MURILO KOUBAY DO AMARAL

AVALIAÇÃO DE DESEMPENHO DE ALGORÍTMOS DE CONSENSO EM CANAIS SEM FIO USANDO MODELO DE *OUTAGE*

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Curitiba
2013

MURILO KOUBAY DO AMARAL

AVALIAÇÃO DE DESEMPENHO DE ALGORÍTMOS DE CONSENSO EM CANAIS SEM FIO USANDO MODELO DE *OUTAGE*

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Marcelo Eduardo Pellenz

Curitiba
2013

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

A485a
2013

Amaral, Murilo Koubay do
Avaliação de desempenho de algoritmos de consenso em canais sem fio usando modelo de outage / Murilo Koubay do Amaral ; orientador: Marcelo Eduardo Pellenz. – 2013.
55 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2013
Bibliografia: f. 52-55

1. Redes sensoriais. 2. Algoritmos de computador. I. Pellenz, Marcelo Eduardo. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 22. ed. 005.73

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

PUCPR
GRUPO MARISTA

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 03/2013

Aos 07 dias do mês de Março de 2013 realizou-se a sessão pública de Defesa da Dissertação **"Avaliação de Desempenho de Algoritmos de Consenso em Canais sem Fio Usando Modelo de *Outage*"**.apresentado pelo aluno **Murilo Koubay do Amaral,** como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Marcelo E. Pellenz
PUCPR (Orientador) _____ APROVADO
(assinatura) (Aprov/Reprov.).

Prof. Dr. Manoel Camillo de O. Penna Neto (co-orientador)
PUCPR _____ APROV.
(assinatura) (Aprov/Reprov.).

Prof. Dr. Richard Demo Souza
UTFPR Richard Demo Souza APROV
(assinatura) (Aprov/Reprov.).

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado aprovado (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Fabrício Enembreck
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática
PPGIa
PUCPR

Dedico este trabalho a Deus, à minha Família
e a meus Amigos.

Agradecimentos

A Deus, pela força espiritual para a realização deste trabalho. Ao professor Marcelo Eduardo Pellenz, pela orientação, exemplo, confiança e amizade. Aos meus pais João e Aracy, pelo amor incondicional e por sempre acreditarem em mim. Um agradecimento especial para minha esposa Tatiely e meu filho Gabriel por todo carinho, paciência, incentivo e por compreender minhas ausências e omissões. A todas as pessoas que de alguma maneira contribuíram para a realização deste trabalho.

Sumário

Capítulo 3

Capítulo 4

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

n	Número de Nós de uma RSSF
$\mathbf{G}$	Grafo de uma RSSF
$\mathbf{V}$	Conjunto de Vértices do Grafo
$\mathbf{E}$	Conjunto de Arestas do Grafo
N_i	Conjunto de Vizinhos do Nó i
e_{ij}	Arestas entre os Nós i e j
$\mathbf{A}$	Matriz de Adjacência do Grafo
a_{ij}	Elementos da Matriz de Adjacência
d_i	Grau do Nó i
$\mathbf{L}$	Matriz Laplaciana do Grafo
$\mathbf{D}$	Matriz Diagonal
λ_2	Conectividade Algébrica da Rede
p	Probabilidade que define a conectividade dos nós
$\mathbf{W}$	Matriz de Pesos
$x_i(0)$	Valor de Medida Local do Nó i
x_{ave}	Média Real (Teórica) das Medidas do Nós
$x_i(t)$	Estimativa do Nó i no Tempo t
$\mathbf{x}(t)$	Vetor contendo as Estimativas Correntes dos Nós
$w_{ij}(t)$	Valor de Peso entre os Nós i e j
$\mathbf{W}^G$	Matriz de Pesos com base no Grafo G
d_{max}	Valor do Grau do Nó
$S(t)$	Subconjunto de Nós Sensores
L	Comprimento da Rota em Saltos
m	Mensagem Transmitida pelo Nó
β	Parâmetro de Ajuste de Medida
O	Ordem de Complexidade
R	Taxa de Transmissão

B	Largura de Banda do Sistema
$\mathcal{P}_{outage}$	Probabilidade de *Outage*
δ	Eficiência Espectral
P_i	Potência de Transmissão da Fonte
γ_{ij}	Perda de Percurso entre a fonte e o destino
h_{ij}	Coeficiente de Desvanecimento
x	Vetor de Dados Transmitido
$\mathbf{k}_j$	Ruído Aditivo Gaussiano Branco na Transmissão
d_{ij}	Distância em metros entre os nós i e j
α	Expoente de Perda de Percurso
Gn	Ganho Total das Antenas de Transmissão e Recepção
λ	Comprimento de Onda
f_c	Frequência da Portadora
M_l	Margem de Enlace
N_f	Figura do Ruído no Receptor
I_{ij}	Informação Mútua no Enlace
N	Potência do Ruído
N_0	Densidade Espectral de Potência
Hz	Hertz
E_b	Energia por Bit de Informação
$Pr\{\theta\}$	Probabilidade do Evento θ
d	Espaçamento da *grade*
e	Erro Médio Normalizado

Lista de Abreviações

RSSFs	Redes de Sensores Sem Fio
MAC	*Media Access Control*
GPS	*Global Positioning System*
SNR	*Signal-to-Noise Ratio*
FER	*Frame error rate*
AWGN	*Additive White Gaussian Noise*

Resumo

As *Redes de Sensores Sem Fio* (RSSF) têm sido amplamente utilizadas para a coleta, processamento e monitoramento de grandezas físicas nos mais variados ambientes e áreas de aplicação. Em determinadas aplicações pode haver a necessidade dos nós da RSSF se comunicaram entre si para realizarem tarefas distribuídas ou tomarem decisões coordenadas. Neste contexto surge o *problema do consenso*, pois os nós da rede precisam concordar sobre uma determinada decisão, de maneira que possam coordenar suas operações. Diversos algoritmos distribuídos de consenso foram propostos na literatura, contudo os limites de desempenho destes algoritmos em redes sem fio ainda é um tema pouco explorado. Neste trabalho investigamos o desempenho dos principais algoritmos de consenso assíncronos (*gossip*), utilizando uma modelagem de canal sem fio baseada em capacidade, usando probabilidade de *outage*. Neste estudo avaliamos os algoritmos do tipo *pairwise, neighborhood, geographic, path averaging e broadcasting*. Os resultados obtidos permitem identificar a estratégia mais adequada a ser utilizada em função das características do ambiente sem fio, de maneira a minimizar o consumo de energia para se atingir o consenso de média na rede.

Palavras-chave: Algoritmos de Consenso, Probabilidade de *Outage*, Redes de Sensores Sem Fio.

Abstract

The Wireless Sensor Networks (RSSF) have been widely used for collection, processing and monitoring of physical quantities in various environments and application areas. In some applications there may be the need for sensors RSSF communication with each other to perform tasks distributed or take coordinated decisions. In this context arises the problem of consensus, because the sensors network must agree on a particular decision, so they can coordinate their operations. Several distributed algorithms for consensus have been proposed in the literature, however the limits of performance of these algorithms in wireless networks is still a relatively unexplored subject. In this paper we investigate the performance of key algorithms Asynchronous consensus (gossip), using a wireless channel modeling based on ability, using probability of outage. In this study we evaluated the algorithms type pairwise, neighborhood, geographic, averaging path and broadcasting. The results allow to identify the most appropriate to be used according to the characteristics of the wireless environment, so as to minimize the energy consumption to achieve consensus in the network medium.

Keywords: Consensus algorithms, Probability of *Outage*, Wireless Sensor Networks.

Capítulo 1

Introdução

Com os recentes avanços ocorridos na área de microprocessadores e comunicações sem fio, cada vez mais tem se estimulado a utilização de sensores de menor porte e potência, sendo amplamente utilizados em *redes de sensores sem fio* (RSSFs) - do inglês: *Wireless Sensor Networks*. Como não necessitam de infraestrutura de hardware, são adotados nas mais diversas áreas, como médica, ambiental e militar, podendo ser aplicados por exemplo, no monitoramento de pacientes, detecção de incêndio em florestas, áreas de desastres, fundo de oceanos e campos de atividades nucleares.

De forma geral, estes sensores possuem capacidades de sensoriamento, processamento e comunicação, e, são responsáveis quando aplicados em RSSF em realizar a coleta das informações do ambiente, processá-las realizando cálculos simples e transmití-las para um nó de destino. À medida que estes sensores coletam as informações do ambiente e trocam com seus vizinhos, há repetidos gastos de energia, o que com o tempo pode ocasionar interrupções na rede devida sua capacidade ser limitada.

Por outro lado, em algumas aplicações pode haver a necessidade dos nós da RSSF se comunicaram entre si para realizarem tarefas distribuídas ou tomarem decisões coordenadas. Neste contexto surge o *problema do consenso*, onde os nós da rede precisam concordar sobre uma determinada decisão, de maneira que possam coordenar suas operações, necessitando de um algoritmo distribuído. Nesse caso, os algoritmos de consenso são normalmente os mais utilizados, devido sua simplicidade e estratégia descentralizada, sendo classificados em síncronos ou assíncronos (BéNéZIT, 2009), (AVRACHENKOV; CHAMIE; NEGLIA, 2011).

A partir da troca de mensagens entre os nós sensores, estes algoritmos procuram encontrar um valor médio de medida entre todos os valores de medidas iniciais dos nós. No momento em que todos os nós atualizam para esse valor médio, dizemos que o consenso foi atingido. Como exemplo de algoritmo síncrono temos o algoritmo de consenso médio

e para algoritmos assíncronos o *gossip*. Em aplicações de RSSFs, devido a restrições de sincronismo dos nós, os algoritmos síncronos não são adequados. Por outro lado, os algoritmos *gossip*, desempenham um papel importante no estudo do consenso neste tipo de rede. Os principais algoritmos *gossip* são: *pairwise, neighborhood, geographic, path averaging e broadcasting* (BéNéZIT, 2009).

Para o cálculo da média, o *pairwise* requer apenas um par de vizinhos ativos a cada iteração, onde estes atualizam suas estimativas para a média de seus valores iniciais. O comportamento do *neighborhood* é muito semelhante ao *pairwise*, uma vez que realiza o cálculo das estimativas da vizinhança inteira a cada iteração. Diferentemente, os algoritmos *geographic* e *path averaging* possuem algumas premissas complexas, como por exemplo, conhecer a posição dos nós e a topologia da rede. Estas premissas são importantes pois durante a execução do algoritmo é necessário sortear uma coordenada na rede fazendo com que os nós troquem informações com os nós mais próximos dessa coordenada.

No caso do *geographic*, este realiza a média par a par entre os nós, de maneira que o nó que iniciou a iteração em conjunto com o nó de destino mais próximo da coordenada atualizam suas estimativas para a média de seus valores iniciais. Já em relação ao *path averaging*, este cálcula a média realizando uma agregação de todas as estimativas dos nós que estão na rota até a dada coordenada. O algoritmo *broadcasting* não se preocupa com estas premissas, visto que é um algoritmo mais simples, onde a cada iteração um nó transmite sua estimativa para os vizinhos que estão no seu alcance, sem a necessidade de concordar com uma ação comum a qualquer momento. Todos os nós que receberam a estimativa atualizam os seus valores para o valor médio das medidas iniciais. Estes algoritmos serão estudados em detalhe no Capítulo 3.

1.1 Objetivo Geral

O objetivo desse trabalho é investigar o desempenho dos principais algoritmos de consenso assíncronos *gossip*, utilizando uma modelagem de canal sem fio baseada em capacidade, usando probabilidade de *outage*. Portanto, levando em conta que o canal pode falhar, através dos resultados obtidos identificaremos a estratégia mais adequada a ser utilizada em função das características do ambiente sem fio, de maneira a minimizar o consumo de energia para se atingir o consenso de média na rede.

1.2 Objetivos Específicos

Além do objetivo geral já enunciado, este trabalho também proporcionará as seguintes contribuições:

- Detalhamento dos algoritmos de consenso síncronos e assíncronos;
- Estudo do *problema de consenso* em RSSF;
- Avaliação de desempenho de diferentes algoritmos propostos na literatura para o caso de um modelo de canal de rádio baseado em *probabilidade de outage*;
- Investigação de qual estratégia é a mais adequada para RSSFs com topologia em *grade*, utilizando diferentes graus de conectividade e diferentes números de nós.

1.3 Estrutura do Trabalho

O restante deste documento está organizado da seguinte maneira. O Capítulo 2 contém o levantamento bibliográfico necessário para o estudo e desenvolvimento do documento. Neste capítulo apresentamos os conceitos básicos de RSSFs e iniciamos o estudo do *problema do consenso* e seus algoritmos. O Capítulo 3 apresenta em detalhes os algoritmos de consenso síncronos e assíncronos bem como suas aplicações. No Capítulo 4 é realizada a avaliação de desempenho dos algoritmos propostos, permitindo-se através dos resultados identificar a estratégia mais adequada a ser utilizada em função das características do ambiente sem fio. Por fim, estão presentes as conclusões e discussões sobre os resultados obtidos, sugestões de possíveis trabalhos futuros e as referências bibliográficas utilizados no trabalho.

Capítulo 2

Fundamentação Teórica

Neste Capítulo apresentamos alguns conceitos de RSSFs com enfoque nos algoritmos de consenso distribuídos. Tomamos como objetivo abordar os fundamentos que serão empregados nas propostas dos Capítulos 3 e 4. Iniciamos a Seção 2.1 descrevendo o conceito de RSSFs, utilizada para monitorar zonas de interesses com o propósito final de estimar parâmetros físicos e detectar situações de emergência em diversas aplicações. Na Seção 2.2, descrevemos como as matrizes de adjacência e laplaciana podem representar um grafo e quais suas propriedades a respeito da conectividade. Na Seção 2.3 abordamos o problema do consenso em RSSFs e suas aplicações. Na Seção 2.4 apresentamos o conceito de *Algoritmos de Consenso Médio Distribuído* e suas estratégias para resolver o problema de consenso. Por fim, na Seção 2.5, descrevemos os trabalhos relacionados que serviram de base para o desenvolvimento deste documento.

2.1 Redes de Sensores Sem Fio

As RSSFs são compostas de um grande número de dispositivos autônomos chamados nós sensores, utilizados para monitorar e controlar um determinado ambiente, podendo ser implantados tanto dentro como próximos ao fenômeno a ser investigado (AKYILDIZ et al., 2002). Geralmente instalados em locais de difícil acesso, podem sofrer com problemas de falhas de comunicação, necessitando possuir capacidade de autoconfiguração e adaptação, já que muitas vezes não possuem nenhum tipo de intervenção humana. Como possuem limitações de energia, capacidade computacional e memória, à medida que vão coletando informações tem de haver um esforço cooperativo entre eles a fim de processá-las, realizar cálculos simples e transmití-las.

Segundo *Akyildiz* e *Kazi*, (AKYILDIZ et al., 2002), (KAZI, 2010), as RSSFs podem ser constituídas de diversos sensores, tais como sísmico, térmico, acústico, umidade, pressão

e níveis de ruído que são aplicados em diversas áreas como:

- Na área militar, para detecção de intrusos em bases militares, detecção de movimentos de inimigos em terra ou mar e para vigilâncias em campo de batalha;
- Na área médica, para o fluxo de sangue, frequência respiratória, pressão arterial, medição de oxigênio e monitorar a localização de pacientes;
- Na área industrial, para a automação industrial, monitoramento e controle de equipamentos industriais;
- Na área automotiva, para a monitoração da pressão dos pneus e rastreamento de veículos;
- Na área física, para o monitoramente ambiental da água e do solo, observação dos sistemas biológicos e artificiais e monitoramento de pequenos animais;
- Na área de segurança pública, para detecção de locais afetados por catástrofes;
- Na área agrária, para a detecção da umidade do solo, pesticidas e herbicidas;
- Na área oceânica, para o monitoramento de peixes.

Sua implantação pode ser feita de diversas maneiras, como: espalhados aleatoriamente ou posicionados um a um, lançados por aviões, mísseis, foguetes ou por robôs. Após a implantação, podem sofrer alterações de topologia causadas pela mudança da posição do nó, fenômenos de propagação no canal sem fio, obstáculos móveis, disponibilidade de energia e mau funcionamento. Mesmo após a fase de implantação, é possível reimplantar sensores adicionais para executar outras tarefas e também para substituir sensores defeituosos.

Os protocolos e algoritmos utilizados devem ser tolerantes a falhas, já que com essa funcionalidade mesmo que ocorram falhas nos nós sensores não haverá interrupção na rede. Para tanto, os protocolos de roteamento propostos para redes convencionais ou para redes *ad hoc* não são adequados para redes de sensores, pois não consideram as limitações e características dessa rede, necessitando de adaptações para seu funcionamento.

Na Figura 2.1, apresentamos um exemplo de uma RSSF centralizada onde os nós encaminham suas medidas para uma unidade central de processamento, denotada por *sink*, responsável pela coleta e processamento dos dados da rede. Uma estrutura centralizada exige uma organização adequada dos nós e a implementação do protocolo de controle de acesso ao meio (MAC) - do inglês: *Media Access Control*, assim como de protocolos de

roteamento para encaminhar os dados para o nó *sink*. Isso pode eventualmente tornar a rede ineficiente e cara dependendo da aplicação, pois existem situações em que há alterações na rede, tanto pela falta de bateria como pela adição de um novo nó sensor, devendo os protocolos de roteamento e *MAC* serem reorganizados. Outra situação está relacionada com os requisitos de *hardware* para a comunicação sem fio, pois podem levar a um aumento no custo dos dispositivos, e assim, um aumento no custo global da rede, especialmente quando o número de nós é grande.

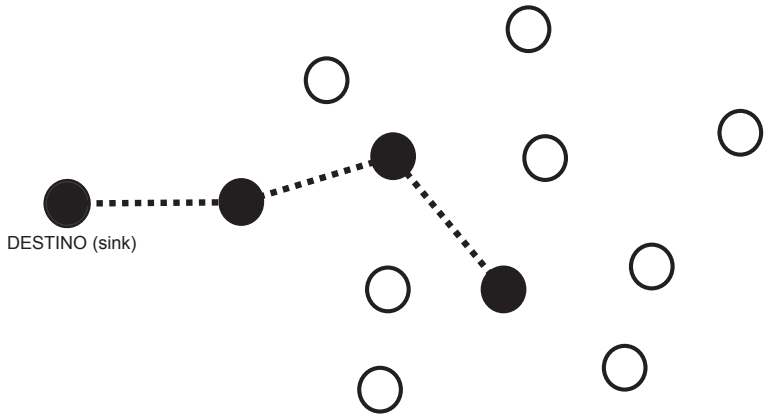


Figura 2.1: RSSF centralizada (AKYILDIZ et al., 2002)

Para certas aplicações, há a necessidade de utilizar redes descentralizadas, onde todos os nós possuem as mesmas capacidades e são capazes de realizar as mesmas tarefas. Nesse tipo de rede, os nós se organizam localmente e realizam os cálculos sem a necessidade de transmitir a informação para o nó central *sink*. Cada nó se comunica com seus vizinhos para trocar suas informações e tomar decisões. Uma rede descentralizada pode utilizar a estrutura hierárquica, por exemplo em uma rede de *cluster*, onde um nó de cada *cluster* funciona como um nó *sink*, sendo responsável por estabelecer a comunicação com os outros *sinks*. Nesse tipo de arquitetura, onde os nós cooperam entre si, pode haver a necessidade dos nós se comunicarem para realizar tarefas distribuídas. Para que isso aconteça, surge a necessidade da utilização de algoritmos distribuídos que serão estudados na Seção 2.4.

Em uma RSSF, a conectividade para comunicação de um par de nós (ou entre o nó i e o nó j) depende basicamente da potência de transmissão, da distância dos nós e das características do ambiente onde os nós estão implantados. O grau de conectivade da rede, é definido pelo nível de potência de transmissão, onde quanto maior for a potência de transmissão, maior será a quantidade de conexões entre os nós. O parâmetro da potência pode ser ajustado para um melhor controle da topologia, entretanto, na prática, normalmente os nós sensores possuem potência fixa. Para casos eventuais, a potência pode ser pré-estabelecida de um conjunto de valores finitos (PEREIRA; PAGES-ZAMORA, 2009).

