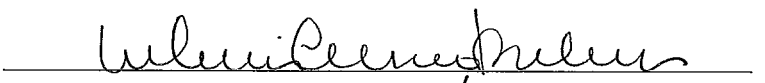
UM NOVO ALGORITMO PARA SINCRONIZAÇÃO COM GRADIENTE DOS RELÓGIOS DE UMA REDE DE SENSORES

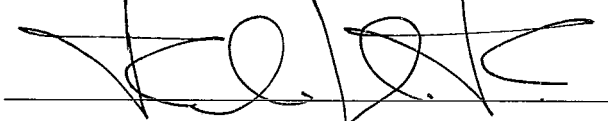
Rodolfo de Mello Pussente

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMS E COMPUTAÇÃO

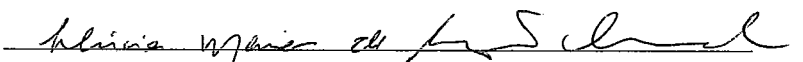
Aprovada por:



Prof. Valmir Carneiro Barbosa, Ph.D.



Prof. Felipe Maia Galvão França, Ph.D.



Profª. Lúcia Maria de Assumpção Drummond, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2008

PUSSENTE, RODOLFO DE MELLO

Um novo algoritmo para sincronização com gradiente dos relógios de uma rede de sensores [Rio de Janeiro] 2008

VII, 51 p. 29,7 cm (COPPE/UFRJ M.Sc. Engenharia de Sistemas e Computação, 2008)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1. Sincronização de relógios 2. Propriedade gradiente em sincronização de relógios

3. Redes de sensores

I. COPPE/UFRJ II. Título (série)

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM NOVO ALGORITMO PARA SINCRONIZAÇÃO COM GRADIENTE DOS RELÓGIOS DE UMA REDE DE SENSORES

Rodolfo de Mello Pussente

Junho/2008

Orientador: Valmir Carneiro Barbosa

Programa: Engenharia de Sistemas e Computação

Neste trabalho apresentamos um novo algoritmo para sincronização de relógios em redes de sensores. O algoritmo possui a propriedade gradiente apresentada em [1], mas o limite inferior de [1,2] que proíbe diferenças constantes entre vizinhos não é valido, pois o algoritmo não garante um limite inferior constante para a taxa dos relógios lógicos. Obtemos uma diferença de pior caso entre relógios lógicos de nós vizinhos de $O(1)$ assumindo que os nós conhecem os seus vizinhos e um limite superior para o diâmetro da rede. O algoritmo utiliza a comunicação já existente para a troca dos dados de sincronização, não aumentando assim a quantidade de mensagens existentes.

Abstract of Dissertation presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A NEW ALGORITHM FOR CLOCK SYNCHRONIZATION WITH THE GRADIENT PROPERTY IN SENSOR NETWORKS

Rodolfo de Mello Pussente

June/2008

Advisor: Valmir Carneiro Barbosa

Department: Systems Engineering and Computer Science

In this work we introduce a new algorithm for clock synchronization in sensor networks. The algorithm has the gradient property presented in [1], but the lower bound of [1,2] that forbids constant skew between neighbors does not apply. This happens because our algorithm does not guarantee constant lower bound for the rate of logical clocks. We achieve a worst-case clock skew between neighbors of $O(1)$ by assuming that nodes know their neighbors and an upper bound for the network's diameter. The algorithm uses already existing communication to exchange synchronization data, so we do not raise the number of messages in the network.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Sincronização de relógios

A sincronização de relógios é um problema da ciência da computação que trata da diferença entre os relógios internos de diversos dispositivos computacionais e da diferença entre os relógios destes dispositivos e o tempo real. Estas diferenças ocorrem devido a imperfeições de *hardware* dos dispositivos que levam estes a contabilizar o tempo com uma taxa variável. Em geral os dispositivos computacionais possuem um oscilador responsável por gerar um sinal elétrico com uma freqüência constante, através da qual é possível contabilizar o tempo. As imperfeições de *hardware*, citadas anteriormente, fazem com que a freqüência deste sinal não seja constante, causando assim uma diferença entre o tempo contabilizado e o tempo real.

O problema da sincronização de relógios vem ganhando cada vez mais importância à medida que os sistemas distribuídos evoluem e passam a ser utilizados para executar as mais variadas aplicações. Esses sistemas são formados por múltiplos dispositivos computacionais que se comunicam entre si através de uma rede e que podem ser utilizados para executar partes diferentes de uma mesma tarefa simultaneamente. Podemos citar como exemplos de sistemas distribuídos uma rede de computadores, uma rede de telefonia celular, uma rede de sensores, entre outros.

Os sistemas distribuídos compostos por dispositivos móveis em geral utilizam uma rede sem fio como forma de comunicação. Neste caso, estes dispositivos precisam atuar de maneira sincronizada para realizar determinadas tarefas específicas dos protocolos de rede, como a sincronização do envio de pacotes para controle de acesso ao meio de comunicação por exemplo. Em geral os dispositivos móveis possuem restrições quanto ao consumo de energia; nesse sentido a sincronização dos relógios permite que os sistemas de comunicação dos dispositivos sejam ligados e desligados de maneira coordenada visando à economia de energia.

Uma rede de sensores é um exemplo de sistema distribuído. Ela é formada por dispositivos autônomos que usam sensores para monitorar cooperativamente as condições físicas ou ambientais e que possuem um sistema de comunicação sem fio

para a troca de mensagens. A função dessas redes depende de um esforço colaborativo dos dispositivos que a compõem, o que gera a necessidade de sincronização dos sensores para permitir a comparação na escala do tempo entre eventos que ocorrem em diferentes dispositivos. Estes por sua vez possuem restrições quanto ao consumo de energia; dessa forma o uso da sincronização para o acionamento dos dispositivos de comunicação como citamos acima é de extrema valia.

A importância da sincronização dos nós de uma rede de sensores pode ser mais bem entendida através do seguinte exemplo. Imagine um conjunto de sensores que medem a temperatura ambiente em uma determinada região e se comunicam através de uma rede sem fio. Os dados colhidos por sensores diferentes no mesmo instante de tempo real precisam ser marcados com o mesmo tempo estimado pelos nós que os mediram para que possam ser comparados posteriormente. Em determinadas aplicações esta requisição é ainda mais forte e o tempo utilizado para marcar os dados precisa ser extremamente próximo do tempo real no qual a medida foi obtida.

As redes de sensores podem possuir um nó *gateway*, responsável por transmitir as informações captadas na rede para um meio externo, conforme representado na Figura 1.1. Este nó também pode ser responsável por armazenar o tempo padrão que será utilizado por todos os nós da rede e enviar essa informação para todos os nós para que a sincronização seja feita. Um problema para esta solução é a constante troca de mensagens que aumentaria o consumo de energia dos dispositivos. Outro problema é que as redes podem ser implantadas em locais de difícil acesso, impossibilitando assim a utilização de um nó *gateway*.

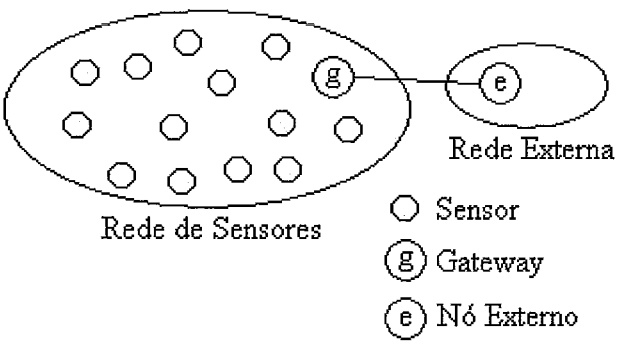


Figura 1.1 – Rede de Sensores com Nó Gateway

As redes sem fio utilizadas pelas redes de sensores, em geral, possuem uma comunicação ponto a ponto. Os nós se conectam diretamente uns com os outros e são responsáveis pelo roteamento das mensagens dentro da rede, uma vez que não existem equipamentos destinados especificamente para esta função. Para que dois nós distintos

se comuniquem é necessário que exista um caminho de nós conectados entre si que possam levar a mensagem do nó de origem até o nó de destino. Na Figura 1.2 temos a representação de uma rede de sensores ponto a ponto e da área de alcance dos nós, o que determina a possibilidade de comunicação entre eles.

Uma das características das redes sem fio ponto a ponto é a facilidade de mudança de sua topologia, pois as conexões entre os nós podem ser feitas ou desfeitas com muita facilidade. Essas mudanças na topologia podem ocorrer pela movimentação dos nós, por mudanças no ambiente que alteram o alcance dos sinais ou pela redução da reserva de energia que também pode reduzir o alcance dos dispositivos de comunicação. Um algoritmo de sincronização deve ser capaz de se adequar a estas situações sem que a sincronização dos relógios seja afetada de forma que prejudique o correto funcionamento das aplicações que dependem desta sincronização.

Os nós das redes de sensores também são vulneráveis a falhas, podendo parar de funcionar durante a execução de um algoritmo de sincronização, portanto estes algoritmos precisam tolerar estas falhas mantendo a sincronização dos nós que continuam em funcionamento. Outro fator adverso é que alguns protocolos de rede utilizados não garantem a entrega das mensagens, o que também deve ser levado em consideração durante o desenvolvimento dos algoritmos.

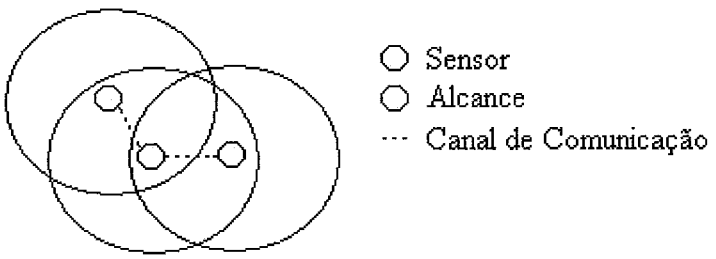


Figura 1.2 – Rede de Sensores Ponto a Ponto

O objetivo de um algoritmo distribuído de sincronização de relógios é garantir que todos os dispositivos participantes no sistema possuam um valor aproximado para os seus relógios. Isto é feito através da criação de um relógio lógico para cada dispositivo. O valor deste relógio lógico é calculado pelo dispositivo através do valor do seu relógio de *hardware* e das mensagens recebidas de seus vizinhos, procurando sempre minimizar a diferença entre os relógios lógicos dos dispositivos. O relógio de *hardware* de um dispositivo é um contador de tempo que utiliza somente o oscilador existente no dispositivo, ou qualquer outra tecnologia destinada a esta função, para estimar o tempo real. O relógio lógico é um contador de tempo que utiliza um algoritmo

e o relógio de *hardware* existente no dispositivo para tentar obter uma melhor aproximação do tempo real.

A sincronização dos relógios pode ser externa ou interna. Na primeira forma, todos os relógios dos dispositivos são sincronizados para um tempo fornecido de fora da rede, tarefa que pode ser realizada pelo nó *gateway*. Na segunda forma não ocorre interferência externa à rede e, portanto, pode existir apenas uma consistência entre os relógios dos dispositivos; este valor comum por sua vez pode diferir do tempo real. Nas redes de sensores a sincronização interna é suficiente para garantir a ordenação dos eventos na escala do tempo e para a coordenação das ações distribuídas.

Algumas aplicações precisam de uma sincronização global, onde todos os nós da rede de sensores estão sincronizados para um tempo comum. Este tipo de sincronização é necessário quando a rede como um todo precisa atuar de uma maneira coordenada. Podemos novamente citar como exemplo a comparação de dados medidos por diversos sensores. De acordo com o nosso conhecimento, os algoritmos desenvolvidos até o momento para este tipo de sincronização possuem uma diferença de pior caso entre os relógios lógicos de dois nós igual a $O(D)$, onde D é o diâmetro da rede. Esse valor de pior caso é válido até para nós vizinhos, o que para algumas aplicações não é suficiente, já que para essas aplicações a sincronização com a propriedade gradiente, a ser descrita agora, é mais adequada.

A sincronização de relógios com a propriedade gradiente consiste em sincronizar de forma mais precisa o relógio dos nós que estão mais próximos entre si na rede, enquanto os nós que estão mais distantes entre si podem possuir uma sincronização mais dispersa. A propriedade gradiente, que será apresentada mais adiante, é caracterizada pela existência de uma função que limita superiormente a diferença máxima entre o relógio de dois nós de acordo com a distância entre eles. Determinadas aplicações precisam de um nível de sincronização entre nós vizinhos melhor do que a sincronização global pode oferecer até o momento e podem aceitar uma sincronização menos eficaz para nós mais distantes.

Trabalhos anteriores, como em [5,6,7,8], focaram em minimizar a diferença entre os relógios lógicos de dois vértices quaisquer realizando assim uma sincronização global. Ao contrário destes, o trabalho desenvolvido em [3], que será discutido mais adiante, apresenta um algoritmo com a propriedade gradiente, mas que necessita de uma alta freqüência de comunicação para atingir os seus objetivos. Consideramos esta

freqüência inadequada para redes de sensores, pois os dispositivos que compõem estas redes possuem restrições quanto ao consumo de energia.

Ao longo deste trabalho iremos apresentar um novo algoritmo de sincronização com a propriedade gradiente dos relógios de redes de sensores, que utiliza a troca de mensagens já existente entre os nós para trafegar os dados necessários para a sincronização, não aumentando assim o número de mensagens que trafegam pela rede. Este algoritmo obtém uma diferença de pior caso entre vizinhos de $O(1)$, que contraria o limite inferior apresentado em [1], mas como veremos adiante o algoritmo não possui uma das propriedades necessárias para a validade do limite inferior.

1.2 Notação

Um algoritmo de sincronização de relógios, denotado por A, será executado sobre uma rede de sensores que será representada por um grafo conectado e não direcionado $G = (N, E)$. N é o conjunto dos nós que representam os sensores e E é o conjunto das arestas não direcionadas que representam canais bidirecionais de comunicação entre os sensores da rede. Faremos $n = |N|$, representando assim o número de nós da rede, e usaremos $N_i \subset N$ para representar o conjunto de todos os nós vizinhos de um nó i qualquer. Consideramos também que uma determinada execução de um algoritmo A será representada por α.

O tempo real será denotado por $t \geq 0$ e nenhum nó da rede terá acesso ao valor de t em momento nenhum. Para simplificar alguns conceitos iremos assumir que a execução do algoritmo sempre começa com tempo real igual a zero. Os nós da rede possuem um relógio de *hardware* cujo valor será representado por $H_i(t)$ para um nó i no instante de tempo real t. Cada nó irá calcular de maneira contínua o seu relógio lógico a partir do seu relógio de *hardware* e das mensagens trocadas com outros nós e este relógio lógico será representado por $L_i(t)$. Iremos assumir valores iniciais iguais a zero para ambos os relógios de um nó. Dessa forma, teremos $H_i(0) = 0$ e $L_i(0) = 0$. Quando for necessário identificar os relógios de um nó i em uma execução α iremos utilizar a notação $L_i^{\alpha}(t)$ e $H_i^{\alpha}(t)$.

