

Uma representação de *Bluetooth* usando algoritmos distribuídos e grafos randômicos

Célio Márcio Soares Ferreira, Ricardo Augusto Rabelo Oliveira
PPGCC - Programa de Pós-Graduação em Ciência da Computação
UFOP - Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brasil
email: celio@linuxplace.com.br, rrabelo@gmail.com

Resumo—O cenários de redes sem fio de curto alcance que possuem como características, conexão ponto multi ponto, comunicação em salto de frequência, para minimizar interferências de outras redes, e sincronização de canal usando o *clock* do nó mestre, pode ser representado pela figura dos grafos randômicos. Onde os vértices caracterizam os nós da rede, as arestas os links e a presença ou possibilidade de conexão entre eles. Onde esta representação do ambiente de conexão e mapeamento de seu *layout*, apontando ausência ou presença de conexão, nós ativos e inativos e sua capacidade de troca de mensagens, são representados. Em nossa pesquisa mostramos detalhes desta negociação inicial, durante a estabilização de conexão com os nós adjacentes em seu *range* de comunicação. Esta demonstração ajudará a entender os motivos do tempo excessivo durante a fase de *discovery* de nós, e propor novos cenários e alternativas ao processo de *inquiry*. Utilizando o *framework* Daj, mostramos utilizando algoritmos distribuídos, a complexidade das trocas de mensagens, detalhando as fases envolvidas na descoberta e conexão de novos pontos, ilustrada e comparada em uma tecnologia prática como o *bluetooth*.

Keywords—computação móvel, algoritmos distribuídos, grafos randômicos, *mobile*, algoritmos, complexidade, *Bluetooth*.

I. INTRODUÇÃO

O uso de dispositivos móveis vem ganhando cada vez mais espaço no cotidiano. Um visível crescimento na comercialização e fabricação destes dispositivos vem levando a uma rápida disseminação e adesão em quase todos os níveis de nossa sociedade. Boa parte deste fenômeno comercial e tecnológico, advém da grande popularização no uso diário de telefones celulares, *smartphones* e recentemente *tablets*. As promessas futuristas de antes, nas quais teríamos no futuro eletrodomésticos, games, automóveis, roupas e monitoramento de suporte a vida totalmente capazes de se comunicar sem cabos já se tornaram realidade.

Frente a este quadro, surge cada vez mais a necessidade de simularmos novos cenários e seus mais variados problemas típicos das redes sem fio. A característica pervasiva destes dispositivos vem levando a um surgimento de tecnologias e técnicas cada vez mais complexas e apuradas, afim de propiciar técnicas mais apuradas de gerenciamento, monitoramento do comportamento destes dispositivos.

Ao tentar conectar os mesmos dispositivos, via rede sem fio *bluetooth*, em muitos casos ocorrem latências e as vezes até mesmo a não identificação de um dispositivo durante a requisição de conexão. Surgem então indagações como quem

tentou conectar primeiro? Quem conseguiu autenticar? Em qual ordem?

Questões estas, se tornam cruciais em situações que a conexão entre equipamentos fazem o tráfego de informações de segurança, mas não podem ser respondidas somente com uso de uma tecnologia somente. [1] recomenda o uso do infra-vermelho para identificar os elementos da comunicação. [2] recomenda o uso do RFID para a identificação por proximidade. Ambos os casos são usados para lidar com este problema na tecnologia *bluetooth*.

Neste ponto entra a relevância de testar cenários alternativos ao atual processo de conexão destes dispositivos, por meio de simulação da tecnologia, usando algoritmos distribuídos e grafos randômicos.

Neste artigo será abordada a modelagem e simulação do comportamento de redes de curto alcance que tenham como característica a negociação de canal inicial, mais conhecido como *frequency hopping* – salto de frequência, e e que possuam características de conexão *ad hoc*, como o *Bluetooth* e o *Zigbee*.

Quando modelamos este ambiente em um grafo, temos cada dispositivo como um vértice e as arestas como os alcances, $G(V, A)$, no qual $|V| = n$ e $|A| = m$, no qual o grau de cada vértice n_i é definido como o tamanho conjunto de nós que estão dentro do raio de alcance da comunicação de n_i

Levando para nosso objetivo que é simular o processo de *discovery* de uma rede *ad hoc* com comunicação de salto de frequência, teremos uma representação no simulador de grafos contendo os nós no *range* de alcance e dos nós que conseguiram fazer algum dos estados da máquina de estado *bluetooth*: *inquiry*, *page* e *connect*.