Como os nós de uma RSSF possuem uma fonte limitada de energia, a energia torna-se um recurso escasso, devendo ser administrada adequadamente e sendo essencial para garantir uma vida mais longa para toda a rede. Uma maneira de otimizar esse problema é utilizar nós sensores auto recarregáveis, como por exemplo, modelos que utilizam energia solar. Em uma RSSF é desejável que os nós sejam simples, baratos, confiáveis e de preferência possuirem um tamanho reduzido, sendo capazes de executar cálculos simples e implementar protocolos de baixa complexidade (SILVANA, 2011).

2.2 Representação em Grafo de uma RSSF

O fluxo de informações entre os nós de uma RSSF é geralmente descrito através de grafos. Um grafo é uma abstração matemática usada para representar relações binárias entre os elementos de um conjunto (SILVANA, 2011). Esses elementos são chamados vértices e as relações são descritas por arestas entre pares de vértices. Em uma RSSF, os vértices representam os sensores e as arestas representam enlaces de comunicação sem fio entre esses nós.

Para uma RSSF com n nós, um grafo $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ consiste de um conjunto de vértices, $\mathbf{V}$, e um conjunto de arestas, $\mathbf{E}$, onde os vértices em $\mathbf{V} = \{1, ..., n\}$ correspondem aos nós da rede e as arestas $e_{ij} = (i, j) \in \mathbf{E}$ correspondem à possibilidade de comunicação entre os nos i e j (AVRACHENKOV; CHAMIE; NEGLIA, 2011). O conjunto de vizinhos N_i do nó i é denotado por:

$$N_i = \{j \in V : (i, j) \in E\}. \tag{2.1}$$

Eventualmente em uma RSSF, onde os nós operam com potências distintas, podem aparecer enlaces de comunicações unidirecionais. Nessas topologias, a rede pode ser representada por um grafo direcionado. Em um grafo direcionado, $e_{ij} \neq e_{ji}$. Em um grafo não direcionado, $\forall e_{ij} \in \mathbf{E}$ existe um $e_{ji} \in \mathbf{E}$. A Figura 2.2 apresenta um exemplo de ambos os grafos, direcionados e não direcionados, composto de 5 vértices e 6 arestas. No contexto desse trabalho vamos assumir uma RSSF onde os nós operam com potências iguais e os enlaces de comunicações são representados por grafos não direcionais.

A conectividade dos grafos é definida através de caminhos, onde o caminho de um vértice i para o vértice j é uma sequência de vértices distintos iniciando com o vértice i e terminando com o vértice j, de tal modo que os vértices consecutivos são adjacentes. Há diversos tipos de caminhos, podendo ser diferenciados por caminhos simples, direcionados, fortes, fracos, em círculos e em círculos direcionados. Por exemplo, para o

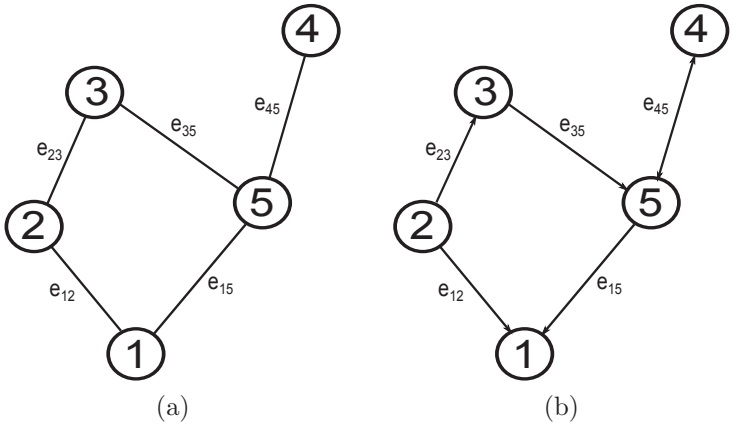


Figura 2.2: Exemplo de grafos: (a) Grafo não direcionado; (b) Grafo direcionado;

grafo direcionado da Figura 2.2b, a sequência de vértices $\{2, 3, 5\}$ com o conjunto de arestas $\{e_{23}, e_{35}\}$ representa um caminho forte e a sequência $\{2, 1, 5\}$ com o conjunto de arestas $\{e_{21}, e_{51}\}$ representa um caminho fraco. Um caminho forte no grafo é representado por uma sequência de vértices distintos com ordem consecutiva $1, ..., q \in \mathbf{V}$ tal que $e_{i,i-1} \in \mathbf{E}, \forall i = 2, ..., q$. Já um caminho fraco é uma sequência de vértices distintos com ordem consecutiva $1, ..., q \in \mathbf{V}$ tal que ambos $e_{i-1,i} \in \mathbf{E}$ ou $e_{i,i-1} \in \mathbf{E}$.

Em um grafo não direcionado $\mathbf{G}$, dois vértices i e j estarão conectados, se existir um caminho de i para j ou, equivalente, de j para i. Se todos os vértices tem o mesmo número de vizinhos, dizemos que é um grafo regular, já se para qualquer par de vértices há uma aresta conectada, denominamos grafo completo.

Para representar o grafo $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, pode-se utilizar a matriz de adjacência $\mathbf{A} = [a_{ij}]$, que possui dimensão $n \times n$, onde os elementos a_{ij} são definidos como:

$$a_{ij} = \begin{cases} 1, \ se \ \ (i,j) \in E \\ 0, \ caso \ contrario. \end{cases} \tag{2.2}$$

Na Figura 2.2a, o conjunto de vértices do grafo é $\mathbf{V} = \{1, 2, 3, 4, 5\}$ e o conjunto de arestas é $\mathbf{E} = \{e_{12}, e_{23}, e_{35}, e_{45}, e_{15}\}$. Nesse caso, a aresta $e_{12} = (1, 2)$ indica que os vértices 1 e 2 são adjacentes. Todo grafo simples $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, não direcionado, pode ser representado por matrizes. Desse modo, a matriz de adjacência da Figura 2.2a é:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{2.3}$$

O grau do nó i, denominado d_i, representa o número de arestas que existem entre o vértice i e seus vizinhos $j \in N_i$. O parâmetro d_i pode ser calculado a partir da matriz de adjacência, como:

$$d_i = \sum_{j=1}^{n} a_{ij} = \sum_{j=1}^{n} a_{ji} \tag{2.4}$$

Considerando o exemplo da Figura 2.2, os nós $\{1, 2, 3, 4, 5\}$ possuem grau $d_1 = d_2 = d_3 = 2$, $d_4 = 1$, $d_5 = 3$ respectivamente. Outra forma de representar um grafo é através da matriz Laplaciana $\mathbf{L} = [l_{ij}]$. É uma matriz simétrica $\mathbf{L} = \mathbf{D} - \mathbf{A}$ de dimensão $n \times n$, onde $\mathbf{D}$ é a matriz diagonal cujo valor dos elementos representa o grau de cada vértice e $\mathbf{A}$ é a matriz de adjacência do grafo (PEREIRA; PAGES-ZAMORA, 2008), (XIAO; BOYD, 2003) e (CHEN; FROLIK, 2011). Portanto, os elementos l_{ij} da matriz laplaciana $\mathbf{L}$ são definidos como:

$$l_{ij} = \begin{cases} d_i, \ se \ i = j \\ -1, \ se \ i \neq j \qquad e \quad (i,j) \in E \\ 0, \ caso \ contrario \end{cases} \tag{2.5}$$

Considerando o grafo da Figura 2.2a, sua matriz laplaciana é:

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & -1 & -1 & 3 \end{bmatrix} \tag{2.6}$$

Podemos notar que para cada linha da matriz $\mathbf{L}$, temos uma entrada com o grau do vértice representado por essa linha e a mesma quantidade de entradas com valor -1, cuja soma é igual a zero. Ou seja, $\mathbf{L} \cdot \mathbf{1} = 0 = 0 \cdot \mathbf{1}$. Conforme uma das propriedades da matriz laplaciana, todos os seus autovalores são reais e não negativos.

Para um grafo conectado, os autovalores são ordenados da seguinte maneira: $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$. Se o grafo é conexo, ou seja, se há pelo menos um enlace ligando um par de vértices, demonstra-se em (SARDELLITTI; BARBAROSSA; SWAMI, 2010),(FIEDLER, 1973), que o segundo menor autovalor (λ_2) é sempre não-nulo e positivo, ou seja, $\lambda_2 > 0$. Portanto, λ_2 é conhecido como *conectividade algébrica* e reflete o grau de conectividade do grafo (SARDELLITTI; BARBAROSSA; SWAMI, 2010), (KAR; MOURA, 2006) e (BARBAROSSA; SCUTARI; SWAMI, 2007). A *conectividade algébrica* está relacionada com o número de ligações do grafo, ou seja, $\lambda_2 > 0$ se e somente se $\mathbf{G}$ está conectado (FIEDLER, 1973). Na

verdade, a multiplicidade do autovalor Laplaciano zero é exatamente igual ao número de arestas conectadas em G e está relacionada com as condições para atingir o consenso.

O espectro Laplaciano, composto pelo conjunto de autovalores da matriz Laplaciana, descreve o nível de conectividade do grafo e é através dele que se obtém o λ_2. Como não é possível identificar se o grafo é conexo ou não conexo através do espectro do grafo, utilizam-se os autovalores da matriz Laplaciana para identificar o número de componentes conexas do grafo. As propriedades espectrais da matriz Laplaciana desempenham um papel importante na análise de convergência dos algoritmos de consenso. Os algoritmos de consenso serão descritos na Seção 2.3.

Para o estudo dos grafos, assume-se topologias que podem ser aplicadas tanto em grafo direcionado como não direcionado. Para isso, há diversos modelos, sendo as mais comuns as topologias em anel, *grade*, aleatória, *small-world* e *scale-free* representadas na Figura 2.3.

Uma rede em anel, Figura 2.3a, é uma rede unidimensional, cujos vértices são distribuídos espacialmente formando um círculo. Nesse tipo de rede cada nó tem exatamente dois vizinhos (SILVANA, 2011). No modelo de *grade*, Figura 2.3b, os vértices são espacialmente distribuídos de acordo com uma grade bidimensional e a distância entre um nó e seu vizinho é sempre a mesma (SILVANA, 2011). Redes aleatórias, Figura 2.3c, consistem de um conjunto de vértices espalhados aleatoriamente em uma área bidimensional, onde cada par de vértices estão conectados quando a distância euclidiana (métrica) entre eles é inferior a um determinado raio (PENROSE, 2003). Nessa topologia, o raio precisa ser assintoticamente maior que $\sqrt{\log n/n}$ para garantir a conectividade do grafo (GIRIDHAR; KUMAR, 2005). Em redes *small-world*, Figura 2.3d, a maior parte dos vértices não são vizinhos um do outro, mas podem ser alcançados por uma quantidade pequena de saltos. A partir de um grafo regular, as conexões aleatórias podem ser estabelecidas entre os vértices através das arestas existentes ou pela adição de novas arestas com probabilidade (p) diferente de zero (WATTS; STROGATZ, 1998). Por fim, em redes *scale-free*, Figura 2.3e, a idéia básica do modelo é que as redes não são construídas de uma só vez, mas sim com o tempo. Neste tipo de rede, alguns vértices estão fortemente conectados mas a maioria dos vértices têm um baixo número de ligações. Considera-se que à medida que a rede cresce e que novos nós são inseridos, estes irão se ligar preferencialmente aos nós com maior grau com uma probabilidade proporcional ao grau desses nós (BARABASI; ALBERT, 1999). Exemplos desta estrutura são comumente encontradas na natureza e na tecnologia, como por exemplo, a Internet (SILVANA, 2011).

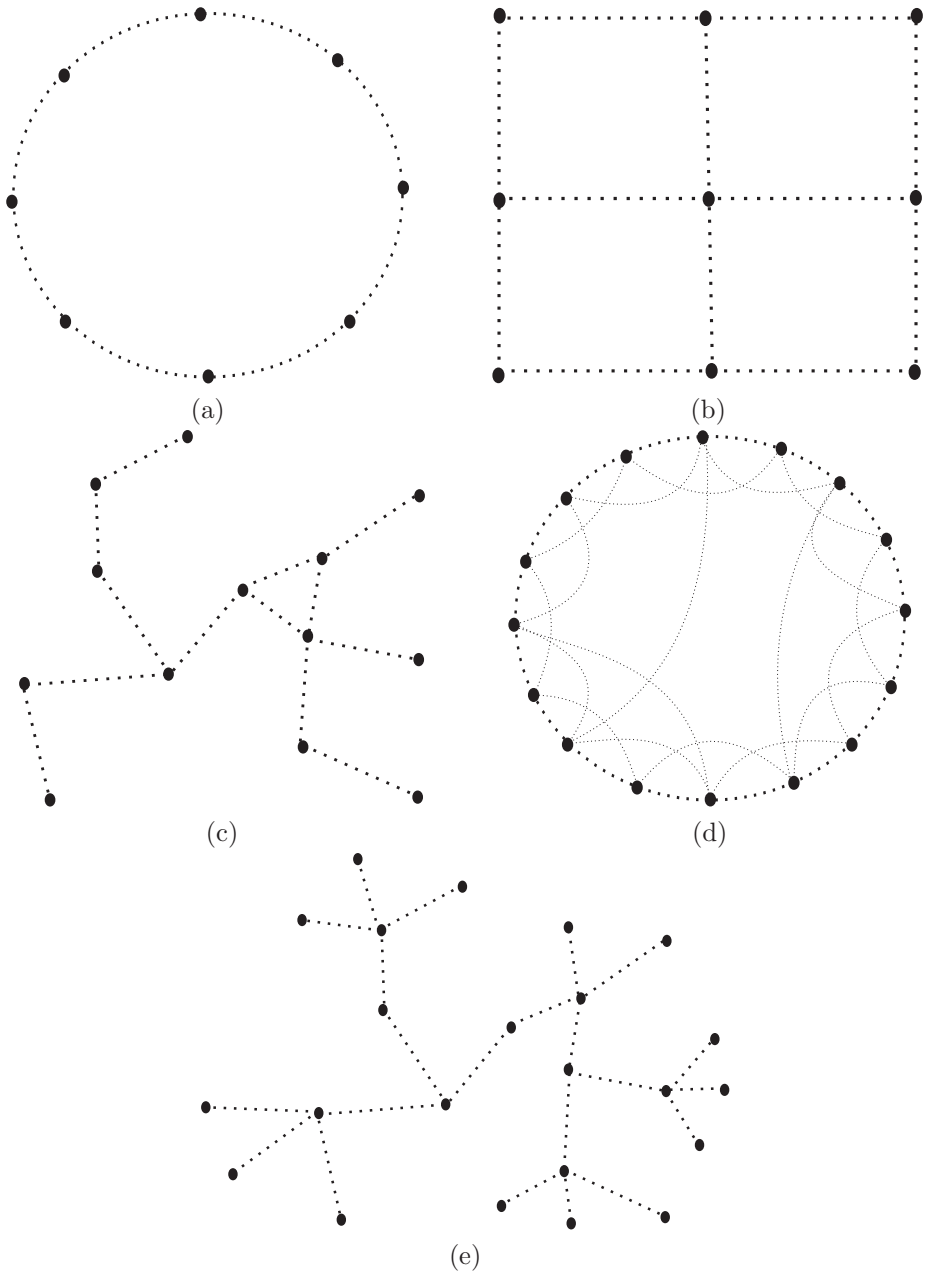


Figura 2.3: Exemplo de topologias: (a) Anel; (b) *Grade*; (c) Aleatória; (d) Small-world; (e) Scale-free;

2.3 Consenso em RSSFs

Também conhecido como o problema do acordo (BORKAR; VARAIYA, 1982), o consenso foi estudado por Tsitsiklis (TSITSIKLIS, 1984) e (TSITSIKLIS; BERTSEKAS; ATHANS, 1986). Seu primeiro trabalho foi em 1984 utilizando aplicações em computação distribuída com diversos processadores. O problema do consenso não aparece em redes centralizadas a qual possui a figura do nó central, mas sim em redes descentralizadas (BéNéZIT, 2009), onde todos os nós devem possuir a mesmas funcionalidades. O problema de acordo ou de consenso surge quando os nós da rede precisam concordar sobre uma decisão de maneira

que possam coordenar suas operações. Com isso, faz com que ele apareça em algumas aplicações, tais como:

- Monitoramento ambiental (AVRACHENKOV; CHAMIE; NEGLIA, 2011), (BRACA; MARANO; MATTA, 2008), (BOYD et al., 2006);
- Controle de formação distribuído (veículos autônomos) (EREN; BELHUMEUR; MORSE, 2002), (FAX; MURRAY, 2004);
- Fusão de sensores distribuídos (XIAO, 2005);
- Problema de encontro no espaço (robôs móveis) (ANDO et al., 1999), (LIN; MORSE; ANDERSON, 2003);
- Coordenação multiagente (JADBABAIE; LIN; MORSE, 2003), (REN; BEARD; ATKINS, 2005), (FAX; MURRAY, 2004), (OLFATI-SABER; FAX; MURRAY, 2007);
- Balanceamento de carga distribuído (processadores/computadores) (CYBENKO, 1989);
- Sincronização de osciladores acoplados (ERMENTROUT; KOPELL, 1984), (KURAMOTO, 1984), (OLFATI-SABER; FAX; MURRAY, 2007).

Para aplicações que envolvem o problema de encontro no espaço, consideramos um grupo de robôs móveis que possuem visão limitada e desejam se mover para um único ponto. Cada robô é um processador móvel omnidirecional que repetidamente observa as posições dos robôs que estão ao alcance e calcula a sua nova posição a partir desta observação, em seguida move-se para esta posição (ANDO et al., 1999), (LIN; MORSE; ANDERSON, 2003). Como segundo exemplo, citamos o controle cooperativo na formação de veículos, onde tem crescido o interesse em veículos que interagem de forma autônoma com o meio ambiente e veículos que realizam, na presença de incerteza e adversidade, tarefas além da capacidade de veículos individuais (FAX; MURRAY, 2004). As áreas de aplicação incluem veículos aéreos não tripulados (YEH; NELSON; SPARKS, 2000), veículos submarinos autônomos (BUZOGANY; PACHTER; D'AZZO, 1993) e sistemas rodoviários automatizados (BENDER, 1991). Como exemplo de uma aplicação de monitoramento ambiental, podemos observar a Figura 2.4 onde sensores são implantados para medir a temperatura T de uma determinada origem. Devido ao ruído gaussiano, cada sensor tem uma medida diferente da temperatura da origem. Portanto, se calcular a média das medidas iniciais, podemos ter uma boa estimativa da temperatura da origem.

A vantagem dos algoritmos de consenso é que eles podem calcular iterativamente este valor médio de uma forma completamente distribuída, através da troca de informações

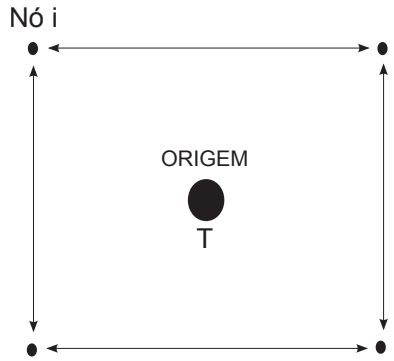


Figura 2.4: Exemplo de aplicação de consenso

locais entre os vizinhos e de cálculos simples de médias em cada sensor (AVRACHENKOV; CHAMIE; NEGLIA, 2011).

