A figura 22 mostra no eixo-y o maior erro médio em porcentagem entre os itens V_s ao utilizarmos a regra R'_{val} para a avaliação do comportamento dos dados, onde variamos no eixo-x o número de sensores em $50, 150, 250$ e 350 bytes. Além disso, fixamos o número de dados monitorados em $|V_s| = 1024$. As curvas nas figuras representam os resultados para a utilização do algoritmo OGK-*multivar* com $|V'| \in \{\log|V|, |V|/3, |V|/2\}$.

Como no resultado anterior, à medida que aumentamos a taxa de redução (amostra-($\log|V|$)), o erro médio também aumenta para 25%. Como esse erro é aceitável por um conjunto de aplicações, conclui-se que a regra R'_{val} nos leva a tomar a decisão D'_{val} de forma correta. Esses resultados reforçam a afirmação a respeito da escalabilidade de nossa solução, uma vez que o erro se mantém praticamente constante. Novamente, se precisarmos manter uma alta qualidade nos dados, aconselha-se utilizar a amostra-($|V|/3$) que obteve um erro próximo a zero.

Para as avaliações apresentadas acima, executamos o OGK-*multivar* com diferentes valores para $|V_s|$ e s. O erro médio, em porcentagem, para todos os valores utilizados são apresentados na tabela 3.

Como podemos observar na tabela 3 obtivemos resultados similares aos apresentados anteriormente. Ou seja, à medida que aumentamos a redução dos dados $|V'_s|$ o

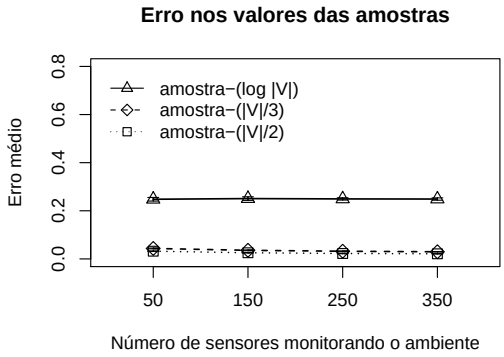


FIGURA 22 – Avaliação do comportamento dos dados reduzidos, considerando o erro médio ao aplicar a regra R'_{val} sobre os dados multivariados com diferentes número de sensores.

TABELA 3 – Erro médio em porcentagem da redução de dados multivariados utilizando a solução OGK.

Sensores	$\|V_s\|$ = 256			$\|V_s\|$ = 512			$\|V_s\|$ = 1024			$\|V_s\|$ = 2048		
	$1/2$	$1/3$	log	$1/2$	$1/3$	log	$1/2$	$1/3$	log	$1/2$	$1/3$	log
50	3.6	5.1	21.7	3.4	4.8	16.6	3.2	4.3	24.8	3.0	4.1	20.1
150	3.3	4.6	22.4	2.8	4.0	16.1	2.6	3.6	25.1	2.3	3.2	19.4
250	3.0	4.3	22.3	2.7	3.8	16.3	2.3	3.2	24.9	2.1	2.8	19.4
350	3.0	4.4	22.6	2.5	3.6	16.1	2.1	3.0	24.9	1.9	2.5	20.0

erro também aumenta. Além disso, independentemente da quantidade de sensores e da taxa de redução utilizada, a qualidade dos dados reduzidos se mostra satisfatória, mantendo o mesmo padrão.

5.2.3 Comportamento da rede vs. comportamento dos dados reduzidos

Considerando o comportamento dos dados separadamente, os resultados para a avaliação dos dados multivariados são os esperados. Porém, é importante destacar que, se juntarmos o comportamento da rede com a qualidade dos dados, a nossa solução se mostra bastante eficiente, uma vez que a redução “inteligente” dos dados nos leva a uma taxa de erro aceitável para parte das aplicações que precisam economizar energia e diminuir o atraso.

Como pode ser observado, se precisarmos manter a qualidade nos dados, não precisamos enviar todos os dados, mas apenas $|V|/3$, que possui um erro próximo à zero. Porém se analisarmos o mesmo tamanho de amostra na avaliação do comportamento da rede, uma redução de $|V|/3$ se torna proibitiva para algumas aplicações. Logo uma

redução de $\log|V|$ pode ser utilizada, pois apesar de perdermos em 25% a qualidade dos dados, consumimos pouca energia e temos um atraso reduzido.

5.2.4 Conclusões parciais

Este capítulo apresentou a utilização da arquitetura OGK para as aplicações de redução de dados baseadas em técnicas de *stream* de dados no momento do sensoriamento considerando dados univariados e multivariados. Como pode ser observado, os resultados mostram que a solução OGK pode ser utilizada satisfatoriamente quando tivermos dados univariados e multivariados em redes de sensores, podendo ser aplicada como solução parcial para o problema apresentado na definição 1 (pg. 53), onde podemos aceitar a hipótese 2 (pg. 55) que diz respeito à economia de recursos da rede sem perder a qualidade nos dados. Além disso, podemos encontrar os resultados aqui presentes para as soluções OGK aplicadas no momento do sensoriamento nos seguintes trabalhos:

1. AQUINO, A. L. L. et al. Data stream based algorithms for wireless sensor network applications. In: *21st IEEE International Conference on Advanced Information Networking and Applications (AINA'07)*. Niagara Falls, Canada: IEEE Computer Society, 2007. p. 869–876.

2. AQUINO, A. L. L. et al. A sampling data stream algorithm for wireless sensor networks. In: *IEEE International Conference on Communications (ICC'07)*. Glasgow, Scotland: IEEE Computer Society, 2007. p. 3207–3212.

Dando continuidade à análise da utilização da arquitetura para aplicações gerais, no próximo capítulo, apresenta-se um estudo referente ao problema de redução de dados em redes hierárquicas.

6 REDUÇÃO DE DADOS EM REDES HIERÁRQUICAS

"*A dúvida é o principio da sabedoria.*" (Aristóteles)

ESTE capítulo apresenta um estudo, através de simulação, da redução de dados baseada nas técnicas de *stream* em redes hierárquicas. Além disso, apresentamos uma formulação matemática que comprova que as aplicações gerais têm um melhor desempenho quando modeladas para uma rede hierárquica ao invés de plana. Nas simulações consideramos diferentes cenários onde os dados que representam os fenômenos monitorados pelos nós sensores são sempre univariados.

Os resultados são utilizados para aceitar a hipótese 1 (pg. 55) que diz respeito ao projeto de uma solução genérica onde é possível aplicar a redução de dados Ψ e aceitar a hipótese 3 (pg. 55) que diz respeito à economia de recursos da rede sem perder a qualidade nos dados através da redução de dados em redes hierárquicas. Ambas hipóteses são derivadas do problema relacionado à redução de dados em redes de sensores baseada nas técnicas de *stream* de dados apresentado na definição 1 (pg. 53).

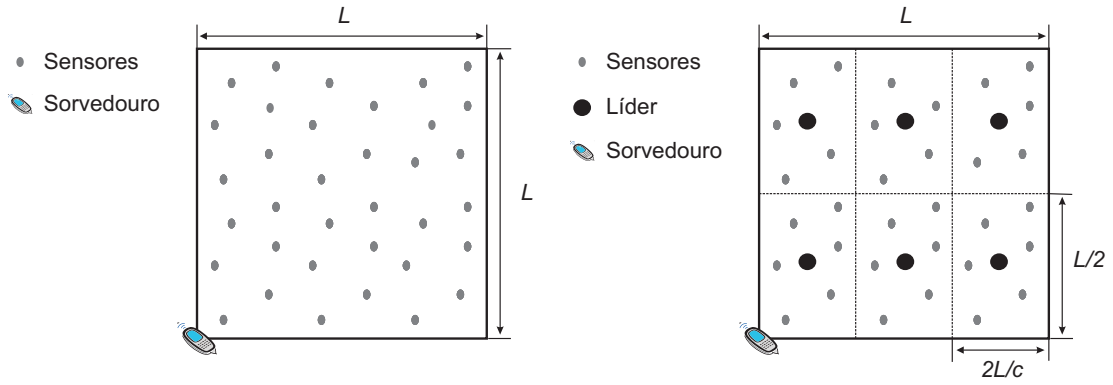
6.1 Caracterização da redução em redes hierárquicas

Como apresentado anteriormente, de forma geral, para efetuarmos a redução de dados nas redes de sensores consideramos as aplicações gerais com o seguinte comportamento de sensoriamento

$$\mathscr{N} \rightarrow \mathscr{V}^* \xrightarrow{S} \mathscr{V} \rightarrow D.$$

Nessas aplicações quanto mais leituras tivermos no conjunto $\mathscr{V}$ mais precisa e eficiente pode ser a decisão D. Seguindo esse comportamento de sensoriamento consideramos duas topologias de rede, nas aplicações gerais, uma plana e outra hierárquica. Para uma organização plana, o conjunto de sensores $S = (S_1, \ldots, S_s)$ é distribuído alea-

toriamente numa área quadrada $Area = L \times L$ e pode existir apenas um nó sorvedouro localizado em (0, 0). Essa topologia é ilustrada na figura 23(a). Para a organização hierárquica o conjunto de sensores S é distribuído numa área dividida em um conjunto de a agrupamentos $A = (A_1, \ldots, A_a)$, $a \leq s$. Cada agrupamento A_j é responsável por monitorar os fenômenos de uma área $Area_{A_j} = L/2 \times 2L/a$. O conjunto A_j é composto por um líder de agrupamento e todos os nós $S = (S_1, \ldots, S_{s_a})$ presentes na $Area_{A_j}$, onde s_a corresponde ao número de sensores por agrupamento. Como nas redes planas, pode existir apenas um nó sorvedouro que está localizado na posição $(0,0)$. Essa topologia é ilustrada na figura 23(b).



(a) Rede plana. (b) Rede hierárquica.

FIGURA 23 – Topologias de rede consideradas nas aplicações gerais.

Para cada tipo de aplicação devemos definir como o item *stream* deve ser representado no sistema, ou seja, como representar os valores do conjunto $\mathscr{V}_i = \{V_1, V_2, \ldots\}$. Sendo assim, para as nossas aplicações gerais temos, para as redes hierárquicas $\mathscr{V} = \mathscr{V}_{Aj}$, onde $\mathscr{V}_{Aj}$ é descrito pelo item V representado por um conjunto $V = \{v_1, \ldots, v_{s_a \times n}\}$, onde n é o número de leituras dos s_a nós do agrupamento Aj e v é no formato numérico tal que $v \in \mathbb{R}$, $0 \leq v \leq 1$ e segue uma distribuição normal com os parâmetros $\mu = 0.5$ e $\sigma = 0.1$, tais parâmetros foram escolhidos de forma arbitrária. A leitura e o processamento de V ocorre quando n itens forem obtidos por todos os s_a. Temos o tamanho máximo do pacote $pct_t = 20$, logo $|V'| > pct_t \rightarrow V' = \{V'^1, \ldots, V'^{n_f}\}$, onde n_f é o número de fragmentos.

Com o objetivo de mostrar que o desempenho em número de bits transmitidos das redes hierárquicas com redução é melhor que o desempenho das redes planas sem redução, adaptamos o modelo analítico proposto por Vlajic e Xia (2006). Isso foi feito comparando o custo operacional em bits transmitidos de cada uma das redes. Para obter o custo operacional das redes planas sem efetuar redução, fizemos as seguintes

considerações:

- Existem $s = |S|$ nós sensores na rede.
- Os nós são distribuídos uniformemente sobre uma área $Area = L \times L$ dividida em grade, onde cada célula da grade com tamanho $a \times a$ contém apenas um nó. Além disso, o sorvedouro está localizado no centro da área.
- Cada nó comunica-se apenas com os oito nós localizados nas células vizinhas.
- $\overline{d}$ representa a média da distância entre os nós e o sorvedouro. Como estamos considerando os nós dispostos numa área quadrada dividida em grade, Vlajic e Xia (2006) mostram que temos, em número de saltos, $\overline{d} = \sqrt{s}/3$.
- Cada nó envia um conjunto de tamanho $|V|$ para o sorvedouro.

Baseado nessas considerações o número de bits (T_p) transmitidos numa rede plana sem redução é definido como

$$T_p = \overline{d}\,|V|\,s. \tag{6.1}$$

Para derivar o custo operacional de uma rede hierárquica temos as seguintes considerações:

- Existem $a = |A|$ agrupamentos na rede.
- A área da rede é dividida em zonas quadradas fixas, iguais e não sobrepostas de tamanho $L/a \times L/a$, onde cada zona quadrada contém um líder no centro. O sorvedouro está localizado no centro da área.
- Como a representa o número de agrupamentos, s/a é o número de nós por agrupamento.
- Segundo Vlajic e Xia (2006), a média da distância em saltos dentro do agrupamento é $\overline{d_a} = \overline{d}/\sqrt{a}$.
- O fator médio de redução de dados dentro do agrupamento por cada leitura de sensoriamento reportada é dada por α e a quantidade média de dados enviados para o sorvedouro é $\alpha\,|V|\,s/a$.