O relógio de *hardware* deveria evoluir com a mesma taxa que o tempo real, mas devido às imperfeições dos equipamentos o progresso desse relógio se dá em função de uma taxa que varia com o tempo real. A diferença entre a taxa do tempo real e a taxa do relógio de *hardware* de um nó é chamada de *drift*. Iremos assumir que este *drift* é limitado por uma constante

$$\hat{\rho} \in [0,1) \tag{1.1}$$

e para um nó i no instante t ele será denotado pela função

$$\rho_i(t) \in [-\hat{\rho}, \hat{\rho}]. \tag{1.2}$$

Considerando a característica aditiva do *drift*, a taxa de um relógio de *hardware* será a soma da taxa do tempo real, que é igual a um, com o *drift*; temos então

$$\frac{dH_i(t)}{dt} = 1 + \rho_i(t). \tag{1.3}$$

De acordo com os limites do *drift* podemos calcular os valores mínimo e máximo para a taxa do relógio de *hardware* de um nó, o que nos dá

$$\frac{dH_i(t)}{dt} \geq 1 - \hat{\rho} \tag{1.4}$$

e

$$\frac{dH_i(t)}{dt} \leq 1 + \hat{\rho}. \tag{1.5}$$

O valor exato do relógio de *hardware* de um nó i no instante t pode ser calculado pela integral de sua taxa desde o tempo real zero até o instante desejado. Este valor será a estimativa de tempo real utilizada pelo nó naquele momento.

$$H_i(t) = \int_{x=0}^{t} [1 + \rho_i(x)]dx. \tag{1.6}$$

A Figura 1.3 permite uma comparação visual entre o valor do relógio de *hardware* de três nós em relação ao tempo real no intervalo entre 0 e 100 segundos. Nesta comparação avaliamos o progresso de acordo com as taxas máxima e mínima para o relógio de *hardware*. Consideramos que os *drifts* dos nós são constantes no intervalo mencionado e seus valores estão de acordo com a Tabela 1.1, sendo $\hat{\rho} = 0{,}2$.

Tabela 1.1 – Valores das Taxas de *Hardware* para Comparação

Nó	$\rho_i(t), \forall t \mid 0 \leq t \leq 100$	$\frac{dH_i(t)}{dt}$	$H_i(t)$
A	$\hat{\rho}$	$1+\hat{\rho}$	$t(1+\hat{\rho})$
B	0	1	t
C	$-\hat{\rho}$	$1-\hat{\rho}$	$t(1-\hat{\rho})$

Podemos perceber que após os 100 segundos a diferença entre os relógios dos nós A e B e dos relógios dos nós B e C é de 20 segundos, enquanto a diferença entre os relógios dos nós A e C é de 40 segundos. Esses dados permitem uma melhor avaliação da importância dos algoritmos de sincronização de relógios, uma vez que após um curto período de tempo e mesmo considerando um valor de *drift* baixo dentro dos limites estipulados em (1.1) e (1.2), a diferença entre os valores dos relógios é considerável.

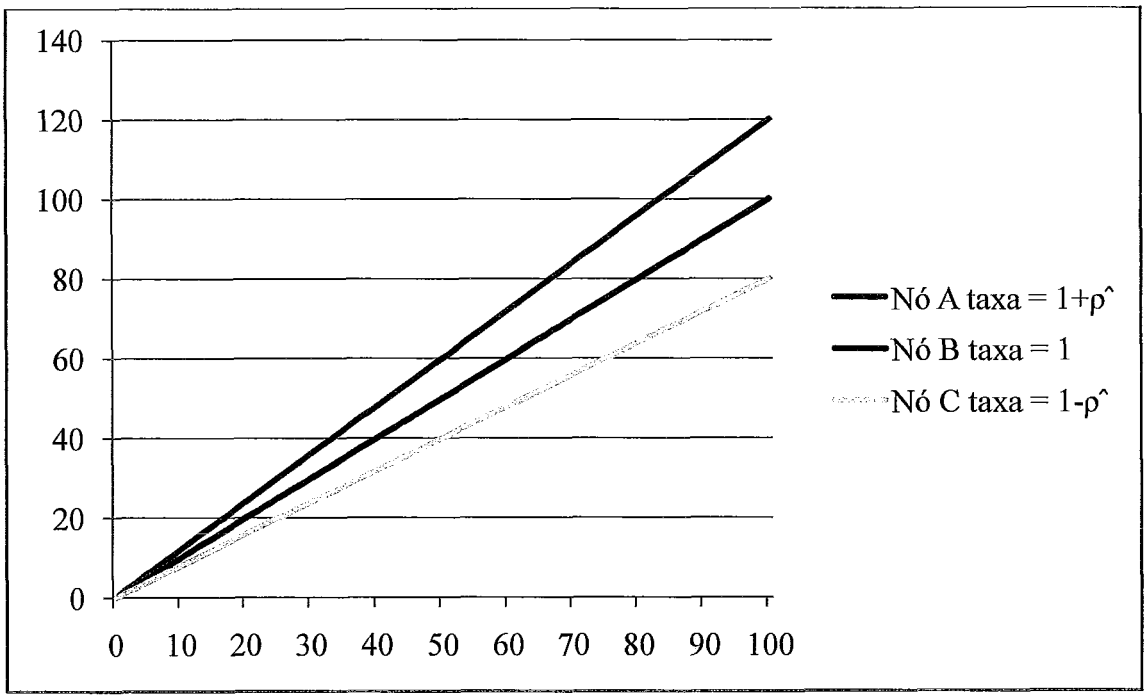


Figura 1.3 - Comparação entre Taxas de Relógios de *Hardware* para $\hat{\rho} = 0{,}2$

A Figura 1.4 permite a comparação entre o tempo real, um relógio de *hardware* com taxa máxima, um relógio de *hardware* com taxa mínima e um relógio de *hardware* com taxa variável. Para esta comparação utilizamos o tempo real entre 0 e 300 segundos e a constante $\hat{\rho} = 0{,}8$.

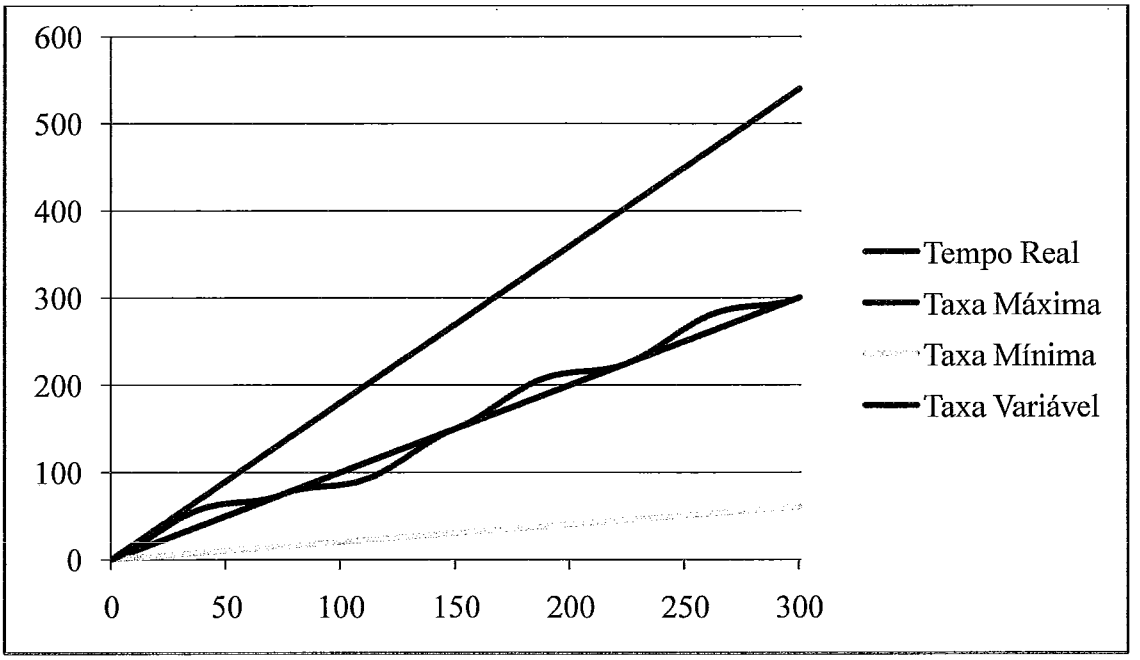


Figura 1.4 - Exemplo de um Relógio de *Hardware* com Taxa Variável e $\hat{\rho} = 0{,}8$

O valor do relógio lógico, assim como sua taxa, será computado de acordo com o algoritmo aplicado. Neste momento iremos assumir apenas que esta taxa deve ser maior ou igual a zero para todo o tempo real, portanto

$$L_i(t') > L_i(t) \text{ para todo } t' > t \tag{1.7}$$

Consideramos que a forma mais simples para calcular a taxa do relógio lógico é usar o mesmo valor da taxa do relógio de *hardware*, teremos então

$$\frac{dL_i(t)}{dt} = \frac{dH_i(t)}{dt}. \tag{1.8}$$

Uma das conseqüências de utilizar a taxa lógica como mostrado em (1.8) é que dependendo do *drift* do relógio de *hardware* a taxa do relógio lógico será muito baixa. Este problema pode ser evitado caso os nós conheçam o valor de $\hat{\rho}$, bastando aplicar um multiplicador ao valor do relógio de *hardware*. Considerando um limite inferior $b \in (0,1]$ para a taxa do relógio lógico teremos

$$\frac{dL_i(t)}{dt} = x\frac{dH_i(t)}{dt} \geq b \tag{1.9}$$

logo

$$x \geq \frac{b}{\frac{dH_i(t)}{dt}}. \tag{1.10}$$

Considerando o valor mínimo para a taxa do relógio de *hardware* como vimos em (1.4) é suficiente utilizar

$$x = \frac{b}{1-\hat{\rho}}. \tag{1.11}$$

Portanto

$$\frac{dL_i(t)}{dt} = \frac{b}{1-\hat{\rho}} \frac{dH_i(t)}{dt}. \tag{1.12}$$

A inclusão desse multiplicador acelera o relógio lógico em relação ao relógio de *hardware* garantindo assim o limite inferior constante para a taxa do relógio lógico. Como conseqüência o relógio lógico pode andar muito rápido caso a taxa do relógio de *hardware* esteja alta, pois aplicando os limites da taxa de *hardware* apresentados em (1.4) e (1.5) na expressão (1.12), teremos

$$b \leq \frac{dL_i(t)}{dt} \leq \frac{b(1+\hat{\rho})}{1-\hat{\rho}}. \tag{1.13}$$

Outra opção seria somar uma constante à taxa de *hardware* da seguinte forma:

$$\frac{dL_i(t)}{dt} = x + \frac{dH_i(t)}{dt} \geq b. \tag{1.14}$$

Logo

$$x \geq b - \frac{dH_i(t)}{dt}. \tag{1.15}$$

Considerando o valor mínimo para a taxa do relógio de *hardware* como vimos em (1.4) é suficiente utilizarmos

$$x = b - (1-\hat{\rho}). \tag{1.16}$$

Teremos então

$$\frac{dL_i(t)}{dt} = b - (1-\hat{\rho}) + \frac{dH_i(t)}{dt}. \tag{1.17}$$

Podemos agora calcular os limites para taxa de um relógio lógico.

$$b \leq \frac{dL_i(t)}{dt} \leq b + 2\hat{\rho}. \tag{1.18}$$

Este breve estudo do problema da taxa mínima para o relógio lógico de um nó serve para nos envolver um pouco mais com o problema da sincronização de relógios em redes de sensores, permitindo um melhor entendimento dos conceitos que serão utilizados ao longo deste trabalho.

A incorporação da propriedade gradiente, que será apresentada mais adiante, na sincronização de relógios, aumenta a importância do conceito da distância entre dois nós dentro de uma rede. Essa distância será denotada por d_{ij} e pode ser calculada pela incerteza no tempo de propagação da mensagem nos canais de comunicação existentes entre os nós, ou pelo número de arestas da rede nos casos onde o tempo de propagação é constante ou desprezível. A incerteza no tempo de propagação das mensagens consiste no tempo máximo que uma mensagem pode levar para percorrer uma aresta do grafo G definido acima. O diâmetro do grafo G, denotado por D, é a maior distância possível entre dois vértices distintos em uma determinada rede. Temos então

$$D = \max_{i,j} d_{ij}. \tag{1.19}$$

1.3 Esta tese

Ao longo desta tese iremos apresentar um algoritmo distribuído para sincronização de relógios em redes de sensores com a propriedade gradiente. O algoritmo desenvolvido assume que os nós conhecem seus vizinhos imediatos e um limite superior para o diâmetro da rede onde estão sendo utilizados. Os dados de sincronização são trocados entre os nós através das mensagens que já trafegam na rede. A propriedade gradiente apresentada em [1] é respeitada pelo algoritmo, que obtém uma diferença entre relógios lógicos de nós vizinhos de pior caso igual a $O(1)$.

Iremos estudar no Capítulo 2 o limite inferior para algoritmos distribuídos de sincronização de relógios com a propriedade gradiente apresentado em [1,2]. Este depende diretamente do diâmetro da rede onde o algoritmo está sendo executado. Veremos que este limite não se aplica ao algoritmo desenvolvido, pois este não determina um limite inferior constante para o progresso do relógio lógico de um nó com relação ao tempo real, fator que é condição necessária para a validade do limite inferior

apresentado. Veremos também que o modelo utilizado na apresentação do limite inferior é ligeiramente diferente do modelo utilizado no desenvolvimento do nosso algoritmo, mas que o segundo é uma simplificação do primeiro.

O algoritmo A^{root} de sincronização de relógios com a propriedade gradiente apresentado em [3] será analisado no Capítulo 2. O algoritmo alcança uma diferença de pior caso entre nós vizinhos igual à $O(\sqrt{D+1})$ através de uma alta freqüência de comunicação. Este algoritmo satisfaz a propriedade gradiente através de uma função limite que satura de acordo com a distância entre os nós como veremos mais adiante.

Vamos estudar o algoritmo desenvolvido no Capítulo 3, juntamente com o modelo proposto para este desenvolvimento, que se assemelha ao modelo utilizado em [2], onde é feito um estudo do limite inferior de [1] em redes de sensores. Por fim vamos apresentar as propriedades desse novo algoritmo e discutir sobre algumas questões que permanecem em aberto.

Capítulo 2

Sincronização com a Propriedade Gradiente

2.1 A propriedade gradiente

Os algoritmos de sincronização de relógios vêm sofrendo uma constante evolução como pode ser visto em [5,6,7,8], mas de acordo com o nosso conhecimento, todos os algoritmos desenvolvidos até então, com uma única exceção em [3], possuem uma diferença entre relógios lógicos de pior caso igual a $O(D)$. Esta diferença é independente da distância entre dois nós distintos, e dessa forma a diferença entre relógios lógicos de pior caso é a mesma para nós vizinhos e para nós com uma distância D entre si. Determinados problemas em redes de sensores requerem uma sincronização mais restrita para nós próximos, enquanto nós mais distantes entre si podem ter uma sincronização mais dispersa.

A propriedade gradiente, introduzida em [1], consiste na existência de um limite superior para a diferença entre os relógios de dois nós, determinado por uma função positiva não-decrescente da distância entre estes. A distância entre os nós pode ser determinada pela incerteza no tempo de propagação das mensagens ou pelo número de saltos entre dois nós caso o tempo de propagação das mensagens seja constante ou desprezível. Um algoritmo para sincronização com gradiente de relógios deve respeitar o limite determinado por esta função para qualquer par de nós em qualquer rede e em qualquer execução.

Sejam A um algoritmo para sincronização com gradiente de relógios em redes de sensores, n qualquer inteiro positivo, N um conjunto de nós $1, \ldots, n$ e as distâncias entre estes nós definidas por $\{d_{ij} | 1 \leq i, j \leq n\}$. Para toda execução α de A em N e durante todo o tempo desta execução o seguinte deve ser verdadeiro

$$\forall t \forall i, j: |L_i^{\alpha}(t) - L_j^{\alpha}(t)| \leq f(d_{ij}). \tag{2.1}$$

A principal implicação desta propriedade é a formação de um gradiente da diferença entre os relógios de dois nós distintos com relação à distância entre eles. Dessa forma a propriedade gradiente complementa a sincronização de relógios e atende

aos problemas que necessitam de uma sincronização mais precisa entre nós próximos e permitem que nós mais distantes possuam uma sincronização mais dispersa.