2.4 Algoritmos de Consenso Médio Distribuído

A maioria das aplicações que envolvem o consenso são voltadas para a determinação de um valor médio para uma certa grandeza. Esse valor médio é conhecido através de cálculos distribuídos executados pelos nós sensores à medida que trocam estimativas. No entanto, para que isso seja possível é necessário a utilização de *algoritmos de consenso médio distribuído*, onde cada nó sensor da rede tem um determinado valor e quer aprender a média de todos os valores da rede. Em geral, nestas situações os nós não conhecem a topologia da rede e nem a quantidade de nós sensores espalhados. Tudo o que eles podem aprender com facilidade é a identidade dos seus vizinhos e em alguns casos, através de um Sistema de Posicionamento Global (GPS) - do inglês: *Global Position System* ou um algoritmo de localização distribuído conhecer suas localizações. Todos os algoritmos médios distribuídos são executados de uma maneira iterativa a fim de calcular a média local até que cada nó tenha uma estimativa global precisa. Em cada iteração, um nó envia sua estimativa para um ou mais nós sensores da rede, em seguida, os nós que receberam os dados atualizam suas estimativas locais para a média com base nas informações recebidas.

Para projetar um algoritmo de cálculo de média distribuído, deve-se primeiramente especificar como os nós enviarão suas estimativas na rede. Em seguida deve-se projetar mecanismos para decidir como os nós atualizaram suas estimativas locais de maneira a convergir ao longo das iterações para uma média global. De acordo com (BéNéZIT, 2009), (AVRACHENKOV; CHAMIE; NEGLIA, 2011), os algoritmos de consenso médio são divididos em duas classes: síncronos e assíncronos. Em algoritmos síncronos, todos os nós na rede acordam a cada iteração e enviam suas estimativas. Antes do fim da iteração, todos os nós atualizam suas estimativas para a média dos valores recebidos. Já no caso dos algoritmos

assíncronos, a cada iteração somente um ou mais nós trasmitem para os nós escolhidos, o que faz com que somente um subconjunto de nós atualizem suas estimativas de média ao final de cada iteração. Para aplicações de RSSFs, devido a restrições de sincronismo dos nós, o algoritmo síncrono não é adequado.

2.5 Trabalhos Relacionados

Nesta seção são descritos alguns dos principais focos de pesquisa na área de consenso em RSSFs, levando em consideração os diferentes tipos de abordagens, tanto de algoritmos síncronos como assíncronos.

A velocidade com que os algoritmos de consenso convergem para a média, seja síncrono ou assíncrono, tem uma forte dependência da conectividade da rede. Por sua vez, a conectividade depende basicamente da potência de transmissão, da distância dos nós e das características do ambiente onde os nós estão implantados. A maioria das pesquisas nesta área visa acelerar o tempo de convergência ao passo que busca minimizar o consumo de energia e também maximizar a *conectividade algébrica* (XIAO; BOYD, 2003), (AVRACHENKOV; CHAMIE; NEGLIA, 2011). A *conectividade algébrica*, representada pelo λ_2, é definida como o segundo menor autovalor da matriz laplaciana. Conforme explicado na Seção 2.2, este valor é proporcional ao nível de conectividade da rede e é utilizado para medir a velocidade de convergência.

Quando tratamos de RSSF, existe um tradeoff entre a velocidade de convergência e o consumo de energia, não havendo na literatura um ajuste ideal de qual seria o valor ótimo para ambos. Tomamos como exemplo, um cenário onde o enlace entre qualquer par de nós é não simétrico a cada instante de tempo. Nesse caso, o consumo de energia da rede pode ser balanceado entre os nós quando o algoritmo de consenso é executado. Para isso, os nós devem transmitir usando diferentes níveis de potência a cada iteração, que podem ser escolhidos aleatoriamente em um conjunto finito de valores (PEREIRA; PAGES-ZAMORA, 2009). O consumo de energia pode ser visto ainda como no caso de uma rede densamente conectada, a qual requer um alto gasto de energia para garantir enlaces confiáveis entre os nós. Isso se deve ao fato de que ter uma rede totalmente conectada é equivalente a ter muitos nós transmissores, o que faz com que o tempo de convergência seja reduzido, embora o consumo de energia para manter essa rede conectada é máximo.

Por outro lado, a rede com o mínimo de conectividade requer pouca energia para manter a topologia, mas tem uma lenta convergência. Desse modo, uma maneira de minimizar esta situação é a escolha de uma topologia ótima, onde os nós utilizam uma potência de transmissão comum de forma que o gasto com energia é reduzido e a topologia

da rede é mantida por mais tempo. O total de energia gasto para alcançar o consenso pode ser medido como o produto das potências de transmissão necessárias para estabelecer os enlaces entre os nós vizinhos e do tempo de convergência (SARDELLITTI; BARBAROSSA; SWAMI, 2010).

Se os enlaces são simétricos, ou seja, se o grafo que descreve a topologia da rede é não direcionado, a velocidade de convergência pode ser medida através da *conectividade algébrica*. A maximização da conectividade algébrica de um grafo não direcionado, pode ser realizada pela escolha adequada dos pesos de cada aresta, ou, através da adição de arestas, a partir de um conjunto de arestas candidatas (GHOSH; BOYD, 2006). Dependendo do tipo de grafo, a velocidade de convergência pode ser mais rápida pela adição de pesos maiores em determinados enlaces. Os pesos são representados através da *matriz de pesos* $\mathbf{W}$, onde os elementos da matriz $\mathbf{W}$ representam os pesos w_{ij} dos nós i e j. A *matriz de pesos* é aplicada em algoritmos síncronos e assíncronos, onde a topologia da rede varia com o tempo. Para a escolha dos pesos, há algumas estratégias como: *uniform weights*, *metropolis weights* e *adapted weights*, que serão melhor explicadas no capítulo 3.

Para exemplificar a aplicação dos pesos, podemos representar uma rede com estrutura em *cluster*, onde os nós sensores tendem a formar redes ou ilhas isoladas que normalmente são conectadas por um único enlace. Como a comunicação se dá por meio de uma única via e o peso do enlace é igual para toda rede, o número de iterações para o algoritmo de consenso convergir é elevado. As informações tendem a ficar confinadas dentro de cada *cluster* causando lentidão na propagação de um *cluster* a outro. Geralmente dentro do *cluster*, as estimativas locais convergem rapidamente para a média dos valores iniciais, já para atingir a média global, a convergência é mais lenta por possuir apenas uma via de comunicação.

Neste tipo de cenário, uma das soluções é aumentar o peso do enlace que interliga os *clusters*, conforme apresentado em (AVRACHENKOV; CHAMIE; NEGLIA, 2011). Como os nós não conhecem toda a topologia da rede, necessitam selecionar seus pesos correspondentes com base em informações locais. Com um maior peso no enlace, a troca de mensagens é acelerada entre os *clusters* minimizando o tempo de convergência do algoritmo, embora o consumo de energia seja elevado com o aumento do número de transmissões.

Como em RSSF a sincronicidade das iterações leva a uma complexidade extra, o algoritmo de consenso síncrono acaba não sendo adequado para este cenário sem fio. Neste caso, os algoritmos de consenso assíncronos têm sido amplamente adotados. A principal classe de algoritmos assíncronos é a *gossip*, sendo divididos em: *pairwise*, *neighborhood*, *geographic*, *path averaging* e *broadcast*. Dependendo do tipo da aplicação, há a necessidade que os nós troquem informações de maneira distribuída e cheguem no valor médio

das estimativas iniciais. Para isso, podemos utilizar por exemplo o algoritmo *pairwise*, onde a cada iteração, um nó ativo envia sua estimativa para outro nó vizinho que é escolhido aleatoriamente na rede. Essa comunicação é realizada par a par e a velocidade de convergência é fortemente influenciada pelo tamanho da rede.

Outra maneira de chegar ao consenso pode ser através do algoritmo *neighborhood*, bastante similiar ao *pairwise*. Neste algoritmo, o nó que iniciou a iteração juntamente com seus vizinhos atualizam suas estimativas. Outra possibilidade é com a utilização do *geographic*, o qual permite rotear informações entre dois nó aleatórios que estão a distâncias maiores. Como premissa deste algoritmo, todos os nós devem conhecer as localizações dos outros nós e podem explorar essas localizações através do roteamento. Mesmo que o custo de roteamento seja caro em termos de RSSF, a rapidez na difusão das estimativas compensa esse custo. Outro algoritmo que utiliza roteamento é o *path averaging*, que diferentemente do *geographic*, realiza a agregação dos valores dos nós a medida que a rota é traçada até a posição alvo.

Em algoritmos *broadcast*, todos os nós no alcance do nó origem, atualizam suas estimativas sem a necessidade de qualquer tipo de roteamento complexo ou operações par a par. Nesse modelo, um nó da rede, escolhido aleatoriamente, transmite a sua estimativa e esse valor é recebido com sucesso por todos os nós em um raio pré-definido de conectividade. Os nós que receberam as estimativas enviadas atualizam seus próprios valores e os nós restantes que não receberam as estimativas mantém seus valores de origem. Nesse tipo de algoritmo *gossip*, é importante atingir o consenso com um custo de comunicação mínimo (BOYD et al., 2006). Uma vez que a cada iteração do algoritmo *broadcast* vários nós recebem as estimativas e atualizam seus valores, é esperado uma diminuição significativa nos custos de comunicação para atingir o consenso em relação aos algoritmos *gossip* relatados anteriormente.

2.6 Considerações Finais

Neste capítulo foram apresentados os conceitos básicos de RSSFs e sua representação em grafo através das matrizes de adjacência e Laplaciana. Discutimos algumas pesquisas que estudam a relação do espectro Laplaciano com a conectividade da rede e descrevemos a ideia do algoritmo de consenso e suas aplicações. Descrevemos também as principais estratégias utilizadas para resolver o problema do consenso. Enfim, apresentamos toda a fundamentação teórica que contextualizará os próximos capítulos.

Capítulo 3

Algoritmos de Consenso

Atingir o consenso sob parâmetros comuns globais através de algoritmos totalmente descentralizados é um assunto que tem atraído bastante atenção nos últimos anos. Diversos algoritmos têm sido desenvolvidos, sendo divididos em duas classes principais, síncronos e assíncronos (AVRACHENKOV; CHAMIE; NEGLIA, 2011), (BéNéZIT, 2009). Como exemplo de algoritmo síncrono podemos citar o de *consenso médio* e para o caso assíncrono, o *gossip*. Em algoritmos de consenso distribuídos, os nós se comunicam uns com os outros para chegar a um acordo a respeito de um determinado valor de interesse, sem a necessidade de transmitir qualquer informação para um nó central ou *sink*.

Considere uma RSSF com n nós, onde no instante inicial $t = 0$, cada nó i possui um valor local $x_i(0) \in \Re$. O objetivo do algoritmo de consenso médio é calcular a média x_{ave} de uma maneira iterativa e distribuída e que após um determinado tempo todos os nós da rede convirjam para o valor médio comum, onde

$$x_{ave} = \frac{1}{n} \sum_{i=0}^{n} x_i(0). \tag{3.1}$$

De maneira geral, uma iteração de um algoritmo de consenso pode ser expressada como:

- Um ou mais nós acordam;
- Os nós ativos enviam suas estimativas para um ou mais vizinhos na rede;
- Os nós atualizam suas estimativas para a média de sua estimativa e da estimativa recebida.

Assim, cada nó i mantém na iteração t uma estimativa $x_i(t)$ da média baseada nas estimativas recebidas dos nós vizinhos. Todas as estimativas são reunidas em um vetor n denotado por $\mathbf{x}(t)$. O algoritmo de consenso médio tem que garantir que todas

as estimativas convergem para a x_{ave}:

$$\lim_{t\to\infty} \mathbf{x}(t) \;=\; \frac{\mathbf{1}\mathbf{1}^T}{n}\mathbf{x}(0) \;=\; x_{ave}\cdot\mathbf{1} \tag{3.2}$$

onde $\mathbf{1}$ é um vetor coluna com todas as entradas iguais a 1.

A cada iteração, um nó recebe as estimativas de seus vizinhos, ao passo que com as estimativas recebidas, atualiza sua medida. Independente do algoritmo ou da iteração t, há sempre uma matriz $\mathbf{W}(t)$, cujos elementos correspondem a seus pesos, tal que:

$$\mathbf{x}(t) = \mathbf{W}(t)\,\mathbf{x}(t-1). \tag{3.3}$$

Em resumo, há três restrições principais para a matriz $\mathbf{W}(t)$ (XIAO; BOYD, 2003), (BéNéZIT, 2009), (AVRACHENKOV; CHAMIE; NEGLIA, 2011):

- O consenso tem que ser estável. Se no tempo t, todos os nós concordam com uma estimativa de média, então os nós devem manter esse acordo da estimativa para tempos maiores que t. Conforme a equação anterior, esta condição pode ser representada matematicamente por

$$\mathbf{W}(t)\cdot\mathbf{1} \;=\; \mathbf{1} \tag{3.4}$$

- Segundo, garantir que a média das estimativas, a qualquer momento deve ser igual a x_{ave}, ou seja, a soma das estimativas devem permanecer constante.

$$\mathbf{1}^T\cdot\mathbf{W}(t) \;=\; \mathbf{1}^T \tag{3.5}$$

- Finalmente, as estimativas devem convergir para a média verdadeira.

Mesmo se a matriz $\mathbf{W}(t)$ satisfazer estas condições para garantir a convergência das estimativas para a x_{ave}, a maneira como é realizada a escolha dos pesos difere de algoritmo para algoritmo.

3.1 Algoritmo de Consenso Médio Síncrono

Em algoritmos de consenso médio síncrono, todos os nós acordam a cada iteração t, se comunicam com seus vizinhos e atualizam suas estimativas. Cada elemento da matriz

$\mathbf{W}(t)$ é denotado por w_{ij} e o conjunto de vizinhos do nó i é denotado por $N_i(t)$. Assim, para qualquer instante t e qualquer nó i, a regra de atualização local pode ser escrita como:

$$x_i(t) = w_{ii}(t) \cdot x_i(t-1) + \sum_{j \in N_i(t)} w_{ij}(t) \cdot x_j(t-1) \tag{3.6}$$

Para exemplificar, consideramos o pseudo-código do Algoritmo 1, onde cada nó i possui sua estimativa inicial $x_i(0)$ e t_{fim} representa o número de iterações do algoritmo. A construção da matriz de pesos $\mathbf{W}$ pode ser realizada através de algumas estratégias como: *uniform weights*, *metropolis weights* e *adaptive weights*.

Considerada como a estratégia mais simples, *uniform weights* (BéNéZIT, 2009), para qualquer par de nós i e j, o peso $w_{ij}(t)$ é escolhido para ser igual a uma constante α, onde $\alpha = \frac{1}{n}$, sendo comum para toda rede. O valor dos pesos $w_{ii}(t)$ são adaptados localmente pelos nós contando quantos vizinhos existem no instante t:

$$w_{ij}(t) = \begin{cases} \alpha, \ se \ j \in N_i(t) \\ 1 - \alpha|N_i(t)|, \ se \ i = j \\ 0, \ caso \ contrario \end{cases} \tag{3.7}$$

Outro algoritmo é o *Metropolis weights*, também conhecido como *Local Degree* (AVRACHENKOV; CHAMIE; NEGLIA, 2011). Nesse algoritmo os nós necessitam conhecer o grau de seus vizinhos e para isso assume-se que a rede não tem enlaces unidirecionais. Uma maneira possível de implementar o *Metropolis weights* é a seguinte: os nós enviam via *broadcast* suas estimativas, em seguida, eles transmitem o número de estimativas que receberam. Por fim, os pesos são calculados como:

$$w_{ij}(t) = \begin{cases} \frac{1}{1+max\{|N_i(t)|,|N_j(t)|\}}, \ se \ j \in N_i(t) \\ 1 - \sum_{k \in N_i(t)} \ w_{ik}(t), \ se \ i = j \\ 0, \ caso \ contrario \end{cases} \tag{3.8}$$

Já para o caso do algoritmo *Adapted weights*, este considera que os enlaces da rede podem falhar ao longo do tempo. Nessa estratégia, é definida uma matriz de pesos $\mathbf{W}^G$ que realiza o consenso médio com êxito no grafo fixo $\mathbf{G}$. A cada iteração, os pesos $w_{ij}(t)$ são projetados baseando-se nas falhas atuais e em $\mathbf{W}^G$. O nó $j \in N_i(t)$ se e somente se entre os nós i e j não há falha no instante t. Se o enlace (i, j) não falha, então $w_{ij}(t) = w^G_{ij}$. Se o enlace $e_{ij} = (i, j)$ falha, então necessariamente $w_{ij}(t) = 0$. Então, os nós i e j, adicionam o peso w^G_{ij} para $w_{ii}(t)$ e $w_{jj}(t)$.

$$w_{ij}(t) = \begin{cases} w_{ij}^{G},\ se\ j \in N_i(t) \\ 1 - \sum_{k \in N_i(t)}\ w_{ik}(t),\ se\ i = j \\ 0,\ caso\ contrario \end{cases} \tag{3.9}$$

Dois outros algoritmos podem ser utilizados para a escolha dos pesos, o *Max Degree* e o *Best Constant* (AVRACHENKOV; CHAMIE; NEGLIA, 2011). Para o caso do *Max Degree*, cada nó assume conhecer o grau máximo de seus vizinhos ($d_{max} = max_i\{d_i\}$) e o mesmo peso é associado a cada enlace. Os pesos w_{ii} são determinados da seguinte forma:

$$w_{ij}(t) = \begin{cases} \frac{1}{d_{max}\{d_i, d_j\}},\ se\ (i,j) \in E\ e\ i \neq j \\ 0,\ se (i,j) \notin E\ e\ i \neq j \end{cases} \tag{3.10}$$

O algoritmo *Best Constant* utiliza um valor constante de peso. Em (XIAO; BOYD, 2003), é demonstrado que sua velocidade de convergência é maior que aqueles que usam pesos uniformes e que cada peso w_{ij}, com $i \neq j$ tem que ser selecionado igual a $\frac{2}{\lambda_1(L)+\lambda_{n-1}(L)}$, onde L é a matriz laplaciana do grafo e λ_i é seu autovalor.

Algoritmo 1: Algoritmo de Consenso Médio Síncrono (BéNéZIT, 2009)

1: **for** $t = 1 : t_{fim}$ **do**
2: Cada nó i transmite sua estimativa $x_i(t-1)$ na rede e recebe as estimativas $x_j(t-1)$ dos nós vizinhos, $j \in N_i(t)$.
3: Os nós que receberam as estimativas atualizam para:

$$x_i(t) \leftarrow w_{ii}(t)\ x_i(t-1) + \sum_{j \in N_i(t)}\ w_{ij}(t)\ x_j(t-1)$$

onde $N_i(t)$ é o conjunto de nós de que i recebeu a estimativa.
4: **end for**

O principal desafio para qualquer nó i na construção da matriz $\mathbf{W}$ é a capacidade em aprender localmente seus pesos $\mathbf{W}_i$, onde $\mathbf{W}_i = [w_{i1}, w_{i2}, ..., w_{in}]$. Baseado no conhecimento dos vizinhos $N_i(t)$, o nó i deve escolher seu peso coordenado com o peso dos outros nós, de modo que as condições 3.4 e 3.5 sejam verificadas. Como a rede varia com o tempo em configurações sem fio, então os nós devem concordar previamente com uma estratégia para a escolha dos pesos.

Nessa Seção descrevemos o funcionamento do algoritmo de consenso síncrono e as principais estratégias de cálculos do pesos, que são utilizados para a construção da matriz $\mathbf{W}$. Como o fator do sincronismo em RSSFs pode ser bem restritivo, muitas pesquisas abordam o problema do consenso utilizando algoritmos assíncronos, os quais serão detalhados na próxima Seção.