Baseado nessas considerações o número de bits (T_h) transmitidos numa rede hierárquica é definido como

$$T_h = a\Big(\frac{s}{a} - 1\Big)\overline{d_a}\,|V| + a\,\overline{d}\Big(\alpha\,|V|\,\frac{s}{a}\Big), \tag{6.2}$$

onde o primeiro termo da soma representa a quantidade de bits que trafegam dentro dos agrupamentos e o segundo a quantidade de bits que trafegam na rede até o sorvedouro. Simplificando e substituindo $\overline{d_a}$ por $\overline{d}/\sqrt{a}$, na equação (6.2) temos

$$T_h = (s-a)\frac{\overline{d}}{\sqrt{a}}\,|V| + a\,\overline{d}\Big(\alpha\,|V|\,\frac{s}{a}\Big). \tag{6.3}$$

Baseado nas equações (6.1) e (6.3), Vlajic e Xia (2006) mostram que

$$T_h < T_p \iff \frac{s-a}{\sqrt{a}} + \alpha\, s < s, \tag{6.4}$$

ou seja, a quantidade de bits transmitidos pela rede hierárquica com redução só será inferior a quantidade de bits transmitidos pela rede plana sem redução se a inequação acima for satisfeita.

Para efetuarmos a redução baseada em *stream* de dados nos nossos cenários de aplicações gerais de redução, devemos considerar $|V'| = \alpha\,|V|\,s/a$. Com isso,

$$T_h = (s-a)\frac{\overline{d}}{\sqrt{a}}\,|V| + a\,|V'|\overline{d}, \tag{6.5}$$

onde $|V'|$ é a quantidade de dados reduzidos no agrupamento e enviados até o sorvedouro. Com isso, para os nossos cenários de redução, a inequação (6.4) pode ser reescrita da seguinte forma

$$T_h < T_p \iff \frac{s-a}{\sqrt{a}} + \frac{|V'|a}{|V|} < s. \tag{6.6}$$

Como $a < s$, a inequação acima sempre será satisfeita, ou seja, o consumo em bits transmitidos numa rede hierárquica com redução sempre será inferior à quantidade de bits transmitidos na rede plana sem redução. No entanto essa formulação pode ser utilizada para encontrarmos o número ideal de agrupamentos numa aplicação de redução de dados.

Como exemplo, considere uma rede de sensores com $s = 1024$ nós, $a = (4, 9, 16, 25)$ agrupamentos, cada nó do agrupamento gerando um item com o tamanho $|V| = 256$ e $|V'| \in \{\log|V|, |V|/2, |V|\}$. Para todas as possíveis combinações desse cenário, através das equações (6.1) e (6.5), foi calculado a razão entre T_h/T_p. A tabela 4 mostra que,

em todos os casos, a rede hierárquica aplicando redução, tem um melhor desempenho quando comparadas a uma rede plana sem redução.

TABELA 4 – Razão de bits transmitidos numa rede com 1024 nós.

Agrupamentos	$\log\|V\|$	$\|V\|/2$	$\|V\|$
4	0.49	0.50	0.50
9	0.33	0.33	0.33
16	0.24	0.25	0.26
25	0.21	0.20	0.19

Considere outro cenário, um pouco mais real, onde a rede de sensores possui $s = 160$ nós, $a = (4,9,16,25)$ agrupamentos, cada nó do agrupamento gerando um item com o tamanho $|V| = 256$ e o líder fazendo reduções $|V'| \in \{\log|V|, |V|/2, |V|\}$. Mais uma vez em todos os casos, através das equações (6.1) e (6.5), calculamos a razão T_h/T_p, que nos levam aos resultados apresentados na tabela 5. Como nos resultados anteriores, em todos os casos a redução de dados na rede hierárquica possui um melhor desempenho quando comparada a uma rede plana sem redução.

TABELA 5 – Razão de bits transmitidos numa rede com 160 nós.

Agrupamentos	$\log\|V\|$	$\|V\|/2$	$\|V\|$
4	0.48	0.50	0.51
9	0.33	0.36	0.38
16	0.22	0.27	0.32
25	0.14	0.24	0.34

Como visto, as redes hierárquicas com redução possuem um melhor desempenho quando comparadas às redes planas. Na próxima seção, apresentaremos as avaliações feitas para aceitar a hipótese 3 (pg. 55) que diz respeito à economia de recursos da rede sem perder a qualidade nos dados através da redução de dados em redes hierárquicas.

6.2 Avaliação do comportamento da rede

As aplicações gerais nas redes hierárquicas, modeladas utilizando a arquitetura OGK, possuem uma função de redução Ψ_F composta pelas seguintes funções definidas a seguir:

- *Classificação* (ψ_c): Utilizamos a priori dados *Univariados* para representar o *stream* $\mathscr{V}^*$.

- *Processamento* (ψ_p): Utilizamos diretamente sobre os itens os algoritmos definidos na API para dados univariados (OGK-*amostragem*$_{aleatoria}$ e OGK-*rascunho*). Esses algoritmos, bem como seus parâmetros, foram definidos a priori. Para o OGK-*amostragem*$_{aleatoria}$ consideramos diferentes tamanhos para o conjunto de amostra $|V'| = \{|V|/2, \log|V|\}$ e para o OGK-*rascunho*, fixamos o tamanho do rascunho $|V'| = pct_t$, onde $pct_t = 20$.

- *Dados de saída* (ψ_s): O dado reduzido V', sem informações adicionais é enviado do líder do agrupamento até o sorvedouro de forma *ad-hoc*. Nos casos onde $|V'| > pct_t$ os V'^j, com $1 < j < n_f$, serão enviados seqüencialmente.

Com isso, as simulações realizadas para a avaliação do comportamento das redes hierárquicas são baseadas nas seguintes considerações:

- A avaliação foi feita na ferramenta de simulação NS-2 (*Network Simulator 2*) versão 2.32. Cada cenário simulado foi executado em 33 diferentes topologias aleatórias. Ao fim, para cada cenário apresentamos os resultados utilizando a média das 33 execuções com intervalo de confiança de 95%.

- Utilizamos um algoritmo de roteamento baseado em árvore chamado EF-Tree (NAKAMURA et al., 2005), tanto dentro do agrupamento como para propagar a informação do líder para o sorvedouro. A densidade da rede é mantida constante e todos os nós têm a mesma configuração de *hardware*. A árvore é construída apenas uma vez na fase de estabelecimento da rede, pois o consumo para montagem da árvore pode interferir nos resultados.

- Os parâmetros de configuração variados são: número de nós no agrupamento, tamanho do item *stream* em bytes suportado pelo líder e número de agrupamentos. Além disso, todos os nós presentes nos agrupamentos enviam seus dados monitorados para o líder. Para cada parâmetro de configuração, analisamos o comportamento da rede hierárquica, utilizando um conjunto de amostras de tamanhos $|V|/2$ e $\log|V|$. Alguns parâmetros importantes utilizados nas simulações são apresentados na tabela 6. Nós variamos o tamanho da rede para manter a densidade da rede constante em $8,48$. Para isso, consideramos a densidade da rede com a relação $\pi r^2 |S|/A$, onde r é o alcance do rádio, $|S|$ é o número de sensores e A é a área da rede. O tamanho da fila suportada por cada nó varia com o tamanho do *stream* para que não haja descarte de pacotes, i.e., cada nó suporta no máximo o tamanho do item *stream* utilizado pela aplicação. O tempo

de simulação é 5000s, onde os 1000s iniciais são utilizados para montagem e configuração da rede e da estrutura de roteamento, os 1000s finais são utilizados para permitir que a rede escoe os pacotes restantes na rede. O tráfego real de dados dura 3000s onde cada nó em cada agrupamento envia 300 *streams* na rede um a cada 10s como determinado pelo período do *stream*, esse alto tráfego no agrupamento é para possibilitar sucessivas reduções no líder de agrupamento. O alcance do rádio e a largura de banda seguem a especificação do MicaZ[†]. Por fim, a energia inicial utilizada é 100, com isso os nós nunca terão suas reservas de energia esgotadas.

TABELA 6 – Parâmetros de simulação para redução de dados nas redes hierárquicas.

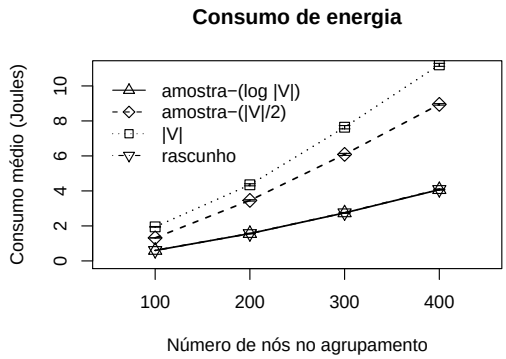
Parâmetro	**Valor**
Tamanho da rede	varia com a densidade
Tamanho da fila	varia com $\|V\|$
Tempo de simulação (s)	5000
Tráfego (s)	[1000, 4000]
Período do *stream* (s)	10
Alcance do rádio (m)	50
Largura de banda (kbps)	250
Energia inicial (Joules)	100
Localização do sorvedouro	coordenadas (0, 0)

Inicialmente considere a avaliação do comportamento da rede hierárquica ao utilizarmos a função de redução Ψ_F nas aplicações gerais. Para essa avaliação identificamos o consumo total de energia na rede e a média do atraso para entregar um pacote dos líderes até o sorvedouro. Além disso, utilizamos para comparação os resultados, referentes à eficiência, sem utilizar nenhuma solução de redução ($|V|$ amostras). As curvas nas figuras 24 e 25 representam os resultados para a utilização dos algoritmo OGK-*rascunho* e OGK-*amostragem*$_{aleatório}$ com $|V'| \in \{\log|V|, |V|/2\}$ e o comportamento da rede sem utilizar redução ($|V|$).

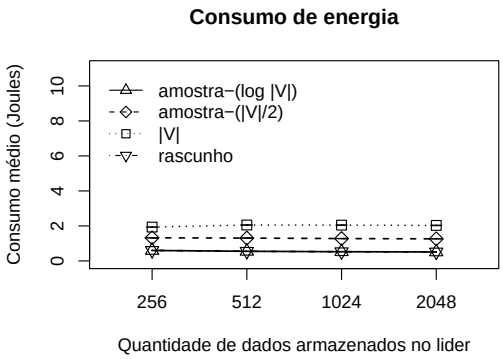
A figura 24 mostra no eixo-y o consumo médio de energia em Joules para a avaliação do comportamento da rede, onde variamos no eixo-x o número de nós no agrupamento em $100, 200, 300$ e 400[‡], o tamanho do item *stream* identificado no líder em $256, 512, 1024$ e 2048 bytes e o número agrupamentos em $4, 8, 12$ e 16. Para todos os cenários, quando não variados, fixamos os respectivos parâmetros em 100, 256 e 4.

[†] `http://www.xbow.com/`

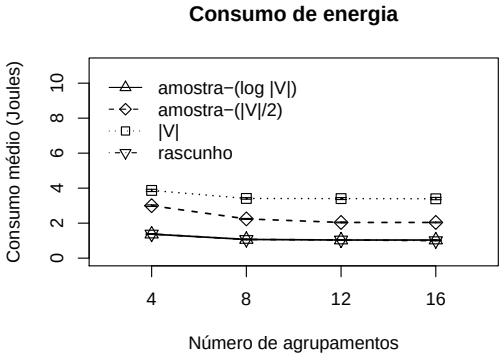
[‡] Utilizamos apenas 400 nós devido às limitações de memória ao executar a ferramenta de simulação NS. Nas simulações mais extremas, a rede gera em torno de 10000 pacotes por cada ciclo de envio do dado monitorado.



(a) Diferentes número de nós no agrupamento.



(b) Diferentes tamanhos do item *stream*, em bytes, suportados pelo líder.



(c) Diferentes números de agrupamentos.

FIGURA 24 – Avaliação do comportamento da rede, considerando a média da energia consumida na rede ao reduzir dados nos nós líderes.

Analisando de forma geral, como esperado, em todos os casos para o OGK-*amostragem*$_{aleatoria}$ observamos que quando $|V'|$ diminui, o consumo de energia também diminui. Para o OGK-*rascunho* observamos um comportamento similar ao resultado para a amostra-($\log|V|$). Além disso, esses resultados estão de acordo com a análise feita na seção 6.1, pois quanto maior $|V'|$ melhor é o desempenho da rede hierárquica.

Analisando esses resultados separadamente, quando o número de nós no agrupamento varia (figura 24(a)), a energia consumida aumenta consideravelmente. Isso ocorre porque o volume de dados dentro de cada agrupamento aumenta, pois temos mais nós monitorando o ambiente e, conseqüentemente, maior será o consumo de energia global da rede. No entanto a amostra-($\log|V|$) e o rascunho tiveram o menor impacto sobre o consumo de energia porque, apesar do alto tráfego dentro dos agrupamentos, os dados que são enviados para o sorvedouro são bastante reduzidos (apenas um pacote por ciclo de envio). Já para os outros resultados, o dado foi fragmentado, gerando mais pacotes, tráfego e consumo de energia na rede.