2.2 Limite inferior

O trabalho realizado em [1] apresenta um limite inferior da diferença entre os relógios lógicos de dois nós distintos que um algoritmo para sincronização com gradiente de relógios é capaz de atingir dentro de um determinado modelo que será apresentado a seguir. O limite é provado através da criação de duas execuções do algoritmo de sincronização de relógios através da manipulação das taxas dos relógios de *hardware* e do tempo de propagação das mensagens. Essas manipulações são feitas de tal forma que as duas execuções são indistinguíveis para todos os nós da rede e que ao final da segunda execução a diferença entre os relógios de *hardware* de um determinado par de nós é maior do que ao final da primeira. Dessa forma nenhum algoritmo seria capaz de realizar ações que compensem essa diferença, pois, como as duas execuções são iguais as ações determinadas pelo algoritmo serão iguais nos dois casos.

O modelo utilizado em [1] para a apresentação do limite inferior é formado por um conjunto fixo de nós que começam a executar no mesmo instante de tempo. Todos os nós podem se comunicar entre si de maneira confiável, ou seja, não há perdas de mensagens. A distância entre dois nós distintos i e j, representada por d_{ij} como vimos anteriormente, é dada pela incerteza do tempo de propagação de uma mensagem entre os nós i e j. Dessa forma uma mensagem enviada de i para j leva no mínimo 0 e no máximo d_{ij} unidades de tempo para ir de i até j. O resultado do limite inferior é dado em termos do diâmetro da rede de acordo com a expressão (1.19). Para que esta definição permaneça consistente os autores definiram um valor mínimo para a distância entre dois nós distintos

$$\min_{i,j} d_{ij} = 1, \tag{2.2}$$

evitando assim que o diâmetro da rede seja igual a zero.

Uma determinada execução de um algoritmo de sincronização com gradiente de relógios identificada por α possui duração em tempo real denotada por $\ell(\alpha)$. O estado de uma execução α no instante de tempo real t, imediatamente antes de qualquer evento ocorrer neste instante, é denotado por $\alpha(t)$. O tempo real onde um evento qualquer π ocorre em α é denotado por $T_\alpha(\pi)$.

Assim como apresentamos anteriormente, os nós são equipados com um relógio de *hardware* cujo valor é definido de acordo com a sua taxa, denotada em [1] por $h_i^{\alpha}(t)$ para uma determinada execução α de um algoritmo qualquer no instante de tempo real t. O valor deste relógio neste mesmo instante t de tempo durante a mesma execução α é dado por

$$H_i^{\alpha}(t) = \int_0^t h_i^{\alpha}(r)dr. \tag{2.3}$$

No desenvolvimento do limite inferior o relógio de *hardware* possui um *drift* limitado. A taxa de *hardware* varia dentro dos seguintes limites

$$\forall i \forall t: 1 - \rho \leq h_i^{\alpha}(t) \leq 1 + \rho, \tag{2.4}$$

onde ρ é constante e $0 \leq \rho < 1$. Podemos perceber que a combinação das expressões (2.3) e (2.4) utilizadas em [1] é equivalente a combinação das expressões (1.3), (1.4) e (1.5) se considerarmos que estas são aplicadas à mesma execução α. De acordo com a expressão (2.4) temos

$$\min h_i(t) = 1 - \rho \tag{2.5}$$

e

$$\max h_i(t) = 1 + \rho. \tag{2.6}$$

Como a constante ρ utilizada na expressão (2.4) é equivalente à constante $\hat{\rho}$ utilizada nas expressões (1.4) e (1.5), pois ambas possuem os mesmos limites e a mesma finalidade, e a derivada da expressão (2.3) em relação ao tempo igual a $h_i(t)$, podemos afirmar que

$$\frac{dH_i(t)}{dt} = h_i(t) = 1 + \rho(t). \tag{2.7}$$

Nesta expressão estamos desconsiderando a identificação da execução α. Gostaríamos de chamar a atenção para a diferença entre o termo $\rho(t)$ em (2.7) que é definido em (1.2) e a constante ρ utilizada em [1].

O valor do relógio lógico de um nó i no instante t em uma execução α é denotado por $L_i^{\alpha}(t)$. Podemos agora mostrar a base do limite inferior em termos matemáticos, sendo α e β duas execuções distintas do mesmo algoritmo e supondo que

as mesmas ações acontecem em α e β na mesma ordem. Se para cada ação π no nó i em α temos uma ação π' correspondente no mesmo nó i em β, tal que,

$$H_i^{\alpha}(T_{\alpha}(\pi)) = H_i^{\beta}(T_{\beta}(\pi')) \tag{2.8}$$

o nó i irá se comportar da mesma forma em ambas as execuções. A expressão (2.8) mostra que o valor do relógio de *hardware* de i no instante em que π ocorre em α deve ser o mesmo que no instante em que π' ocorre em β.

A definição do limite inferior feita em [1] indica que este só é válido para algoritmos que respeitam duas propriedades. Uma delas é a propriedade gradiente, que já foi apresentada anteriormente. A segunda propriedade diz respeito à existência de um valor mínimo para a taxa do relógio lógico dos nós. Para que o limite apresentado seja válido para um determinado algoritmo, este deve garantir que a taxa de crescimento do relógio lógico de um nó seja de pelo menos $1/2$ durante toda a sua execução Temos então que o seguinte deve ser verdadeiro para determinada execução α de A:

$$\forall t \forall i \forall r > 0 : L_i^{\alpha}(t + r) - L_i^{\alpha}(t) \geq \frac{r}{2}. \tag{2.9}$$

Segundo [1] este valor foi escolhido de maneira a simplificar os cálculos e pode ser substituído por qualquer constante positiva. As motivações apresentadas para esta restrição são remover algoritmos triviais que iniciem o relógio lógico com valor zero e nunca atualizem este valor, além da necessidade da maioria das aplicações de que os relógios lógicos não sejam incrementados de maneira muito lenta, ou até mesmo de que os relógios lógicos andem próximos do tempo real.

Gostaríamos de destacar algumas propriedades interessantes do modelo utilizado. Primeiramente, a diferença entre os relógios de dois nós distintos é limitada através da propriedade gradiente. A incerteza do tempo de propagação das mensagens, bem como o tempo de propagação das mensagens em si é limitado, mas não existem restrições quanto à ordem de chegada das mensagens com relação à ordem de envio. Para finalizar, a freqüência de comunicação é ilimitada, o que permite que um algoritmo possa enviar mensagens de sincronização em curtos intervalos de tempo com o objetivo de melhorar a sincronização.

Podemos afirmar que o resultado do trabalho realizado em [1] é um limite inferior para a diferença entre o relógio lógico de nós com distância d entre si que pode ser

obtido durante a execução de um algoritmo de sincronização de relógios A qualquer. O limite é expresso por

$$f(d) = \Omega\left(d + \frac{\log D}{\log \log D}\right). \tag{2.10}$$

Desta forma não existe algoritmo que consiga sincronizar dois nós de tal forma que a diferença entre seus relógios lógicos fique sempre abaixo de (2.10) durante todo o tempo de todas as suas execuções. O mesmo pode ser escrito da seguinte forma: em alguma execução de um algoritmo de sincronização de relógios, em algum momento um determinado par de nós vai possuir uma diferença entre seus relógios maior que o valor reproduzido em (2.10).

Os autores de [1] optaram por dividir a prova deste limite em duas partes. Primeiro foi realizada a prova que para qualquer número real $d \geq 1$ existe uma rede contendo dois nós a uma distância d entre si, de forma que tais nós possuem uma diferença entre seus relógios lógicos igual a $\Omega(d)$ em alguma execução de A. Depois foi necessário mostrar que para qualquer inteiro $D \geq 1$ existe uma rede de diâmetro D e uma execução do algoritmo A neste rede, tal que dois nós vizinhos, ou seja, com distância 1 entre si, possuam uma diferença de relógios lógicos de $\Omega(\log D / \log \log D)$ em algum momento. Temos então

$$f(d) = \Omega(d) \tag{2.11}$$

e

$$f(1) = \Omega\left(\frac{\log D}{\log \log D}\right). \tag{2.12}$$

O restante desta seção contém a prova do limite inferior, sugerimos que a leitura continue a partir da próxima seção caso não haja interesse em se aprofundar nesta assunto.

As execuções foram criadas controlando as taxas dos relógios de *hardware* e os tempos de entrega das mensagens dos nós, criando assim situações indistinguíveis que apresentam as diferenças entre relógios lógicos desejadas. A prova de (2.11) é apresentada em [9] e apenas a construção de duas execuções indistinguíveis nas quais a diferença do relógio lógico entre dois nós é diferente é apresentada em [1]. Reproduzimos agora a configuração destas duas execuções.

Execução α: Os nós i e j possuem valores de relógios iguais. O tempo de propagação de i para j é igual a zero e o tempo de propagação de j para i é igual a d.

Execução β: O nó i possui o relógio atrasado em d unidades de tempo em relação ao relógio de j. O tempo de propagação de i para j é igual a d e o tempo de propagação de j para i é igual a zero.

Considerando que os dois algoritmos começam no instante de tempo real igual a zero, que os nós i e j enviam mensagens um para o outro em α e β assim que a execução é iniciada e sejam $\pi_{i,j}^{\alpha}$ e $\pi_{j,i}^{\alpha}$ respectivamente os eventos determinados pela chegada em j da mensagem enviada por i e da chegada em i da mensagem enviada por j na execução α e $\pi_{i,j}^{\beta}$ e $\pi_{j,i}^{\beta}$ eventos equivalentes para a execução β. Basta manipularmos as taxas dos relógios de *hardware* de i e j de forma que $H_i^{\alpha}(T_{\alpha}(\pi_{j,i}^{\alpha})) = H_i^{\beta}(T_{\beta}(\pi_{j,i}^{\beta}))$ e $H_j^{\alpha}(T_{\alpha}(\pi_{i,j}^{\alpha})) = H_j^{\beta}(T_{\beta}(\pi_{i,j}^{\beta}))$ e fazer o mesmo para os eventos que representam os envios das mensagens para que as execuções sejam indistinguíveis para os nós.

Para demonstrar a parte do limite inferior representada em (2.12) é utilizada uma rede N com diâmetro positivo e inteiro igual a D e nós representados por índices $1, \ldots, D$, tal que $d_{ij} = |i - j|$, para $1 \leq i, j \leq D$, temos então um grafo em linha. Os autores de [1] criaram dois lemas para a parte (2.12) do limite inferior. O lema *Add Skew*, afirma que dados dois nós i e j, uma determinada execução α que apresente certas condições em relação à taxa do relógio de *hardware* dos nós e em relação ao tempo de propagação das mensagens, é possível encontrar uma execução β de forma que os nós i e j apresentem uma diferença de relógios maior ao final de β do que ao final de α. O segundo lema, chamado de *Bounded Increase*, afirma que em qualquer execução de A que satisfaça determinadas condições sobre as taxas dos relógios de *hardware* e sobre o tempo de propagação das mensagens, nenhum nó é capaz de incrementar o seu relógio lógico muito rapidamente de maneira a compensar as diferenças incluídas pelo lema *Add Skew*.

Reproduzimos agora o lema *Add Skew*, este diz que, sejam dois nós i e j com $1 \leq i < j \leq D$, dados:

$$\tau = \frac{1}{\hat{\rho}}; \tag{2.13}$$

$$\gamma = 1 + \frac{\hat{\rho}}{4 + \hat{\rho}}; \tag{2.14}$$

$$T = S + \tau(j - i); \tag{2.15}$$

$$T' = S + \frac{\tau}{\gamma}(j - i); \tag{2.16}$$

Seja α uma execução de A com duração T e supondo que o seguinte é verdadeiro:

(i) O tempo de propagação das mensagens entre dois nós k_1 e k_2 durante o intervalo de tempo $[S, T]$ em α é $\frac{|k_1 - k_2|}{2}$.

(ii) Todos os nós possuem taxa de relógio de *hardware* igual a 1 durante o intervalo $[S, T]$. Ou seja, $\forall i \forall t \in [S, T]: h_i^{\alpha}(t) = 1$.

Então existe uma execução β de A tal que o seguinte é verdadeiro:

(i) $L_i^{\beta}(T') - L_j^{\beta}(T') \geq L_i^{\alpha}(T) - L_j^{\alpha}(T) + \frac{j-i}{12}$.

(ii) O tempo de propagação das mensagens entre dois nós k_1 e k_2 durante o intervalo $[0, S]$ é igual ao da execução α e o tempo de propagação entre k_1 e k_2 está entre $\left[\frac{|k_1 - k_2|}{4}, \frac{3|k_1 - k_2|}{4}\right]$ durante o intervalo $(S, T']$ em β.

O lema indica que dados dois nós $i < j$ em uma execução α que satisfaça determinados limites para o tempo de propagação das mensagens e para a taxa do relógio de *hardware* dos nós durante o intervalo de tempo $[S, T]$, é possível encontrar uma execução β onde os nós i e j possuem uma diferença de relógios maior do que $(j - i)/12$ no tempo real T' em β do que a diferença existente no tempo real T em α.

A idéia básica do lema é a criação de uma execução β na qual os relógios de *hardware* de alguns nós são acelerados em determinados intervalos de tempo. Os tempos de propagação das mensagens são ajustados de forma que a execução β seja indistinguível em relação à execução α. Em conseqüência os nós em β que possuem um relógio de *hardware* acelerado também vão possuir um relógio lógico acelerado, aumentando assim as diferenças entre os relógios lógicos.

A execução β é então definida da seguinte forma, as ações que ocorrem em β são um subconjunto das ações que ocorrem α. Ou seja, uma ação π ocorre em β somente se esta mesma ação ocorre em α. O tempo real no qual as ações ocorrem nas

duas execuções pode ser diferente e alguns nós possuem um relógio de *hardware* mais rápido em β em determinados períodos de tempo. A execução α dura T unidades de tempo real, enquanto a execução β dura T' unidades como foram definidas em (2.15) e (2.16). A taxa do relógio de *hardware* de um nó k em β, para $1 \leq k \leq D$, é definida por

$$h_k^{\beta}(t) = \begin{cases} 1 & ,\forall t \in [0, T_k] \\ \gamma & ,\forall t \in (T_k, T'] \end{cases} \tag{2.17}$$

onde

$$T_k = \begin{cases} S & ,\forall k: 1 \leq k \leq i \\ S + \dfrac{\tau}{\gamma}(k-i) & ,\forall k: i < k < j \\ T' & ,\forall k: j \leq k \leq D \end{cases} \tag{2.18}$$

Uma representação gráfica das taxas lógicas dos nós durante a execução β de acordo com as definições (2.17) e (2.18), obtida em [1], pode ser vista na Figura 2.1. O eixo horizontal representa o tempo real, enquanto o eixo vertical representa os nós da rede. As linhas horizontais grossas representam o intervalo de tempo durante o qual o nó que referencia esta linha no eixo vertical possui taxa de relógio de *hardware* igual a γ, nos outros intervalos a taxa é sempre igual a 1. Partindo do nó 1, à medida que nos aproximamos do nó j o tempo que um determinado nó passa com seu relógio de *hardware* acelerado diminui em τ/γ unidades de tempo, ou seja, um determinado nó k possui taxa igual a γ por τ/γ unidades de tempo a mais que o nó $k+1$, para $k = 1, \ldots, j-1$.