O processo de *discovery* será simulado pelo Daj, [3] um *framework* de modelagem de rede. Nele será reproduzido o modelo proposto para um novo cenário de *inquiry* ao *Bluetooth*. Este cenário será modelado em algoritmos distribuídos.

A análise de complexidade dos algoritmos distribuídos difere dos algoritmos convencionais, nos quais além dos tradicionais tempo e espaço, precisa ser analisada a quantidade de troca de mensagens até que o algoritmo entre em convergência e parem sua execução. A complexidade de troca de mensagens fica em função de n dispositivos e m pontos de conexão entre estes. Levando para o ambiente

de rede sem fio o m é o alcance da rede no qual se existe m , existe comunicação.

Como objetivo específico, o algoritmo de conexão do *Bluetooth* será modelado, sua complexidade analisada.

O artigo está organizado da seguinte forma. A Seção II apresenta um *overview* da tecnologia *Bluetooth*, a Seção III detalha a máquina de estado *Bluetooth* em seu processo de *discovery* de pontos, na Seção IV a análise de complexidade do grafo dinâmico gerado pelo *Bluetooth* é detalhada, a Seção V descreve como simulamos o processo de *inquiry discovery* do *Bluetooth* no *framework* de simulação de algoritmos distribuídos Daj.

II. O Bluetooth

Bluetooth é uma tecnologia que usa rádio frequência para gerar redes de curto alcance, apresentando uma substituição aos tradicionais cabos que interligam dispositivos e periféricos. Sua frequência utiliza a popular banda 2.4 GHz ISM (*Industrial, Scientific, Medical*), e possui velocidades de transferência superiores a 1 Mbps.

Seu ambiente de conexão usa espectro espalhado em salto de frequência, tendo um alcance de conexão de 10 metros. Sua rede de característica tem o nome de piconet, na qual no máximo 8 dispositivos podem estar conectados, um mestre e sete escravos. Cada piconet é definida por um padrão de salto de frequência, este definido pelo mestre e os outros dispositivos que estão usando este padrão para comunicação. A Figura ?? mostra uma piconet. O vértice vermelho é o mestre da rede, os verdes são os escravos em comunicação, os rosa parados e os amarelos fora do alcance.

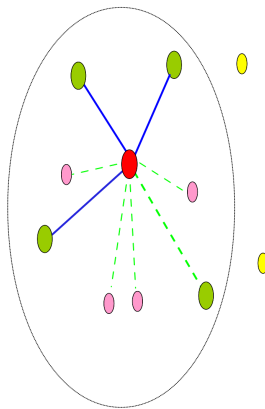


Figura 1. Bluetooth [4]

Possui característica ponto-multi ponto na relação entre os dispositivos *Bluetooth*, onde qualquer um pode funcionar como mestre ou escravo na Piconet. O uso de salto de frequência vem da necessidade de diminuir a interferência de outras redes e de permitir seu trabalho sem visada direta em ambientes nos quais barreiras físicas são uma realidade, dando uma melhor flexibilidade no seu uso em variadas

distâncias e direções. A Figura 2 mostra a topologia esperada de uma piconet.

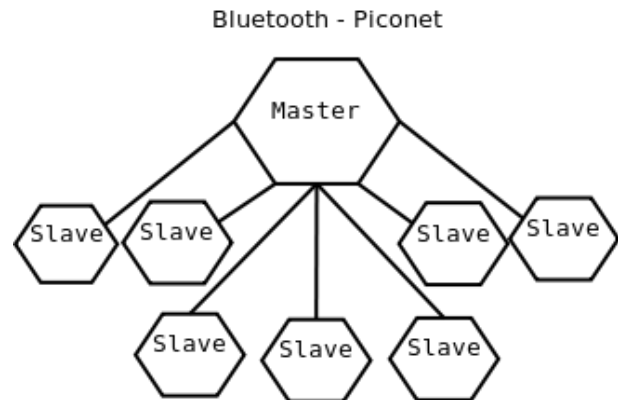


Figura 2. Piconet

Quando dispositivos como fones de ouvido, mouses, teclados, etc, se conectam a um computador ou até mesmo durante a conexão de dois computadores em modo *ad hoc*, observamos que existe um certo controle visual do que está acontecendo durante a conexão, possibilitando a percepção de quem parou ou quem conectou. Muitas vezes, demora-se um certo tempo até que os elementos comunicantes estejam conectados e a validação dessa conexão pode ser demorada [1]. Este processo pode atrapalhar o resultado final da comunicação, que é uma rápida resposta ao usuário.

Na seção a seguir será explorado o processo de descoberta dos elementos da rede e seu impacto na comunicação.