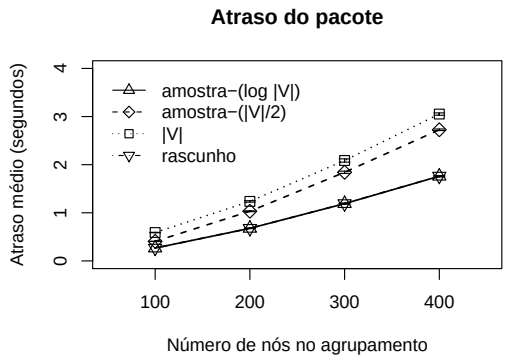
Quando o tamanho do item *stream* suportado pelo líder é variado (figura 24(b)) a amostra-($\log|V|$) e o rascunho tiveram o melhor desempenho em todos os casos. Apesar de aumentarmos o tamanho do item, a energia consumida se manteve quase constante, pois temos um número pequeno de agrupamentos (4), o que leva a um baixo tráfego na rede. Especificamente no caso da amostra-($\log|V|$), isso ocorre porque o tamanho do pacote é incrementado apenas de um elemento, quando aumentamos o tamanho do item *stream* ($256, 512, 1024, 2048$) e no caso do rascunho o tamanho do pacote usado é sempre constante. Os outros resultados (amostra-($|V|/2$) e $|V|$) tiveram um desempenho pior, porque o tamanho dos pacotes aumentam proporcionalmente ao tamanho do item *stream* e mais pacotes são lançados na rede devido à fragmentação.

Quando o número de agrupamentos é variado (figura 24(c)), mais uma vez, a amostra-($\log|V|$) e o rascunho obtiveram um melhor desempenho em todos os casos, uma vez que menos dados são enviados até o sorvedouro. No entanto podemos observar que o consumo de energia, é maior, quando temos poucos agrupamentos. Isso ocorre porque o tráfego dentro do agrupamento é grande, ou seja, o número de agrupamentos ainda não é o ideal, como sugerido na formulação da seção 6.1.

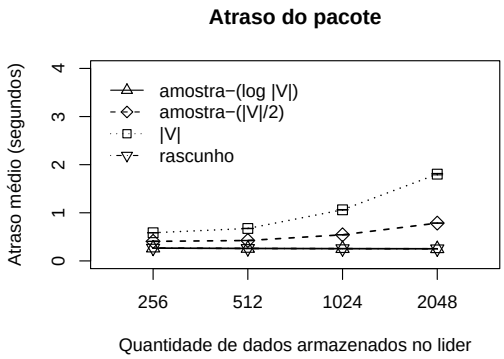
A figura 25 mostra no eixo-y o atraso médio em segundos dos pacotes, onde variamos no eixo-x o número de nós no agrupamento em $100, 200, 300$ e 400, o tamanho do item *stream* identificado no líder em $256, 512, 1024$ e 2048 bytes e o número agrupamentos em $4, 8, 12$ e 16. Para todos os cenários, quando não variados, fixamos os respectivos parâmetros em 100, 256 e 4.

Como nos resultados para o consumo de energia, podemos observar que quando o $|V'|$ diminui temos um menor atraso. Os mesmos efeitos no consumo de energia, para a variação do número de nós no agrupamento e número de agrupamentos, são observados no atraso dos pacotes (figuras 25(a) e 25(c)). No entanto, quando temos a variação no tamanho do item *stream* (figura 25(b)) observamos uma variação no atraso. Isso ocorre porque dentro do agrupamento os nós levam mais tempo para propagar o dado monitorado até o líder, diferente do comportamento da avaliação da energia que apenas depois da redução é que podemos observar a economia de energia.

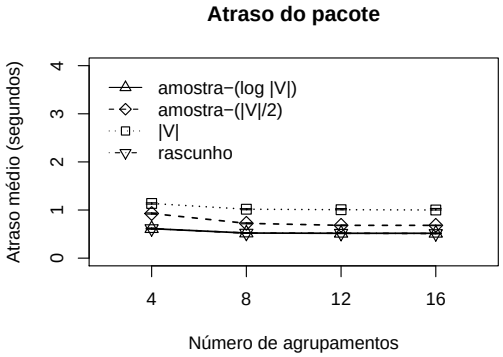
Ao considerar a avaliação do comportamento dos dados reduzidos nas redes hierárquicas, é observado o mesmo comportamento do capítulo 5 referente a redução de dados univariados no momento do sensoriamento. Com isso, podemos utilizar as regras R'_{dist},



(a) Diferentes número de nós no agrupamento.



(b) Diferentes tamanhos do item *stream*, em bytes, suportados pelo líder.



(c) Diferentes números de agrupamentos.

FIGURA 25 – Avaliação do comportamento da rede, considerando a média do atraso do pacote ao reduzir dados nos nós líderes.

R'_{val} e R'_{ord} para tomarmos as respectivas decisões D'_{dist}, D'_{val} e D'_{ord} ao reduzirmos dados em redes hierárquicas.

6.3 Conclusões parciais

Este capítulo apresentou a utilização da arquitetura OGK para as aplicações de redução de dados baseadas em técnicas de *stream* de dados em redes hierárquicas. Como pode ser observado, os resultados mostram que a solução OGK em redes hierárquicas pode ser aplicada como solução parcial para o problema apresentado na definição 1 (pg. 53) relacionado à redução de dados em redes de sensores baseada nas técnicas de *stream* de dados, onde podemos confirmar a hipótese 3 (pg. 55) que diz respeito a economia de recursos da rede sem perder a qualidade nos dados através da redução de dados em redes hierárquicas. Além disso, podemos encontrar os resultados aqui presentes para as soluções OGK aplicadas às redes hierárquicas

no seguinte trabalho:

1. AQUINO, A. L. L. et al. Sensor stream reduction for clustered wireless sensor networks. In: *23rd ACM Symposium on Applied Computing 2008 (SAC'08)*. Fortaleza, Brazil: ACM, 2008. p. 2052–2056.

Dando continuidade à análise da utilização da arquitetura OGK para aplicações gerais, no próximo capítulo apresentaremos um estudo referente ao problema de redução de dados no roteamento em aplicações de tempo real.

7 REDUÇÃO DE DADOS EM APLICAÇÕES DE TEMPO REAL

"*Vale a pena repetir muitas vezes as coisas belas.*" (Platão)

ESTE capítulo apresenta um estudo, através de simulação, da redução de dados baseada nas técnicas de *stream* em aplicações de tempo real em redes de sensores. Esse estudo foi realizado com o objetivo de avaliar como a solução de redução no roteamento pode ser utilizada em aplicações específicas. Além disso, apresentamos uma formulação matemática que estima o tamanho $|V'|$ ideal quando a redução é efetuada no momento do roteamento em aplicações de tempo real.

Os resultados são utilizados para aceitar a hipótese 1 (pg. 55) que diz respeito ao projeto de uma solução genérica onde é possível aplicar a redução de dados Ψ e aceitar a hipótese 4 (pg. 55) que diz respeito à utilização da redução de dados Ψ durante o roteamento para alcançar os prazos determinados pela aplicação e tomar as decisões D. Ambas são derivadas do problema relacionado à redução de dados em redes de sensores, baseada nas técnicas de *stream* de dados apresentado na definição 1 (pg. 53).

7.1 Caracterização da redução em aplicações de tempo real

Sejam as aplicações gerais de tempo real com comportamento de sensoriamento

$$\mathscr{N} \rightarrow \mathscr{V}^* \rightarrow \mathscr{V} \rightarrow D,$$

onde o *stream* $\mathscr{V}$ é descrito pelo item V representado por um conjunto $V = \{v_1, \ldots, v_n\}$, n é o número de leituras do ambiente e v está representado no formato numérico tal que $v \in \mathbb{R}$, $0 \leq v \leq 1$ e segue uma distribuição normal com os parâmetros $\mu = 0.5$ e $\sigma = 0.1$. A leitura e o processamento de V ocorre quando n itens forem obtidos e se tivermos $|V'| > pct_t \rightarrow V' = \{V'^1, \ldots, V'^{n_f}\}$, onde n_f é o número de fragmentos e pct_t é

o tamanho do pacote.

Como previsto pela arquitetura OGK, alguns parâmetros das aplicações gerais precisam ser passados para a redução Ψ_R. No caso das nossas aplicações, temos dois momentos que esses parâmetros são passados: na fase de (re)construção da árvore e na fase de propagação do roteamento. No caso da fase de (re)construção da árvore, os parâmetros recebidos são o número de saltos (s_{dst}) e o tempo gasto para entregar um pacote até o sorvedouro (t_{dst}). Já na fase de propagação, os parâmetros recebidos e/ou atualizados são o prazo da aplicação (p_a), o número de fragmentos do item V (n_f), o instante em que o fragmento foi gerado (t_{gen}) e o número de saltos do nó que gerou o item V (s_{org}), que é sempre incrementado quando o fragmento passa por um nó roteador.

Outro ponto importante a ser considerado é como determinar, nos nós roteadores na fase de propagação, o tamanho do conjunto V' de tal forma que os prazos especificados pela aplicação sejam atendidos. Com isso, para estimar o tamanho de V' faremos as seguintes considerações:

- Para identificar o atraso do item *stream* consideramos que todos os nós estão com os relógios sincronizados[†].
- Cada sensor assume um tamanho máximo para o pacote (pct_t), no nosso caso, $pct_t = 20$.
- Em cada nó roteador, um novo prazo local (p_l), entre o nó roteador e o sorvedouro, é calculado da seguinte forma:

  $$p_l = p_a - t_{atual},$$

  onde t_{atual} é o tempo atual do sistema.
- O tempo estimado (t_{org}) que o fragmento V^j gasta para viajar entre o nó origem e o nó roteador é

  $$t_{org} = t_{atual} - t_{gen}.$$

- Considere t_{dst} o tempo que o fragmento V^1 gasta para viajar do nó roteador até o sorvedouro. O fragmento V^2 gastará o tempo t_{dst}/s_{dst} para viajar entre o nó roteador e o sorvedouro e o tempo t_{org}/s_{org} para viajar entre o nó origem e o nó

[†]Soluções para o problema de sincronização em redes de sensores podem ser encontradas em Elson (2003).

roteador. Dessa forma, o tempo estimado para entrega e recebimento de V é respectivamente:

$$t_{ent} = t_{dst} + (n_f - 1)\, t_{dst}/s_{dst} \tag{7.1}$$

e

$$t_{rec} = (n_f - 1)\, t_{org}/s_{org}. \tag{7.2}$$

O primeiro termo da soma não é considerado para t_{rec} porque V^1 já chegou.

Baseado nessas considerações, $|V'|$ é determinado e usado somente se $gap > 0$, onde o gap é dado por:

$$gap = p_l - atraso, \tag{7.3}$$

com o $atraso$ estimado para entregar V, após receber o primeiro fragmento V^1, até o sorvedouro sendo

$$atraso = t_{rec} + t_{ent}. \tag{7.4}$$

Sendo o $gap > 0$ de (7.3) e (7.4) temos

$$p_l - atraso > 0$$

$$p_l - (t_{rec} + t_{ent}) > 0$$

usando (7.2) e (7.1) nós temos

$$p_l - ((n_f - 1)\, t_{org}/s_{org} + t_{dst} + (n_f - 1)\, t_{dst}/s_{dst}) > 0$$

simplificando

$$n_f < 1 + \frac{s_{org}\, s_{dst}\, (p_l - t_{dst})}{s_{dst}\, t_{org} + s_{org}\, t_{dst}}$$

considerando que $n_f = \lceil |V'|/pct_t \rceil$, temos

$$|V'| < pct_t \left(1 + \frac{s_{org}\, s_{dst}\, (p_l - t_{dst})}{s_{dst}\, t_{org} + s_{org}\, t_{dst}}\right).$$

Finalmente para atender a desigualdade temos

$$|V'| = pct_t \left(1 + \frac{s_{org}\, s_{dst}\, (p_l - t_{dst})}{s_{dst}\, t_{org} + s_{org}\, t_{dst}}\right) - 1. \tag{7.5}$$

De posse das considerações feitas acima, na próxima seção, apresentaremos as avaliações feitas para aceitar a hipótese 4 (pg. 55), que diz respeito a utilizar a redução de dados Ψ durante o roteamento, para alcançar os prazos determinados pela aplicação e tomar as decisões D.

7.2 Avaliação do comportamento da solução OGK em aplicações de tempo real

As aplicações gerais de tempo real, modeladas utilizando a arquitetura OGK, possuem uma função de redução Ψ_R composta pelas seguintes funções definidas a seguir:

- *Classificação* (ψ_c): Os dados recebidos são do tipo *Sensoriamento* e *Roteamento*, sendo que definimos a priori que apenas os dados de *Roteamento* podem ser reduzidos e que o item V é univariado. Caso um dado de *Sensoriamento* seja recebido o mesmo será propagado diretamente sem sofrer nenhuma redução. No caso dos dados de *Roteamento*, se for necessário aplicar a redução, os fragmentos do item V são armazenados e as informações da aplicação de tempo real agregadas aos fragmentos, são lidas e armazenadas no banco de dados de *parâmetros da aplicação*, para serem utilizadas no momento da redução.

- *Processamento* (ψ_p): Inicialmente os parâmetros da aplicação são verificados e utilizados para escolher, através do OGK-*oráculo*, a melhor estratégia de redução que, no nosso caso, é a determinação do tamanho da redução $|V'|$ (equação (7.5)). Com isso, o item V recebido é reduzido, através do OGK-*amostragem*$_{central}$, de acordo com os parâmetros da aplicação.