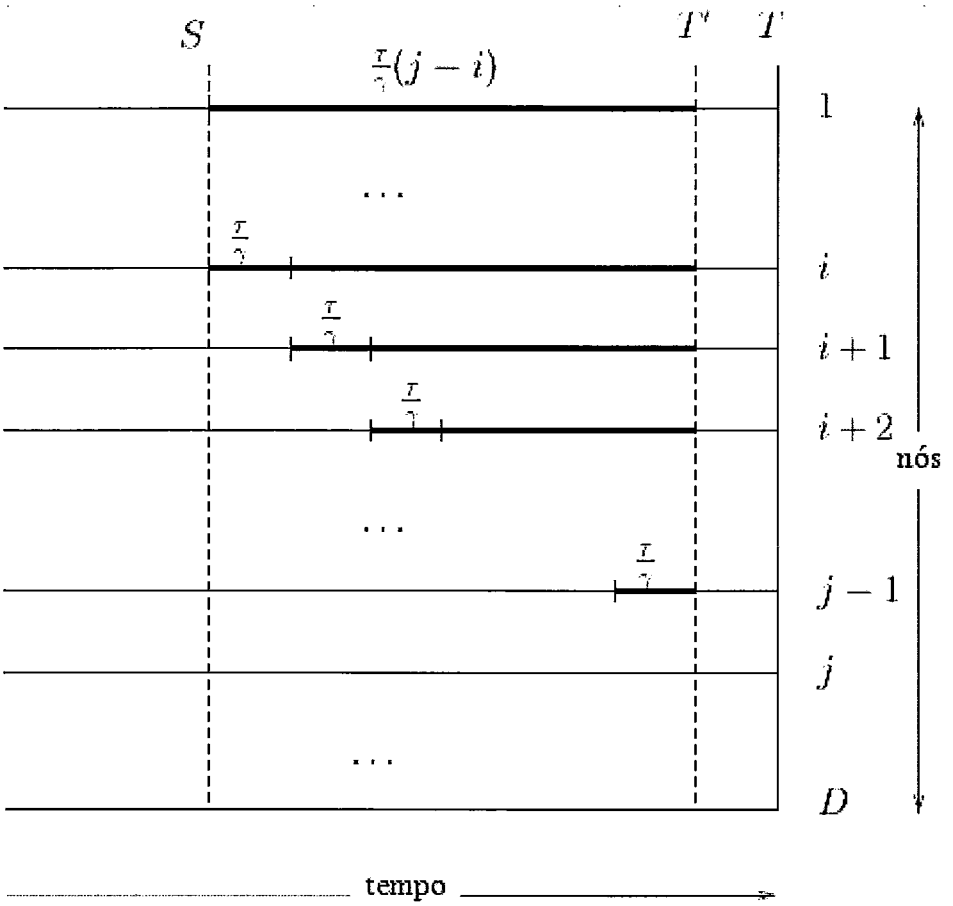


Figura 2.1 – Taxa de Relógio de *Hardware* em uma Execução β

O nó onde uma determinada ação π que ocorre em α é representado por $k(\pi)$ e como já mostramos anteriormente o tempo real no qual essa ação ocorre é representado por $T_\alpha(\pi)$. Para a execução β o tempo real onde essa ação correspondente irá ocorrer será definido por

$$T_\beta(\pi) = \begin{cases} T_\alpha(\pi) & ,\forall T_\alpha \in [0, T_{k(\pi)}] \\ T_{k(\pi)} + \frac{1}{\gamma}\left(T_\alpha(\pi) - T_{k(\pi)}\right) & ,\forall T_\alpha \in \left(T_{k(\pi)}, T'\right]. \end{cases} \quad (2.19)$$

Através destas definições para as taxas dos relógios de *hardware* e para o tempo real no qual as ações ocorrem na execução β, foram criadas duas execuções α e β de um mesmo algoritmo de sincronização de relógios que são indistinguíveis para os nós da rede e que ao final da execução β, dois nós i e j possuem uma diferença entre seus relógios maior do que a diferença existente entre eles ao final da execução α, mostrando assim o lema *Add Skew*. A prova que a execução β satisfaz esse lema é baseada na prova de três afirmações que reproduzimos agora:

a) As execuções α e β são indistinguíveis para todos os nós.

b) A taxa do relógio de *hardware* de todos os nós em β está dentro dos limites corretos, ou seja, $1 - \hat{\rho} \leq h_i^{\beta}(t) \leq 1 + \hat{\rho}$.

c) O tempo de propagação das mensagens é o mesmo em α e β durante o intervalo de tempo $[0, S]$. Durante o intervalo $(S, T']$ o tempo de propagação para um par de nós qualquer k_1 e k_2 está limitado por $\left[\frac{|k_1 - k_2|}{4}, \frac{3|k_1 - k_2|}{4}\right]$ em β.

A combinação das três afirmações acima mostra que β é uma execução de A indistinguível da execução α. As provas destas três afirmações não estão envolvidas na invalidade do limite inferior que veremos mais adiante, portanto, deixaremos a cargo do leitor o estudo destas provas que foram desenvolvidas em [1]. Neste momento resta apenas mostrar que a execução β aumenta a diferença entre os relógios lógicos de dois nós i e j distintos, ou seja,

$$L_i^{\beta}(T') - L_j^{\beta}(T') \geq L_i^{\alpha}(T) - L_j^{\alpha}(T) + \frac{j - i}{12}. \tag{2.20}$$

A prova desta afirmação está diretamente ligada à invalidade do limite inferior para o algoritmo desenvolvido, portanto iremos analisá-la aqui. A partir de (2.3), (2.17) e (2.18) podemos mostrar que $H_i^{\beta}(T') = H_i^{\alpha}(T)$ e $H_j^{\beta}(T') = H_j^{\alpha}(T')$. Calculando inicialmente os valores para a execução α onde

$$\begin{aligned} H_i^{\alpha}(T) &= \int_0^T h_i^{\alpha}(r)dr \\ &= \int_0^T dr \\ &= T \end{aligned} \tag{2.21}$$

e

$$\begin{aligned} H_j^{\alpha}(T') &= \int_0^{T'} h_j^{\alpha}(r)dr \\ &= \int_0^{T'} dr \\ &= T'. \end{aligned} \tag{2.22}$$

Calculando agora os valores para a execução β.

$$H_i^{\beta}(T') = \int_0^{T'} h_i^{\beta}(r)dr$$
$$= \int_0^{T_i} h_i^{\beta}(r)dr + \int_{T_i}^{T'} h_i^{\beta}(r)dr$$

e

$$H_j^{\beta}(T') = \int_0^{T'} h_j^{\beta}(r)dr$$
$$= \int_0^{T_j} h_j^{\beta}(r)dr + \int_{T_j}^{T'} h_j^{\beta}(r)dr.$$

De acordo (2.18) teremos que $T_i = S$ e $T_j = T'$, portanto

$$H_i^{\beta}(T') = \int_0^{S} h_i^{\beta}(r)dr + \int_{S}^{T'} h_i^{\beta}(r)dr$$

e

$$H_j^{\beta}(T') = \int_0^{T'} h_j^{\beta}(r)dr.$$

Substituindo os valores apresentados em (2.17) teremos

$$H_i^{\beta}(T') = \int_0^{S} dr + \int_{S}^{T'} \gamma dr$$
$$= S + \gamma(T' - S)$$
$$= S + \gamma(S + \frac{\tau}{\gamma}(j - i) - S)$$
$$= S + \tau(j - i)$$
$$= T \qquad (2.23)$$

e

$$H_j^{\beta}(T') = \int_0^{T'} dr$$
$$= T'. \qquad (2.24)$$

Juntando (2.21) a (2.23) e (2.22) a (2.24) verificamos que $H_i^{\beta}(T') = H_i^{\alpha}(T)$ e $H_j^{\beta}(T') = H_j^{\alpha}(T')$ respectivamente conforme foi apresentado em [1]. Como as execuções α e β são indistinguíveis, como mostramos anteriormente, teremos também que

$$L_i^{\beta}(T') = L_i^{\alpha}(T) \tag{2.25}$$

e

$$L_j^{\beta}(T') = L_j^{\alpha}(T'). \tag{2.26}$$

A partir da restrição para a taxa do relógio lógico apresentada em [1] e reproduzida em (2.9) teremos

$$L_j^{\alpha}(T) - L_j^{\alpha}(T') \geq \frac{1}{2}(T - T'). \tag{2.27}$$

Substituindo a expressão (2.26) em (2.28) teremos,

$$L_j^{\alpha}(T) - L_j^{\beta}(T') \geq \frac{1}{2}(T - T')$$

$$L_j^{\beta}(T') \leq L_j^{\alpha}(T) - \frac{1}{2}(T - T'). \tag{2.28}$$

Subtraindo (2.28) de (2.25) chegamos à seguinte expressão

$$L_i^{\beta}(T') - L_j^{\beta}(T') \geq L_i^{\alpha}(T) - L_j^{\alpha}(T) + \frac{1}{2}(T - T'), \tag{2.29}$$

onde

$$\begin{aligned} T - T' &= \left(S + \tau(j-i)\right) - \left(S + \frac{\tau}{\gamma}(j-i)\right) \\ &= \tau\left(1 - \frac{1}{\gamma}\right)(j-i) \\ &= \frac{1}{\rho}\left(1 - \frac{1}{1 + \frac{\rho}{4+\rho}}\right)(j-i) \\ &= \frac{1}{4+2\rho}(j-i) \\ &\geq \frac{1}{6}(j-i). \end{aligned} \tag{2.30}$$

Substituindo (2.30) em (2.29) chegamos à prova da expressão (2.20) e em conseqüência do lema *Add Skew*.

O lema *Bounded Increase* diz que, seja α uma execução de A com duração $T \geq \tau$ e i um nó qualquer. Supondo que

(i) Todos os nós possuem taxa de relógio de *hardware* entre $[1, 1 + \frac{\hat{\rho}}{2}]$ durante todo o tempo de execução de α.

(ii) O tempo de propagação das mensagens entre i e j está entre $[\frac{|i-j|}{4}, \frac{3|i-j|}{4}]$ durante todo o tempo de execução de α.

Então, para qualquer $t \geq \tau$, teremos

$$L_i^{\alpha}(t+1) - L_i^{\alpha}(t) \leq 16f(1) \tag{2.31}$$

A prova deste lema não é necessária para a prova da invalidade do limite inferior, portanto deixamos aqui apenas a idéia inicial de como esta prova foi feita em [1] e fica a cargo do leitor o estudo aprofundado desta. Assumindo que um nó i incremente seu relógio lógico muito rapidamente em uma execução α, criamos então outra execução β onde aceleramos o relógio de *hardware* de i. Fazemos β indistinguível de α através dos ajustes do tempo de propagação das mensagens. Como o nó i possui um relógio de *hardware* mais rápido em β ele também irá possuir um relógio lógico mais rápido, possuindo assim uma diferença de tempo com seus vizinhos maior, o que pode então violar a propriedade gradiente. Dessa forma um nó i está limitado a incrementar o seu relógio lógico de forma que não desrespeite a propriedade gradiente.

O teorema principal criado utilizado para a prova do limite inferior de $f(1)$ apresentado em [1] afirma que existe uma execução de um algoritmo A, na qual um par de nós que possuem distância 1 entre si possuirão uma diferença entre seus relógios lógicos de pelo menos $\Omega(\log D / \log\log D)$ ao final desta execução. Ou seja, existe uma execução α de A, com nós i e j, com $d_{ij} = 1$, tal que $L_i^{\alpha}(\ell(\alpha)) - L_j^{\alpha}(\ell(\alpha)) = \Omega(\log D / \log\log D)$. Então, $f(1) = \Omega(\log D / \log\log D)$.

Mais uma vez deixaremos o estudo da prova para o leitor e faremos apenas uma breve explicação de como esta foi realizada. A prova é feita através da criação de uma execução do algoritmo onde o lema *Add Skew* é aplicado repetidamente para incrementar a diferença entre os relógios lógicos de determinados nós vizinhos. Ao

mesmo tempo é utilizado o lema *Bounded Increase* para limitar o quão rápido esta diferença pode ser diminuída através do limite imposto para o incremento dos relógios lógicos dos nós. Dessa forma a diferença entre os relógios lógicos não pode ser diminuída de maneira rápida sem que a propriedade gradiente seja desrespeitada, levando a uma diferença entre nós vizinhos igual ao limite inferior apresentado em [1].

Concluímos assim o estudo do limite inferior para algoritmos de sincronização com gradiente de relógios apresentado em [1]. Focamos apenas nas partes da prova do limite inferior que serão de interesse para provar que este não se aplica ao algoritmo desenvolvido neste trabalho e deixamos a idéia inicial das outras partes da prova para que o leitor se familiarize com estas e possa se aprofundar no assunto caso seja de interesse.

2.3 O algoritmo A^{root}

O trabalho realizado em [3] procura analisar se o limite inferior de [1] é preciso e se existem algoritmos que conseguem alcançar, ao menos assintoticamente, este limite. O trabalho apresenta um conjunto de algoritmos simples, sendo que nenhum deles se aproxima do limite inferior de [1]. Dentro dos algoritmos deste conjunto, vamos analisar apenas o algoritmo chamado A^{root}, o que possui um limite para a diferença de relógios entre nós com distância d entre si igual a $O(d + \sqrt{D})$ durante todo o tempo.

O modelo utilizado em [3] é semelhante ao modelo de [1], a troca de mensagens continua confiável, mas o tempo de propagação das mensagens é limitado em [0,1]. Para todos os nós $i \in N$ e para todo o tempo t, teremos $h_i(t) \in [\mathcal{L}, \mathcal{U}]$, onde $0 < \mathcal{L} < U$. Fazendo um paralelo com o que vimos até agora podemos considerar

$$\mathcal{L} = 1 - \hat{\rho} \tag{2.32}$$

e

$$\mathcal{U} = 1 + \hat{\rho}. \tag{2.33}$$

O grau de sincronização que pode ser alcançado está diretamente relacionado com o tempo de propagação das mensagens utilizado em [3]. Com o interesse de garantir que um nó consiga aumentar o seu relógio em mais de uma unidade de tempo, enquanto uma mensagem leva no máximo uma unidade de tempo para ser propagada os

autores assumiram $\mathcal{U} \geq 1$. Podemos facilmente perceber que o valor apresentado em (2.33) está de acordo com esta consideração.

O valor do relógio lógico de um nó aumenta com a mesma taxa que o relógio de *hardware* deste enquanto nenhuma mensagem for recebida. Um algoritmo de sincronização especifica como o relógio lógico $L_i(t)$ do nó i no tempo t é modificado de acordo com o valor atual deste e do histórico de mensagens recebidas pelo nó i de seus vizinhos até o tempo t. Os algoritmos apresentados são reativos, desta forma eles atualizam o relógio lógico sempre que uma nova mensagem é recebida. Os autores de [3] utilizam $\tilde{L}_j(t)$ para representar o maior valor de relógio lógico recebido de um nó vizinho j até o instante t. Ete valor pode não se referir a última mensagem recebida deste nó uma vez que nada garante a ordem das mensagens. O histórico de mensagens recebidas por um nó i de seus vizinhos é representado por $\psi_i(t)$. Teremos então que para todo algoritmo A o seguinte deve ser sempre verdadeiro

$$\forall i \forall t : \; L_i(t) \leq A\big(L_i(t), \psi_i(t)\big) \leq \max_{j \in N_i} \tilde{L}_j(t). \tag{2.34}$$

Onde $A\big(L_i(t), \psi_i(t)\big)$ representa o valor de relógio lógico calculado pelo algoritmo A de acordo com o valor deste valor atual $L_i(t)$ e com as mensagens recebidas $\psi_i(t)$.Como o relógio lógico não pode voltar no tempo, o algoritmo pode aumentar o valor deste ou não modificá-lo. Além disso, o maior valor que o algoritmo pode dar para um relógio lógico é o maior valor recebido até então. Caso isto não seja respeitado podemos causar um aumento desnecessário nos relógios dos nós vizinhos, que potencialmente poderia resultar em uma diferença maior entre os relógios de determinados nós.