III. O PROCESSO DE Discovery

O processo de estabilização de conexão do *Bluetooth* inicia com uma fase chamada *inquiry*, no qual o dispositivo pesquisa por outros dispositivos que estão em modo de espera de conexão. No processo de *inquiry*, o dispositivo que procura outros envia um código chamado IAC, – *Inquiry Access Code*. Este código define o tipo de dispositivo que está sendo procurado. Um exemplo deste processo, é quando colocamos um *smartphone* em uma busca por "*Human Interfaces Devices*", neste processo só responderiam ao *inquiry* dispositivos de viva voz automotivos e fones de ouvido disponíveis no alcance da transmissão. A Figura 3 mostra a máquina de estados da conexão do *Bluetooth*.

Quando um dispositivo *Bluetooth* precisa descobrir um novo dispositivo, este entra no estado de *Inquiry*, onde este envia em todo o espectro de frequência os quadros de *inquiry* contendo IAC, para todos os dispositivos em seu alcance. Cada envio acontece em um canal de frequência.

Durante todo o *inquiry*, o dispositivo efetua um procedimento chamado *Inquiry Hopping Sequence* que cobre um conjunto de 32 canais dos 79 disponíveis, efetuando um salto a cada 312 μS . O dispositivo gera o *inquiry* e envia mensagem em cada canal definido nesta sequência de salto.

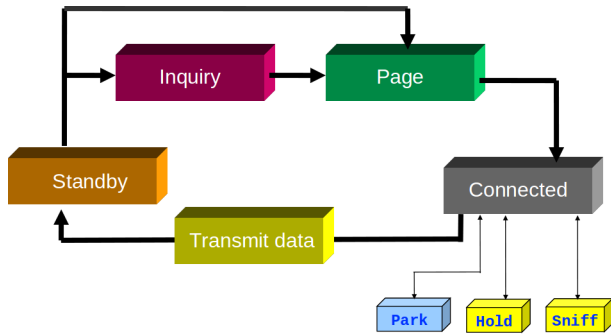


Figura 3. Máquina de Estado Bluetooth [4]

Esta sequência de salto é derivada de um cálculo usando o IAC e seu clock.

Os dispositivos que estão em modo de descoberta, entram periodicamente em um sub-estado chamado *Inquiry Scan*. Neste estado os dispositivos saltam de frequência quando a mesmo critério de salto de frequência do estado de inquiry descrito anteriormente usando o IAC e o clock local do dispositivo.

Podemos ilustrar este estado em um cenário onde dispositivos Bluetooth, como joysticks de videogames, precisam ser descobertos pelo console, durante este processo os joysticks são colocados em estado de descoberta *inquiry scan*, aguardando conexão. Quando um dispositivo em *inquiry scan* recebe uma mensagem de *inquiry* este entra no subestado de *inquiry response* e replica com uma mensagem de *inquiry response*. Esta mensagem inclui o endereço do dispositivo e o seu *textclock*, informações necessárias para que sejam estabelecidas conexões.

Todos estes dispositivos em *inquiry scan* que estão a uma distância teórica de 10 metros respondem ao dispositivo que fez o *inquiry*. Ao receber a resposta, entramos em uma etapa na qual o tempo gasto depende da tarefa de interação do usuário do dispositivo. De posse da lista de dispositivos descobertos, é preciso que o usuário manualmente selecioná-lo, dedicando indeterminado custo de tempo. Quando é selecionado o endereço de rede Bluetooth correspondente ao dispositivo que deseja conectar, este começa a enviar mensagens de *page*.

No estado de *page*, o dispositivo gera uma mensagem estimada no clock do dispositivo remoto, enviando repetidamente no salto de frequência definido. Se este dispositivo quer habilitar outros dispositivo que o tem cadastrado na lista de dispositivos o encontre e estabeleça conexão, precisa entrar em estado de *page scan*.

No subestado *page scan* a sequência de saltos de frequência também é baseado no clock do dispositivo. O ajuste do salto de frequência ocorre por meio de troca de mensagens entre os dispositivos mestre e escravo, sendo esta uma característica do salto de frequência. Neste tipo de rede é necessária a constante sincronização de clock dos nós

escravos com o mestre para existir conexão.

Quando um dispositivo escravo recebe uma mensagem de *page*, este responde ao mestre um *page response*, que responde com um pacote de sincronização de frequência. No conteúdo deste pacote temos o endereço *bluetooth* e o *clock* do mestre.

Após esta sincronização, o escravo então envia um *ack* confirmando recebimento para o mestre, e este regeira uma nova sequência de salto com seu próprio *clock* e endereço. O escravo usa o *clock* e endereço do mestre para gerar a sequência de saltos idêntica ao mestre. Esta sequência idêntica de saltos de frequência permite que os nós se conectem.