- *Dados de saída* (ψ_s): O item reduzido V', com as informações da aplicação atualizadas, é enviado até o sorvedouro de forma *ad-hoc*. Nos casos onde $|V'| > pct_t \rightarrow V'^j$, com $1 < j < n_f$, os dados serão enviados seqüencialmente.

Com isso, as simulações realizadas para a avaliação do comportamento das aplicações de tempo real, ao utilizar a redução na camada de roteamento, são baseadas nas seguintes considerações:

- A nossa avaliação foi feita na ferramenta de simulação NS-2 (*Network Simulator 2*) versão 2.32. Cada cenário simulado foi executado em 33 diferentes topologias

aleatórias. Ao fim, para cada cenário apresentamos os resultados utilizando a média das 33 execuções com intervalo de confiança de 95%.

- Utilizamos um algoritmo de roteamento baseado em árvore EF-Tree. A densidade da rede é mantida constante e todos os nós têm a mesma configuração de *hardware*. A árvore é construída apenas uma vez na fase de estabelecimento da rede, pois o consumo para montagem da árvore pode interferir nos resultados. Além disso, os nós são distribuídos numa $Area = L \times L$ com o sorvedouro localizado na posição $(0,0)$ e um único nó monitorando o ambiente localizado na posição (L,L).

- Utilizamos uma aplicação geral com exigências de prazos para a entrega do item V. Para utilizar prazos mais realistas, determinamos de forma empírica o prazo mínimo (p_{min}) exigido em cada cenário simulado. Com isso, utilizamos nas nossas avaliações os seguintes prazos da aplicação: $p_a = p_{min}/2$ com a rede funcionando sem atraso; e $p_a = p_{min}$ com cada nó roteador na rede gerando um $atraso = p_{min}/10000$ em todos os pacotes recebidos, ou seja, cada nó só repassa cada pacote recebido por $atraso$ segundos.

- Os parâmetros de configuração variados são: número de nós na rede e tamanho do item *stream* em bytes. Para cada parâmetro de configuração, analisamos o comportamento da rede e a qualidade dos dados (decisões D') utilizando um conjunto de amostras de tamanhos $|V|/2$ e $\log|V|$. Alguns parâmetros importantes utilizados nas simulações são apresentados na tabela 7. Nós variamos o tamanho da rede para manter a densidade da rede constante em $8,48$. Para isso, consideramos a densidade da rede com a relação $\pi r^2 |S|/A$, onde r é o alcance do rádio, $|S|$ é o número de sensores e A é a área da rede. O tamanho da fila suportada por cada nó varia com o tamanho do *stream* para que não haja descarte de pacotes, i.e., cada nó suporta no máximo o tamanho do item *stream* utilizado pela aplicação. O tempo de simulação é 1100s, onde os 500s iniciais são utilizados para montagem e configuração da rede e da estrutura de roteamento, os 500s finais são utilizados para permitir que a rede escoe os pacotes restantes na rede. O tráfego real de dados dura 100s onde 10 *streams* são lançados na rede um a cada 10s como determinado pelo período do *stream*. O alcance do rádio e a largura de banda seguem a especificação do MicaZ[†]. Por fim, a energia inicial utilizada é 100, com isso os nós nunca terão suas reservas de energia

[†] `http://www.xbow.com/`

esgotadas.

TABELA 7 – Parâmetros de simulação para redução de dados nas aplicações de tempo real.

Parâmetro	**Valor**
Tamanho da rede	varia com a densidade
Tamanho da fila	varia com $\|V\|$
Tempo de simulação (s)	1100
Tráfego (s)	[500, 600]
Período do *stream* (s)	10
Alcance do rádio (m)	50
Largura de banda (kbps)	250
Energia inicial (Joules)	100

Para as nossas avaliações consideramos dois cenários específicos: um com tráfego e outro sem. Para ambos os cenários avaliamos o comportamento da solução OGK ao utilizarmos prazos da aplicação $p_a = p_{min}/2$ e $p_a = p_{min}$ com os nós atrasando os pacotes. Além disso, uma questão importante a ser considerada ao determinar o prazo da aplicação é o prazo mínimo (p_{min}) permitido pela rede para entregar o item V ao sorvedouro. O valor p_{min} pode ser difícil de se determinar, pelas condições dinâmicas durante a operação da rede, tais como número de nós fontes, quantidade de dados e topologias. Por essas razões, os nossos cenários são avaliados utilizando como base os valores para p_{min} obtidos de forma empírica.

7.2.1 Cenário I – Redes sem tráfego

Neste primeiro cenário, consideramos diferentes número de nós na rede ($128, 256, 512$ e 1024), o tamanho do item *stream* $|V| = \{256, 512, 1024, 2048\}$, e somente um nó é usado para gerar V. Os valores de p_{min} são determinados pela medida do tempo gasto entre o primeiro fragmento enviado pelo nó origem e o último fragmento recebido pelo sorvedouro, ou seja, o tempo para V ser inteiramente recebido pelo sorvedouro. A tabela 8 resume os valores de p_{min} para os diferentes tamanhos $|V|$ em cada topologia da rede. Os valores para p_{min}, com os intervalos de confiança, são ilustrados na figura 26, onde no eixo-x temos a variação do tamanho do item *stream*, no eixo-y temos a média do atraso em segundos e as colunas representam a variação no número de nós na rede.

É importante notar que, se a aplicação tem um prazo menor que os mostrados na tabela 8, os dados não poderão ser entregues no prazo e alguma redução é necessária.

TABELA 8 – Cenário I: Valores mínimos para os prazos exigidos pelas aplicações.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	0.77s	1.51s	3.09s	6.04s
256	1.05s	2.04s	4.10s	8.18s
512	1.42s	2.71s	5.31s	13.76s
1024	1.88s	3.43s	6.72s	17.90s

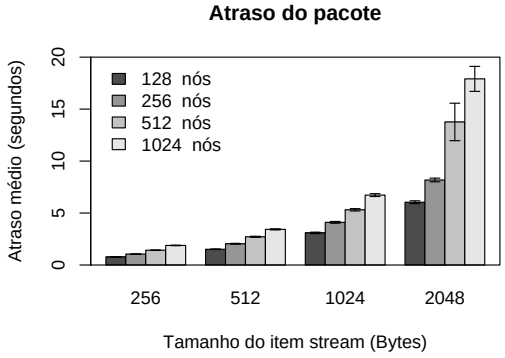


FIGURA 26 – Cenário I: Valores mínimos para os prazos exigidos pelas aplicações.

Avaliação I – Prazos menores que os suportados

Inicialmente, consideramos o prazo da aplicação $p_a = p_{min}/2$. Nesse caso, a rede não pode atender o prazo p_a sem reduzir V. Tal redução ocorre dentro da rede quando o nó roteador percebe que o prazo da aplicação (p_a) não pode ser alcançado. Com isso, a redução é efetuada com um tamanho $|V'|$ obtido através da equação (7.5). A tabela 9 mostra os resultados do $atraso$ para essa avaliação. Os valores para o $atraso$ com os intervalos de confiança são ilustrados na figura 27, onde no eixo-x temos a variação do tamanho do item *stream*, no eixo-y temos a média do atraso em segundos e as colunas representam a variação no número de nós na rede.

TABELA 9 – Cenário I: Atrasos identificados ao utilizar a metade dos prazos suportados pela rede.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	**0.45s**	**0.96s**	**2.15s**	**4.30s**
256	**0.57s**	**1.27s**	**2.69s**	**5.70s**
512	0.56s	1.29s	**3.29s**	**8.23s**
1024	0.42s	0.79s	3.24s	**10.03s**

Como podemos ver na tabela 9, o prazo da aplicação (p_a) não pode ser alcançado em muitos casos (prazos não alcançados em destaque na tabela), quando comparados com a metade dos prazos apresentados na tabela 8. Porém esses prazos perdidos

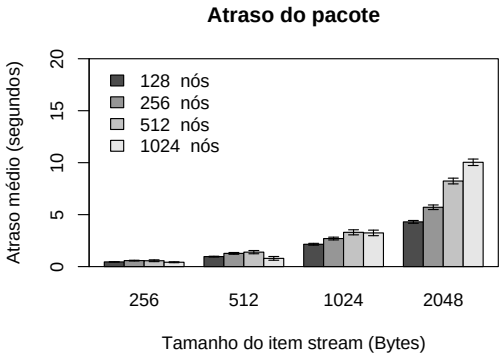


FIGURA 27 – Cenário I: Atrasos identificados ao utilizar a metade dos prazos suportados pela rede.

ainda são aceitáveis, uma vez que estamos tratando de aplicações de tempo real *soft*. Esse comportamento acontece porque a redução pode ser observada no início da propagação dos dados, indicando que a solução OGK é eficiente e escalável para esse cenário. Entretanto, sem a nossa solução de redução esses atrasos seriam maiores. O impacto dos atrasos também podem ser observados na tabela 10 onde é apresentado a razão entre o $atraso$ e o p_a, calculado como $razao = (atraso - p_a)/p_a$ (os prazos alcançados são identificados por "–" na tabela 10).

TABELA 10 – Cenário I: Razão entre os atrasos identificados e os prazos exigidos pela aplicação.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	0.18	0.28	0.39	0.42
256	0.9	0.24	0.31	0.39
512	–	–	0.24	0.19
1024	–	–	–	0.12

A fim de mostrar o comportamento dos dados reduzidos, considere inicialmente a tabela 11, que mostra a porcentagem dos dados recebidos pelo sorvedouro, calculado pela fórmula $rec = 100\,(|V'|/|V|)$. Podemos ver que, em todos os casos menos dados são recebidos quando a rede aumenta. Isso ocorre, pois maiores reduções são feitas e os prazos são mais apertados quando a rede tem muitos nós.

Além do cálculo da porcentagem, aplicamos sobre os dados recebidos as regras R'_{dst} e R'_{val}. Considerando a regra R'_{dst} (tabela 12), os resultados mostram que na maioria dos casos temos um erro menor que 20%. Note que o erro é proporcional à quantidade de dados recebidos (tabela 11). O maior erro ocorre quando menos dados são recebidos pelo sorvedouro, mas a similaridade dos dados é mantida e com isso a decisão D'_{dst}

TABELA 11 – Cenário I: Porcentagem dos dados recebidos pelo sorvedouro, ao utilizar a metade dos prazos suportados pela rede.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	15.19%	22.00%	34.31%	35.43%
256	13.12%	22.21%	26.27%	31.22%
512	13.19%	15.95%	22.87%	25.45%
1024	7.84%	9.09%	18.20%	21.27%

sobre os dados pode ser tomada.

TABELA 12 – Cenário I: Erro do teste KS identificado ao utilizar a metade dos prazos suportados pela rede.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	31%	18%	11%	11%
256	32%	20%	16%	14%
512	26%	21%	17%	16%
1024	21%	21%	17%	18%

Em relação aos valores do erro para a regra R'_{val} (tabela 13), os resultados mostram que o erro é constante e entre 1% e 3%. Isso ocorre porque nossas amostras consideram sempre os elementos centrais na coluna do histograma, através do OGK-*amostragem*$_{central}$. Com isso, a decisão D'_{dst} sobre os dados pode ser tomada.

TABELA 13 – Cenário I: Erro identificado nos valores dos dados ao utilizar a metade dos prazos suportados pela rede.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	3%	2%	1%	1%
256	3%	2%	2%	1%
512	3%	2%	2%	1%
1024	3%	2%	1%	2%

Avaliação II – Atrasos gerados pelos nós roteadores

Agora, consideramos o prazo da aplicação $p_a = p_{min}$, com os nós roteadores atrasando os fragmentos do item (V^j) por $p_{min}/10000$ ou 0.01% do valor de p_{min}, esta fração do prazo mínimo foi escolhida de forma arbitrária. Como na avaliação I, a rede não pode atender o prazo p_a sem reduzir V e a redução ocorre quando o nó roteador percebe que o prazo da aplicação (p_a) não pode ser alcançado. Assim a redução é efetuada com um tamanho $|V'|$ obtido através da equação (7.5). A tabela 14 mostra os resultados do $atraso$ para essa avaliação. Os valores para o $atraso$ com os intervalos de

confiança são ilustrados na figura 28, onde no eixo-x temos a variação do tamanho do item *stream*, no eixo-y temos a média do atraso em segundos e as colunas representam a variação no número de nós na rede.

TABELA 14 – Cenário I: Atrasos identificados ao utilizar atrasos gerados pelos nós roteadores.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	0.77s	1.47s	2.69s	4.35s
256	1.03s	1.97s	3.67s	6.16s
512	1.34s	2.60s	4.96s	10.12s
1024	1.33s	3.11s	6.40s	14.24s

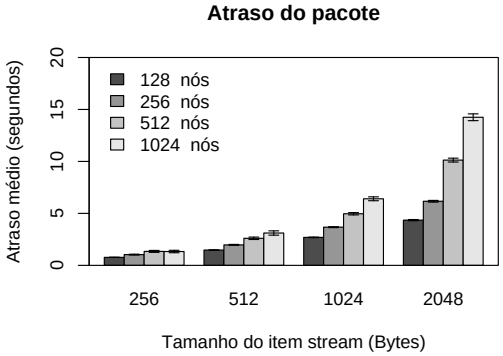


FIGURA 28 – Cenário I: Atrasos identificados ao utilizar atrasos gerados pelos nós roteadores.