Os nós armazenam apenas uma parte do histórico das mensagens recebidas, representado por

$$\tilde{\psi}_i(t) = \{\tilde{L}_i(t)\}^{j \in N_i}, \tag{2.35}$$

ou seja, o conjunto dos maiores valores de relógios lógicos recebidos de cada um dos vizinhos. Sempre que uma nova mensagem é recebida esses valores são atualizados e um novo valor para o relógio lógico é computado de acordo com o histórico armazenado e com o valor do relógio lógico neste momento. Caso o novo valor computado seja menor que o existente, o relógio lógico é mantido, caso contrário, o relógio lógico é atualizado e o seu novo valor é enviado para todos os vizinhos do nó.

Os limites encontrados para os algoritmos apresentados são provados através de um grafo em linha com n nós. Neste tipo de grafo existem canais de comunicação entre os nós k e $k+1$ para $k \in \{1, n-1\}$. O valor inicial dos relógios lógicos é igual a zero. A inicialização do algoritmo A^{root} é feita da seguinte maneira. No tempo real zero o nó n envia uma mensagem de inicialização para o seu vizinho e inicia o seu relógio lógico. Quando um nó recebe uma mensagem de inicialização pela primeira vez ele inicia o seu relógio lógico e envia esta mensagem de inicialização para todos os seus vizinhos. Para a sincronização, cada nó informa todos os seus vizinhos sobre o valor de seu relógio lógico regularmente.

Os algoritmos em [3] assumem que todos os nós transmitem uma mensagem com o seu relógio lógico para todos os vizinhos sempre este alcança um valor inteiro ou quando este é atualizado devido ao recebimento de uma mensagem. Se um nó ainda não recebeu nenhuma mensagem de um determinado vizinho, ele considera o relógio lógico desse vizinho como zero. Esta propagação de mensagens ajuda na sincronização dos relógios, mas pode ser prejudicial em aplicações que possuam restrições quanto ao número de mensagens enviadas.

O algoritmo A^{root} utiliza o conjunto de mensagens recebidas dos nós vizinhos, o valor atual de relógio lógico e o diâmetro da rede para calcular o novo valor para o relógio lógico do nó. Este algoritmo atualiza o valor do relógio lógico de um nó para o maior valor já recebido de um vizinho, desde que este novo valor seja maior do que o seu relógio atual e não ultrapasse o menor valor já recebido em $\mathcal{U}\sqrt{D+1}$ unidades de tempo.

O algoritmo A^{root} garante que a diferença entre dois nós quaisquer está sempre limitada em $O(D)$ e o valor máximo utilizado para evitar que um nó incremente ainda mais o seu relógio lógico caso esteja muito distante de um vizinho garante uma diferença de pior caso para nós vizinhos igual a $O(\sqrt{D})$. O tempo no qual um nó precisa esperar por outros nós para incrementar o seu relógio lógico através do recebimento de mensagens é no máximo de $O(\sqrt{D})$.

O algoritmo A^{root}, como comentamos anteriormente, requer que todos os nós enviem o valor de seus relógios lógicos sempre que este alcance um valor inteiro. Além do diâmetro D, este algoritmo também necessita de que os nós conheçam o limite superior para a taxa de relógio de *hardware*. Em geral é possível obter um limite

superior $\mathcal{U}$ para a taxa de *hardware*. Conforme discutimos anteriormente, utilizamos $\mathcal{U} = 1 + \hat{\rho}$ no novo algoritmo desenvolvido.

Reproduzimos agora a descrição do algoritmo A^{root} apresentado em [3]. Um determinado nó i, ao receber uma nova mensagem de um nó vizinho j, verifica se o valor enviado nessa mensagem é maior que o valor armazenado $\tilde{L}_j(t)$. Em caso afirmativo, esse novo valor é armazenado e o valor antigo descartado. O algoritmo então verifica se o valor do relógio lógico do nó deve ser atualizado. Isto irá ocorrer se o valor atual do relógio for menor que o maior valor já recebido de seus vizinhos,

$$L_i(t) < \max_{j \in N_i} \tilde{L}_j(t),$$

e se o valor atual não estiver mais do que $\mathcal{U}\sqrt{D+1}$ unidades de tempo à frente do menor valor armazenado já recebido de seus vizinhos,

$$L_i(t) < \min_{j \in N_i} \tilde{L}_j(t) + \mathcal{U}\sqrt{D+1}.$$

Caso as condições acima sejam satisfeitas, o valor do relógio lógico do nó i irá receber o menor valor entre o maior valor já recebido de um vizinho e armazenado até então e a soma do menor valor já recebido de um vizinho somado com $\mathcal{U}\sqrt{D+1}$. Ou seja,

$$L_i(t) := \min\left(\max_{j \in N_i} \tilde{L}_j(t), \min_{j \in N_i} \tilde{L}_j(t) + \mathcal{U}\sqrt{D+1}\right). \tag{2.36}$$

Após a atualização, o nó i irá enviar o novo valor de seu relógio lógico para todos os seus vizinhos. Todas as mensagens que não determinem a atualização no relógio lógico do nó são simplesmente ignoradas.

O algoritmo A^{root} possui uma propriedade global que garante uma diferença de pior caso entre os relógios lógicos de dois nós quaisquer igual a $O(D)$. Esta propriedade global é utilizada para mostrar a propriedade gradiente do algoritmo apresentado. A propriedade global é definida em [3] da seguinte forma. Para todas as execuções de A^{root} em qualquer grafo, o seguinte é verdadeiro:

$$\forall i, j \forall t : \left|L_i(t) - L_j(t)\right| < \mathcal{U}D + 1 \in O(D). \tag{2.37}$$

Considerando um grafo com diâmetro D, sejam os nós i e j os nós das extremidades do grafo e i o nó com maior taxa de *hardware* e j com a menor. Com a inicialização do

algoritmo pelo nó i, o nó j irá iniciar o seu relógio lógico no máximo D unidades de tempo depois, considerando o maior tempo de propagação de mensagens possível. Durante esse período o nó i poderá ter um aumento em seu relógio lógico de no máximo $\mathcal{U}D$, caso ele possua a taxa máxima para o seu relógio de *hardware*, resultando em uma diferença máxima entre eles de $\mathcal{U}D$.

A chegada da próxima mensagem ao nó j irá ocorrer no máximo uma unidade de tempo após a chegada da mensagem de inicialização. Durante esse intervalo de tempo, o relógio lógico de i aumenta com a taxa de *hardware*, levando a uma diferença máxima igual a $\mathcal{U}D + 1$. Quando esta nova mensagem chegar ao nó j ele irá atualizar o seu relógio lógico de acordo com o algoritmo e este novo valor será no mínimo igual ao mesmo valor que nó i tinha D unidades de tempo atrás. Dessa forma a diferença não pode aumentar ainda mais e caso o tempo de propagação das mensagens reduza, o nó j pode até mesmo alcançar o nó i, uma vez que ele irá atualizar o seu relógio lógico de maneira mais rápida através do algoritmo.

A propriedade gradiente em A^{root} é garantida da seguinte forma. Para todas as execuções e em qualquer grafo, o seguinte é verdadeiro:

$$\forall t : \left|L_i(t) - L_j(T)\right| < 2\mathcal{U}\sqrt{D+1} \in O\left(\sqrt{D}\right), \tag{2.38}$$

para nós i e j vizinhos.

Essa diferença de pior caso entre nós vizinhos para o algoritmo A^{root} foi obtida em [3] através de uma cadeia de espera entre nós. Esta cadeia é definida de seguinte maneira. Seja a diferença entre os relógios lógicos dos nós i e j igual a $\mathcal{U}\sqrt{D+1}$ no instante de tempo t; teremos

$$L_i(t) = L_j(t) + \mathcal{U}\sqrt{D+1}. \tag{2.39}$$

Se existir um nó $k \in N_j$ tal que

$$L_j(t) = L_k(t) + \mathcal{U}\sqrt{D+1}, \tag{2.40}$$

o nó j terá que esperar pelo nó k para incrementar o seu relógio lógico através das mensagens recebidas. O nó k também pode ter um vizinho com relógio lógico atrasado em relação a ele em $\mathcal{U}\sqrt{D+1}$ unidades de tempo, formando assim uma cadeia de

espera entre os nós. Considerando o tamanho desta cadeia igual a ℓ, serão necessárias no máximo ℓ unidades de tempo até que o nó j possa aumentar o seu relógio lógico em $\mathcal{U}\sqrt{D+1}$ unidades de acordo com a regra (2.36).

A propriedade global do algoritmo A^{root} diz que a diferença máxima entre dois nós quaisquer é limitada. Dessa forma, o tamanho ℓ da cadeia de espera, determinada por uma diferença entre os relógios lógicos de dois nós vizinhos, também é limitado. Esse limite é calculado pela divisão do limite superior determinado na propriedade global pelo valor da diferença entre os relógios lógicos de cada nó vizinho existente na cadeia de espera. Teremos então

$$\ell \leq \frac{\mathcal{U}D+1}{\mathcal{U}\sqrt{D+1}}$$

$$\leq \sqrt{D+1}, \tag{2.41}$$

pois $\mathcal{U} \geq 1$. A Figura 2.2 é uma representação visual de uma cadeia de espera. Podemos perceber que no instante t, onde a cadeia é formada, a diferença entre cada nó que participa dela é exatamente $\mathcal{U}\sqrt{D+1}$ e que a soma das diferenças entre os relógios de todos os nós que participam da cadeia é de $\mathcal{U}D+1$.

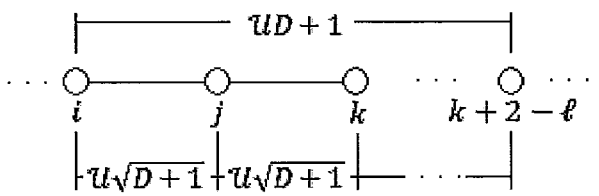


Figura 2.2 – Cadeia de Espera para A^{root}

Como a cadeia de espera possui ℓ nós e considerando o tempo de propagação máximo das mensagens, serão necessárias $\sqrt{D+1}$ unidades de tempo antes que o nó j possa atualizar o seu relógio lógico de acordo com as mensagens recebidas. Durante esse intervalo de tempo o relógio lógico do nó i irá aumentar de acordo com a taxa de seu relógio de *hardware*, que é no máximo $\mathcal{U}$, determinando um aumento máximo de $\mathcal{U}\sqrt{D+1}$ unidades de tempo. Passado esse intervalo de tempo, o nó j poderá aumentar o seu relógio lógico de acordo com o algoritmo e a diferença para o nó i será reduzida. Como a diferença no instante da formação da cadeia de espera era igual a $\mathcal{U}\sqrt{D+1}$, teremos uma diferença máxima de $2\mathcal{U}\sqrt{D+1}$ e a diferença entre dois nós na cadeia de

espera com distância d entre si será no máximo $2d\mathcal{U}\sqrt{D+1}$. É garantida assim a diferença de pior caso de $O(\sqrt{D})$.

O algoritmo A^{root} apresentado em [3] foi o primeiro algoritmo de nosso conhecimento que possui a propriedade gradiente. Através do estudo realizado verificamos que o valor de $f(d_{i,j})$ em (2.1) é

$$f(d_{i,j}) = \begin{cases} O(d_{ij}\sqrt{D}) & \text{, se } d_{ij} \leq \sqrt{D+1}; \\ O(D) & \text{, caso contrário.} \end{cases} \tag{2.42}$$

A primeira parte da expressão (2.42) provém da propriedade gradiente e a segunda da propriedade global. Podemos perceber que a função $f(d_{ij})$ satura em $O(D)$, pois a partir de $d_{ij} > \sqrt{D+1}$ a função permanece com valor constante e garante o limite $O(d+\sqrt{D})$. A forma como o algoritmo alcança esses resultados requer que os nós conheçam o diâmetro da rede onde estão atuando e um limite superior para a taxa de *hardware* dos nós. Além disso, é necessária uma alta freqüência de comunicação entre os nós. Para determinadas redes de sensores, é possível obter uma estimativa sobre o diâmetro da rede e o valor máximo da taxa de *hardware* dos nós, mas acreditamos que a alta freqüência de comunicação seja incompatível com as restrições quanto ao consumo de energia associadas às redes de sensores.

Capítulo 3

Um Novo Algoritmo

3.1 O algoritmo

Apresentamos agora um novo algoritmo distribuído de sincronização com gradiente de relógios em redes de sensores. Este algoritmo possui a propriedade gradiente e suas ações ocorrem de tal forma que o limite inferior apresentado em [1] não se aplica. Assumimos que os nós conhecem um pouco da topologia da rede e os dados de sincronização utilizados são trafegados juntamente com as mensagens que já estão sendo trocadas pelos nós, assumindo que a freqüência de comunicação seja razoável. Dessa forma o algoritmo não aumenta a quantidade de mensagens em tráfego na rede.

Ao contrário do algoritmo A^{root} apresentado em [3], o algoritmo desenvolvido neste trabalho é destinado à sincronização em redes de sensores. Por esta razão utilizamos um modelo que considera as características desses tipos de rede, relativamente diferente do modelo utilizado em [1,3]. Utilizamos o mesmo modelo do trabalho desenvolvido em [2], que analisa o limite apresentado em [1] em redes de sensores, um ambiente onde o tempo de propagação das mensagens pode ser desprezado se compararmos com o intervalo de tempo entre as trocas de mensagens.

As diferenças em relação aos modelos dizem respeito às comunicações que já ocorrem entre os nós e que são utilizadas pelo algoritmo para a propagação dos dados de sincronização. Assumimos que a freqüência de comunicação não é muito alta; dessa forma o efeito do *drift* dos relógios de *hardware* dos nós entre as trocas de mensagens é muito maior do que a incerteza do tempo de propagação das mensagens, permitindo assim que esta incerteza seja considerada igual a zero.

Vamos assumir que as mensagens trocadas entre vizinhos em G são entregues instantaneamente. Como o tempo de propagação das mensagens é igual a zero, utilizamos o número de saltos para calcular as distâncias na rede ao invés de utilizar a incerteza do tempo de propagação. De acordo com a expressão (1.19) e com a definição de distância que será utilizada, o diâmetro de uma rede passa a ser o número máximo de saltos desta rede. Pelos mesmos motivos que consideramos o tempo de propagação das mensagens igual a zero, iremos considerar o tempo computacional gasto para a

execução do algoritmo como zero. Dessa forma todo calculo determinada pelo algoritmo é realizado instantaneamente.

O padrão de troca de mensagens que ocorre na rede não é conhecido pelo algoritmo, mas para obter resultados satisfatórios, assumimos que um par de nós se comunica entre si ao menos uma vez a cada intervalo de d unidades de tempo. Este limite leva em consideração a direção da mensagem, ou seja, se um nó i enviar mensagem para o seu nó vizinho j no instante t, ele terá que enviar outra mensagem para j antes do instante $t + d$ mesmo que o nó j lhe envie uma mensagem nesse intervalo. Garantimos assim que o nó j possui uma informação sobre o relógio lógico do nó i relativamente recente, com atraso máximo de d unidades de tempo. Resumimos os pressupostos utilizados no desenvolvimento do algoritmo da seguinte forma.