Uma vez completo o processo de *page*, o dispositivo passa para o estado de conectado. O mestre envia um *pool* de mensagens para o escravo verificando de a sequência de salto definida no estado de *page* foi bem sucedida. Se ocorre sucesso os dois *nodes* começam a comunicar em um pseudo randômico salto de frequência baseado no endereço e *clock* do mestre.

IV. ANÁLISE DE COMPLEXIDADE

A rede Bluetooth será modelada como um grafo, no qual os dispositivos são representados por n vértices, cuja coordenadas são randomicamente escolhidas dentro da periferia de cada *node* selecionado $c(n)$ vizinho entre todos os *nodes* vizíveis, que está, entre todos os *nodes* dentro de uma distância euclidiana $r(n)$, onde $r(n)$ define o *range* de visibilidade, que é assumido ser o mesmo entre todos os dispositivos.

Este grafo pode ser empregado como um modelo de uma rede real de características *ad hoc* onde *nodes* são forçados a manter pequeno número de conexões simultâneas, devido a limitado recursos energéticos e computacionais, ou onde estabelecer links com cada *node* também possui um custo alto de energia e tempo.

A complexidade do envio de mensagens será proporcional a troca de mensagens em cada algoritmo. Sendo d o grau do vértice, cada vértice do grafo contribuirá com $(\theta(d))$. O meio mais eficiente de comunicação numa rede seria ter cada vértice n_i efetuar um *broadcast*, eliminando o custo de comunicação ponto a multi-ponto, no alcance de forma que o número de mensagens transmitidas em toda rede a cada envio fosse $\theta(n)$. Contudo, isso gera uma quantidade de colisões e interferências que o torna proibitivo.

Segundo [5] podemos definir formalmente a topologia do Bluetooth, como: dado um inteiro positivo n e uma função de números reais $r(n) - (0, \sqrt{2})$ e uma função de inteiros positivos $c(n)$, a topologia do Bluetooth, denotada por $BT(r(n), c(n))$, é um grafo randômico não direcionado $G = (V_n, E_n)$, definido como:

- O conjunto de *nodes* $V(n)$, $|V| = n$, com distribuição espacial descrita por uma variável aleatória N , uniformemente distribuída em $[0, 1]^2$

- O conjunto de arestas $E(n)$, obtidas por meio do seguinte processo: independentemente cada *node* n_u seleciona um conjunto aleatório de $c(n)$ vizinhos entre todos os *node* dentro da distância $r(n)$. Todos em seguida se eles são menores que $c(n)$. Uma aresta $(n_u, n_v) \in E_n$ existe se e somente se n_u selecionou n_v , e vice versa.

Em [6], é mostrado que para qualquer constante $r > 0$ existe uma (larga) constante c tal que $GR(r, c)$ possui com alta probabilidade de conexão de rede. Em [7] o autor mostra que para qualquer constante fixa $r > 0$ e $c \geq 2$ sempre que n se torna suficientemente grande.

Nós dizemos que dois nós se comunicam, se eles estão dentro do range $r(n)$. Considerando o *tesselation* padrão de $[0, 1]^2$ dentro de células quadradas k^2 de tamanho $\frac{1}{k}$ onde $k = \lceil \frac{\sqrt{5}}{r(n)} \rceil$. Consequentemente qualquer dois nós residindo na mesma célula ou em duas adjacentes estão a uma distância no máximo de $r(n)$, por isso comunicam. Caracterizamos assim uma célula como um conjunto de *nodes* ali residentes.

Um evento acontece com alta probabilidade se pelo menos $1 - \frac{1}{poly(n)}$. Fazemos que $m = \frac{n}{k^2} = \theta(nr^2(n))$ sejam o número de *nodes* esperados para residir uma célula.

Fazendo $\alpha = \frac{9}{10}, \beta = \frac{11}{10}$. Existe uma constante $\gamma_1 > 0$ tal que para cada $r(n) = \gamma_1 \sqrt{\log \frac{n}{n}}$ os seguintes dois eventos ocorrem.

- cada célula contém no mínimo αm e no máximo βm *nodes*
- cada *node* tem, ao menos $(\frac{\alpha}{4})\pi nr^2(n)$ e no máximo $\beta\pi nr^2(n)$ *nodes* no range de visada
- cada aresta $e \in E$ possui um valor ϵ_t associado, representando a frequência utilizada na comunicação.

A definição da vizinhança dado um conjunto de vértices $X \subseteq V$ é $\Gamma(X) = \{u \in V(G) : \exists e = \{u, v \in E(G), v \in X\}\}$.