Diferente dos resultados para a avaliação I, a tabela 14 mostra que p_a é alcançado em todos os casos quando comparados com os prazos apresentados na tabela 8. Isso ocorre, pois o tamanho estimado para a redução (equação (7.5)) está diretamente relacionado ao prazo da aplicação p_a, ou seja, se $p_a \geq p_{min}$, a redução consegue ser eficiente independente do atraso da rede.

Mais uma vez, para o comportamento dos dados reduzidos, consideramos a porcentagem dos dados recebidos pelo sorvedouro (tabela 15), calculado por $rec = 100\,(|V'|/|V|)$. Podemos ver que, em todos os casos menos dados são recebidos quando a rede aumenta, porém se comparado com a avaliação I, mais dados foram recebidos. Pois os nós roteadores identificam as reduções mais próximas ao sorvedouro. Com isso a estimativa é mais próxima do esperado e, assim, mais dados podem ser entregues.

Além do cálculo da porcentagem, aplicamos sobre os dados recebidos as regras R'_{dst} e R'_{val}. Considerando a regra R'_{dst} (tabela 16), os resultados mostram que aumentando o número de nós o erro no teste KS também cresce, pois mais dados estão sendo

TABELA 15 – Cenário I: Porcentagem dos dados recebidos pelo sorvedouro ao utilizar atrasos gerados pelos nós roteadores.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	43.55%	36.92%	32.32%	30.83%
256	28.29%	25.00%	25.87%	26.91%
512	19.39%	14.78%	16.65%	28.0%
1024	18.07%	21.31%	13.37%	22.23%

entregues (tabela 15), se comparado com a avaliação I. Mais uma vez, o erro é proporcional à quantidade de dados recebidos (tabela 15). Como esperado, os maiores erros ocorrem quando menos dados são recebidos pelo sorvedouro, mas a similaridade do dado ainda é mantida e, com isso, a decisão D'_{dst} sobre os dados pode ser tomada.

TABELA 16 – Cenário I: Erro no teste KS identificado ao utilizar atrasos gerados pelos nós roteadores.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	12%	10%	9%	12%
256	20%	16%	12%	15%
512	29%	28%	20%	16%
1024	24%	25%	29%	22%

Em relação à regra R'_{val} (tabela 17), como pode ser visto, os resultados e, conseqüentemente, as conclusões são as mesmas apresentadas na avaliação I. Com isso, a decisão D'_{val} sobre os dados pode ser tomada.

TABELA 17 – Cenário I: Erro nos valores dos dados identificado ao utilizar atrasos gerados pelos nós roteadores.

Número de nós	**Tamanho do item *stream***			
	256	512	1024	2048
128	1%	1%	1%	0%
256	2%	1%	1%	0%
512	2%	2%	1%	0%
1024	2%	1%	1%	0%

7.2.2 Cenário II – Redes com tráfego

Neste segundo cenário, além dos nós da nossa aplicação, existem outros nós monitorando alguma variável do ambiente e enviando suas leituras ao sorvedouro, gerando assim tráfego concorrente. Nas avaliações, consideramos uma rede com 128 nós, apenas um nó monitorando o ambiente, variamos o tamanho do item *stream* $|V|$ em 256,

512, 1024 e 2048 e para identificarmos o impacto da nossa solução numa rede com tráfego utilizamos 16%, 20%, 25% e 33% dos nós gerando tráfego até o sorvedouro. Seguindo as mesmas estratégias do cenário I, os valores de p_{min} são as médias dos tempos gastos para V ser inteiramente recebido pelo sorvedouro. A tabela 18 resume os valores de p_{min} para os diferentes tamanhos $|V|$ em cada porcentagem de nós monitorando o ambiente. Os valores para p_{min} com os intervalos de confiança são ilustrados na figura 29, onde no eixo-x temos a variação do tamanho do item *stream*, no eixo-y temos a média do atraso em segundos e as colunas representam a porcentagem dos nós na rede gerando tráfego.

TABELA 18 – Cenário II: Valores mínimos para os prazos exigidos pelas aplicações.

Porcentagem dos nós	**Tamanho do item *stream***			
	256	512	1024	2048
16%	1.47s	2.27s	3.86s	7.73s
20%	1.59s	2.40s	4.00s	7.86s
25%	1.78s	2.58s	4.18s	8.38s
33%	2.09s	2.91s	4.52s	9.08s

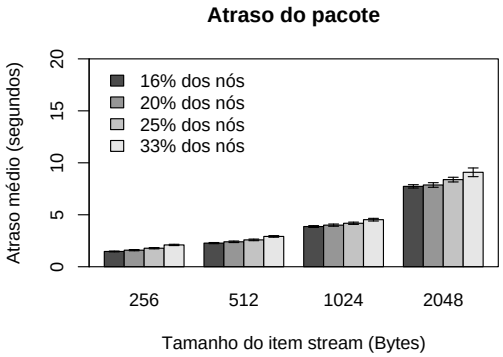


FIGURA 29 – Cenário II: Valores mínimos para os prazos exigidos pelas aplicações.

É importante destacar que, se a aplicação tem um prazo menor que os mostrado na tabela 18, os dados não poderão ser entregues no prazo e alguma redução é necessária.

Avaliação I – Prazos menores que os suportados

Inicialmente consideramos o prazo da aplicação $p_a = p_{min}/2$. A tabela 19 mostra os resultados do $atraso$ para essa avaliação. Os valores para o $atraso$ com os intervalos de confiança são ilustrados na figura 30, onde no eixo-x temos a variação do tama-

nho do item *stream*, no eixo-y temos a média do atraso em segundos e as colunas representam a porcentagem dos nós na rede gerando tráfego.

TABELA 19 – Cenário II: Atrasos identificados ao utilizar a metade dos prazos suportados pela rede.

Porcentagem dos nós	**Tamanho do item *stream***			
	256	512	1024	2048
16%	**0.96s**	**1.19s**	1.82s	3.25s
20%	**1.11s**	**1.48s**	1.84s	3.34s
25%	**1.42s**	**1.44s**	1.91s	3.25s
33%	**1.61s**	**1.69s**	2.12s	3.19s

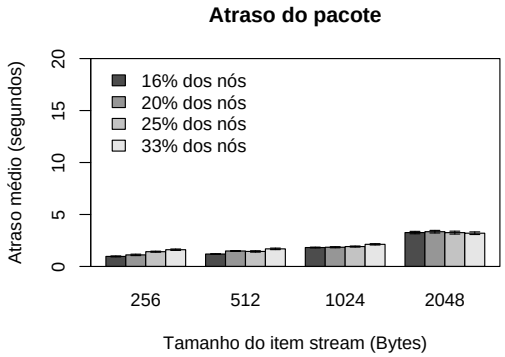


FIGURA 30 – Cenário II: Atrasos identificados ao utilizar a metade dos prazos suportados pela rede.

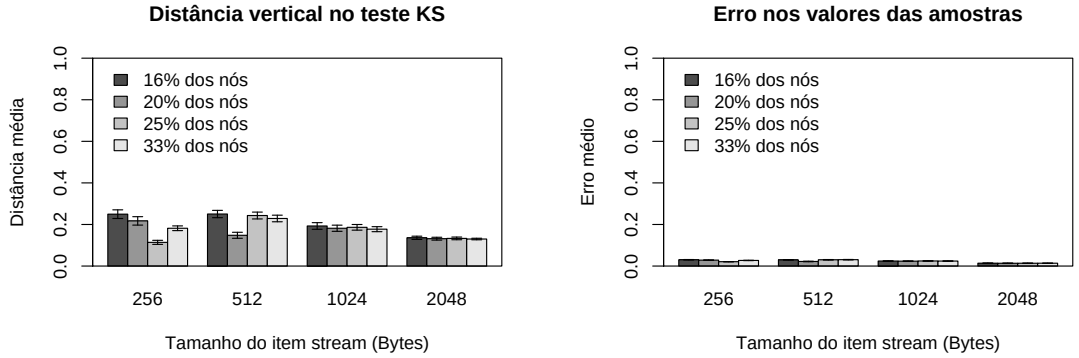
Na tabela 19 apresentamos o prazo da aplicação (p_a). Como podemos ver, quando o tamanho do item *stream* é pequeno ($|V| = 256$) os prazos não são alcançados (prazos não alcançados em destaque na tabela), quando comparados com a metade dos prazos apresentados na tabela 18. Porém esses prazos perdidos ainda são aceitáveis, uma vez que estamos tratando de aplicações de tempo real *soft*. Como no cenário I, isso acontece porque a redução pode ser observada no início da propagação dos dados e o atraso maior é proveniente dos outros nós gerando tráfego, o que mostra que através da nossa solução podemos perceber o atraso gerado pelo item *stream* e efetuar reduções apropriadas através da equação (7.5). O impacto dos atrasos também pode ser observado na tabela 20 onde é apresentada a razão $atraso/p_a$ (os prazos alcançados são identificados por "–" na tabela 10).

Ao aplicarmos as regras R'_{dst} e R'_{val} sobre aos dados recebidos, observamos que o comportamento dos dados é similar ao do cenário I, pois os níveis de redução efetuados são próximos. Os resultados para os erros associados às regras R'_{dst} e R'_{val}, com os intervalos de confiança, podem ser vistos na figura 31, onde no eixo-x temos

TABELA 20 – Cenário II: Razão entre os atrasos identificados e os prazos exigidos pela aplicação.

Porcentagem dos nós	**Tamanho do item *stream***			
	256	512	1024	2048
16%	31%	5%	–	–
20%	40%	25%	–	–
25%	59%	11%	–	–
33%	54%	16%	–	–

a variação do tamanho do item *stream*. No eixo-y temos o erro médio associado às regras R'_{dst} e R'_{val} e as colunas representam a porcentagem dos nós na rede gerando tráfego. Com esses resultados, as decisões D'_{dst} e D'_{val} sobre os dados reduzidos podem ser tomadas.



(a) Erro no teste KS.

(b) Erro nos valores dos dados.

FIGURA 31 – Cenário II: Erros identificados ao utilizar a metade dos prazos suportados pela rede.

Avaliação II – Atrasos gerados pelos nós roteadores

Agora, consideramos o prazo da aplicação $p_a = p_{min}$ com os nós roteadores atrasando os fragmentos do *stream* por $p_{min}/10000$ ou 0.01% do valor de p_{min}. A tabela 21 mostra os resultados do $atraso$ para essa avaliação. Os valores para o $atraso$ com os intervalos de confiança são ilustrados na figura 32, onde no eixo-x temos a variação do tamanho do item *stream*, no eixo-y temos a média do atraso, em segundos, e as colunas representam a porcentagem dos nós na rede gerando tráfego.

Como esperado, diferente dos resultados para a avaliação I, a tabela 21 mostra que p_a é alcançado em todos os casos, quando comparados com os prazos apresentados na tabela 18. Isso acontece, pois o tamanho estimado para a redução (equação (7.5)) está diretamente relacionado ao prazo da aplicação p_a. Esses resultados reforçam a

TABELA 21 – Cenário II: Atrasos identificados ao utilizar atrasos gerados pelos nós roteadores.

Porcentagem dos nós	**Tamanho do item *stream***			
	256	512	1024	2048
16%	1.28s	1.85s	3.01s	5.14s
20%	1.37s	1.89s	3.01s	5.32s
25%	1.53s	2.01s	2.94s	5.54s
33%	1.81s	2.21s	3.05s	5.78s

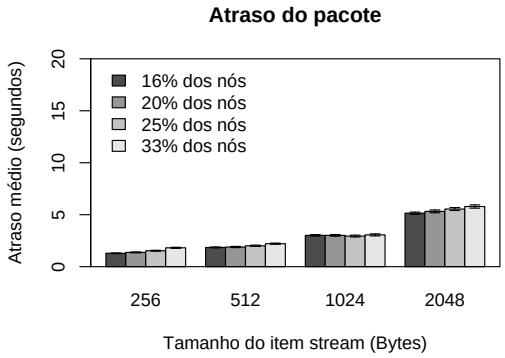


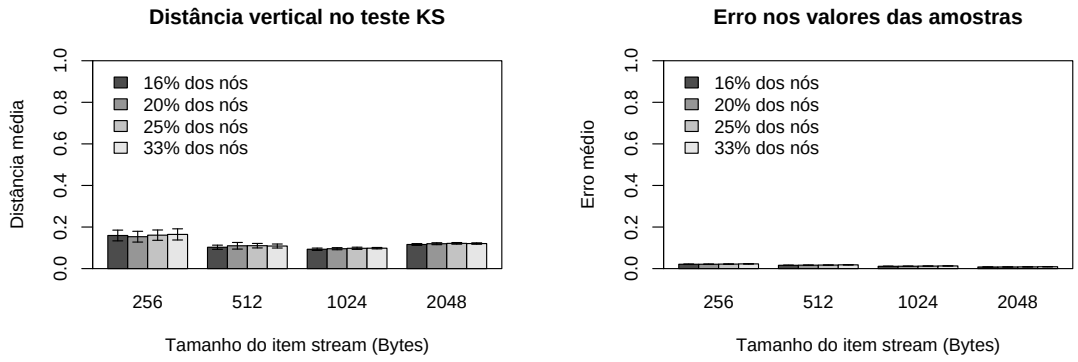
FIGURA 32 – Cenário II: Atrasos identificados ao utilizar atrasos gerados pelos nós roteadores.

eficiência da solução OGK nas aplicações de tempo real, uma vez que a solução se aplica a diferentes cenários avaliados.