(i) O tempo de propagação das mensagens é igual a zero.

(ii) Se t e t' são dois instantes distintos onde o nó i envia uma mensagem para seu nó vizinho j, então teremos $|t - t'| \leq d$ para uma constante $d > 0$.

(iii) O tempo gasto para qualquer calculo computacional do algoritmo é igual a zero.

Podemos fazer um paralelo com o trabalho desenvolvido em [1], onde são criadas execuções de um algoritmo através da manipulação das taxas do relógio de *hardware* dos nós e do tempo de propagação das mensagens. O mesmo pode ser feito neste novo modelo manipulando as taxas dos relógios de *hardware* e o instante de tempo real onde as trocas de mensagens são realizadas.

Os *drifts* dos relógios de *hardware* utilizados são os mesmos apresentados anteriormente em (1.4) e (1.5), assim como as expressões para calculo da taxa e do valor em um determinado instante do relógio de *hardware*, definidos em (1.3) e (1.6) respectivamente. A Tabela 3.1 apresenta um comparativo das características dos dois modelos.

Tabela 3.1 – Características dos Modelos Estudados

	Modelo de [1]	Modelo utilizado
Drift dos relógios de *hardware*	limitado	limitado
Incerteza do tempo de propagação	limitado	0
Tempo de propagação	limitado	0
Freqüência de comunicação	ilimitado	limitado

O algoritmo necessita que os nós possuam algumas informações para garantir a propriedade gradiente e a diferença de pior caso constante. Assumimos que os nós devem conhecer o diâmetro D da rede onde estão atuando, mas devido à dinâmica da topologia de uma rede de sensores essa informação é variável ao longo da execução do algoritmo e de difícil acesso, o que aumenta a complexidade do algoritmo. Consideramos então que os nós conhecem um limite superior para o valor de D, informação que é suficiente para as necessidades do algoritmo.

Assumimos que os nós não possuem nenhum conhecimento sobre os valores que limitam o tempo máximo entre duas comunicações e sobre o *drift* dos relógios de *hardware*, d e $\hat{\rho}$ respectivamente. A sincronização tem início no instante de tempo real $t = 0$ em um único nó, de onde é propagada para os outros nós da rede. Se o nó i é onde a sincronização tem início, este possui relógio lógico igual a zero no tempo real zero, ou seja, $L_i(0) = 0$. Caso contrário, assumimos que o relógio lógico do nó i será igual a zero no instante em que este nó receber a sua primeira mensagem de sincronização, ou seja, $L_i(t) = 0$ para o instante de tempo real t no qual o nó i recebe a sua primeira mensagem de sincronização. Chamamos mensagem de sincronização uma mensagem que contém algum dado de sincronização enviado pelo algoritmo.

Os nós devem conhecer os seus vizinhos. Para um determinado nó i estes são os nós que pertencem ao conjunto N_i, e para cada nó vizinho $j \in N_i$ o nó i deve possuir uma variável L_i^j para armazenar o seu valor conhecido do relógio lógico do nó j. Este será o último valor de relógio lógico enviado de j para i através de uma mensagem de comunicação. Se t é o instante de tempo real no qual o nó j envia o valor de seu relógio lógico $L_j(t)$ para o nó i, então de acordo com o pressuposto (i), $L_i^j = L_j(t)$ desde o instante t até o instante no qual i receber a próxima mensagem de sincronização do nó j. O valor inicial para esta variável que armazena o último valor de relógio lógico recebido de um vizinho é zero Temos então que $L_i^j = 0$ para todo $i \in N$ e para todo $j \in N_i$ antes da chegada da primeira mensagem de sincronização enviada pelo nó j para o nó i.

Ao contrário dos algoritmos apresentados em [3], o algoritmo apresentado aqui não utiliza diretamente a taxa do relógio de *hardware* de um nó como a taxa do relógio lógico deste nó. Para calcular a taxa lógica de um nó, multiplicamos a taxa de *hardware* por uma variável que é atualizada sempre que um nó recebe uma mensagem de sincronização de um de seus vizinhos. Esta variável, denotada por α_i para um

determinado nó i, expressa a proporção da taxa de progresso atual de $L_i(t)$ com relação à taxa de progresso de $H_i(t)$. A taxa do relógio lógico de um determinado nó i será então calculada por

$$\frac{dL_i(t)}{dt} = \alpha_i \frac{dH_i(t)}{dt}. \tag{3.1}$$

O algoritmo é baseado na redução do valor da variável α_i sempre que o nó i percebe que seu relógio lógico está consideravelmente adiantado com relação ao relógio de algum de seus vizinhos. Este consideravelmente adiantado é mensurado através de uma constante $c > 0$, como veremos mais adiante. Sempre que um determinado nó i receber uma mensagem de um nó vizinho j cujo valor de relógio lógico estiver atrasado em mais de c unidades de tempo com relação ao seu relógio lógico, o nó i irá reduzir o valor de α_i, reduzindo assim a taxa de seu relógio lógico e permitindo que o nó j se aproxime novamente dele.

Os nós devem possuir um conjunto de variáveis, uma para cada um de seus vizinhos, que indicam se determinado vizinho possui relógio lógico atrasado a mais de c unidades de tempo com relação ao seu próprio relógio lógico; estes valores serão utilizados para o calculo de α_i. Portanto, cada nó $i \in N$ deve possuir uma variável α_i^j para cada vizinho $j \in N_i$. Uma determinada variável α_i^j será atualizada pelo algoritmo sempre que o nó i receber uma mensagem de sincronização de seu nó vizinho j, e esta permanecerá com este novo valor até que uma nova mensagem seja recebida. Vamos assumir que no início do algoritmo de sincronização em um nó i teremos $\alpha_i^j = 1$ para todo $j \in N_i$.

O valor de α_i para um determinado nó i será igual ao mínimo entre os valores de suas variáveis α_i^j para todo $j \in N_i$. Teremos então

$$\alpha_i = \min_{j \in N_i} \alpha_i^j. \tag{3.2}$$

Como o valor inicial de todo α_i^j para $j \in N_i$ é igual a um, o valor inicial de α_i também será igual a um. Dessa forma, quando o algoritmo de sincronização tiver início para determinado nó i, a taxa de seu relógio lógico será igual à taxa de seu relógio de *hardware*.

Os dados de sincronização são trocados através das mensagens que já ocorrem na rede. Quando um nó j envia uma mensagem para um nó vizinho i no instante t ele envia o valor de seu relógio lógico $L_j(t)$ naquele instante. De acordo com o pressuposto (i) o nó i irá receber essa mensagem no mesmo instante t e pode fazer uma comparação precisa entre o seu relógio lógico e o de seu vizinho j. O nó i verifica então se o nó j está atrasado a mais de c unidades de tempo com relação a ele: se $L_i(t) \geq L_j(t) + c$ for verdadeiro o nó i irá atualizar o valor de α_i^j para $1/D$, caso seja falso o valor de α_i^j será atualizado para 1. Quando uma das variáveis α_i^j for atualizada a variável α_i também será atualizada.

A regra de atualização das variáveis α_i^j para $j \in N_i$ apresentada acima permite que estas variáveis indiquem os nós que estão atrasados em relação ao nó i. De acordo com a expressão (3.2) o valor de α_i será igual a $1/D$ se o nó i possuir ao menos um nó vizinho atrasado a mais de c unidades de tempo com relação a ele e será igual a um apenas se nenhum nó vizinho estiver atrasado a mais de c unidades de tempo com relação a ele.

A simples redução da taxa do relógio lógico de um nó não garante que os relógios lógicos de seus vizinhos irão alcançá-lo em tempo razoável. Sendo assim, optamos por utilizar a mesma idéia utilizada no algoritmo A^{root} onde um nó pode atualizar o valor de seu relógio lógico caso este não ultrapasse o limite da constante c com relação ao seu vizinho mais lento. Nosso algoritmo determina que um nó i atualize o valor de $L_i(t)$ para o menor dos valores entre o maior valor de L_i^j e o menor valor de L_i^j somado à constante c. Lembramos que, assim como no algoritmo A^{root}, o valor de $L_i(t)$ só será atualizado se este for menor que o valor computado através da regra apresentada acima.

O algoritmo pode ser mais bem descrito se pensarmos em como um nó reage ao recebimento de uma mensagem de um de seus vizinhos. Denotamos uma mensagem de sincronização por $\langle L \rangle$, sendo enviada de um nó $j \in N_i$ para o nó i no instante t. L será o valor do relógio lógico de j no instante t em que a mensagem foi enviada, e de acordo com (i), teremos $L = L_j(t)$. Para facilitar a descrição do algoritmo, vamos utilizar

$$L^- = \min_{j \in N_i} L_i^j \tag{3.3}$$

e

$$L^+ = \max_{j \in N_i} L_i^j \tag{3.4}$$

A resposta do nó i é composta dos três passos abaixo que de acordo com (iii) ocorrem de maneira instantânea.

Passo 1:

$$L_i^j := L$$

Passo 2:

Se $L_i(t) \geq L_i^j + c$, então $\alpha_i^j = \frac{1}{D}$, caso contrário $\alpha_i^j = 1$.

$$\alpha_i = \min_{j \in N_i} \alpha_i^j.$$

Passo 3:

$$L_i(t) := \max\{L_i(t), \min\{L^- + c, L^+\}\}$$

O Passo 1 atualiza o valor que o nó i possui do relógio lógico de seu vizinho j. De acordo com o pressuposto (ii), no pior caso esta visão será atualizada no máximo em intervalos de d unidades de tempo, ou seja, um determinado valor de relógio lógico de j recebido por i será mantido por no máximo d unidades de tempo. O Passo 2 realiza a redução do valor de α_i^j caso o nó j esteja atrasado em mais de c unidades de tempo com relação ao nó i. Este passo também é responsável por aumentar o valor de α_i^j para um, caso o nó j já tenha se aproximado do nó i para permitir que este avance com a sua taxa lógica normal e devemos lembrar que isto só irá acontecer se para todo $j \in N_i$ tivermos $\alpha_i^j = 1$. O Passo 2 possui outra característica importante onde valor de determinado α_i^j só será atualizado no instante em que o nó i receber uma mensagem do nó j.

O Passo 3 é responsável pela atualização do valor do relógio lógico de i. Esta atualização só irá ocorrer se o relógio lógico de i for menor que o maior relógio lógico já recebido de seus vizinhos e menor que o menor relógio lógico já recebido de seus vizinhos somado à constante c. A primeira restrição desse passo evita que o relógio lógico de i aumente ainda mais caso ele seja o mais rápido entre os seus vizinhos; isto só pode ocorrer através da taxa de relógio lógico e não através de uma atualização realizada pelo algoritmo. A segunda restrição não permite que o nó i se afaste ainda mais de um nó que já possui diferença maior que c com relação a ele, a não ser através de sua taxa de relógio lógico, que neste caso, de acordo com o Passo 2 será reduzida através do multiplicador $\alpha_i = 1/D$.

A Figura 3.1 contém quatro exemplos da execução do Passo 3 do algoritmo em um nó i. As linhas verticais representam os valores de L_i^j, $L_i(t)$ e L_i^k no momento em que o Passo 3 do algoritmo é executado no nó i. Consideramos o valor L_i^j como o maior valor já recebido pelo nó i e o valor L_k^i o menor valor já recebido por esse mesmo nó i. Os gráficos da esquerda apresentam o valor do relógio lógico do nó i antes da atualização pelo Passo 3, enquanto os gráficos da direita mostram o valor deste relógio após a atualização. Nos exemplos a e b o valor de i não sofre alterações, pois este é maior que o menor dos valores entre L_i^j e $L_i^k + c$. No exemplo c o nó atualiza o seu relógio lógico para o valor de $L_i^k + c$, enquanto no exemplo d o nó i atualiza para o valor de L_i^j.

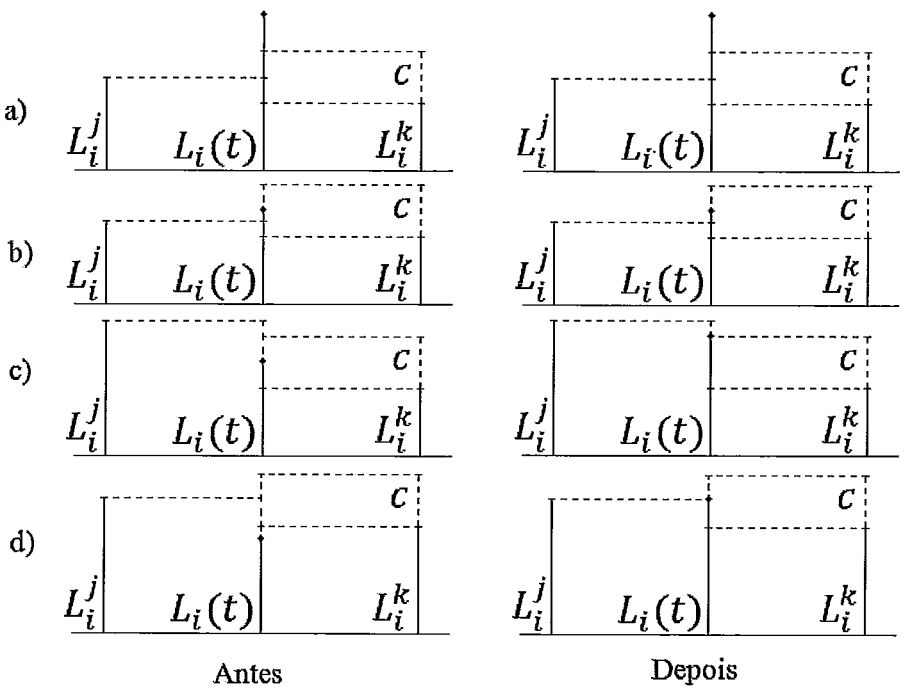


Figura 3.1 – Exemplos de Atualizações pelo Passo 3 do Algoritmo

O valor de $L_i(t)$ resultante dos Passos 1, 2 e 3 continuará a crescer com a sua taxa de relógio lógico multiplicada pelo valor de α_i de acordo com a expressão (3.1) até que uma nova mensagem de sincronização seja enviada por algum dos vizinhos de i. Caso o nó i envie uma mensagem de sincronização para algum de seus vizinhos no instante t' dentro deste intervalo e t seja o instante onde ele recebeu a última mensagem de sincronização de um de seus vizinhos, ele irá calcular o valor de seu relógio lógico de acordo com a seguinte expressão

$$L_i(t') = L_i(t) + \alpha_i[H_i(t') - H_i(t)] \quad (3.5)$$

A Figura 3.2 mostra uma comparação, durante a execução do algoritmo, entre o valor do relógio lógico de um nó i e o tempo real, considerando que a taxa de *hardware* deste nó é igual à taxa do tempo real. Nos intervalos $t \in [0,20)$, $t \in [40,60)$ e $t \in [60,80)$ temos $\alpha_i = 1$. Nos intervalos $t \in [20,40)$ e $t \in [60,80)$ temos $\alpha_i = 1/D$. No instante $t = 60$ ocorre uma atualização do relógio lógico do nó através do Passo 3 do algoritmo.