3.2 Algoritmos de Consenso Médio Assíncronos

Os algoritmos de consenso assíncronos, utilizam o modelo de tempo assíncrono (BOYD et al., 2005),(BOYD et al., 2006), onde cada nó tem um relógio independente e acorda em momentos aleatórios. Esse processo aleatório pode ser modelado com um processo de Poisson. O tempo é medido em termos do número de iterações e assume-se que há sempre uma iteração de cada vez na rede, não sendo necessário tratar problemas de controle de congestionamento. Do ponto de vista prático, esse tempo médio de acordar um nó deve ser um valor suficientemente grande para que uma iteração do algoritmo de consenso seja fechada.

A principal classe de algoritmos de consenso assíncronos é denominada *gossip* (BéNéZIT, 2009), onde os principais algoritmos são:

- *Pairwise* (BOYD et al., 2006), (DENANTES et al., 2008);
- *Neighborhood* (NAZER; DIMAKIS; GASTPAR, 2011), (NAZER; DIMAKIS; GASTPAR, 2009);
- *Geographic* (DIMAKIS; SARWATE; WAINWRIGHT, 2006);
- *Path averaging* (BéNéZIT, 2009);
- *Broadcast* (AYSAL et al., 2008), (AYSAL; YILDIZ; SCAGLIONE, 2008).

A diferença entre o algoritmo *gossip* e o algoritmo de *consenso médio* é que para o de *consenso médio* todos os nós atualizam a cada iteração t e no caso do *gossip*, dependendo do algoritmo, apenas um subconjunto de nós atualizam a cada iteração t. O subconjunto de nós, denotado por $S(t)$, atualizam suas estimativas para a média das estimativas de $S(t)$. Para todo $j \in S(t)$, temos que:

$$x_j(t+1) = \frac{1}{|S(t)|} \sum_{i \in S(t)} x_i(t) \tag{3.11}$$

Em algoritmos *gossip* para se garantir a convergência é necessário obedecer a seguinte *propriedade*: para qualquer algoritmo *gossip*, em qualquer iteração t, a matriz $\mathbf{W}(t)$ é uma matriz de projeção, de modo que os cálculos de média dentro dos algoritmos são representados por essa matriz, tal que:

$$\mathbf{W}(t) \cdot \mathbf{W}(t) = \mathbf{W}(t) \tag{3.12}$$

Adicionalmente, $\mathbf{W}(t)$ deve ser uma matriz simétrica, $\mathbf{W}(t)^T = \mathbf{W}(t)$ (BéNéZIT, 2009).

Conforme mencionamos, existem diferentes algoritmos *gossip* propostos na literatura, estes algoritmos serão descritos em detalhe na Seção subsequente.

3.2.1 Algoritmo Pairwise

Conhecido como algoritmo clássico de *gossip*, o algoritmo *pairwise*, também denominado *standard*, (BOYD et al., 2006), (AYSAL et al., 2008), (BéNéZIT, 2009), requer apenas um par de vizinhos ativos a cada iteração e estes atualizam suas estimativas para a média de seus valores. O pseudocódigo deste algoritmo é demonstrado no Algoritmo 2.

Algoritmo 2: *Pairwise* (BOYD et al., 2006), (BéNéZIT, 2009)

1: **for** $t = 1 : t_{fim}$ **do**
2: O nó i envia sua estimativa $x_i(t-1)$ e qualquer nó que tenha recebido $x_i(t-1)$ seta um tempo de *backoff* aleatório.
3: O vizinho j com o menor tempo de *backoff* envia sua estimativa $x_j(t-1)$ para o nó i.
4: O nó i confirma a recepção.
5: O vizinhos de j ouvem a confirmação e não encaminham suas estimativas.
6: Ambos os nós i e j atualizam suas estimativas para:

$$x_i(t) \leftarrow \frac{x_i(t-1)+x_j(t-1)}{2},$$
$$x_j(t) \leftarrow \frac{x_i(t-1)+x_j(t-1)}{2}.$$

7: **end for**

Considerando uma topologia de rede conectada, quando o nó i está ativo, ele envia sua estimativa para o vizinho j, que é escolhido aleatoriamente pelo nó i. Assumimos que cada nó i possui uma estimativa inicial $x_i(0)$ e que t_{fim} representa o número de iterações. Do ponto de vista do grafo ideal (estático), com as ligações bem definidas, para cada iteração do algoritmo *pairwise*, executada entre os passos $1-7$ numa iteração $(t-1)$, o número de mensagens (pacotes transmitidos) na rede é igual a 3.

Através da Figura 3.1 é possível verificar a execução do algoritmo *pairwise*. No exemplo, os nós 3 e 4 ouvem a mensagem de confirmação e não enviam suas estimativas para o nó 1. Em seguida, os nós 1 e 2 atualizam suas estimativas para a média dos valores.

A matriz de pesos do algoritmo *pairwise*, para o nó transmissor i e o nó receptor j no tempo t é modelada como:

$$\mathbf{W}(k) = \mathbf{I} - \frac{(e_i - e_j)(e_i - e_j)^T}{2} \tag{3.13}$$

onde e_i denota o i^th vetor coluna da matriz identidade $\mathbf{I}$ (BOYD et al., 2006). No *pairwise*, cada nó comunica com seu nós vizinhos, isto é, $j \in N_i$ (SILVANA, 2011).

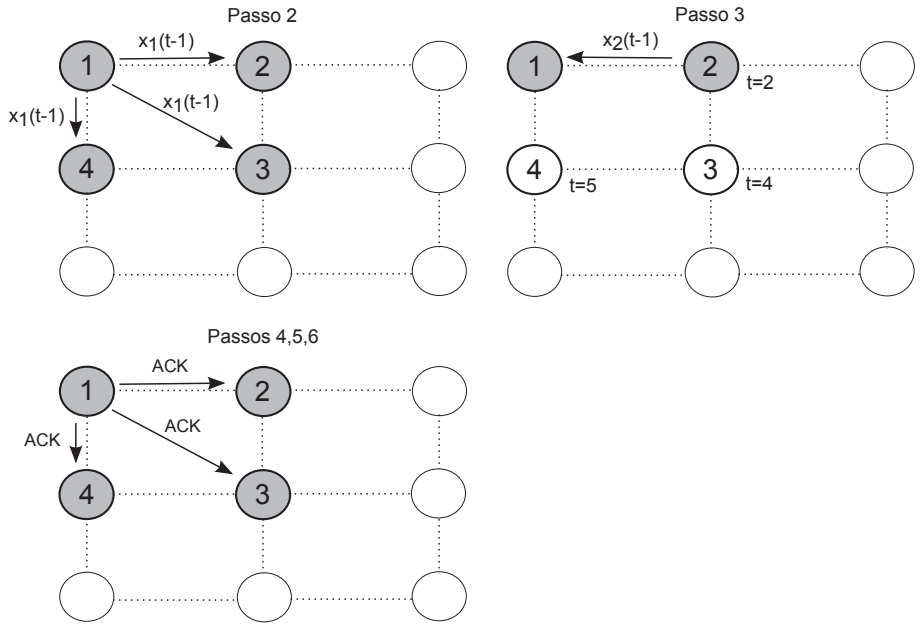


Figura 3.1: Execução do Algoritmo *Pairwise*

3.2.2 Algoritmo Neighborhood

Outro algoritmo bastante similar ao *pairwise* é o *neighborhood*, introduzido por (NAZER; DIMAKIS; GASTPAR, 2009), (NAZER; DIMAKIS; GASTPAR, 2011), realiza o cálculo das estimativas da vizinhança inteira a cada iteração. O pseudocódigo deste algoritmo é demonstrando no Algoritmo 3.

Algoritmo 3: *Neighborhood* (NAZER; DIMAKIS; GASTPAR, 2009), (NAZER; DIMAKIS; GASTPAR, 2011), (BéNéZIT, 2009)

1: **for** $t = 1 : t_{fim}$ **do**
2: O nó i transmite uma mensagem que está ativo.
3: Qualquer nó que recebeu a mensagem, envia sua estimativa para o nó i.
4: Considerando que $N_i(t)$ é o conjunto de nós de que i recebeu as estimativas, então o nó i calcula

$$y \leftarrow \frac{1}{|N_i(t)|} \sum_{k \in N_i(t)} x_k(t-1).$$

5: Após calcular y, o nó i envia y e os identificadores dos nós pertecentes ao conjunto $N_i(t)$.
6: Então, o nó i e todos os nós pertencentes a $N_i(t)$ (que receberam y) atualizam suas estimativas para y.
7: **end for**

Considerando uma topologia de rede conectada, onde o nó i envia uma mensagem informando seus vizinhos j de que está ativo, todos os nós que receberam esta mensagem enviam suas estimativas para ele. Assumimos que cada nó i possui uma estimativa inicial $x_i(0)$ e que t_{fim} representa o número de iterações. Do ponto de vista do grafo ideal

(estático), com as ligações bem definidas, para cada iteração do algoritmo *neighborhood* o número de mensagens (pacotes transmitidos) varia dependendo do número de vizinhos do nó. Ao final de uma iteração, houve $|N_i|+1$ mensagens enviadas, onde $|N_i|$ representa a quantidade de vizinhos, contando com o nó i.

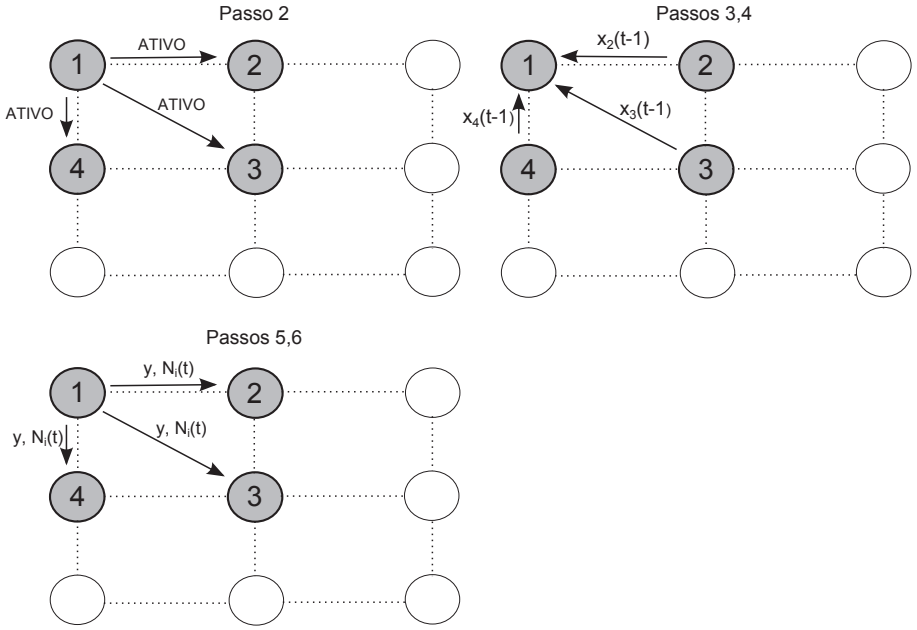


Figura 3.2: Execução do Algoritmo *Neighborhood*

Através da Figura 3.2 é possível verificar a execução do algoritmo *neighborhood*. No exemplo, N_i representa o conjunto dos nós de que o nó 1 recebeu as estimativas. O nó 1 calcula o valor de y. Em seguida, o nó 1 e todos os nós em N_i (que receberam y) atualizam seus estados para o valor de y.

3.2.3 Algoritmo Geographic

Conforme discutido na Seção 3.2.1, uma das limitações do algoritmo *pairwise* é a lenta convergência. Nesse sentido foi proposto um algoritmo alternativo, denominado *geographic*, (DIMAKIS; SARWATE; WAINWRIGHT, 2006) onde qualquer par de nós pode trocar suas estimativas através do roteamento, como se a rede fosse um grafo completo. O algoritmo assume atuar em redes onde os nós conhecem suas localizações bem como a localização de seus vizinhos.

Em relação ao funcionamento do algoritmo, a cada iteração, um nó acorda e gera uma rota aleatória para um nó de destino j. Tanto o nó que iniciou a rota e o nó de destino atualizam suas estimativas para a média. Assim, a rota precisa ser utilizada para encaminhar primeiro em uma direção e após no sentido contrário. Este roteamento pode ser feito, por exemplo, usando a estratégia do algoritmo *greedy routing* (BéNéZIT, 2009),

cujo pseudocódigo é mostrado no Algoritmo 5.

Através do roteamento é possível chegar o mais próximo do nó alvo, então, no passo 5 do Algoritmo 4, o nó j pode rotear novamente a nova estimativa y usando o roteamento também. No algoritmo *geographic*, cada iteração custa L mensagens, onde L é o comprimento da rota, em saltos, gerada pelo roteamento. É possível demonstrar em (BéNéZIT, 2009) que o custo em mensagens em uma iteração para uma rede com topologia em grade é $L \leq 2\sqrt{n}$. Considerando como exemplo uma RSSF com $n = 9$ nós, o custo em mensagens é $L \leq 6$.

Algoritmo 4: *Geographic* (DIMAKIS; SARWATE; WAINWRIGHT, 2006), (BéNéZIT, 2009)

1: **for** $t = 1 : t_{fim}$ **do**
2: O nó i seta a posição alvo T de forma uniforme e aleatória.
3: O nó i inicia o *greedy routing* com o alvo $\mathbf{T}$ e mensagem $m = (x_i(t-1), l_i)$, onde l_i é a posição do nó i.
4: Seja j o nó de destino da *greedy route* anterior. Assim, o nó j calcula
$$y = \frac{x_i(t-1)+x_j(t-1)}{2},$$
e atualiza sua estimativa para $y : x_j(t) \leftarrow y$.
5: O nó j envia a atualização y para o nó i, usando o caminho de volta.
6: O nó i atualiza sua estimativa para $y : x_i(t) \leftarrow y$.
7: **end for**

Dentro do cenário da rede, o nó i sorteia uma posição alvo de maneira aleatória e uniforme, ou seja, qualquer posição de maneira equiprovável dentro da rede. Através do *greedy routing*, tenta chegar até o nó que está mais próximo dessa posição. Quando se atinge esse nó com o envio da mensagem m, este atualiza a sua média. Os passos do algoritmo são representados através da Figura 3.3. Assumimos que cada nó i possui uma estimativa inicial $x_i(0)$, t_{fim} é o número de iterações, $\mathbf{T} = (x, y)$ é a posição alvo na rede e $\mathbf{m} = [x_i, l_i]$ representa a mensagem. Do ponto de vista do grafo ideal (estático), com as ligações bem definidas, para cada iteração do algoritmo *geographic* o número de mensagens (pacotes transmitidos) varia dependendo da quantidade de vizinhos existentes próximo ao alvo.

A matriz de pesos do algoritmo *geographic*, para o nó transmissor i e o nó receptor j no tempo t é modelada como:

$$\mathbf{W}(k) = \mathbf{I} - \frac{(e_i - e_j)(e_i - e_j)^T}{2} \tag{3.14}$$

onde e_i denota o i^th vetor coluna da matriz identidade $\mathbf{I}$ (BOYD et al., 2006). No algoritmo *geographic* a comunicação ocorre com qualquer $j \in \mathbf{V}$ através do roteamento (SILVANA,

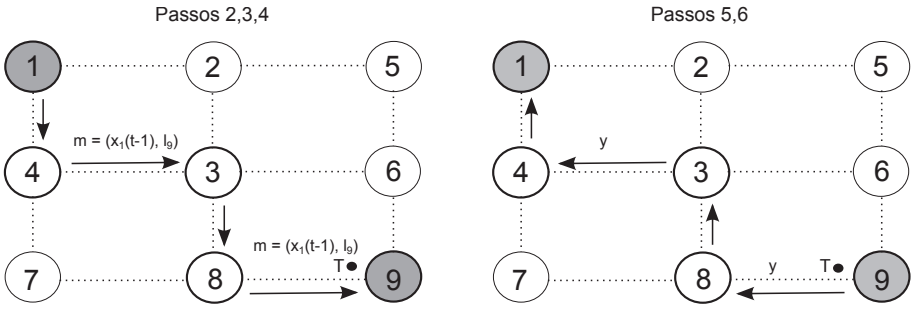


Figura 3.3: Execução do Algoritmo *Geographic*

2011).

3.2.3.1 Greedy Routing

O algoritmo de roteamento *greedy routing*, assume que o nó s tem uma mensagem m para transmitir para o seu vizinho que está mais próximo do alvo, onde s representa um nó intermediário entre o nó de origem i e o nó de destino j.

Algoritmo 5: *Greedy Routing* (BéNéZIT, 2009)

1: Inicializa $i \leftarrow s$
2: O nó i pesquisa seus vizinhos j que estão mais próximos da posição **T**.
3: **if** $j = i$ **then**
4: O nó i é o receptor final da mensagem m e o algoritmo de roteamento termina.
5: **else**
6: O nó i envia a mensagem m e a posição alvo **T** para o nó j.
7: $i \leftarrow j$ e vai para o passo 2.
8: **end if**

3.2.4 Algoritmo Path Averaging

Em relação ao algoritmo *path averaging* (BéNéZIT, 2009), toda a rota pode participar do cálculo da média, onde todas as estimativas dos nós na rota podem ser calculadas juntas sem custo extra na comunicação. Para melhorar o desempenho do algoritmo *path averaging* é utilizado o algoritmo de roteamento chamado *random greedy routing*, muito parecido ao algoritmo de roteamento utilizado pelo *geographic*. Nesse algoritmo, assume-se que o nó s tem uma mensagem m para transmitir, e uma posição alvo **T**. Assim, o nó s envia a mensagem m para o vizinho que foi escolhido aleatoriamente que está mais próximo do alvo. Uma vantagem do *random greedy routing* é que o nó não precisa manter o controle de seus vizinhos. Para evitar colisões nas rotas, inicialmente todos os nós

permanecem no estado *inativo*.

Algoritmo 6: *Path Averaging* (BéNéZIT, 2009)

1: **for** $t = 1 : t_{fim}$ **do**
2: No instante t, o nó s acorda e altera seu estado para *ativo*.
3: O nó s seta o alvo $\mathbf{T}$ de forma uniforme e aleatória.
4: Então o nó s envia a mensagem $m = (y, l_s, \mathbf{T}, C)$, onde l_s é a posição do nó s, C é um contador inicializado em $C = 1$, e y é a medida inicializada em $y = x_s(t)$.
5: $i \leftarrow s$
6: Considera-se $N_{vizinhos}$ como o conjunto de nós inativos que receberam a mensagem m e que estão mais próximos de $\mathbf{T}$ do que i. Então, qualquer nó em $N_{vizinhos}$ sorteia um tempo de *backoff*.
7: **if** $N_{vizinhos} \neq \emptyset$ **then**
8: Seja j o vizinho com o menor tempo de *backoff*. Quando o tempo expira, o nó j envia uma mensagem de notificação para o nó i e configura seu estado para *ativo*.
9: Os vizinhos que ouviram a mensagem de notificação retiram-se da iteração.
10: O nó i envia uma confirmação para o nó j. (Se o nó i recebe outra mensagem de notificação, ele confirma de forma negativa para que voltem ao estado *inativo*).
11: Quando o nó j recebe a confirmação, ele envia a mensagem $m = (y + x_j(t), l_j, \mathbf{T}, C + 1)$, e guarda a localização l_i de seu predecessor i.
12: $i \leftarrow j$ e vai para o passo 6.
13: **else**
14: Depois de um tempo sem mensagem de notificação, o nó i considera-se como o receptor final da mensagem m e o roteamento *random greedy routing* termina.
15: O nó i calcula $z = \frac{y}{C}$, atualiza sua estimativa para z, configura seu estado para *inativo*, e envia z de volta para o nó s pela mesma rota, o que é possível uma vez que todos os nós na rota memorizada são seus predecessores.
No caminho, todos os nós atualizam suas estimativas para z e configuram seus estados para *inativos*.
16: **end if**
17: **end for**

Em ambos os algoritmos *geographic* e *path averaging*, para que haja coordenação e acordo entre os nós diretamente conectados assim como os nós que estão distantes uns dos outros, a informação precisa ir e vir na rede. No passo 11 do Algoritmo 6, se o nó i não ouvir a mensagem m enviada pelo nó j, então o nó i reenvia uma mensagem de confirmação. Com o estado ativo, nenhum nó é envolvido em duas diferentes rotas ao mesmo tempo (BéNéZIT, 2009).