Como na avaliação anterior, ao aplicarmos aos dados recebidos as regras R'_{dst} e R'_{val}, observamos que o comportamento dos dados é similar ao cenário I, uma vez que a redução efetuada sobre os dados possuem um tamanho próximo em ambos os cenários. Os resultados para os erros associados às regras R'_{dst} e R'_{val}, com os intervalos de confiança, podem ser vistos na figura 33, onde no eixo-x temos a variação do tamanho do item *stream*, no eixo-y temos o erro médio associado às regras R'_{dst} e R'_{val} e as colunas representam a porcentagem dos nós na rede gerando tráfego. Com esses resultados, as decisões D'_{dst} e D'_{val} sobre os dados reduzidos podem ser tomadas.

7.3 Conclusões parciais

Este capítulo apresentou a utilização da arquitetura OGK para as aplicações de redução de dados baseadas em técnicas de *stream* de dados em aplicações com exigências de prazos. Como pôde ser observado, os resultados mostraram que a solução OGK em aplicações de tempo real pode ser aplicada como solução parcial para o problema



(a) Erro do teste KS. (b) Erro nos valores dos dados.

FIGURA 33 – Cenário II: Erros identificados ao utilizar atrasos gerados pelos nós roteadores.

apresentado na definição 1 (pg. 53) e que podemos aceitar a hipótese 4 (pg. 55) que diz respeito a utilizar a redução de dados Ψ durante o roteamento para alcançar os prazos determinados pela aplicação e tomar as decisões D' de forma correta. Além disso, podemos encontrar os resultados aqui presentes para a solução OGK em aplicações de tempo real nos seguintes trabalhos:

1. AQUINO, A. L. L. et al. On the use data reduction algorithms for real-time wireless sensor networks. In: *IEEE Symposium On Computers and Communications (ISCC'07)*. Aveiro, Potugal: IEEE Computer Society, 2007. p. 583–588.

2. AQUINO, A. L. L.; CABRAL, R. da S.; FERNANDES, A. O. Um algoritmo de redução de dados para aplicações de tempo real em redes de sensores sem fio. In: *26st Brazilian Symposium on Computer Networks (SBRC'08)*. Rio de Janeiro, Brazil: SBC, 2008.

8 CONCLUSÃO

"*O homem nada pode aprender senão em virtude do que já sabe.*" (Aristóteles)

REDES de sensores sem fio possuem limitações de energia. Assim, aumentar seu tempo de vida é uma das tarefas mais importantes no projeto dessas redes. Normalmente, as redes de sensores sem fio coletam uma grande quantidade de dados do ambiente que, por suas características, podem ser modelados como *stream* de dados.

Neste trabalho aplicamos reduções de dados em redes de sensores sem fio através da utilização das técnicas de *stream* de dados. Para isso, propusemos uma arquitetura para redução de dados que possui uma API de redução que pode ser aplicada em diversas aplicações nas redes planas, hierárquicas e em aplicações de tempo real. Dentre as principais contribuições deste trabalho podemos destacar:

1. A definição e avaliação da arquitetura OGK para redução de dados que pode ser utilizada em diversos cenários e aplicações de redes de sensores onde os dados tenham características de *stream*.
2. A disponibilização da API-OGK utilizada para dar suporte a nossa arquitetura e que permite reduzir os dados nas redes de sensores de forma autônoma.
3. A utilização da arquitetura em topologias de redes planas e hierárquicas e para o roteamento.
4. A utilização da arquitetura como solução no desenvolvimento de aplicações de tempo real, onde é proposto uma formulação matemática para estimar o quanto deve ser reduzido para podermos atender aos prazos exigidos pela aplicação.

Baseado no problema, apresentado na definição 1 (pg. 53), relacionado à redução de dados em redes de sensores baseada nas técnicas de *stream* de dados, vimos através dos resultados, que a nossa solução pode ser aplicada ao problema proposto, uma vez que foi possível aceitar as hipóteses levantadas à respeito do problema. Para a hipótese 1 (pg. 55) que diz respeito ao projeto de uma solução genérica

onde é possível aplicar a redução de dados Ψ, vimos que através da arquitetura OGK (capítulo 4), do ponto de vista conceitual podemos aceitar essa hipótese. Além disso, os demais resultados apresentados nos capítulos 5 a 7 reforçam essa afirmação.

Considerando a hipótese 2 (pg. 55) que diz respeito à economia de recursos da rede sem perder a qualidade nos dados através da redução de dados em redes planas vimos, no capítulo 5, que essa hipótese pode ser aceita, uma vez que a utilização do algoritmo OGK-*rascunho* reduz o consumo de energia e o atraso mantendo constante o tamanho dos dados transmitidos $|V'|$. Além disso, como os dados podem ser gerados artificialmente no sorvedouro a qualidade dos dados não é afetada em relação as regras R'_{dist} e R'_{val}. Para o OGK-*amostragem*$_{aleatório}$ com amostra-($\log|V|$) temos economia no consumo de energia e no atraso por diminuir a quantidade de dados transmitidos. Entretanto, a qualidade dos dados é bastante afetada nas regras R'_{dist} e R'_{val} com erros de 20% e 10%, respectivamente. Para o OGK-*amostragem*$_{aleatório}$ com amostra-($|V|/2$) podemos utilizá-lo nos casos onde a prioridade da aplicação de sensoriamento é reduzir o erro quando aplicamos a regra R'_{val} ou nos cenários em que o tamanho do item *stream* e o número de nós monitorado o ambiente não variam.

Considerando a hipótese 3 (pg. 55) que diz respeito à economia de recursos da rede sem perder a qualidade nos dados através da redução de dados em redes hierárquicas vimos, no capítulo 6, que o comportamento da rede em relação ao consumo de energia e atraso em diferentes cenários foram satisfatórios ao utilizar a solução OGK, uma vez que, em todos os casos para o OGK-*amostragem*$_{aleatório}$ quando $|V'|$ diminui o consumo de energia e atraso diminuem e para o OGK-*rascunho* o comportamento é similar ao resultado para a amostra-($\log|V|$), que são os melhores resultados considerando esses cenários. Além disso, foi apresentado uma formulação matemática que confirma que as aplicações gerais têm um melhor desempenho quando modeladas para uma rede hierárquica ao invés de plana.

Considerando a hipótese 4 (pg. 55) que diz respeito a utilizar a redução de dados Ψ durante o roteamento para alcançar os prazos determinados pela aplicação e tomar as decisões D' vimos, no capítulo 7, que podemos utilizar a solução OGK para modelar as aplicações de tempo real. Além disso, a utilização do OGK-*oráculo* com o OGK-*amostragem*$_{central}$ se mostrou bastante satisfatória uma vez que para os diferentes cenários avaliados a maioria dos prazos das aplicações foram atendidos. Além disso, devido às moderadas taxas de redução efetuadas ao longo do roteamento a qualidade dos dados em relação às regras R'_{dist} e R'_{val} foi satisfatória, o que nos ajudou a tomar as decisões D' corretamente. Além disso, foi apresentado uma formulação matemática

que estima o tamanho $|V'|$ ideal quando é necessário efetuar a redução no momento do roteamento para atender aos prazos exigidos pelas aplicações.

De forma geral, os resultados revelaram que é possível utilizar a nossa solução para as diferentes aplicações gerais modeladas. Em todas as aplicações e cenários apresentados foi possível economizar recursos da rede (cerca de 90% de economia nos melhores casos) sem perder a representatividade dos fenômenos monitorados (um erro máximo de 20% nos piores casos). Especificamente, quando a arquitetura OGK foi integrada à fase de roteamento em aplicações de tempo real vimos através dos resultados que na maioria dos cenários foi possível atender aos prazos exigidos pela aplicação, uma vez que consideramos aplicações de tempo real *soft*, e ainda assim manter a representatividade dos fenômenos monitorados.

Apesar do seu grande potencial a solução OGK possui algumas limitações. De forma geral, temos que a solução OGK, não é aplicável à aplicações que utilizam sensores de imagem, como vídeo ou foto. Em relação as soluções de sensoriamento não resolvemos o problema de temporização, ou seja, identificação de dados mais recentes ou mais antigos para efetuar uma amostragem mais precisa em relação ao momento do sensoriamento. Considerando a solução para redes hierárquicas uma limitação é a forma com que os sensores são agrupados, caso utilizemos sensores com informações correlacionadas o resultados poderá se apresentar mais satisfatório. Por fim, para a solução em aplicações de tempo real uma limitação é que sempre estamos considerando que todos os itens chegarão ao nó roteador, ou seja, estamos desconsiderando a perda de pacotes.

Finalmente, podemos concluir que as aplicações gerais em redes de sensores sem fio podem utilizar a solução OGK, tanto para projetar suas aplicações, como para efetuar a redução em casos onde os nós sensores ou agregadores são os responsáveis pela redução e nos casos que se necessite reduzir os dados durante o roteamento, como nas aplicações de tempo real. Por fim, como possíveis trabalhos futuros podemos destacar:

1. Incorporar a arquitetura OGK a um mecanismo de filtragem e identificação do tipo do item de entrada, uma vez que até o momento isso é feito a priori na fase de projeto das redes.

2. Utilizar a arquitetura OGK para recuperação de falhas, através da redução de dados, nas aplicações gerais.

3. Utilizar a arquitetura OGK para prover garantias em aplicações que exijam qualidade de serviço (QoS).

4. Aplicar a arquitetura OGK como solução para problemas de controle de densidade e montagem de agrupamento baseado nos dados. No caso do controle de densidade, podemos considerar como critério de desativar um nó, não só a cobertura e o alcance mas também a importância do dado sensoriado para a aplicação.

5. Incorporar à arquitetura OGK o armazenamento de informações relacionadas ao *stream*. Com o objetivo de prover uma abstração da rede para que ela funcione como um banco de dados, aceitando consultas externas sobre dados monitorados.

6. Refinar o algoritmo OGK-*multivar*, através de estudos relacionados com dados multivariados para avaliarmos melhor a qualidade dos dados reduzidos por esse algoritmo.

7. Aplicar a redução de dados com o OGK-*multivar* em cenários com redes hierárquicas, para reduzir os dados do agrupamento baseado na sua correlação.

8. Variar a solução de roteamento com o objetivo de identificar diferentes propriedades no comportamento dos dados e assim efetuar reduções sensíveis à organização da rede.

9. Guardar informações dos itens já processados como, por exemplo, o erro, para utilizá-las nas reduções a serem efetuadas nos dados futuros. Ou seja, elaborar um mecanismo de realimentação dos algoritmos para tomarmos decisões de redução mais refinadas.

REFERÊNCIAS

ABADI, D. J. et al. An integration framework for sensor networks and data stream management systems. In: *30th International Conference on Very Large Data Bases (VLDB'04)*. Toronto, Canada: Morgan Kaufmann, 2004. v. 30, p. 1361–1364.

AFONSO, J. A. et al. Design and implementation of a real-time wireless sensor network. In: *1st International Conference on Sensor Technologies and Applications (SENSORCOMM'07)*. Valencia, Spain: IEEE Computer Society, 2007. p. 496–501.

AKCAN, H.; BRONNIMANN, H. A new deterministic data aggregation method for wireless sensor networksstar, open. *Signal Processing*, v. 87, n. 12, p. 2965–2977, May 2007.

AKYILDIZ, I. F. et al. A survey on sensor networks. *IEEE Communications Magazine*, v. 40, n. 8, p. 102–114, August 2002.

AL-KATEB, M.; LEE, B. S.; WANG, X. S. Adaptive-size reservoir sampling over data streams. In: *19th International Conference on Scientific and Statistical Database Management (SSDBM'07)*. Banff, Canada: IEEE Computer Society, 2007. p. 22–31.

ALIPPI, C. et al. Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications. In: *4th IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS'07)*. Pisa, Italy: IEEE Computer Society, 2007. p. 1–6.

ALON, N.; MATIAS, Y.; SZEGEDY, M. The space complexity of approximating the frequency moments. In: *28th Annual ACM Symposium on Theory of Computing (STOC'96)*. Philadephia, Pennsylvania, USA: ACM, 1996. p. 20–29.

ALTIPARMAK, F.; TUNCEL, E.; FERHATOSMANOGLU, H. Incremental maintenance of online summaries over multiple streams. *IEEE Transactions on Knowledge and Data Engineering*, v. 20, n. 2, p. 216–229, February 2008.

ANDRADE, A. V. et al. Analysis of selection and crossover methods used by genetic algorithm-based heuristic to solve the lsp allocation problem in mpls networks under capacity constraints. In: *International Conference on Engineering Optimization (EngOpt'08)*. Rio de Janeiro, Brazil: Springer, 2008. p. 1–15.

AQUINO, A. L. L.; CABRAL, R. da S.; FERNANDES, A. O. Um algoritmo de redução de dados para aplicações de tempo real em redes de sensores sem fio. In: *26st Brazilian Symposium on Computer Networks (SBRC'08)*. Rio de Janeiro, Brazil: SBC, 2008.