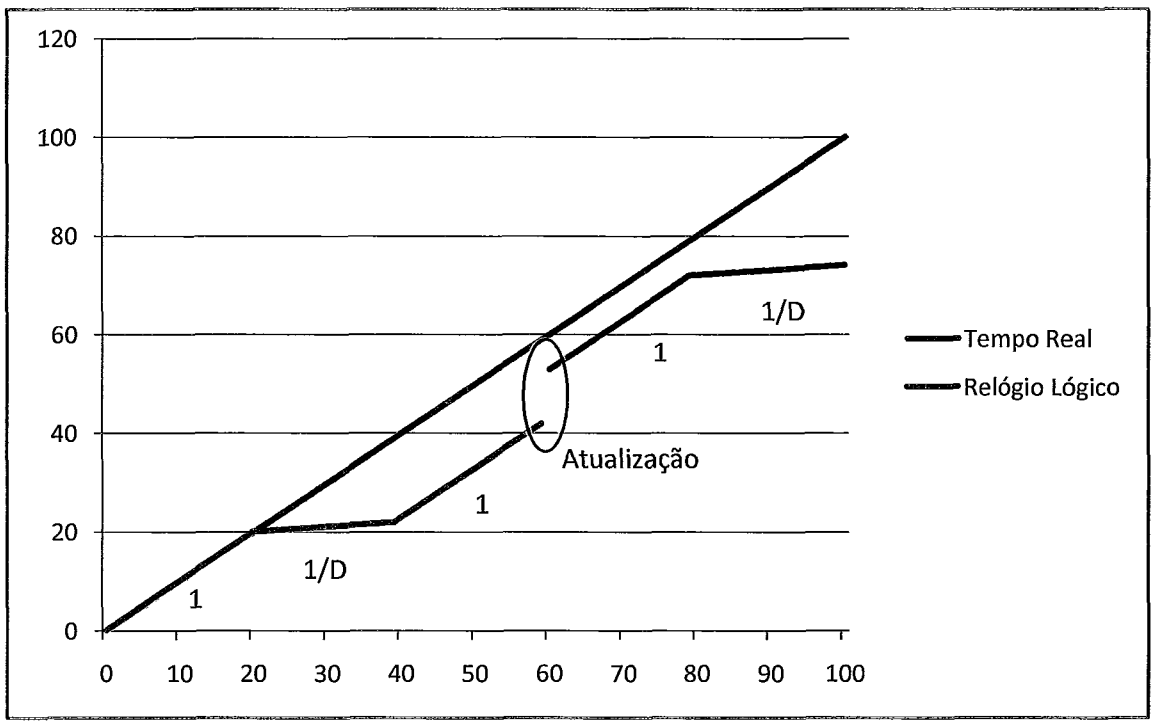


Figura 3.2 – Comparação entre o Relógio Lógico de um Nó e o Tempo Real

Definimos assim o novo algoritmo de sincronização gradiente de relógios em redes de sensores, agora podemos estudar as suas características e apresentar o motivo que invalida o limite inferior apresentado em [1] para este algoritmo.

3.2 Propriedades

Após a apresentação do algoritmo podemos estudar algumas de suas propriedades, vamos mostrar as diferenças de pior caso entre nós vizinhos e entre dois nós quaisquer, vamos comparar as diferenças entre o novo algoritmo desenvolvido e o algoritmo A^{root} para percebermos como a inclusão de um novo passo é capaz de reduzir a diferença de pior caso entre nós vizinhos para $O(1)$. Também iremos mostrar que o

algoritmo respeita a propriedade gradiente e vamos estudar outras características de menor importância.

O valor utilizado para a constante c no algoritmo A^{root}, $\mathcal{U}\sqrt{D+1}$, é a principal razão para o resultado de pior caso igual a $O(\sqrt{D})$ obtido para a diferença de relógios entre nós vizinhos. Existe um balanceamento da diferença de pior caso determinado pela expressão (2.41), se diminuirmos o valor de c estamos diminuindo o valor utilizado pelo algoritmo para garantir que um nó não vai atualizar o seu relógio lógico se estiver muito distante do seu vizinho mais lento, mas ao mesmo tempo diminuímos o denominador da expressão (2.41), aumentando assim o tamanho máximo da cadeia de espera e em conseqüência o tempo máximo que o nó mais rápido pode se afastar até que o seu vizinho atualize o relógio lógico de acordo com o algoritmo. Se aumentarmos o valor de c estamos aumentando diretamente a diferença máxima entre dois vizinhos.

O estudo que realizamos sobre o algoritmo A^{root} mostra que não é possível utilizar um valor de $O(1)$ para a constante c para este algoritmo, pois teríamos a formação de uma cadeia de espera de comprimento $O(D)$. Esta cadeia de espera faria com que o nó com maior valor de relógio lógico, que é o nó que inicializa o algoritmo, se afaste de seu vizinho durante um intervalo de tempo na ordem de $O(D)$ antes que seu vizinho possa atualizar o seu relógio lógico de acordo com o algoritmo, alcançando assim uma diferença de pior caso igual a $O(D)$.

Para possibilitar um valor constante para c foi necessário criar um mecanismo capaz de reduzir a taxa de relógio lógico dos nós que estão a mais de c unidades de tempo à frente de ao menos um de seus vizinhos. O Passo 2 do nosso algoritmo é responsável por essa redução através da comparação do valor recebido de um vizinho e do valor atual do relógio lógico do nó em questão. Esta comparação só é possível devido ao pressuposto (i), pois caso contrário estaríamos comparando valores de relógios lógicos obtidos em instantes de tempo real diferentes.

Podemos agora calcular os valores máximo e mínimo para a taxa de relógio lógico de um nó em nosso algoritmo de acordo com a variável α_i e com a taxa de *hardware*. De acordo com (3.1), que mostra a expressão da taxa do relógio lógico de um nó, com (3.2) que mostra a expressão de α_i e com os valores possíveis para as variáveis α_i^j, como podemos verificar no Passo 2 do algoritmo temos os seguintes valores:

$$\frac{dL_{i(t)}}{dt} \leq (1+\hat{\rho}) \tag{3.6}$$

e

$$\frac{dL_{i(t)}}{dt} \geq \frac{1}{D}(1-\hat{\rho}). \tag{3.7}$$

Utilizando estes valores e o pressuposto (ii) podemos verificar que um determinado nó pode aumentar no máximo em $(1+\hat{\rho})d$ unidades de tempo o seu relógio lógico durante o intervalo de recebimento de duas mensagens de sincronização de um mesmo vizinho. A Figura 3.3 mostra a escala de tempo durante a inicialização do algoritmo que leva a maior diferença entre dois nós quaisquer. As linhas verticais representam os valores dos relógios lógicos dos nós de acordo com o tempo real.

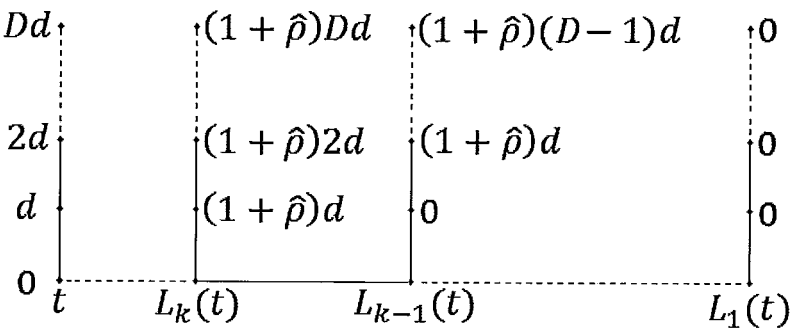


Figura 3.3 – Diferença Máxima entre os Relógios Lógicos de Dois Nós Quaisquer

Vamos agora começar o estudo das diferenças de pior caso que podem ocorrer entre dois nós quaisquer e entre dois nós vizinhos. A diferença de pior caso entre os relógios lógicos de dois nós quaisquer é igual a $O(D)$. Para entendermos esta diferença basta considerarmos uma cadeia com $D+1$ nós em uma rede com diâmetro D. Considerando que o algoritmo tem início em apenas um nó e este é o primeiro da cadeia, irão se passar no máximo Dd unidades de tempo até que todos os nós tenham iniciado os seus relógios lógicos. Durante este intervalo, o relógio lógico do primeiro nó a iniciar o algoritmo progrediu no máximo $(1+\hat{\rho})Dd$, de acordo com a taxa máxima de relógio lógico de um nó apresentada em (3.6). Passado esse intervalo de tempo todos os nós terão iniciado os seus relógios lógicos e de acordo com as ações do algoritmo essa diferença não pode aumentar ainda mais, pois os outros nós irão aumentar as suas taxas de relógio lógico e também irão aumentar o valor deste relógio de acordo a atualização determinada pelo Passo 3 do algoritmo.

A diferença de pior caso para relógios lógicos de nós vizinhos é da ordem de $O(1)$ conforme já citamos anteriormente. O Passo 2 do algoritmo é responsável pela garantia do limite constante como veremos agora. Vamos considerar uma cadeia de espera de tamanho ℓ com nós k e $k-1$ vizinhos para $k \in \{\ell+1,2\}$. Os valores dos relógios lógicos dos nós desta cadeia no instante t em que ela é formada são definidos da seguinte forma:

$$L_k(t) = L_{k-1}(t) + c \text{ para } k-1 \in N_k, \forall k \in [\ell+1, 2]. \tag{3.8}$$

O nó $\ell+1$ possui a maior taxa de *hardware*, igual a $1+\hat{\rho}$, enquanto todos os outros nós da cadeia possuem a taxa mínima, igual a $1-\hat{\rho}$. Consideramos também que todos os nós da cadeia já descobriram que seu vizinho mais lento está atrasado em mais de c unidades de tempo, portanto $\alpha_k = 1/D$ para $k \in \{\ell+1,1\}$. A Figura 3.4 representa os nós da cadeia de espera e os canais de comunicação existentes entre os nós vizinhos.

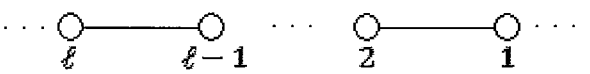


Figura 3.4 – Nomenclatura dos Nós em uma Cadeia de Espera

O número máximo de saltos de uma cadeia de espera será igual ao nosso resultado de limite para a diferença entre qualquer nó, $(1+\hat{\rho})Dd$, dividido pela constante c de acordo com a definição de uma cadeia de espera dada em (3.8). Podemos verificar que a mesma conta foi feita para o algoritmo A^{root} em (2.41). Teremos então

$$\ell \leq \min\left\{D, \frac{(1+\hat{\rho})Dd}{c}\right\}. \tag{3.9}$$

Utilizamos o mínimo entre os dois valores, pois não é possível existir uma cadeia de espera maior que o diâmetro da rede.

No máximo a cada xd unidades de tempo, para $x \in [1, \ell-1]$, o nó x irá enviar uma mensagem para o seu vizinho $x+1$ com um valor de relógio lógico que faz com que o Passo 2 do algoritmo altere o valor de α_{x+1} de volta para 1, isto ocorre pois o nó x terá recebido uma valor do vizinho $x+1$ no instante $(x-1)d$ com c unidade de tempo a frente o que irá igualar o relógio dos dois nós. Este conjunto de mensagens demora no máximo ℓd unidades de tempo para chegar ao nó ℓ e permitir que este

aumente o seu relógio lógico de acordo com o Passo 3 do algoritmo. Durante este intervalo de tempo a diferença entre os relógios lógicos de $\ell + 1$ e ℓ irá aumentar em

$$\frac{1}{D}(1 + \hat{\rho}) - \frac{1}{D}(1 - \hat{\rho}) = \frac{2d\ell\hat{\rho}}{D}. \tag{3.10}$$

Somando este valor à expressão ao valor da diferença entre eles no momento em que a cadeia de espera foi formada, conforme definido em (3.8), chegamos ao seguinte resultado:

$$L_{\ell+1}(t + \ell d) - L_{\ell}(t + \ell d) = \frac{2d\ell\hat{\rho}}{D} + c. \tag{3.11}$$

Este é o maior valor para a diferença entre os relógios lógicos de dois nós vizinhos, pois após o instante $t + d + \ell d$ o nó ℓ irá aumentar o seu relógio lógico, pois α_{ℓ} irá voltar para um e ele poderá atualizar o seu relógio de acordo com o Passo 3 do algoritmo. Uma garantia ainda mais forte é que o nó ℓ irá se aproximar do nó $\ell + 1$ de forma que estes fiquem a menos de c unidades de tempo de diferença e o nó $\ell + 1$ possa também aumentar o valor $\alpha_{\ell+1}$.

Devemos agora analisar os possíveis valores para a constante c. Considerando a expressão (3.11) devemos garantir que

$$c > \frac{2d\ell\hat{\rho}}{D}, \tag{3.12}$$

para que o nó ℓ se aproxime do nó $\ell + 1$ de tal forma que a distância entre eles fique menor do que c, liberando assim o nó $\ell + 1$ da cadeia de espera. Como veremos mais adiante o tamanho máximo para a nossa cadeia de espera é o diâmetro da rede, portanto podemos considerar $\ell = D$ em (3.12). Considerando a expressão (3.9) podemos perceber que se o valor de c for maior que $(1 + \hat{\rho})d$ teremos um valor menor do que D para a expressão $(1 + \hat{\rho})Dd/c$ o que iria limitar o tamanho máximo da cadeia de espera em um valor menor do que o diâmetro D da rede. Para evitar esta limitação, consideramos o valor $(1 + \hat{\rho})d$ como valor máximo para c. De acordo com estas duas limitações, chegamos à seguinte faixa de valores para a constante c:

$$2d\hat{\rho} < c \leq (1 + \hat{\rho})d \tag{3.13}$$

Utilizando um valor para a constante c de acordo com (3.13) garantimos que o nó ℓ vai se aproximar do nó $\ell + 1$ de tal forma que a diferença entre eles será menor que c. O nó $\ell + 1$ poderá então aumentar novamente o valor de $\alpha_{\ell+1}$, de uma maneira mais geral, garantimos que um nó sempre vai ser aproximar de seu vizinho mais rápido, fazendo com que este possa aumentar a sua relação entre taxa lógica e a taxa de *hardware*. Desta forma garantimos que o maior valor possível para a diferença entre dois nós vizinhos é expresso por (3.11) e de uma maneira mais geral podemos dizer que para qualquer par de nós i e j vizinhos durante toda a execução do algoritmo teremos

$$\begin{aligned} \left|L_i(t) - L_j(t)\right| &\leq \frac{2d\ell\hat{\rho}}{D} + c \\ &\leq \frac{2dD\hat{\rho}}{D} + c \\ &= 2d\hat{\rho} + c \\ &< 2c. \end{aligned} \tag{3.14}$$

Conseguimos assim um limite máximo constante para a diferença de pior caso entre os relógios lógicos de dois nós vizinhos, garantimos também que dois nós vizinhos sempre vão se aproximar novamente, pois o valor da constante c garante que após uma atualização de acordo com o Passo 3 a diferença máxima entre dois nós vizinhos vai ser reduzida para um valor menor do que c, o que permite que o nó mais rápido acelere seu relógio lógico novamente. O algoritmo garante a propriedade gradiente de acordo com a seguinte função

$$f(d_{ij}) = O(d_{ij}) \tag{3.15}$$

para todo $d_{ij} \in [1, D]$, dessa forma a diferença de pior caso para dois nós vizinhos é $O(1)$ e para dois nós quaisquer é $O(D)$. A diferença de pior caso entre dois nós com distância d_{ij} será dada pelo limite garantido em (3.14) multiplicado pela distância entre eles.

A cadeia de espera utilizada para mostrar a diferença de pior caso entre os relógios lógicos de dois nós vizinhos é suficiente pois o algoritmo garante que a diferença entre os relógios dos nós irá passar por c em algum momento durante a execução do algoritmo. Isto ocorre porque quando o relógio lógico é atualizado de acordo com o Passo 3 isto é feito de maneira instantânea de acordo com os pressupostos

(i) e (ii) e este passo garante que no máximo um nó vai aumentar o seu relógio lógico em c unidades de tempo com relação ao vizinho mais lento. A outra única forma de aumentar um relógio lógico é de acordo com a sua taxa de progresso, o que ocorre de forma contínua e também faz com que a diferença passe por c unidades de tempo.