Dentro do cenário da rede, o nó i sorteia uma posição alvo de maneira aleatória e uniforme, ou seja, qualquer posição de maneira equiprovável dentro da rede. Através

do *random greedy routing*, tenta chegar até o nó que está mais próximo dessa posição. Todos os nós que pertencem à rota atualizam suas estimativas para a média. Os passos do algoritmo são representados através da Figura 3.4. Assumimos que cada nó i possui uma estimativa inicial $x_i(0)$, t_{fim} é o número de iterações, $\mathbf{T} = (x, y)$ é a posição alvo na rede e $\mathbf{m} = [y, l_s, \mathbf{T}, C]$ representa a mensagem. Do ponto de vista do grafo ideal (estático), com as ligações bem definidas, para cada iteração do algoritmo *path averaging* o número de mensagens (pacotes transmitidos) varia dependendo da quantidade de nós na rota.

O algoritmo de roteamento *random greedy routing*, assume que o nó s tem uma mensagem m e uma posição alvo $\mathbf{T}$ para transmitir, onde s representa um nó intermediário entre o nó de origem i e o nó de destino j.

3.2.4.1 Random Greedy Routing

Algoritmo 7: *Random Greedy Routing* (BéNéZIT, 2009)

1: Inicializa $i \leftarrow s$
2: O nó i transmite m e $\mathbf{T}$.
3: Seja $N_{vizinhos}$ o conjunto de nós que receberam a mensagem m e que estão mais perto de $\mathbf{T}$ do que i. Qualquer nó em $N_{vizinhos}$ inicia um tempo de *backoff*.
4: **if** $N_{vizinhos} \neq \emptyset$ **then**
5: Seja j o nó com menor tempo de *backoff*, então o nó j confirma para o nó i.
6: O nó i transmite a mensagem m e a posição alvo $\mathbf{T}$ para o nó j.
7: $i \leftarrow j$ e vai para o passo 2.
8: **else**
9: O nó i é o receptor final da mensagem m e o roteamento termina.
10: **end if**

Como as rotas são mais diversificadas no *random greedy routing*, a convergência do algoritmo é mais rápida.

Em ambos os algoritmos *pairwise* e *geographic*, $S(t)$ sempre tem um par de nós se comunicando, enquanto que para os algoritmos *neighborhood* e *path averaging*, $S(t)$ possui um número aleatório de nós, pois caminhos e vizinhanças inteiras são calculadas a cada iteração.

3.2.5 Algoritmo Broadcast

Em algoritmos *gossip*, se ocorrer um erro, este não impacta fortemente no resultado final, pois apenas um subconjunto de nós participam de uma rodada *gossip*. Em alguns cenários de redes, como por exemplo em RSSFs, podem ocorrer erros durante as transmissões e isso pode impactar nos algoritmos *gossip*, pois a média pode não ser atualizada

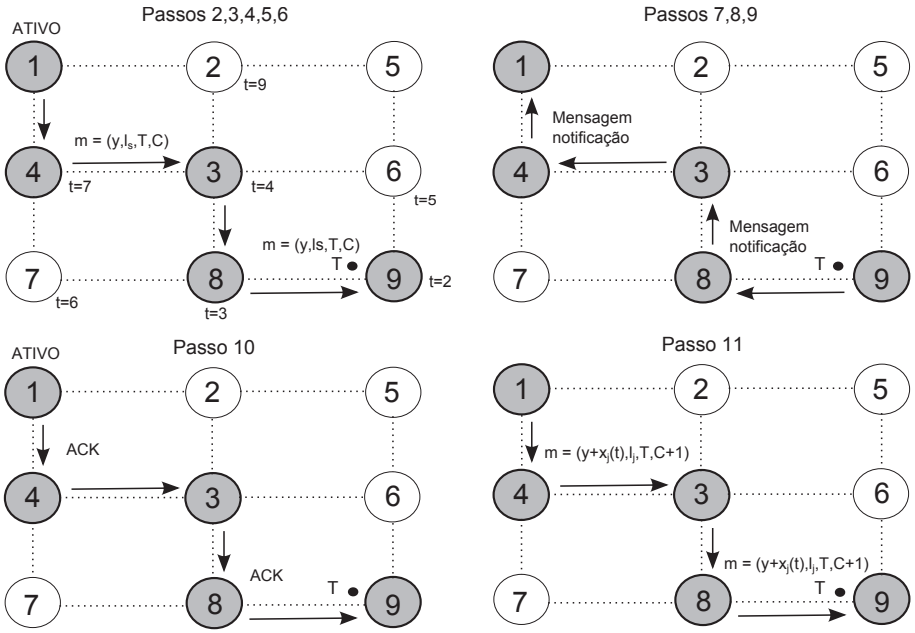


Figura 3.4: Execução do Algoritmo *Path Averaging*

de maneira correta em determinados nós.

Introduzido por Fagnani (FAGNANI; ZAMPIERI, 2006) e desenvolvido por Aysal (AYSAL; YILDIZ; SCAGLIONE, 2008), a ideia do *broadcast* é que o nó transmita seu estado para os vizinhos dentro do alcance de conectividade, utilizando comunicações em um único sentido, o que faz com que nenhum par de nós precise concordar com uma ação comum a qualquer momento.

Além disso, leva vantagem da natureza *broadcast* na comunicação sem fio, onde uma única transmissão é necessária para atualizar vários nós.

Algoritmo 8: *Broadcast* (AYSAL et al., 2008), (BéNéZIT, 2009)

1: **for** $t = 1 : t_{fim}$ **do**

2: O nó i ativo envia sua estimativa $x_i(t-1)$ para seus vizinhos j e qualquer nó que tenha recebido $x_i(t-1)$ atualiza sua estimativa para:

$$x_j(t) \leftarrow \beta x_j(t-1) + (1-\beta)x_i(t-1).$$

3: **end for**

Este algoritmo exige apenas um esforço de comunicação para atualizar vários nós em uma iteração. Seu principal desafio é otimizar o parâmetro β, de modo que os nós cheguem a um consenso sobre um valor perto o suficiente da média x_{ave}. O parâmetro β representado no Algoritmo 8 é um peso para definir a proporção da sua medida em relação à medida recebida no momento da da combinação (atualização) e pode ser expressado por: $0 < \beta < 1$. Quanto mais conectada a rede, mais o valor de β tende a 0, ou seja, para um $\beta = 0$ o consenso ocorre na primeira iteração (AYSAL et al., 2008). Conforme em (AYSAL

et al., 2008), o valor ideal de β a ser considerado é $\beta = 0.5$.

Seja $\mathbf{x}(t)$ o vetor de valores no final da iteração, então:

$$x(t+1) = \mathbf{W}(t)\mathbf{x}(t). \tag{3.15}$$

onde a matriz aleatória $\mathbf{W}(t)$, com probabilidade $\frac{1}{n}$ is:

$$\mathbf{W}_{ij}(k) = \begin{cases} 1, \ se \ \ i \notin N_q, \ j = i \\ \beta, \ se \ \ i \in N_q, \ j = i \\ 1 - \beta, \ se \ \ i \in N_q, \ j = q \\ 0, \ caso \ contrario. \end{cases} \tag{3.16}$$

onde $\mathbf{W}_{ij}(k)$ denota a matriz de pesos correspondente para o caso do nó i ativo.

Em um cenário de rede, o nó i envia sua estimativa $x_i(t)$ para seus vizinhos e todos os vizinhos que tenham recebido $x_i(t)$ atualizam para a média com base no parâmetro β.

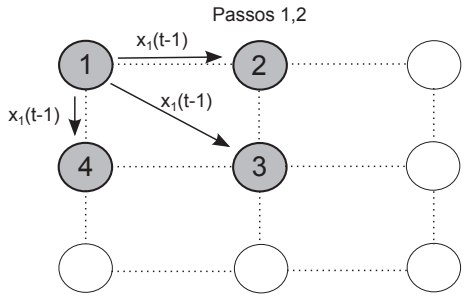


Figura 3.5: Execução do Algoritmo *Broadcast*

Através da Figura 3.5 é possível verificar a execução do algoritmo *broadcast*.

3.3 Análise de Complexidade dos Algoritmos

A seguir apresentamos um síntese dos estudos de complexidade desses algoritmos usando topologias representadas por grafo completo em *grade* e aleatórias. Foi demonstrado em (BOYD et al., 2006), que o algoritmo *pairwise* converge lentamente para a média de todos os valores armazenados. Em redes aleatórias com n nós, o *pairwise* demanda $O(n^2)$ mensagens, enquanto que em grafos completos em *grade* demanda somente $O(n \log n)$ mensagens. Devido a cada nó enviar pelo menos uma vez sua estimativa, o *pairwise* não converge em menos que $O(n)$ mensagens. No caso do algoritmo *geographic* proposto em (DIMAKIS; SARWATE; WAINWRIGHT, 2006), este usa informações geográficas dos nós para reduzir o tempo de convergência para $O(n^{1.5}\sqrt{\log(n)})$ mensagens em grafos aleatórios, o que é uma melhoria em relação ao *pairwise* (AYSAL et al., 2008). Outro algoritmo que possui a premissa de conhecer a localização dos nós é o *path averaging*, responsável por reduzir o tempo de convergência para $O(n \log n)$ mensagens em grafos

aleatórios toroidais. Contrariamente ao algoritmo *pairwise*, onde apenas um vizinho entre todos que receberam a estimativa do nó i envia sua estimativa, no *neighborhood* todos os vizinhos enviam suas estimativas para o nó i, tendo um custo esperado em um grafo completo de $O(n)$ mensagens.

Os valores referentes a complexidade desses algoritmos são definidos na Tabela 3.1.

Tabela 3.1: Complexidade dos Algoritmos *gossip*.

Algoritmo	Topologia Grafo	Média nós por iteração	Complexidade
Pairwise	Completo	2	$O(n \log n)$
Pairwise	Aleatório	2	$O(n^2)$
Neighborhood	Completo	n	$O(n)$
Geographic	Aleatório	2	$O(n^{1.5}\sqrt{\log n})$
Path Averaging	Aleatório Toroidal	$O(\sqrt{n/\log n})$	$O(n \log n)$

Observando a Tabela 3.1 podemos verificar a complexidade dos algoritmos *gossip*. A complexidade está relacionada ao número de mensagens que precisam ser enviadas a cada iteração do algoritmo.

3.4 Considerações Finais

Neste Capítulo foram apresentados os principais algoritmos de consenso médio síncronos e assíncronos. Diversos algoritmos distribuídos de consenso foram propostos na literatura, contudo os limites de desempenho destes algoritmos em redes sem fio ainda é um tema pouco explorado. Em relação ao funcionamento, alguns algoritmos de consenso como o *geographic* e o *path averaging* possuem premissas complexas, como conhecer a região da rede e por conseguinte as posições dos nós. Outros, como o *pairwise* e o *neighborhood* não necessitam conhecer toda a rede, sendo mais simples de implementar. Durante o capítulo exploramos os pseudocódigos dos algoritmos, exemplificando através de grafos os seus comportamentos. Contudo ainda existem algumas questões em relação ao modelo de canal que não foram tratadas. Esse trabalho propõe a análise dos algoritmos de consenso assíncronos descritos anteriormente em um modelo de canal mais realístico, através da probabilidade de *outage*.

Capítulo 4

Desempenho dos Algoritmos em RSSFs

4.1 Introdução

Nesse capítulo apresentamos a avaliação de desempenho dos algoritmos de consenso em RSSFs, usando o modelo de canal, baseado em probabilidade de *outage*. Os algoritmos de consenso *gossip* descritos no Capítulo 3 foram avaliados em termos de complexidade para cenários de rede idealizados, ou seja, onde o grafo é predefinido e nenhum tipo de variação na comunicação é considerada. No canal de comunicações sem fio, além do efeito do ruído e da atenuação, (propagação de larga escala), afinal ainda sofre o efeito de desvanecimento (propagação de pequena escala). O modelo de propagação de larga escala faz uma previsão da potência média recebida, contudo no canal sem fio existe uma variação da potência instantânea recebida, devido ao efeito da propagação de pequena escala, que é o desvanecimento.

Neste trabalho propomos avaliar o desempenho dos algoritmos de consenso assíncronos *gossip* usando um modelo de comportamento de canal mais aprimorado que é baseado em probabilidade de *outage* (GOLDSMITH, 2005). Esse modelo é adequado porque modela o canal de uma maneira mais realística levando em conta os efeitos da propagação de pequena escala, tendo sido empregado na avaliação de diferentes esquemas de transmissão digital (BRANTE; KAKITANI; SOUZA, 2011).

A capacidade do canal pode ser definida de duas formas, pela *capacidade de Shannon*, também conhecida como *capacidade ergódica* e pela *capacidade com outage* (GOLDSMITH, 2005). A *capacidade de Shannon* define a máxima taxa de dados que podem ser enviados através do canal com uma probabilidade de erro assintoticamente pequena. Neste modelo, a taxa transmitida através do canal é constante. A capacidade do canal com *outage* aplica-se em canais lentos, onde a relação sinal-ruído instântanea (SNR) - do

inglês: *Signal-to-Noise Ratio*, é constante ao longo de uma ou várias transmissões (um *burst* de transmissão) e, em seguida muda para um novo valor com base na distribuição de desvanecimento (GOLDSMITH, 2005). Uma vez que o transmissor não sabe o valor da SNR, ele precisa fixar uma taxa de transmissão independente da SNR. Especificamente, o transmissor fixa uma SNR_{min} e codifica para uma taxa de transmissão $R = B \cdot \log_2(1 + \mathrm{SNR}_{min})$, onde B representa a largura de banda do canal. Se os dados são corretamente recebidos, então a SNR é maior ou igual a SNR_{min}. Se a SNR está abaixo da SNR_{min}, então os *bits* recebidos ao longo do *burst* de transmissão não puderam ser decodificados corretamente e o receptor declara uma *outage*. Se uma *outage* ocorre, o pacote é considerado perdido. Portanto, a probabilidade de *outage* é definida como:

$$\mathcal{P}_{outage} = \Pr(\mathrm{SNR} < \mathrm{SNR}_{min}). \tag{4.1}$$

A taxa média recebida depois de muitas transmissões é:

$$R_m = (1 - \mathcal{P}_{outage}) B \log_2(1 + \mathrm{SNR}_{min}) \tag{4.2}$$

uma vez que os dados somente são recebidos corretamente em $1- \mathcal{P}_{outage}$ transmissões. O valor da SNR_{min} é determinada de acordo com a aplicação e estrutura do transmissor/receptor (SADEK; YU; LIU, 2009), podendo ser definida quando se considera a capacidade como o limite

$$SNR_{min} = 2^{\delta} - 1 \tag{4.3}$$

onde δ é a eficiência espectral utilizada. A premissa básica da capacidade com *outage* é que uma alta taxa de dados pode ser enviada e decodificada corretamente através do canal, exceto quando o canal está em desvanecimento profundo. A *outage* modela bem a FER de esquemas práticos de comunicação (HEDAYAT; NOSRATINIA, 2007).

4.2 Modelo de Outage

No canal sem fio quando há uma transmissão do nó i para o nó j, sob uma condição de forte desvanecimento, esta transmissão pode não ocorrer com sucesso. Em uma RSSF, onde há pouca mobilidade ou os nós estão estacionários, o efeito de desvanecimento pode durar por um longo período de tempo. O modelo de *outage* permite capturar esses efeitos do ponto de vista de capacidade de canal.

Seguindo a notação de (BRANTE; KAKITANI; SOUZA, 2011), nesta Seção apresenta-

mos o modelo matemático de *outage* assumindo a transmissão sem fio entre um nó fonte (i) e um nó de destino (j). De forma geral, o pacote recebido pelo nó j, após a transmissão do nó fonte i pode ser modelado como:

$$\mathbf{y}_{ij} = \sqrt{P_i\ \gamma_{ij}}\ h_{ij}\ \mathbf{x} + \mathbf{k}_j \tag{4.4}$$

onde P_i é a potência de transmissão da fonte, γ_{ij} representa a perda de percurso entre a fonte e o destino, h_{ij} representa o coeficiente de desvanecimento *Rayleigh* do enlace $i \rightarrow j$, x é a informação a ser transmitida e $\mathbf{k}_j$ representa o ruído no receptor j, modelado como um ruído aditivo Gaussiano branco (AWGN) - do inglês *Additive White Gaussian Noise*.

A perda de percurso entre o nó i e o nó j é dada por:

$$\gamma_{ij} = \frac{Gn\lambda^2}{(4\pi)^2\ d_{ij}^{\alpha}\ M_l\ N_f} \tag{4.5}$$

onde d_{ij} é a distância em metros entre os nós i e j, α é o expoente de perda de percurso, Gn é o ganho total das antenas de transmissão e recepção, $\lambda = \frac{3.10^8}{f_c}$ é o comprimento de onda, f_c é a frequência da portadora, M_l é a margem de enlace e N_f é a figura do ruído no receptor (GOLDSMITH, 2005).

Uma *outage* ocorre na transmissão direta quando $I_{ij} < R$, onde I_{ij} é a informação mútua no enlace $i \rightarrow j$ e R é a taxa de transmissão da informação. A SNR instantânea no enlace $i \rightarrow j$ é dada por

$$\mathrm{SNR}_{ij} = \frac{|h_{ij}|^2\ \gamma_{ij}\ P_i}{N} \tag{4.6}$$

onde $N = N_0\ B$ é a potência do ruído (Watts), N_0 é a densidade espectral de potência unilateral do ruído (Watts/Hz) e B é a largura de banda do sistema (Hertz).

Considerando entradas Gaussianas complexas e largura de banda unitária (B = 1 Hz), a informação mútua pode se escrita como

$$I_{ij} = \log_2\left(1 + \frac{P_i\ \gamma_{ij}\ |h_{ij}|^2}{N}\right) = \log_2\left(1 + \frac{R\ E_b\ \gamma_{ij}\ |h_{ij}|^2}{N_0}\right) \tag{4.7}$$

onde $P_i = R\ E_b$ e E_b é a energia por bit de informação e a capacidade é o máximo da informação mútua. A probabilidade de *outage* em desvanecimento *Rayleigh* no enlace $i \rightarrow i$ é dada por

$$\mathcal{P}_{outage} = \Pr\{I_{ij} < R\} = \Pr\left\{\frac{R\ E_b\ \gamma_{ij}\ |h_{ij}|^2}{N_0} < 2^{\delta} - 1\right\} = \Pr\left\{|h_{ij}|^2 < \frac{2^{\delta} - 1}{R\ \frac{E_b}{N_0}\ \gamma_{ij}}\right\} \tag{4.8}$$

$$= 1 - \exp\left[\frac{(1-2^{\delta})\,N_0}{R\,E_b\,\gamma_{ij}}\right] = 1 - \exp\left[\frac{(1-2^{\delta})\,N_0}{P_i\,\gamma_{ij}}\right] \tag{4.9}$$

onde $Pr\{\theta\}$ é a probabilidade do evento θ. Notamos que $\mathcal{P}_{outage} = \Pr\{I_{ij} < R\}$ depende da distribuição de $|h_{ij}|^2$. Quando h_{ij} é descrito segundo uma distribuição de *Rayleigh*, $|h_{ij}|^2$ tem uma distribuição exponencial.

Em nosso cenário de simulação dos algoritmos de consenso *gossip* assumimos que as transmissões do nó i e j ocorrem par a par, ou seja, assumimos a transmissão direta modelada pela probabilidade de *outage* sem retransmissões. Estamos considerando neste trabalho um canal com desvanecimento *Rayleigh*, embora o modelo possa ser estendido para outros modelos estatísticos de canal como *Nakagami* (GOLDSMITH, 2005).