AQUINO, A. L. L. et al. Data stream based algorithms for wireless sensor network applications. In: *21st IEEE International Conference on Advanced Information*

Networking and Applications (AINA'07). Niagara Falls, Canada: IEEE Computer Society, 2007. p. 869–876.

AQUINO, A. L. L. et al. On the use data reduction algorithms for real-time wireless sensor networks. In: *IEEE Symposium On Computers and Communications (ISCC'07)*. Aveiro, Potugal: IEEE Computer Society, 2007. p. 583–588.

AQUINO, A. L. L. et al. A sampling data stream algorithm for wireless sensor networks. In: *IEEE International Conference on Communications (ICC'07)*. Glasgow, Scotland: IEEE Computer Society, 2007. p. 3207–3212.

AQUINO, A. L. L. et al. Sensor stream reduction for clustered wireless sensor networks. In: *23rd ACM Symposium on Applied Computing 2008 (SAC'08)*. Fortaleza, Brazil: ACM, 2008. p. 2052–2056.

ARAMPATZIS, T.; LYGEROS, J.; MANESIS, S. A survey of applications of wireless sensors and wireless sensor networks. In: *13th IEEE Mediterranean Conference on Control and Automation (MED'05)*. Hawaii, USA: IEEE Computer Society, 2005. p. 719–724.

ARTIGUENAVE, F. et al. The tropical biominer project: Mining old sources for new drugs. *OMICS: A Journal of Integrative Biology*, v. 9, n. 2, p. 30–138, June 2005.

BABCOCK, B. et al. Models and issues in data stream systems. In: *21th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02)*. Madison, Wisconsin, USA: ACM, 2002. p. 1–16.

BABU, S.; SUBRAMANIAN, L.; WIDOM, J. A data stream management system for network traffic management. In: *Workshop on Network-Related Data Management (NRDM'01)*. Santa Barbara, California, USA: ACM, 2001. p. 1–2.

BANERJEE, S.; KHULLER, S. A clustering scheme for hierarchical control in wireless networks. In: *20th IEEE Conference on Computer Communications (INFOCOM'01)*. Anchorage, Alaska: IEEE Computer Society, 2001. p. 1028–1037.

BAR-YOSSEFF, Z.; KUMAR, R.; SIVAKUMAR, D. Reductions in streaming algorithms, with an application to counting triangles in graphs. In: *13th ACM-SIAM Symposium on Discrete algorithms (SODA'02)*. San Francisco, CA: ACM, 2002. p. 623–632.

BASAGNI, S. Distributed clustering for ad hoc networks. In: *4th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'99)*. Fremantle, Australia: IEEE Computer Society, 1999. p. 310–315.

BROOKS, R. R.; IYENGAR, S. S. *Multi-Sensor Fusion: Fundamentals and Applications*. Upper Saddle River, New Jersey, USA: Prentice Hall, Inc., 1997. ISBN 0-13-901653-8.

BROWN, S.; SREENAN, C. J. A study on data aggregation and reliability in managing wireless sensor networks. In: *4th IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems (MASS'07)*. Pisa, Italy: IEEE Computer Society, 2007. p. 1–6.

BURIOL, L. S. et al. Using data stream algorithms for computing properties of large graphs. In: *Workshop on Massive Geometric Data Sets (MASSIVE'05)*. Pisa, Italy: On-line, 2005. p. 9–14.

BURIOL, L. S. et al. Estimating clustering indexes in data streams. In: *15th Annual European Symposium on Algorithms (ESA'07)*. Eilat, Israel: Springer, 2006. (Lecture Notes in Computer Science, v. 4698), p. 618–632.

BURIOL, L. S. et al. Counting triangles in data streams. In: *25th ACM Symposium on Principles of Database Systems (PODS'06)*. Chicago, Illinois, USA: ACM, 2006. p. 253–262.

CAMMERT, M. et al. A cost-based approach to adaptive resource management in data stream systems. *IEEE Transactions on Knowledge and Data Engineering*, v. 20, n. 2, p. 202–215, February 2008.

CHAN, T. H.; KI, C. K.; NGAN, H. *Real-time Support for Wireless Sensor Networks. Technical Report*. Hong Kong, JP, September 2005.

CHANG, Y.-C.; LIN, Z.-S.; CHEN, J.-L. Cluster based self-organization management protocols for wireless sensor networks. *IEEE Transactions on Consumer Electronics*, v. 52, n. 1, p. 75–80, February 2006.

CHARIKAR, M.; CHEN, K.; FARACH-COLTON, M. Finding frequent items in data streams. In: *29th International Colloquium on Automata, Languages and Programming (ICALP'02)*. Malaga, Spain: Springer, 2002. (Lecture Notes in Computer Science, v. 2380), p. 693–703.

CHEN, B. et al. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, v. 8, n. 5, p. 481–494, September 2002.

CHEN, M.; KNOW, T.; CHOI, Y. Energy-efficient differentiated directed diffusion (eddd) in wireless sensor networks. *Computer Communications*, v. 29, n. 2, p. 231–245, January 2006.

CHEN, Y. P.; LIESTMAN, A. L.; LIU, J. A hierarchical energy-efficient framework for data aggregation in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, v. 55, n. 3, p. 789–796, May 2006.

CORMODE, G. et al. Comparing data streams using hamming norms (how to zero in). *IEEE Transactions on Knowledge and Data Engineering*, v. 15, n. 3, p. 529–540, May/June 2003.

CVEJIC, N.; BULL, D.; CANAGARAJAH, N. Improving fusion of surveillance images in sensor networks using independent component analysis. *IEEE Transactions on Consumer Electronics*, v. 53, n. 3, p. 1029–1035, August 2007.

DASGUPTA, K.; KALPAKIS, K.; NAMJOSHI, P. Improving the lifetime of sensor networks via intelligent selection of data aggregation trees. In: *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'03)*. Orlando, Florida, USA: Springer, 2003. (Lecture Notes in Computer Science, v. 3397), p. 508–517.

DATAR, M. et al. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, v. 31, n. 6, p. 1794–1813, 2002.

DATAR, M.; MUTHUKRISHNAN, S. Estimating rarity and similarity over data stream windows. In: *10th Annual European Symposium on Algorithms (ESA'02)*. Rome, Italy: Springer, 2002. (Lecture Notes in Computer Science), p. 323–334.

DIAMOND, S. M.; CERUTI, M. G. Application of wireless sensor network to military information integration. In: *5th IEEE International Conference on Industrial Informatics (INDIN'07)*. Vienna Austria: IEEE Computer Society, 2007. p. 317–322.

DURRANT-WHYTE, H. F. Sensor models and multisensor integration. *International Journal of Robotics Research*, v. 7, n. 6, p. 97–113, December 1988.

ELNAHRAWY, E. *Research Directions in Sensor Data Streams: Solutions and Challenges. Technical Report*. New Brunswick, NJ, USA, May 2003.

ELSON, J. E. *Time Synchronization in Wireless Sensor Networks*. Tese (Doctor of Philosophy in Computer Science) — University of California, University of California, Los Angeles, 2003. Chair-Deborah L. Estrin.

ESTRIN, D. et al. Instrumenting the world with wireless sensor networks. In: *26th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'01)*. Salt Lake City, Utah, USA: IEEE Computer Society, 2001. v. 4, p. 2033–2036.

ESTRIN, D. et al. Next century challenges: Scalable coordination in sensor networks. In: *5th ACM/IEEE International Conference on Mobile Computing and Networks (MOBICOM'99)*. Seattle, Washington, USA: ACM, 1999. p. 263–270.

FIGUEIREDO, C. M. S. et al. Um esquema de gerenciamento para redes de sensores sem fio auto-organizáveis: Atuando sobre regras locais. In: *25st Brazilian Symposium on Computer Networks (SBRC'07)*. Belém, PA, Brazil: SBC, 2007. p. 1–12.

FIGUEIREDO, C. M. S. et al. Policy-based adaptive routing in autonomous wsns. In: *16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM'05)*. Barcelona, Spain: Springer, 2005. (Lecture Notes in Computer Science, v. 3775), p. 206–219.

FLAMMINI, A. et al. Sensor networks for industrial applications. In: *2nd International Workshop on Advances in Sensors and Interface (IWASI'07)*. Bari, Italy: IEEE Computer Society, 2007. p. 1–15.

GANESAN, D. et al. Coping with irregular spatio-temporal sampling in sensor networks. *ACM SIGCOMM Computer Communication Review*, v. 34, n. 1, p. 125–130, January 2004.

GAO, J. et al. Sparse data aggregation in sensor networks. In: *6th International Conference on Information Processing in Sensor Networks (IPSN'07)*. Cambridge, Massachusetts, USA: ACM, 2007. p. 430–439.

GEDIK, B.; LIU, L.; YU, P. S. Asap: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, v. 18, n. 12, p. 1766–1783, December 2007.

GEHRKE, J.; MADDEN, S. Query processing in sensor networks. *IEEE Pervasive Computing*, v. 3, n. 1, p. 46–55, January–March 2004.

GOLAB, L.; OZSU, M. T. Issues in data stream management. *ACM SIGMOD Record*, v. 32, n. 2, p. 5–14, June 2003.

GOUSSEVSKAIA, O. et al. Data dissemination based on the energy map. *IEEE Communications Magazine*, v. 43, n. 7, p. 134–143, July 2005.

GUHA, S. et al. Clustering data streams: Theory and practice. *IEEE Transactions on Knowledge and Data Engineering*, v. 15, n. 3, p. 515–528, May/June 2003.

GUIDONI, D. L. et al. Sistemas do tipo eixo-raio aplicados à redes de sensores sem fio modeladas como redes small world. In: *39th Brazilian Symposium on Operational Research (SBPO'07)*. Fortaleza, CE, Brasil: SOBRAPO, 2007. p. 1–12.

GUITTON, A.; SKORDYLIS, A.; TRIGONI, N. Utilizing correlations to compress time-series in traffic monitoring sensor networks. In: *IEEE Wireless Communications and Networking Conference (WCNC'07)*. Las Vegas, USA: IEEE Computer Society, 2007. p. 2479–2483.

HE, T. et al. Speed: A stateless protocol for real-time communication in sensor networks. In: *23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03)*. Providence, Rhode Island, USA: IEEE Computer Society, 2003. p. 46–55.

HEIDEMANN, J.; SILVA, F.; ESTRIN, D. Matching data dissemination algorithms to application requirements. In: *1st ACM International Conference on Embedded Networked Sensor Systems (SENSYS'03)*. Los Angeles, CA, USA: ACM, 2003. p. 218–229.

HEINZELMAN, W.; KULIK, J.; BALAKRISHNAN, H. Adaptive protocols for information dissemination in wireless sensor networks. In: *5th ACM/IEEE International Conference on Mobile Computing and Networks (MOBICOM'99)*. Seattle, Washington, USA: ACM, 1999. p. 174–185.

HEINZELMAN, W. R.; CHANDRAKASAN, A.; BALAKRISHNAN, H. Energy-efficient communication protocol for wireless microsensor networks. In: *33rd HAWAII International Conference on System Sciences (HICSS'00)*. Sland of Maui, Hawaii, USA: IEEE Computer Society, 2000. v. 2, p. 8020–8030.

HENZINGER, M. R.; RAQHAVAN, P.; RAJAGOPALAN, S. *Computing on data stream. Technical Report*. Palo Alto, CA, USA, 1998.

INDYK, P. A small approximately min–wise independent family of hash functions. In: *10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*. Baltimore, Maryland, United States: ACM, 1999. p. 454–456.

INTANAGONWIWAT, C. et al. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, v. 11, n. 1, p. 2–16, February 2003.

IOANNIDIS, Y. E.; POOSALA, V. Histogram-based approximation of set-valued query-answers. In: *25th International Conference on Very Large Data Bases (VLDB'99)*. Edinburgh, Scotland: Morgan Kaufmann, 1999. p. 174–185.

JACKSON, J. E. *A User's Guide to Principal Components*. Los Angeles, CA, USA: Wiley-Interscience, 2003. (Wiley Series in Probability and Statistics). ISBN 0471622672.

JAIN, A.; CHANG, E. Y. Adaptive sampling for sensor networks. In: *1st International Workshop on Data Management For Sensor Networks (DMSN'04)*. Toronto, Canada: ACM, 2004. v. 72, p. 10–16.

JUANG, P. et al. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. *ACM SIGPLAN Notices*, v. 37, n. 10, p. 96–107, October 2002.

KARAATA, M. H. Self-stabilizing clustering of tree networks. *IEEE Transactions On Computers*, v. 55, n. 4, p. 416–427, April 2006.

KIM, H.; PARK, J.; CHO, G. Statistical data aggregation protocol based on data correlation in wireless sensor networks. In: *International Symposium on Information Technology Convergence (ISITC'07)*. Jeonju, Republic of Korea: IEEE Computer Society, 2007. p. 130–134.

KRISHANAMACHARI, B.; ESTRIN, D.; WICKER, S. The impact of data aggregation in wireless sensor networks. In: *22nd IEEE International Conference on Distributed Computing Systems (ICDCS'02)*. Vienna, Austria: IEEE Computer Society, 2002. p. 575–578.

KRISHNA, P. et al. A cluster-based approach for routing in dynamic networks. *SIGCOMM Computer Communication Review*, v. 27, n. 2, p. 49–64, April 1997.