Podemos utilizar os limites para a constante c expressos em (3.13) para desenvolver um pouco mais a expressão (3.9), poderemos mostrar então que o tamanho máximo para uma cadeia de espera é igual ao diâmetro da rede. Utilizando o maior valor para a constante c, para obter o maior valor possível para ℓ, chegamos então ao seguinte resultado

$$\begin{aligned} \ell &\leq \min\left\{D, \frac{(1+\hat{\rho})Dd}{c}\right\} \\ &= D\min\left\{1, \frac{(1+\hat{\rho})d}{c}\right\} \\ &= D\min\left\{1, \frac{(1+\hat{\rho})d}{(1+\hat{\rho})d}\right\} \\ &= D \end{aligned} \tag{3.16}$$

Garantimos assim que não existe um limite para o tamanho de nossa rede de espera, a não ser o diâmetro da rede na qual estamos aplicando o algoritmo. Dessa forma a nossa função que define a propriedade gradiente, já definida por $f(d_{ij})$ e cuja expressão vamos apresentar mais adiante, não satura de acordo com a distância entre dois nós, conforme acontece com o algoritmo A^{root} para distâncias maiores que $\sqrt{D+1}$.

A Figura 3.5 permite uma análise visual da diferença entre os relógios lógicos dos nós em uma cadeia de espera que pode ocorrer no algoritmo apresentado neste trabalho. Podemos verificar que não existe limite para o tamanho da cadeia de espera, pois esta pode possuir o mesmo tamanho do diâmetro da rede, e que no exato momento de formação da cadeia a diferença entre dois nós vizinhos é igual a c.

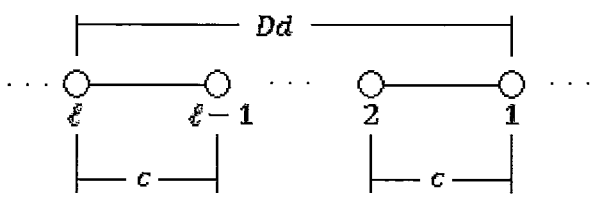


Figura 3.5 – Cadeia de Espera para o Novo Algoritmo

O valor $1/D$ utilizado para reduzir a taxa do relógio lógico de um nó com relação à taxa de relógio de *hardware* foi escolhido de acordo com o tamanho máximo de uma cadeia de espera que pode se formar durante a execução do algoritmo. Como vimos anteriormente esse tamanho máximo é igual a D, o que permite que no máximo um nó fique esperando por Dd unidades de tempo antes que possa atualizar o valor do seu relógio lógico através do algoritmo. Portanto se dividirmos a taxa lógica de um nó por D conseguimos compensar o tempo que este tem para se distanciar de um vizinho mais lento.

A redução para uma diferença constante entre nós vizinhos se dá da seguinte forma, considerando a mesma cadeia de espera utilizada acima, enquanto o nó j aguarda no máximo por Dd unidades de tempo até que possa atualizar o seu relógio lógico de acordo com o algoritmo, o seu vizinho i mais rápido poderia se afastar dele com a sua taxa máxima durante Dd unidades de tempo. Dividindo a taxa de i por D reduzimos esta diferença para um valor que independe do tamanho da rede, temos então uma diferença de $O(1)$ para nós vizinhos.

Concluímos assim o estudo das principais características do algoritmo, mostramos que a diferença de pior caso para nós com distância d entre si é igual a $O(d)$, mostramos que para nós vizinhos a diferença de pior caso é igual a $O(1)$. Também verificamos as taxas mínimas e máximas com que um relógio lógico pode possuir e analisamos a formação de uma cadeia de espera e que estas estruturas não possuem limites como no algoritmo A^{root}. Para finalizar mostramos que o nosso algoritmo respeita a propriedade gradiente através de uma função não-decrescente igual a $O(d)$ sem a utilização de mensagens destinas exclusivamente à sincronização.

3.3 Invalidade do limite inferior

O trabalho realizado em [1] provou que $f(d_{ij})$ é $\Omega(d_{ij} + \log D/\log\log D)$ para todos os algoritmos de sincronização de relógios que possuem a propriedade gradiente e que respeitem o limite inferior constante b para a taxa do relógio lógico de todos os nós com relação ao tempo real como vimos anteriormente. O mesmo foi mostrado no trabalho realizado em [2] para modelos que apresentam os pressupostos (i) e (ii) que são diferentes das características presentes no modelo de [1]. Esse limite inferior é contrariado pela diferença de pior caso entre nós vizinhos alcançada pelo nosso algoritmo. Vamos mostrar agora que as ações realizadas pelo nosso algoritmo não

permitem uma taxa mínima constante para a taxa de relógio lógico dos nós. Dessa forma o limite inferior em $f(d_{ij})$ não é válido para o algoritmo apresentado aqui.

Primeiramente vamos analisar a taxa de relógio lógico de um nó de acordo com as ações do algoritmo e com a sua definição apresentada em (3.1). Vamos verificar o que ocorre com a taxa lógica no exato momento em o valor de α_i é modificado e vamos verificar que esta mudança inviabiliza a existência de um limite inferior constante para a taxa lógica. Considerando um determinado valor de tempo real t, seja $t_1 < t$ e seja $t_2 > t$ de tal forma que o valor de α_i não é alterado nos intervalos $[t_1, t)$ e $(t, t_2]$. Podemos então definir o valor de $dL_i(t)/dt$ de acordo com

$$\lim_{t_1 \to t} \frac{L_i(t) - L_i(t_1)}{t - t_1} = \alpha_i \lim_{t_1 \to t} \frac{H_i(t) - H_i(t_1)}{t - t_1} = \alpha_i \frac{dH_i(t)}{dt} \tag{3.17}$$

ou

$$\lim_{t_2 \to t} \frac{L_i(t_2) - L_i(t)}{t_2 - t} = \alpha_i \lim_{t_2 \to t} \frac{H_i(t_2) - H_i(t)}{t_2 - t} = \alpha_i \frac{dH_i(t)}{dt}. \tag{3.18}$$

Se o valor de α_i mudar de 1 para $1/D$ exatamente no instante t devido ao recebimento de uma mensagem que indique que um dos vizinhos de i está atrasado com relação a ele, teremos $\alpha_i = 1$ no intervalo $[t_1, t)$ e $\alpha_i = 1/D$ no intervalo $(t, t_2]$. Substituindo esses valores nas expressões (3.17) e (3.18) teremos respectivamente

$$\lim_{t_1 \to t} \frac{L_i(t) - L_i(t_1)}{t - t_1} = \lim_{t_1 \to t} \frac{H_i(t) - H_i(t_1)}{t - t_1} \tag{3.19}$$

e

$$\lim_{t_2 \to t} \frac{L_i(t_2) - L_i(t)}{t_2 - t} = \frac{1}{D} \lim_{t_2 \to t} \frac{H_i(t_2) - H_i(t)}{t_2 - t} \tag{3.20}$$

O mesmo pode ocorrer caso o nó i esteja esperando por apenas um de seus vizinhos e receba uma mensagem deste, indicando que eles já estão próximos o suficiente, neste caso teremos $\alpha_i = 1/D$ no intervalo $[t_1, t)$ e $\alpha_i = 1$ no intervalo $(t, t_2]$ e os limites serão

$$\lim_{t_1 \to t} \frac{L_i(t) - L_i(t_1)}{t - t_1} = \frac{1}{D} \lim_{t_1 \to t} \frac{H_i(t) - H_i(t_1)}{t - t_1} \tag{3.21}$$

e

$$\lim_{t_2 \to t} \frac{L_i(t_2) - L_i(t)}{t_2 - t} = \lim_{t_2 \to t} \frac{H_i(t_2) - H_i(t)}{t_2 - t}. \tag{3.22}$$

Podemos verificar que os limites (3.17) e (3.18) são inconsistentes pois

$$\lim_{t_1 \to t} \frac{L_i(t) - L_i(t_1)}{t - t_1} \neq \lim_{t_2 \to t} \frac{L_i(t_2) - L_i(t)}{t_2 - t}. \tag{3.23}$$

Dessa forma não é possível determinar o valor de $dL_i(t)/dt$ no exato instante t. Através da Figura 3.2 podemos verificar a ocorrência destas indeterminações nos instantes $t = 20$, $t = 40$, $t = 60$ e $t = 80$.

Os instantes para os quais o valor da taxa lógica de um nó é indefinida ocorrem apenas quando este nó modifica o valor de α_i de acordo com as ações do Passo 2 do algoritmo. Durante o resto do tempo a taxa lógica pode ser definida de acordo com (3.1). O Passo 2 do algoritmo determina que o menor valor possível para a variável α_i é $1/D$ e de acordo com a expressão (1.4) o menor valor para a taxa de *hardware* de um nó é igual a $1 - \hat{\rho}$. Juntando esses dois valores na expressão para calculo da taxa lógica de um nó apresentada em (3.1) chegamos à expressão (3.7) que mostra o menor valor possível para a taxa lógica de um nó no algoritmo apresentado. Podemos então considerar que existe um limite inferior para a taxa lógica de um nó em nosso algoritmo e que o valor deste limite b é dado por

$$b \leq \frac{(1 - \hat{\rho})}{D}. \tag{3.24}$$

No entanto, este limite não é constante pois ele depende ao diâmetro da rede que varia de acordo com o grafo onde o algoritmo está sendo utilizado.

A expressão (2.27) que faz parte do lema *Add Skew* e tem origem na condição expressa em (2.9), que por sua vez é utilizado na prova do limite inferior apresentado em [1] utiliza a constante b, nesta expressão foi utilizado o valor $1/2$ para a constante, então podemos reescrever a expressão (2.27) da seguinte forma

$$L_j^{\alpha}(T) - L_j^{\alpha}(T') \geq b(T - T'). \tag{3.25}$$

A expressão (2.28) é derivada a partir da expressão (2.27), podemos então fazer a mesma alteração resultando na seguinte expressão

$$L_j^{\beta}(T') \leq L_j^{\alpha}(T) - b(T - T'). \tag{3.26}$$

A expressão (2.28) por sua vez é utilizada para derivar a expressão (2.29), realizando a mesma alteração nesta expressão chegamos a

$$L_i^{\beta}(T') - L_j^{\beta}(T') \geq L_i^{\alpha}(T) - L_j^{\alpha}(T) + b(T - T'). \tag{3.27}$$

Como vimos anteriormente, estas expressões são utilizadas para provar a expressão (2.20) do lema *Add Skew*. Substituindo (2.30) em (3.27) ao invés de utilizar a expressão (2.29) chegamos ao seguinte resultado

$$L_i^{\beta}(T') - L_j^{\beta}(T') \geq L_i^{\alpha}(T) - L_j^{\alpha}(T) + \frac{b}{6}(j - i). \tag{3.28}$$

Substituindo (3.24) em (3.28) teremos

$$L_i^{\beta}(T') - L_j^{\beta}(T') \geq L_i^{\alpha}(T) - L_j^{\alpha}(T) + \frac{(1 - \hat{\rho})}{6D}(j - i). \tag{3.29}$$

Como o valor de D é o diâmetro da rede, lembrando que um limite superior conhecido pelos nós é aceitável pelo algoritmo, essa expressão terá diferentes avaliações para diferentes grafos, mostramos assim que o limite inferior não é aplicável ao nosso algoritmo uma vez que este não garante um limite inferior constante para a taxa dos relógios lógicos durante a maior parte do tempo e em determinados instantes de tempo a taxa lógica é indefinida. Dessa forma a diferença de pior caso constante para nós vizinhos é possível.

Capítulo 4

Conclusões e Trabalhos Futuros

Ao longo deste trabalho apresentamos um algoritmo de sincronização com gradiente de relógios em redes de sensores. O algoritmo possui uma diferença de pior caso para nós com distância d entre si de $O(d)$ e de nós vizinhos de $O(1)$. Mostramos que o limite inferior apresentado em [1] não é valido para este algoritmo e estudamos um conjunto de algoritmos apresentados em [3] que também possuem a propriedade gradiente.

Os Passos 2 e 3 do algoritmo apresentado nesse trabalho possuem funções complementares. O Passo 2 do algoritmo é destinado a reduzir a velocidade dos nós cujos relógios lógicos estão a frente do relógio lógico de algum de seus vizinhos a mais do que uma determinada constante unidades de tempo. O Passo 3 força que um nó atualize o valor de seu relógio lógico de acordo com os valores recebidos de seus vizinhos sempre que possível.

O algoritmo sem o Passo 2 não iria atingir a diferença de pior caso de $O(1)$ não seria alcançada pois dois nós vizinhos poderiam possuir uma diferença entre seus relógios lógicos da ordem do diâmetro da rede. O Passo 3 é indispensável da mesma forma, uma vez que sem ele a presença de um único relógio lógico lento iria atrasar todos os outros relógios lógicos, o que os tornariam péssimas aproximações do tempo real. O Passo 3 também é responsável por aproximar nós que estão distantes a mais que a constante c unidades de tempo entre si.

O algoritmo é fortemente baseado na possibilidade de alterar a taxa com que os relógios lógicos acompanham os relógios de *hardware* dos nós. Apesar do Passo 2 do algoritmo garantir que esta relação volte para 1 em algum momento, ainda não sabemos com qual freqüência um nó irá reduzir a sua taxa lógica e se ele levará o tempo máximo determinado por uma cadeia de espera para aumentar esse valor novamente. A forma como a redução da taxa lógica afeta a sincronização dos relógios lógicos com o tempo real é outra característica que deve ser estudada. Estas análises, a implementação do algoritmo para estudo de desempenho e o desenvolvimento de uma versão do algoritmo para sincronização externa, que pode ser feita através de um sistema GPS, ficam como sugestões de trabalhos futuros.

Referências Bibliográficas

[1] FAN, R., LYNCH, N., "Gradient clock synchronization", *Distributed Computing*, v.18, pp. 255-266, 2006.

[2] MEIER, L., THIELE, L., "Brief announcement: gradient clock synchronization in sensor networks", In: *Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing*, pp. 238, 2005.

[3] LOCHER, T., WATTENHOFER, R., "Oblivious gradient clock synchronization", In: *Proceedings of the Twentieth Internation Symposium on Distributed Computing 4168 of Lecture Notes in Computer Science*, pp. 520-533, Springer-Verlag, Berlin, Germany, 2006.

[4] QI, H., WANG, X., IYENGAR, S., et. al., "Multisensor data fusion in distributed sensor networks using mobile agents", In: *Proceedings of the Internation Conference on Information Fusion*, pp. 11-16, 2001.

[5] ELSON, J., GILROD, L., ESTRIN, D., "Fine-grained network time synchronization using reference broadcasts", *Operation Systems Review*, v. 36, pp. 147-163, 2002.

[6] LAMPORT, L., MELLIAR-SMITH, P. M., "Synchronizing clocks in the presence of faults", *Journal of the ACM*, v. 32, pp. 52-78, 1985.

[7] SRIKANTH, T. K., TOUEG, S., "Optimal clock synchronization", *Journal of the ACM*, v. 34, pp. 626-645, 1987.

[8] WELCH, J. L., LYNCH, N., "A new fault-tolerant algorithm for clock synchronization", *Information and Computation*, v. 77, pp. 1-36, 1988.

[9] WELCH, J. L., LYNCH, N., "An upper and lower bound for clock synchronization", *Information and Control,* v. 62, pp. 190-204, 1984.