4.3 Descrição do Cenário de Simulação

Para o cenário de simulação escolhemos a topologia de rede em *grade*, com dimensão $\sqrt{n} \times \sqrt{n}$, com n nós, conforme apresentado na Figura 4.1, onde o espaçamento de cada nó i para seu vizinho j, na horizontal e vertical, é definido pelo parâmetro d.

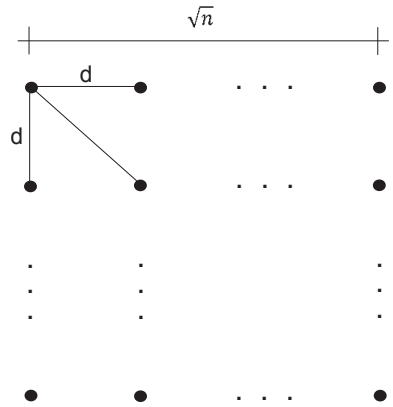


Figura 4.1: Topologia geral de uma rede em *grade*

Como a topologia em *grade* é geométrica, fica mais fácil avaliar o comportamento da dinâmica dos algoritmos de consenso para diferentes espaçamentos da *grade*, não havendo a necessidade de fazer muitas simulações para calcular a média das topologias como no caso da topologia aleatória (SILVANA, 2011). A implementação dos algoritmos de consenso, tanto no modelo convencional *sem outage* como no modelo *com outage*, foram desenvolvidos no *software* MATLAB® (MATHWORKS, 2013).

Nos referimos ao modelo convencional *sem outage* como sendo o cenário onde a RSSF é definida por um grafo estático. Neste cenário os algoritmos de consenso executam sobre um grafo de forma que as ligações são pré-definidas, onde assumimos que dois

nós estão sempre ligados e a transmissão é sempre com sucesso. Em relação ao modelo *com outage*, existe uma probabilidade de comunicação com sucesso de qualquer nó para qualquer nó da rede, podendo inclusive ser nula quando os nós estão muito distantes.

Avaliamos os dois cenários de simulação, *sem modelo de outage* e *com modelo de outage* ambos com 25 e 49 nós. Para cada cenário são realizadas 200 simulações onde em cada simulação são permitidas 400 iterações do algoritmo a ser avaliado. Os valores iniciais de medida de cada nó são estabelecidos no início da simulação. Para o cenário de simulação do modelo *sem outage* é construído uma *grade* com espaçamento normalizado entre os nós. É definido um *range* de transmissão que varia de $1, 2, 4$ a 10 e isso define o grafo de conectividade. Como exemplo, apresentamos na Figura 4.2 uma *grade* com *range* de transmissão igual a 1.

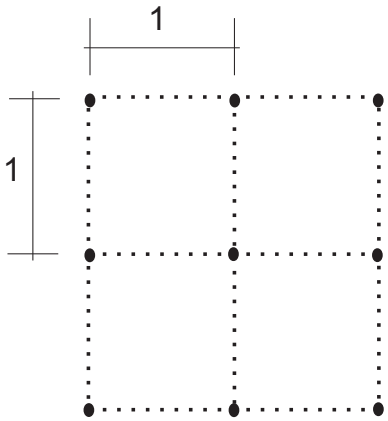


Figura 4.2: Conectividade dos nós na *grade*

No caso do cenário *com outage*, não há uma ligação fixa no grafo, e sim, há uma probabilidade do nó i alcançar qualquer outro nó, resultando assim em uma matriz de adjacência aleatória. Consideramos como estratégia de implementação uma única tentativa em cada evento de transmissão e deixamos cessar o processo de consenso daquela determinada rodada quando ocorre alguma *outage*. Para os cenários de simulação definimos o espaçamento da *grade* em $d = 50m$, $d = 100m$, $d = 150m$ e $d = 200m$. Esses valores foram escolhidos porque geram diferentes graus de conectividade para a RSSF quando utilizados os parâmetros de simulação detalhados na Tabela 4.1.

4.3.1 Métrica de Avaliação dos Algoritmos

Nesta subseção definimos a métrica de avaliação dos algoritmos de consenso *gossip*. Como é um consenso de média, assume-se que cada nó tem um valor de medida que é definido aleatoriamente dentro de uma faixa pré-definida de valores. Como a informação trocada entre os nós nos diferentes algoritmos de consenso não envolvem um volume

significativo de informação, podemos assumir um tamanho de pacote padrão que é capaz de ser utilizado em todos os algoritmos, contudo o tamanho do pacote fica embutido no modelo de capacidade.

Quando um determinado algoritmo está executando, as mensagens são trocadas entre os nós, e isso gera um consumo de transmissão em termos de energia para cada nó, onde o algoritmo que trocar menos pacotes terá um desempenho em termos de energia melhor do que outro de maneira geral. Para nossa avaliação de desempenho, adotamos as métricas de *número de pacotes transmitidos* em função do *erro médio normalizado*. Expressamos o *erro médio normalizado* por e, conforme definido em (AVRACHENKOV; CHAMIE; NEGLIA, 2011):

$$e(t) = \frac{\|\mathbf{x}(t) - x_{ave}\|_2}{\|\mathbf{x}(0) - x_{ave}\|_2} \tag{4.10}$$

onde $\mathbf{x}(t)$ representa um vetor contendo as estimativas correntes de todos os nós sensores, x_{ave} é a média real (teórica) das medidas dos nós, $\mathbf{x}(0)$ é a estimativa inicial de todos os nós e $\|\cdot\|_2$ denota a norma 2 do vetor.

Portanto, por exemplo, se um dado algoritmo transmitiu 50 pacotes para atingir o valor de média e chegar ao consenso e o outro 100 para atingir o mesmo valor em uma mesma topologia, o que gastou 50 convergiu primeiro, obtendo um melhor desempenho. Em nosso cenário de avaliação consideramos os parâmetros típicos de um nó sensor real conforme (CUI; GOLDSMITH; BAHAI, 2005) e eles são apresentados na Tabela 4.1.

Tabela 4.1: Parâmetros do Cenário *com outage*.

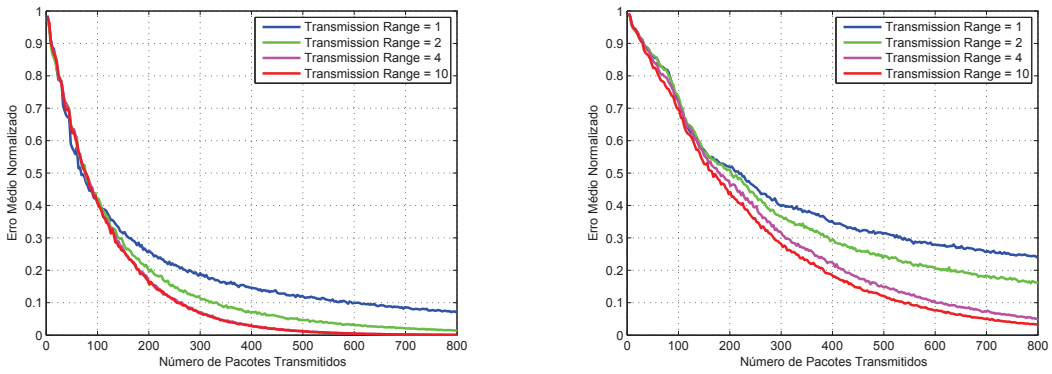
Parâmetro	Valor	Descrição
B	10 kHz	Largura de Banda
R	10 kbps	Taxa de Transmissão
$\delta = R/B$	1 bits/s/Hz	Eficiência Espectral
P_s	1mW	Potência de Transmissão
F	10 dB	Figura de Ruído
M	40 dB	Margem de Operação
N_0	$10^{(-204+F+M)/10}$	Densidade Espectral de Potência do Ruído
α	3	Expoente de Perda de Percurso
F_c	2.4GHz	Frequência da Portadora
$PL(d_0)$		Perda de Percurso em d_0
d_0	1m	Distância de Referência

4.4 Desempenho dos Algoritmos de Consenso sem modelo de outage

Nesta seção apresentamos a avaliação dos algoritmos de consenso *gossip*, *pairwise*, *neighborhood*, *geographic*, *path averaging* e *broadcast*, sobre o modelo convencional, conforme explicado na Seção 4.3. Definimos um alcance de transmissão (*Transmission Range*) que afeta o grau de conectividade no cenário convencional e varia de $1, 2, 4$ e 10.

4.4.1 Pairwise

O desempenho do algoritmo *pairwise* é representado nas Figuras 4.3a e 4.3b.



(a) (b)

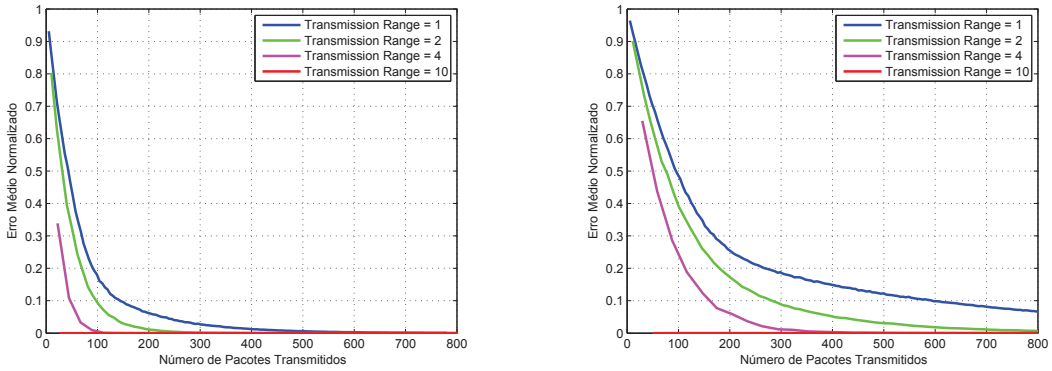
Figura 4.3: Algoritmo *Pairwise*: (a) 25 nós; (b) 49 nós;

Como era de se esperar a velocidade de convergência é afetada de maneira diretamente proporcional ao *transmission range*. Como nesse algoritmo as iterações são par a par, uma topologia com mais nós leva mais tempo para a convergência para a média ir se propagando ao longo da topologia, isso pode ser observado se comparado nas figuras com o mesmo *transmission range*. A convergência para a média real (teórica) é atingida para uma conectividade alta, em média com 600 pacotes transmitidos o que equivale a 200 iterações. No caso com mais nós, levaria aproximadamente 330 iterações, onde uma iteração no *pairwise* equivale a 3 transmissões. Quando há uma variação no número de nós da rede, como para o caso $n = 49$, o comportamento das figuras basicamente se mantém, mas a velocidade de convergência é mais lenta.

4.4.2 Neighborhood

As Figuras 4.4a e 4.4b apresentam o desempenho do método *neighborhood* na RSSF. Este algoritmo é bastante similiar ao *pairwise*, com a diferença que realiza o cálculo

das estimativas da vizinhança inteira a cada iteração.



(a) (b)

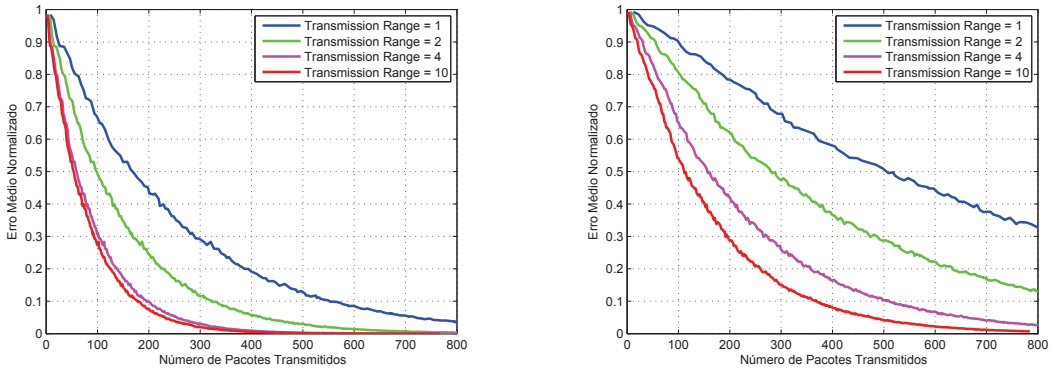
Figura 4.4: Algoritmo *Neighborhood*: (a) 25 nós; (b) 49 nós;

Para o cenário da Figura *a* e *b*, observamos que com um alcance maior (*transmission range* = 10), o consenso é atingido com poucas transmissões, visto que o nó escolhido alcança todos os outros nós na primeira iteração. Neste algoritmo, a velocidade de convergência também é diretamente afetada pelo *transmission range*. Em comparação com o *pairwise*, esse algoritmo se mostra bem mais eficiente, pois com a mesma quantidade de nós e o mesmo alcance do nó para seus vizinhos, por exemplo, 25 nós e *transmission range = 2*, o *neighborhood* necessita apenas 200 transmissões para convergir, enquanto que o *pairwise* aproximadamente 800.

4.4.3 Geographic

Observando as Figuras 4.5a e 4.5b podemos notar que a interpretação dos comportamentos das curvas se mantém, embora no *geographic* haja uma diferença maior entre os valores de *transmission range* quando comparado com o *pairwise*.

Por exemplo, para obter um *erro medio normalizado* de $0,1$, no *pairwise* com 49 nós e *transmission range* = 10, é necessário em média 550 tansmissões. Podemos observar que no *geographic* para a mesma quantidade de nós e o mesmo *transmission range*, há uma redução no número de transmissões. Para o cenário com 25 nós e um valor de *transmission range* = 4, em comparação com o algoritmo *pairwise*, podemos observar que o *geographic* consegue uma convergência mais rápida, estando de acordo com o que foi estudado em (DIMAKIS; SARWATE; WAINWRIGHT, 2006), (BéNéZIT, 2009). Contudo é importante observar que o *geographic* possui uma premissa complicada de implementação onde cada nó tem que conhecer a localização dos outros nós.

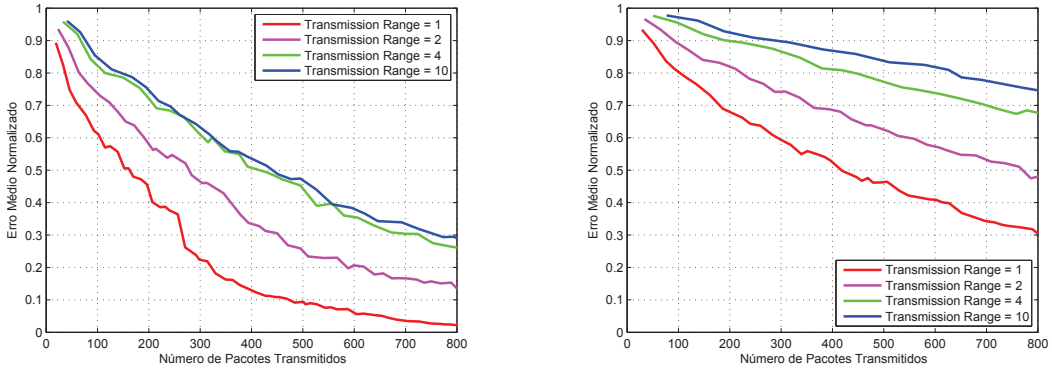


(a) (b)

Figura 4.5: Algoritmo *Geographic*: (a) 25 nós; (b) 49 nós;

4.4.4 Path Averaging

Similarmente ao *geographic*, o método *path averaging* também possui a premissa de conhecer a localização dos nós na rede. Seu desempenho pode ser observado nas Figuras 4.6a e 4.6b.



(a) (b)

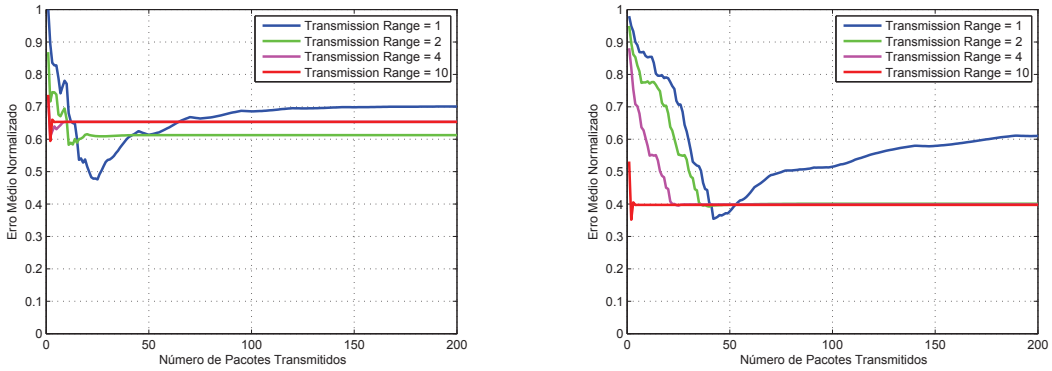
Figura 4.6: Algoritmo *Path Averaging*: (a) 25 nós; (b) 49 nós;

Analisando as Figuras *a* e *b*, podemos notar que a velocidade de convergência deste algoritmo é inversamente proporcional ao *transmission range*. Isto é explicado devido à necessidade do algoritmo de agregar valores ao longo da rota, fazendo com que possua velocidade de convergência lenta quando comparado aos demais métodos.

4.4.5 Broadcast

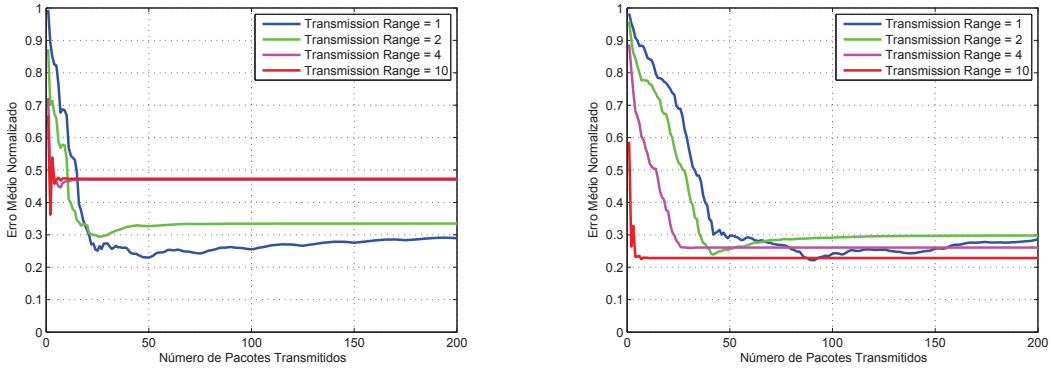
O desempenho do algoritmo *broadcast* é apresentado nas Figuras 4.7a e 4.7b. É um algoritmo bastante simples que não garante convengência para a média, por esta razão as curvas não decaem. Se uma rede é muito conectada, no primeiro momento as

estimativas são enviadas para todos os nós da rede de modo que todos atualizam. O mesmo comportamento acontece com o erro, uma vez que a rede está muito conectada, um único erro é propagado, permanecendo invariável. Já quando há pouca conectividade na rede, a medida que os nós transmitem suas estimativas propagam mais erros. Quando o *transmission range* = 1 , a informação leva mais tempo para se propagar na rede. Depois de uma quantidade de transmissões a curva permanece constante, não ocorrendo variações. Este algoritmo depende do parâmetro β, utilizado para definir a proporção da medida do nó em relação à medida recebida na hora da atualização. Pode variar de $0 < \beta < 1$. Para as Figuras 4.7a e 4.7b assumimos $\beta = 0.2$ e para as Figuras 4.8a e 4.8b um $\beta = 0.5$.