KRZANOWSKI, W. J. *Recent Advances in Descriptive Multivariate Analysis*. Oxford: Claredon Press, 1995. (Royal Statistical Society Lecture Notes Series, v. 2). ISBN 978-0-19-852285-0.

LEDLIE, J.; NG, C.; HOLLAND, D. A. Provenance-aware sensor data storage. In: *21st IEEE International Conference on Data Engineering Workshops (ICDE'05)*. Tokyo, Japan: IEEE Computer Society, 2005. p. 1189–1193.

LEE, S.; CHUNG, T. Data aggregation for wireless sensor networks using self-organizing map. In: *Artificial Intelligence and Simulation*. Jeju Island, Korea: Springer Berlin/Heidelberg, 2005. (Lecture Notes in Computer Science, v. 3397), p. 508–517.

LI, H.; SHENOY, P. J.; RAMAMRITHAM, K. Scheduling communication in real-time sensor applications. In: *10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'04)*. Toronto, Canada: IEEE Computer Society, 2004. p. 10–18.

LI, J.; ZHANG, Y. Interactive sensor network data retrieval and management using principal components analysis transform. *Smart Materials and Structures*, v. 15, n. 11, p. 1747–1757, December 2006.

LI, P.; GU, Y.; ZHAO, B. A global-energy-balancing real-time routing in wireless sensor networks. In: *2nd IEEE Asia-Pacific Service Computing Conference (APSCC'07)*. Tsukuba, Japan: IEEE Computer Society, 2007. p. 89–93.

LIAN, J.; NAIK, K.; AGNEW, G. B. A framework for evaluating the performance of cluster algorithms for hierarchical networks. *IEEE/ACM Transactions on Networking*, v. 15, n. 16, p. 1478–1489, December 2007.

LIAN, X.; CHEN, L. Efficient similarity search over future stream time series. *IEEE Transactions on Knowledge and Data Engineering*, (Accepted for publication) 2008.

LINS, A. et al. Beanwatcher: A tool to generate multimedia monitoring applications for wireless sensor networks. In: MARSHALL, A.; AGOULMINE, N. (Ed.). *6th IFIP/IEEE Management of Multimedia Networks and Services (MMNS'03)*. Belfast, Northern Ireland: Springer, 2003. (Lecture Notes in Computer Science, v. 2839), p. 128–141.

LINS, A. et al. Semi-automatic generation of monitoring applications for wireless networks. In: *9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'03)*. Lisbon, Portugal: IEEE Computer Society, 2003. p. 506–511.

LIU, C.; WU, K.; PEI, J. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Transactions on Parallel and Distributed Systems*, v. 18, n. 7, p. 1010–1023, July 2007.

LOUREIRO, A. A. F. et al. Wireless sensors networks (in portuguese). In: *21st Brazilian Symposium on Computer Networks (SBRC'03)*. Natal, RN, Brazil: SBC, 2003. p. 179–226.

LU, C. et al. Rap: A real-time communication architecture for large-scale wireless sensor networks. In: *8th IEEE Real-Time Technology and Applications Symposium (RTAS'02)*. San Jose, California, USA: IEEE Computer Society, 2002. p. 55–66.

LUO, H.; LIU, Y.; DAS, S. K. Routing correlated data in wireless sensor networks: A survey. *IEEE Network*, v. 21, n. 6, p. 40–47, November/December 2007.

LUO, R. C.; YIH, C.-C.; SU, K. L. Multisensor fusion and integration: Approaches, applications, and future research directions. *IEEE Sensors Journal*, v. 2, n. 2, p. 107–119, April 2002.

MADDEN, S. R. et al. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, v. 30, n. 1, p. 122–173, March 2005.

MAINWARING, A. et al. Wireless sensor networks for habitat monitoring. In: *1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*. Atlanta, Georgia, USA: ACM, 2002. p. 88–97.

MARBINI, A. D.; SACKS, L. E. Adaptive sampling mechanisms in sensor networks. In: *London Communications Symposium (LCS'03)*. London, UK: University College London, 2003. p. 1–4.

MENEZES, G. C. et al. Uma abordagem paralela para os problemas de cobertura e conectividade em redes de sensores sem fio. In: *37th Brazilian Symposium on Operational Research (SBPO'05)*. Gramado, RS, Brasil: SOBRAPO, 2005. p. 1–15.

MUDUNDI, S.; ALI, H. A robust scalable cluster-based multi-hop routing protocol for wireless sensor networks. In: *5th International Symposium Parallel and Distributed Processing and Applications (ISPA'07)*. Niagara Falls, Canada: Springer Berlin/Heidelberg, 2007. (Lecture Notes in Computer Science, v. 4742/2007), p. 895–907.

MUTHUKRISHNAN, S. *Data Streams: Algorithms and Applications*. Hanover, MA, USA: Now Publishers Inc, 2005. ISBN 1-933019-14-X.

NAKAMURA, E. F.; LOUREIRO, A. A. F.; FRERY, A. C. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys*, v. 39, n. 3, p. 9/1 – 9/55, April 2007.

NAKAMURA, E. F. et al. Using information fusion to assist data dissemination in wireless sensor networks. *Telecommunication Systems*, v. 30, n. 1–3, p. 237–254, November 2005.

NASRAOUI, O. et al. A web usage mining framework for mining evolving user profiles in dynamic web sites. *IEEE Transactions on Knowledge and Data Engineering*, v. 20, n. 2, p. 202–215, February 2008.

PAN, L. et al. Real-time monitoring system for odours around livestock farms. In: *IEEE International Conference on Networking, Sensing and Control (ICNSC'07*. London, UK: IEEE Computer Society, 2007. p. 883–888.

PENG, H. et al. An adaptive real-time routing scheme for wireless sensor networks. In: *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*. Niagara Falls, Ontario, Canada: IEEE Computer Society, 2007. v. 2, p. 918–922.

PHAM, T.; KIM, E. J.; MOH, M. On data aggregation quality and energy efficiency of wireless sensor network protocols. In: *First International Conference on Broadband Networks (BroadNets'04)*. San Jose, California, USA: IEEE Computer Society, 2004. p. 730–732.

PHUNG, N. D.; GABER, M. M.; ROHM, U. Resource-aware online data mining in wireless sensor networks. In: *IEEE Symposium on Computational Intelligence and Data Mining (CIDM'07)*. Honolulu, Hawaii, USA: IEEE Computer Society, 2007. p. 139–146.

POTTIE, G. J.; KAISER, W. J. Wireless integrated network sensors. *Communications of the ACM*, v. 43, n. 5, p. 51–58, May 2000.

REIS, I. A. et al. Data-aware clustering for geosensor networks data collection. In: *13th Brazilian Symposium on Remote Sensing (SBRS'07)*. Florianopolis, SC, Brazil: SBC, 2007. p. 6059–6066.

RESCHENHOFER, E. Generalization of the kolmogorov-smirnov test. *Computational Statistics & Data Analysis*, v. 24, n. 4, p. 422–441, June 1997.

ROHM, U.; SCHOLZ, B.; GABER, M. M. On the integration of data stream clustering into a query processor for wireless sensor networks. In: *International Conference on*

Mobile Data Management (MDM'07). Mannheim, Germany: IEEE Computer Society, 2007. p. 331–335.

ROMER, K.; MATTERN, F. The design space of wireless sensor networks. *IEEE Wireless Communications*, v. 11, n. 6, p. 54–61, December 2004.

ROYER, E. M.; TOH, C.-K. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, v. 6, n. 2, p. 46–55, April 1999.

RUIZ, L. B.; NOGUEIRA, J. M. S.; LOUREIRO, A. A. Manna: A management architecture for wireless sensor networks. *IEEE Communications Magazine*, v. 41, n. 2, p. 116–225, February 2003.

SANTINI, S.; ROMER, K. An adaptive strategy for quality-based data reduction in wireless sensor networks. In: *3rd International Conference on Networked Sensing Systems (INSS'06)*. Chicago, IL, USA: Transducer Research Foundation, 2006. p. 29–36.

SCHIZAS, I. D.; GIANNAKIS, G. B.; LUO, Z.-Q. Distributed estimation using reduced-dimensionality sensor observations. *IEEE Transactions on Signal Processing*, v. 55, n. 8, p. 4284–4299, August 2007.

SCHURGERS, C. et al. Topology management for sensor networks: Exploiting latency and density. In: *3rd ACM International Symposium on Mobile Ad-Hoc Networking & Computing (MOBIHOC'02)*. Lausanne, Switzerland: ACM, 2002. p. 135–145.

SEO, S.; KANG, J.; RYU, K. H. Multivariate stream data reduction in sensor network applications. In: *2nd International Symposium on Ubiquitous Intelligence and Smart Worlds (UISW'05)*. Nagasaki, Japan: Springer, 2005. p. 198–207.

SHARMA, A.; PALIWAL, K. K. Fast principal component analysis using fixed-point algorithm. *Pattern Recognition Letters*, v. 28, n. 10, p. 1151–1155, July 2007.

SHEN, X.; WANG, Z.; SUN, Y. Wireless sensor networks for industrial applications. In: *5th World Congress on Intelligent Control and Automation (WCICA'04)*. Hangzhou, China: IEEE Computer Society, 2004. v. 4, p. 3636–3640.

SIEGEL, S.; CASTELLAN, J. N. J. *Nonparametric Statistics for the Behavioral Sciences*. 2. ed. Columbus, OH, USA: McGraw-Hill Humanities/Social Sciences/Languages, 1988. ISBN 0070573573.

SOHRABI, K. et al. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, v. 7, n. 5, p. 16–27, October 2000.

SOLTAN, M.; MALEKI, M.; PEDRAM, M. Lifetime-aware hierarchical wireless sensor network architecture with mobile overlays. In: *IEEE Radio and Wireless Symposium (RWS'07)*. Long Beach, CA, USA: IEEE Computer Society, 2007. p. 325–328.

TILAK, S.; ABU-GHAZALEH, N. B.; HEINZELMAN, W. A taxonomy of wireless microsensor network models. *ACM SIGMOBILE Mobile Computing and Communications Review*, v. 6, n. 2, p. 28–36, April 2002.

VASS, D.; VIDACS, A. Distributed data agregation with geographical routing in wireless sensor networks. In: *IEEE International Conference on Pervasive Services (ICPS'07)*. Istanbul, Turkey: IEEE Computer Society, 2007. p. 68–71.

VLAJIC, N.; XIA, D. Wireless sensor networks:to cluster or not to cluster? In: *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM'06)*. Niagara-Falls, Buffalo-NY: IEEE Computer Society, 2006. p. 258–268.

WILLETT, R.; MARTIN, A.; NOWAK, R. Backcasting: adaptive sampling for sensor networks. In: *3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*. Berkeley, California, USA: ACM, 2004. p. 124–133.

WINTERLE, P.; STEINBRUCH, A. *Álgebra Linear*. 2. ed. São Paulo: Makron Books, 1987. ISBN 0074504126.

XU, J.; TANG, X.; LEE, W.-C. A new storage scheme for approximate location queries in object-tracking sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, v. 19, n. 2, p. 262–275, February 2008.

YADAV, P.; YADAV, N.; VARMA, S. Cluster based hierarchical wireless sensor networks (chwsn) and time synchronization in chwsn. In: *International Symposium on Communications and Information Technologies (ISCIT'07)*. Sydney, Australia: IEEE Computer Society, 2007. p. 1149–1154.

YU, Y.; KRISHNAMACHARI, B.; PRASANNA, V. K. Data gathering with tunable compression in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, v. 19, n. 2, p. 276–287, February 2008.

YUEN, K.; LIANG, B.; LI, B. A distributed framework for correlated data gathering in sensor networks. *IEEE Transactions on Vehicular Technology*, v. 57, n. 1, p. 578–593, January 2008.

ZHANG, Z.; MA, M.; YANG, Y. Energy-efficient multihop polling in clusters of two-layered heterogeneous sensor networks. *IEEE Transactions on Computers*, v. 57, n. 2, p. 231–245, February 2008.

ZHAO, J.; GOVINDAN, R.; ESTRIN, D. Residual energy scans for monitoring wireless sensor networks. In: *IEEE Wireless Communications and Networking Conference (WCNC'02)*. Orlando, Florida, USA: IEEE Computer Society, 2002. p. 356–362.

ZHAO, Y. J.; GOVINDAN, R.; ESTRIN, D. Computing aggregates for monitoring wireless sensor networks. In: *1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)*. Anchorage, AK, USA: IEEE Computer Society, 2003. p. 139–148.

ZHENG, R.; BARTON, R. Toward optimal data aggregation in random wireless sensor networks. In: *26th IEEE International Conference on Computer Communications (INFOCOM'07)*. Anchorage , Alaska , USA: IEEE Computer Society, 2007. p. 249–257.

ZHOU, Q.; XIONG, H.; LIN, H. Real-time performance analysis for wireless sensor networks. In: *IFIP International Conference on Network and Parallel Computing (NPC'07)*. Dalian, China: IEEE Computer Society, 2007. p. 337–342.

ZHU, J.; PAPAVASSILIOU, S. A resource adaptive information gathering approach in sensor networks. In: *IEEE Sarnoff Symposium on Advances in Wired and Wireless Communication (SARNOFF'04)*. Princeton, NJ, USA: IEEE Computer Society, 2004. p. 115–118.