(a) (b)

Figura 4.7: Algoritmo *Broadcast*: (a) 25 nós; (b) 49 nós - $\beta = 0.2$;



(a) (b)

Figura 4.8: Algoritmo *Broadcast*: (a) 25 nós; (b) 49 nós - $\beta = 0.5$;

Notamos através das figuras que quanto maior o valor de β, mais a convergência tende a ir para um erro médio normalizado menor. Também é possível notar que quanto mais conectada a rede e menor o valor de β, mais rápida será a sua convergência. Para o cenário da Figura 4.7a, observamos que quanto menor o *transmission range*, ou seja,

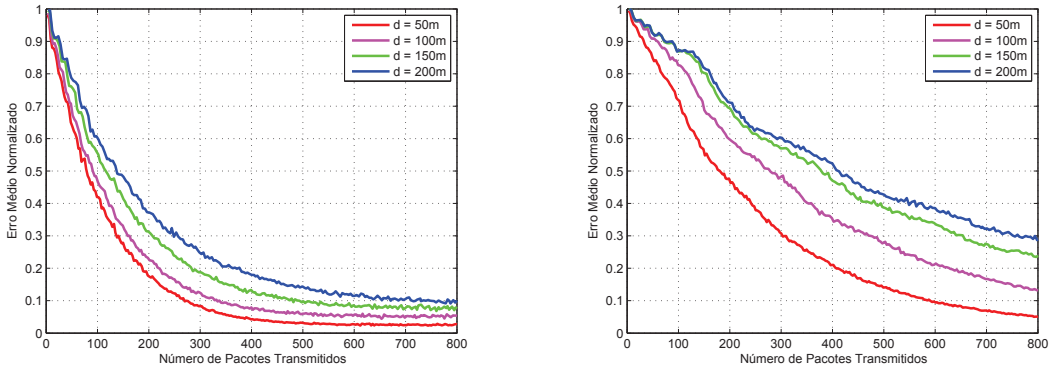
menor a conectividade da rede, mais transmissões serão necessárias para o algoritmo convergir.

4.5 Desempenho dos Algoritmos de Consenso com o modelo de outage

Nesta Seção apresentamos a avaliação dos algoritmos de consenso *gossip*, *pairwise*, *neighborhood*, *geographic*, *path averaging* e *broadcast*, sobre o *modelo de outage*, conforme explicado na Seção 4.3. Consideramos para o cenário uma largura de banda de $B = 10kHz$, taxa de transmissão $R = 10kbps$ e eficiência espectral $\delta = 1bits/s/Hz$.

4.5.1 Pairwise com Outage

É importante observar comparando as curvas não em valores mas em comportamento, que para o cenário *com outage*, estas passam a não convergir para o *erro zero*, devido ao fato de termos rodadas incompletas. Nesse caso, alguns nós atualizam para a média das estimativas e outros não.



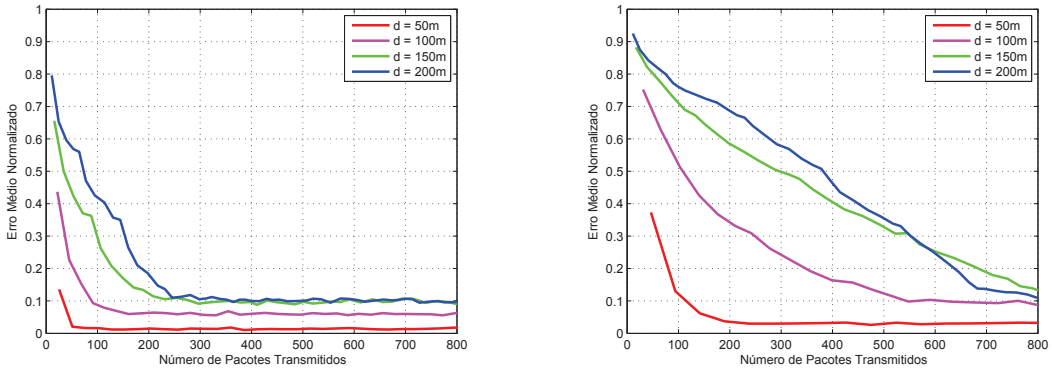
(a) (b)

Figura 4.9: Algoritmo *Pairwise com outage*: (a) 25 nós; (b) 49 nós;

Podemos observar nas Figuras 4.9a e 4.9b que a interpretação do comportamento das curvas se mantém mesmo quando há uma variação no número de nós da rede, como para o caso $n = 49$. Nota-se ainda, que a velocidade de convergência é inversamente proporcional ao espaçamento da *grade*, ou seja, quanto menor for o espaçamento, mais rápida será a convergência. Analisando a Figura *b*, e adotando um erro médio normalizado alvo de 0.3, o algoritmo necessita em média 300 transmissões.

4.5.2 Neighborhood com Outage

Através das Figuras 4.10a e 4.10b podemos observar o desempenho do algoritmo *neighborhood*.



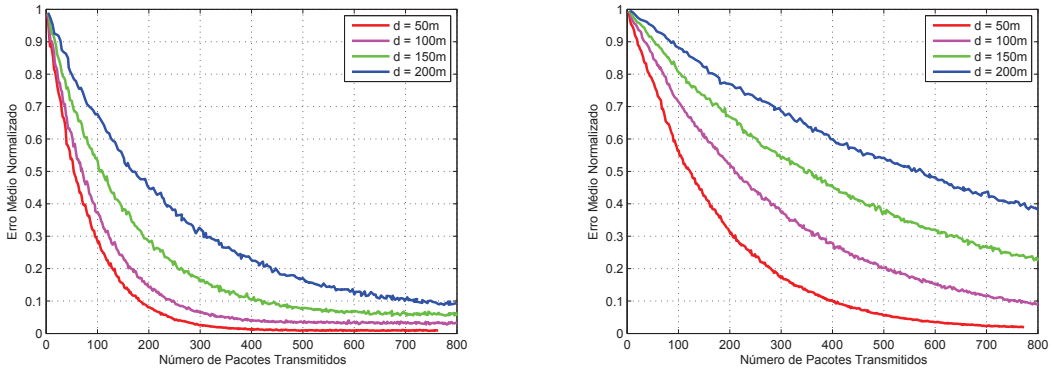
(a) (b)

Figura 4.10: Algoritmo *Neighborhood com outage*: (a) 25 nós; (b) 49 nós;

Observa-se que quanto menor for o espaçamento da *grade*, menor é o erro médio residual. Para o cenário da Figura a, podemos notar que o algoritmo necessita no pior caso em média de 300 transmissões para convergir, independentemente do espaçamento. Já para $d = 50m$, aproximadamente 50 transmissões, uma vez que o nó de origem atinge quase todos os nós nas primeiras iterações..

4.5.3 Geographic com Outage

O desempenho do algoritmo *geographic com outage* é observado através das Figuras 4.11a e 4.11b.



(a) (b)

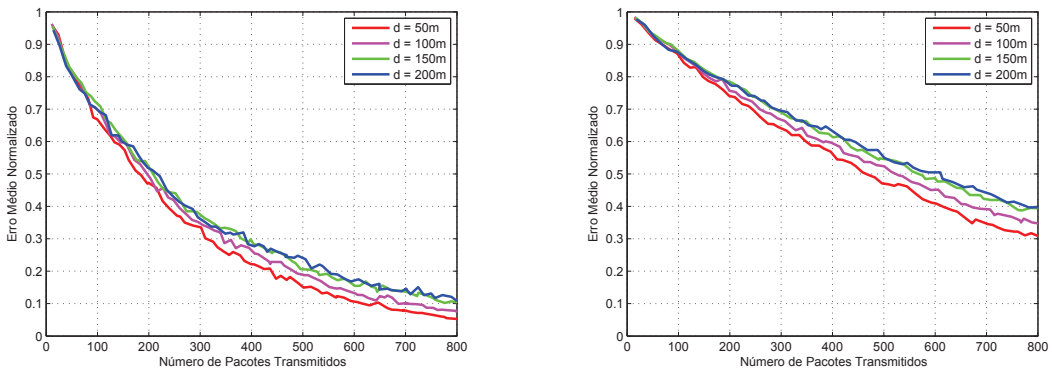
Figura 4.11: Algoritmo *Geographic com outage*: (a) 25 nós; (b) 49 nós;

Analisando a Figura b, para um erro médio normalizado alvo de 0.3 e $d = 50m$, o

algoritmo necessita de 200 transmissões. Já a velocidade de convergência para este cenário é atingida em média com 750 transmissões. Em comparação com o método *pairwise*, Figuras 4.9a e 4.9b, o *geographic* se mostra mais eficiente.

4.5.4 Path Averaging com Outage

Podemos visualizar nas Figuras 4.12a e 4.12b o desempenho do algoritmo *path averaging*. Verifica-se que para o cenário das Figuras *a* e *b*, a interpretação do comportamento das curvas se mantém e que o erro médio normalizado diminui com respeito para ambos os espaçamentos da *grade*. Uma vez que esse algoritmo necessita conhecer a localização dos nós na rede e que o cálculo da média se dá através de roteamento, mesmo com esse custo extra de complexidade, o algoritmo apresenta um bom desempenho quando comparado com o *geographic* que possui basicamente as mesmas premissas.



(a) (b)

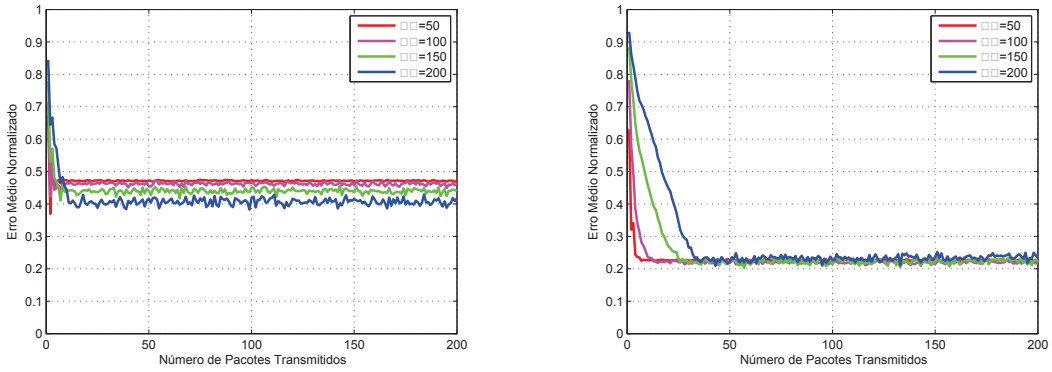
Figura 4.12: Algoritmo *Path Averaging com outage*: (a) 25 nós; (b) 49 nós;

Se observarmos a Figura *a* e considerarmos um erro médio normalizado alvo de 0.3 para um espaçamento igual a $d = 50$, o algoritmo necessitaria de aproximadamente 300 transmissões.

4.5.5 Broadcast com Outage

Último algoritmo analisado, o *broadcast* possui como característica convergir rapidamente para um erro não nulo à medida que a conectividade da rede aumenta.

Conforme comentando na Seção 3.2.5, este algoritmo utiliza-se do parâmetro β para a atualização da medida. É observado na Figura *b* que a medida que o algoritmo vai convergindo, independentemente do espaçamento da *grade*, há uma tendência em todas as curvas irem para o mesmo erro médio normalizado.



(a) (b)

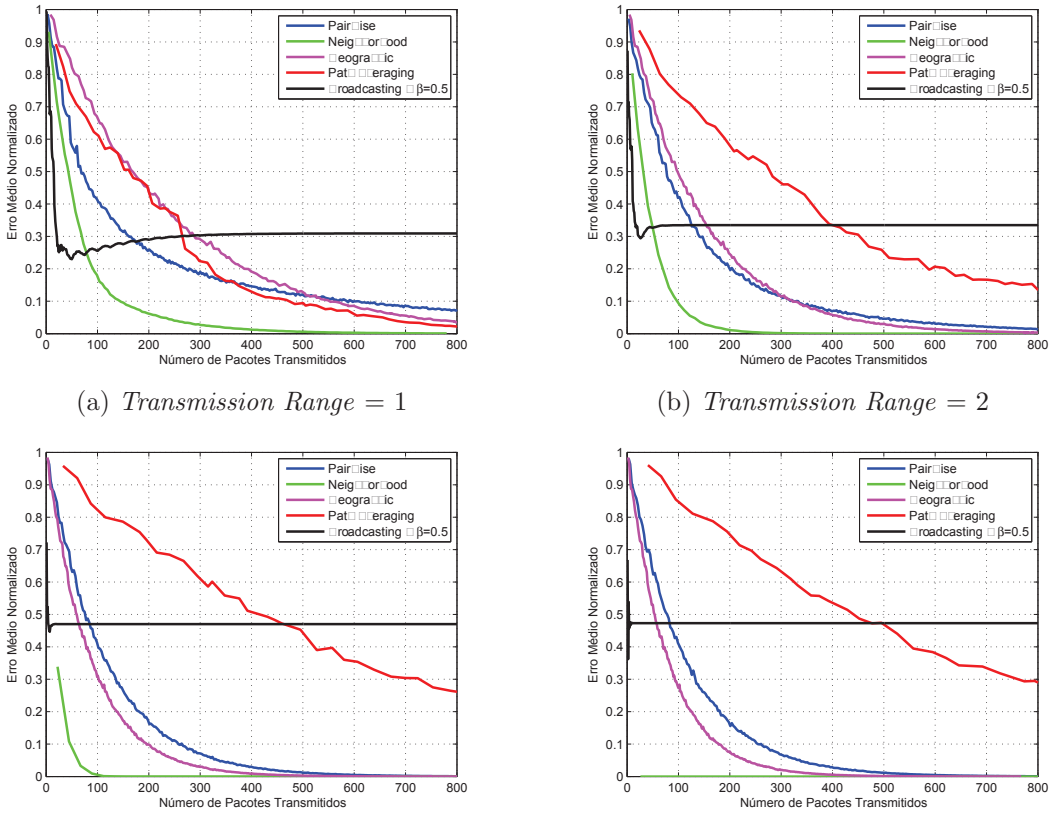
Figura 4.13: Algoritmo *Broadcast com outage* - $\beta = 0.5$: (a) 25 nós; (b) 49 nós;

4.6 Comparação dos Resultados

Nesta seção apresentamos os resultados comparativos dos algoritmos de consenso *gossip*. Primeiramente realizamos a comparação desses algoritmos sobre um cenário idealizado, com ligações pré-definidas em um grafo estático, conforme definido na Seção 4.3. Para o cenário da Figura 4.14a, observamos que o algoritmo *neighborhood* apresenta o melhor desempenho por ter uma convergência mais rápida para a média que os outros algoritmos.

Temos o *pairwise* como sendo o segundo melhor método, se considerarmos um erro médio normalizado alvo de aproximadamente 0.18. Para um erro médio normalizado alvo abaixo de 0.1, em geral podemos observar que os métodos *geographic* e *path averaging* estariam superior ao *pairwise*. Se o objetivo é atingir um erro médio normalizado muito próximo a 0, o algoritmo *neighborhood* tem a menor velocidade de convergência. Inicialmente podemos observar que o *broadcast* converge para um erro médio normalizado não nulo de 0.3, devido à própria natureza desse algoritmo, pois suas trocas são sempre unidirecionais. Contudo, se eventualmente a aplicação tolerar um determinado erro em relação à média, ele pode ser um algoritmo interessante.

Observando a Figura 4.14b, podemos notar que o comportamento do método *broadcast* não varia tanto em relação a Figura *a*, devido a ele atingir nas transmissões um percentual maior de nós na rede. O *neighborhood* continua sendo a melhor estratégia, assim como no cenário da Figura *a*, contudo podemos observar uma degradação do método *path averaging*. Essa degradação se dá não pelo próprio algoritmo, mas sim pela conectividade ser maior, o que faz com que os outros algoritmos atinjam um maior número de nós na rede. Para um erro médio normalizado alvo inferior a 0.1, o *geographic* acaba superando o método *pairwise*.



(a) *Transmission Range* = 1 (b) *Transmission Range* = 2

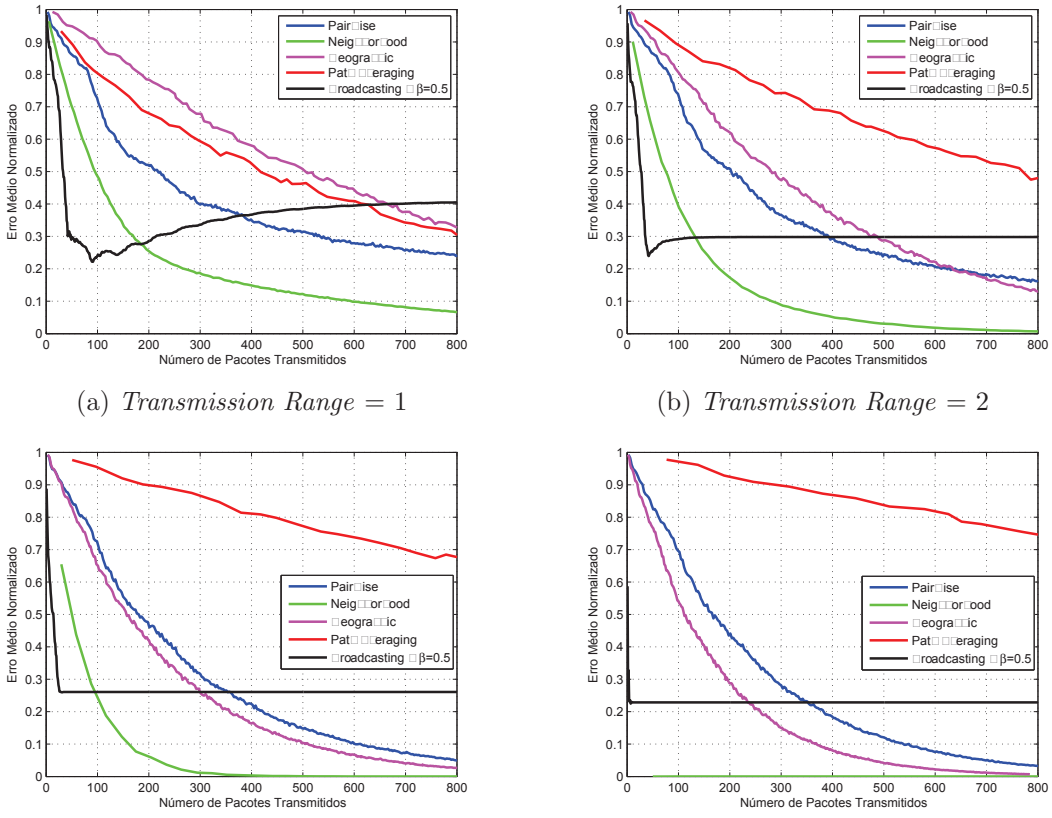
(c) *Transmission Range* = 4 (d) *Transmission Range* = 10

Figura 4.14: Cenário convencional para 25 nós

A interpretação do comportamento das curvas se mantém na Figura 4.14c, contudo, com a diferença de que para essa conectividade o método *geographic* se mostra mais eficiente que o *pairwise*, sendo a segunda melhor estratégia. Com um maior alcance na rede, o algoritmo *broadcast* necessita de poucas transmissões para todos os nós irem para o erro médio normalizado não nulo. Já em relação ao cenário da Figura 4.14d, embora o comportamento se mantenha dos outros métodos, notamos que o algoritmo *neighborhood* necessita de apenas uma única iteração para atingir todos os nós, convergindo rapidamente para a média dos valores iniciais.

É importante observar que devido a natureza de iteração de cada algoritmo, envolveu diferentes números de pacotes, alguns médodos começam com número inicial maior de pacotes transmitidos até ter uma atualização da média.

Para o cenário da Figura 4.15a, observamos que o *neighborhood* continua sendo a melhor estratégia e que o método *path averaging* se beneficia sobre o *geographic* devida a baixa conectividade da rede (*Transmission Range = 1*). Já para o cenário das Figuras *b*, *c* e *d*, podemos observar que como a conectividade aumenta, o método *geographic* apresenta um ganho significativo em relação ao *path averaging*. Em comparação, para



(a) *Transmission Range* = 1

(b) *Transmission Range* = 2

(c) *Transmission Range* = 4

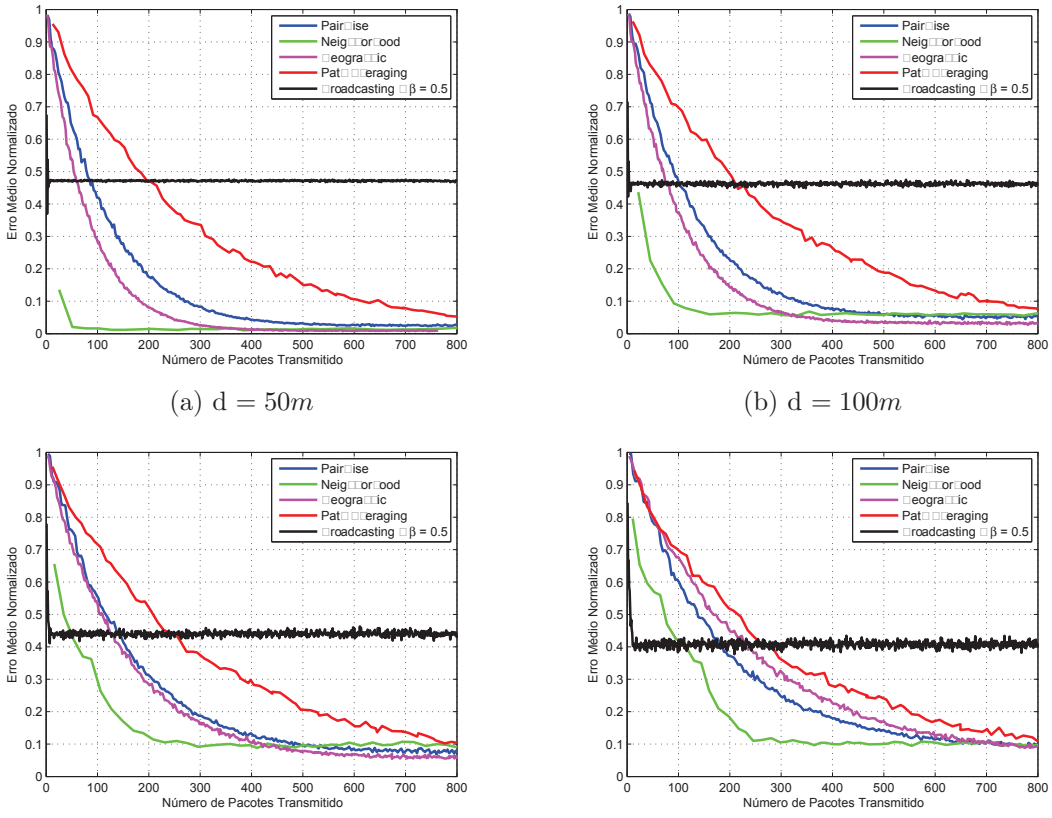
(d) *Transmission Range* = 10

Figura 4.15: Cenário convencional para 49 nós

baixa conectividade, Figuras *a* e *b*, podemos observar que o *pairwise* é uma boa opção, já para conectividades maiores, *c* e *d*, notamos que o melhor método é o *neighborhood*. Contudo, podemos verificar que o *geographic* e o *pairwise* apresentam um desempenho bem próximo.

Apresentamos a seguir a os resultados comparativos dos algoritmos de consenso *gossip* para o cenário *com outage*.

Observando as Figuras 4.16a, *b*, *c* e *d*, notamos que o método *neighborhood* apresenta o melhor desempenho entre os algoritmos simulados e que a medida que o espaçamento da *grade* aumenta, seu erro médio normalizado aumenta. Outra situação verificada é que quanto menor o espaçamento da *grade*, melhor é o desempenho do método *geographic* quando comparado ao *pairwise*. Já, à medida que o espaçamento da *grade* aumenta, $d = 100$, $d = 150m$, $d = 200m$, o algoritmo *pairwise* se mostra como uma estratégia mais eficiente. Para um espaçamento igual a $d = 200m$, o método *path averaging* apresenta uma velocidade de convergência razoável em comparação com o *pairwise* e *geographic*. É importante observar que para o cenário *com outage*, os algoritmos não convergem para erro médio normalizado igual a 0, uma vez que há probabilidade de ocorrer falha na trans-



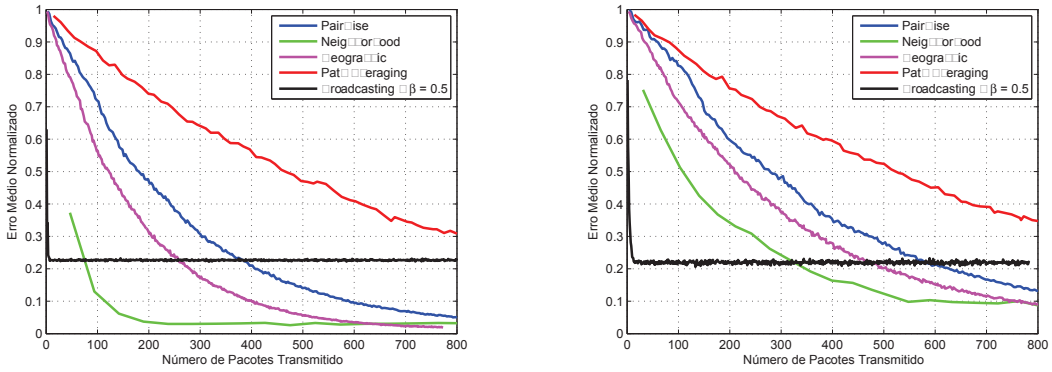
(a) d = 50m (b) d = 100m

(c) d = 150m (d) d = 200m

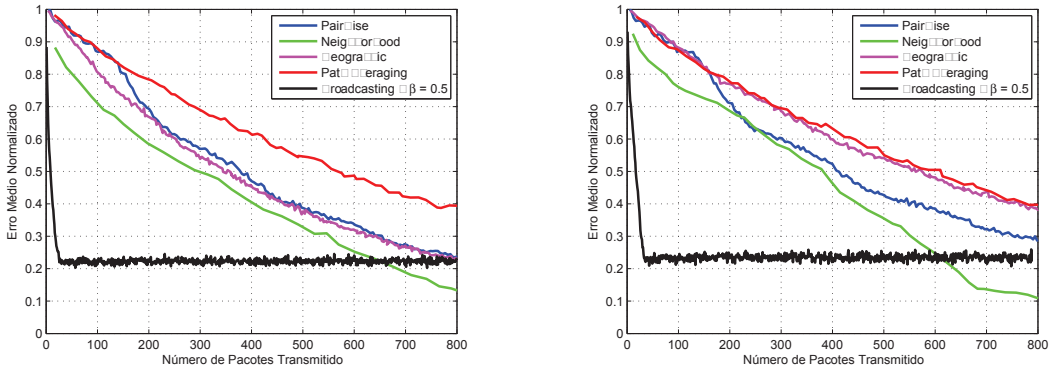
Figura 4.16: Cenário com *outage* para 25 nós

missão. Notamos também, que o erro residual é inversamente proporcional à conectividade da rede, ou seja, aumenta a medida que à conectividade diminui.

Em relação ao cenário da Figura 4.17a, verificamos que a interpretação do comportamento das curvas se mantém, embora os algoritmos levem mais tempo para a convergência pois a rede é maior. Neste cenário, observando a Figura 4.17d notamos que em um determinado momento, aproximadamente entre o erro de 0.6 e 0.7, pela primeira vez o método *pairwise* obtém um melhor desempenho em relação ao *neighborhood*, contudo, com o aumento das transmissões o *neighborhood* acaba recuperando sua posição de melhor estratégia. É importante observar que para o cenário com espaçamento igual a $d = 150m$, o *geographic* apresenta melhor velocidade de convergência quando comparado ao método *pairwise*. Já para um cenário com espaçamento maior da *grade*, $d = 200m$, o comportamento do *path averaging* praticamente se iguala ao *geographic*.



(a) d = 50m (b) d = 100m



(c) d = 150m (d) d = 200m

Figura 4.17: Cenário com *outage* para 49 nós

4.7 Considerações Finais

Neste Capítulo foram apresentados os resultados dos principais algoritmos de consenso (*gossip*). Observamos através das figuras que o comportamento dos algoritmos variam em função da quantidade de nós e de iterações. No cenário *com modelo de outage* notamos que os algoritmos não convergem para um erro médio normalizado igual a 0, uma vez que podem existir transmissões incompletas. Todos os algoritmos foram avaliados sobre os mesmos parâmetros de configurações dos nós sensores.

Capítulo 5

Conclusão

Neste trabalho investigamos o *problema do consenso*, avaliando diferentes algoritmos propostos na literatura para o caso de um modelo de canal de rádio baseado em *probabilidade de outage*. O modelo de canal baseado em *probabilidade de outage* permite modelar o comportamento do canal de rádio levando em conta os efeitos de propagação de pequena e larga escala. Por ser um modelo baseado em capacidade de canal, a *probabilidade de outage* permite uma análise teórica de desempenho que pode ser atingida por esquemas práticos.

Os algoritmos de consenso assíncronos (*gossip*) que foram comparados nesse trabalho são: *pairwise*, *neighborhood*, *geographic*, *path averaging* e *broadcast*. Todos esses algoritmos foram simulados em um mesmo cenário de rede, com topologia em *grade*, utilizando os mesmos parâmetros de configuração dos nós sensores. Como a topologia em *grade* é geométrica, fica mais fácil avaliar o comportamento da dinâmica dos algoritmos de consenso para diferentes espaçamentos da *grade*, não havendo a necessidade de fazer muitas simulações para calcular a média das topologias como no caso da topologia aleatória. A métrica de comparação dos algoritmos foi a quantidade de pacotes transmitidos, pois na prática o consumo de energia dos nós é afetado diretamente pela quantidade de pacotes transmitidos pela execução dos algoritmos de consenso.

Na primeira fase de análise, avaliamos o comportamento dos algoritmos de consenso sem o efeito do canal de rádio em topologia *sem modelo de outage*, em um grafo pré-definido, sem falhas nas transmissões. Já na segunda fase de análise, investigamos os efeitos do canal de rádio sobre os algoritmos de consenso em topologia *com modelo de outage*. Através da análise dos resultados, constatamos que a melhor estratégia a ser utilizada para o canal sem fio no cenário proposto com topologia em *grade* é o método *neighborhood*.

Embora o algoritmo *neighborhood* tenha se destacado neste cenário, é importante

lembrar que outros algoritmos, como o *geographic*, *pairwise* e *path averaging* também se mostraram eficientes. Vale lembrar, que os métodos *geographic* e o *path averaging* possuem premissas fortes como conhecer a localização dos nós. Embora em algumas literatura, o *geographic* e o *path averaging* sejam considerados os melhores para determinados cenários, como por exemplo topologias aleatórias, foi constatado por meio deste trabalho que o *neighborhood* obteve o melhor desempenho para este tipo de cenário proposto. Já em situações de aplicações onde fosse possível tolerar um erro medio não nulo, o algoritmo *broadcast* seria considerado interessante.

5.1 Trabalhos Futuros

A partir dos resultados obtidos no Capítulo 4, identificamos alguns estudos adicionais que podem ser feitos em relação ao comportamento destes algoritmos no canal sem fio.

- Investigar outro algoritmo de consenso assíncrono *gossip*, conhecido como *one-way path averaging* e analisar seu desempenho no cenário proposto;
- Investigar o comportamento dos algoritmos de consenso para outras topologias, como por exemplo as aleatórias;
- Propor algoritmos híbridos que utilizem em conjunto diferentes estratégias de modo que o erro médio residual siga um limiar próximo a 0;
- Investigar o desempenho dos algoritmos usando *probabilidade de outage* para outros modelos de canal, como por exemplo canais com desvanecimento *Nakagami*;
- Incluir os efeitos da camada de múltiplo acesso (MAC) na avaliação dos desempenhos dos algoritmos.

Referências Bibliográficas

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

ANDO, H. et al. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, v. 15, p. 818 – 828, 1999.

AVRACHENKOV, K.; CHAMIE, M. E.; NEGLIA, G. A local average consensus algorithm for wireless sensor networks. *2011 International Conference on Distributed Computing in Sensor Systems and Workshops*, p. 1 – 6, 2011.

AYSAL, T. et al. Broadcast gossip algorithms: Design and analysis for consensus. *47th IEEE Conference on Decision and Control*, p. 4843 – 4848, 2008.

AYSAL, T.; YILDIZ, M.; SCAGLIONE, A. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, p. 343 – 347, 2008.

BARABASI, A. L.; ALBERT, R. Emergence of scaling in random networks. *Science*, v. 286, p. 509 – 512, 1999.

BARBAROSSA, S.; SCUTARI, G.; SWAMI, A. Achieving consensus in self-organizing wireless sensor networks: The impact of network topology on energy consumption. *IEEE International Conference on Acoustics, Speech and Signal Processing*, v. 2, p. 841 – 844, 2007.

BENDER, J. G. An overview of systems studies of automated highway systems. *IEEE Transactions on Vehicular Technology*, v. 40, p. 82 – 99, 1991.

BéNéZIT, F. *Distributed Average Consensus for Wireless Sensor Networks*. Tese (Doutorado) — École Polytechnique Fédérale de Lausanne, 2009.

BORKAR, V.; VARAIYA, P. Asymptotic agreement in distributed estimation. *IEEE Transactions on Automatic Control*, v. 27, p. 650 – 655, 1982.

BOYD, S. et al. Gossip algorithms: Design, analysis and applications. *Proceedings IEEE Infocomm*, p. 1653 – 1664, 2005.

BOYD, S. et al. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, v. 14, p. 2508 – 2530, 2006.

BRACA, P.; MARANO, S.; MATTA, V. Running consensus in wireless sensor networks. *11th International Conference on Information Fusion*, p. 1 – 6, 2008.

BRANTE, G.; KAKITANI, M. T.; SOUZA, R. D. Energy efficiency analysis of some cooperative and non-cooperative transmission schemes in wireless sensor networks. *IEEE Transactions on Communications*, v. 59, p. 2671 – 2677, 2011.

BUZOGANY, L. E.; PACHTER, M.; D'AZZO, J. J. Automated control of aircraft in formation flight. *In Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, p. 1349 – 1370, 1993.

CHEN, L.; FROLIK, J. Self-aware distributed consensus building for sensor networks. *ISRN Communications and Networking*, v. 2011, 2011.

CUI, S.; GOLDSMITH, A.; BAHAI, A. Energy-constrained modulation optimization. *IEEE Transactions on Wireless Communications*, v. 4, p. 2349 – 2360, 2005.

CYBENKO, G. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, v. 7, p. 279 – 301, 1989.

DENANTES, P. et al. Which distributed averaging algorithm should i choose for my sensor network? *The 27th Conference on Computer Communications - IEEE INFOCOM 2008*, p. 986 – 994, 2008.

DIMAKIS, A.; SARWATE, A.; WAINWRIGHT, M. Geographic gossip: efficient aggregation for sensor networks. *The Fifth International Conference on Information Processing in Sensor Networks*, p. 69 – 76, 2006.

EREN, T.; BELHUMEUR, P.; MORSE, A. Closing ranks in vehicle formations based on rigidity. *41st IEEE Conference on Decision and Control*, v. 3, p. 2959 – 2964, 2002.

ERMENTROUT, G. B.; KOPELL, N. Frequency plateaus in a chain of weakly coupled oscillator. *SIAM Journal on Mathematical Analysis*, v. 15, p. 215 – 237, 1984.

FAGNANI, F.; ZAMPIERI, S. Average consensus with packet drop communication. *45th IEEE Conference on Decision and Control*, p. 1007 – 1012, 2006.

FAX, J.; MURRAY, R. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, v. 49, p. 1465 – 1476, 2004.

FIEDLER, M. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, v. 23, p. 298 – 305, 1973.

GHOSH, A.; BOYD, S. Growing well-connected graphs. *45th IEEE Conference on Decision and Control*, p. 6605 – 6611, 2006.

GIRIDHAR, A.; KUMAR, P. Computing and communicating functions over sensor networks. *IEEE Journal on Selected Areas in Communications*, v. 23, p. 755 – 764, 2005.

GOLDSMITH, A. *Wireless Communications*. [S.l.]: Cambridge University Press, 2005.

HEDAYAT, A.; NOSRATINIA, A. Outage and diversity of linear receivers in flat-fading mimo channels. *IEEE Transactions on Signal Processing*, v. 55, p. 5868 – 5873, 2007. ISSN 1053-587X.

JADBABAIE, A.; LIN, J.; MORSE, A. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, v. 48, p. 988 – 1001, 2003.

KAR, S.; MOURA, J. Topology for global average consensus. *Fortieth Asilomar Conference on Signals, Systems and Computers*, p. 276 – 280, 2006.

KAZI, C. R. A survey on sensor network. *International Journal of Computer and Informational Technology*, v. 01, 2010.

KURAMOTO, Y. *Chemical Oscillators, Waves, and Turbulance*. [S.l.]: Springer-Verlag, 1984.

LIN, J.; MORSE, A.; ANDERSON, B. The multi-agent rendezvous problem. *42nd IEEE Conference on Decision and Control*, v. 2, p. 1508 – 1513, 2003.

MATHWORKS. Disponível em: http://www.mathworks.com. *Acesso em 01/2013*, 2013.

NAZER, B.; DIMAKIS, A.; GASTPAR, M. Neighborhood gossip: Concurrent averaging through local interference. *IEEE International Conference on Acoustics, Speech and Signal Processing*, p. 3657 – 3660, 2009.

NAZER, B.; DIMAKIS, A.; GASTPAR, M. Local interference can accelerate gossip algorithms. *IEEE Journal of Selected Topics in Signal Processing*, v. 5, p. 876 – 887, 2011.

OLFATI-SABER, R.; FAX, J.; MURRAY, R. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, v. 95, p. 215 – 233, 2007.

PENROSE, M. *Random Geometric Graphs*. [S.l.]: Oxford University Press, 2003.

PEREIRA, S.; PAGES-ZAMORA, A. Distributed consensus in wireless sensor networks with quantized information exchange. *IEEE 9th Workshop on Signal Processing Advances in Wireless Communications*, p. 241 – 245, 2008.

PEREIRA, S.; PAGES-ZAMORA, A. Randomized transmission power for accelerated consensus in asymmetric wsns. *3rd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, p. 348 – 351, 2009.

REN, W.; BEARD, R.; ATKINS, E. A survey of consensus problems in multi-agent coordination. *Proceedings of the 2005 American Control Conference*, v. 3, p. 1859 – 1864, 2005.

SADEK, A. K.; YU, K.; LIU, K. J. R. On the energy efficiency of cooperative communications in wireless sensor networks. *ACM Transactions on Sensor Networks*, v. 6, p. 1 – 21, 2009.

SARDELLITTI, S.; BARBAROSSA, S.; SWAMI, A. Average consensus with minimum energy consumption: Optimal topology and power allocation. *18th European Signal Processing Conference*, 2010.

SILVANA, S. P. *Distributed Consensus Algorithms for Wireless Sensor Networks - Convergence Analysis and Optimization*. Tese (Doutorado) — Universitat Politècnica de Catalunya, 2011.

TSITSIKLIS, J.; BERTSEKAS, D.; ATHANS, M. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, v. 31, p. 803 – 812, 1986.

TSITSIKLIS, J. N. *Problems in decentralized decision making and computation*. Tese (Doutorado) — Institute of Technology Massachusetts, 1984.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of small-world networks. *International Weekly Journal of Science*, v. 393, p. 440 – 442, 1998.

XIAO, L. A scheme for robust distributed sensor fusion based on average consensus. *Proceedings of the International Conference on Information Processing in Sensor Networks*, p. 63 – 70, 2005.

XIAO, L.; BOYD, S. Fast linear iterations for distributed averaging. *Systems and Control Letters*, v. 53, p. 65 – 78, 2003.

YEH, H.-H.; NELSON, E.; SPARKS, A. Nonlinear tracking control for satellite formations. *39th IEEE Conference on Decision and Control*, p. 328 – 333, 2000.