O diâmetro de G , definido com $diam(G)$, é o máximo de distância entre qualquer dois *nodes* $u, v \in V$, onde a distância entre os dois *nodes* é o número de arestas do caminho mais curto entre eles.

Dada essa definição, o grafo BT pode ser classificado como um grafo dinâmico. Um grafo dinâmico é um grafo no qual alguma propriedade p de um dado grafo $G = (V, E)$ é considerada verdadeira. As arestas do grafo G não são fixas. Elas podem ser inseridas ou retiradas durante a verificação de p . O algoritmo que mantém essa propriedade classifica os vértices e arestas em diferentes estados, com operações que alteram estes estados. Estas operações definem um modelo dinâmico de arestas, as quais podem ser retiradas e inseridas em E . Esta inserção representa o estabelecimento de conexão.

O processo de discovery pode ser definido como a operação $Disc(n_u, n_v, w_e)$, operação de inserir uma aresta (n_u, n_v) em $BT=(V, E')$, com o custo w_e de encontrar do

salto de frequência correto para a conexão. A propriedade a ser verificada é se a inserção desta aresta é verdadeira. Dada a execução distribuída, essa propriedade será verificada se $Disc(n_v, n_u, w_e)$, a inserção da aresta (n_v, n_u) em $BT=(V, E')$, acontecer, significando que n_u e n_v estão executando a operação de maneira síncrona. A operação $Disc(n_u, n_v, w_e)$ não executa em tempo linear, uma vez que a busca do salto de frequência segue uma variável aleatória X .

Considerando esta variável aleatória, o custo de troca de mensagens em cada execução de $Disc$ é definido por $O(E(X) * (\theta(d_u)))$, onde $\theta(d_u) = \frac{\sqrt{5}}{r(n)}$ é o grau do vértice n_u .

Devido ao impacto da variável aleatória na complexidade final, neste trabalho iremos investigar os efeitos da aleatoriedade do salto de frequência na complexidade de troca de mensagens.

V. MÉTODOS

Em todos os cenários de testes foi usado o framework DAJ. Segundo própria definição, ele é um conjunto de ferramentas, que permite o desenho, implementação, teste, simulação e visualização de algoritmos distribuídos em linguagem Java.

Descreveremos a tecnologia *Bluetooth* e identificaremos seus problemas por meio de algoritmos distribuídos. Em um algoritmo distribuído consideramos todos pontos de execução como iguais sem distinção, em um ambiente onde todos podem enviar mensagens para todos. Todos processam o mesmo algoritmo, esperando em um rede que os pontos estejam com o mesmo código, possui como forma de distinção, identificadores individuais como descrito em [8].

O DAJ possui uma biblioteca de classes Java que permitiu a nossa simulação da máquina de estado do *Bluetooth* em diversos cenários, usando algoritmos distribuídos em um modelo de passagem de mensagem, possibilitando tirar conclusões do resultado dos testes e simulações próximas do mundo real.

Em sua interface vemos nós que são ligados por canais e que operam em modo assíncrono e de forma independente, o mecanismo base para uma comunicação ponto a ponto e que possibilite envio e recebimento de mensagens pelo conjunto de canais. Cada nó tem um relógio local para limitar tempos de bloqueio e outras operações necessárias em ambiente de simulação necessária durante a implementação. A Figura 4 mostra uma simulação em execução.

Em seu *frontend*, visualizamos a rede e os nós, o estado dos nós e canais são indicados por cores e suas mensagens em caixas de diálogo e notas de rodapé. No centro do nó e no canto esquerdo superior temos a hora local em tempo cronológico a partir do início da simulação.

Para implementação da máquina de estado do *Bluetooth*, foram criadas as seguintes classes: *BluetoothFrame*, *Prog*, *SimpleLog* e *CountMsg*.

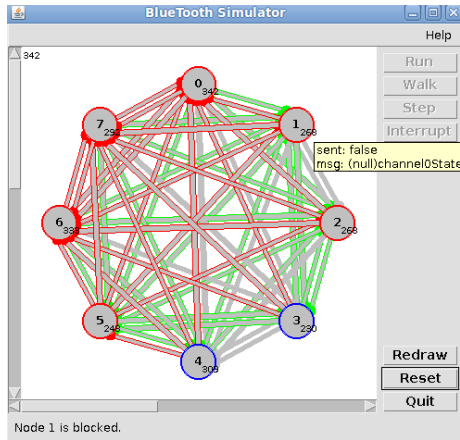


Figura 4. DAJ - Distributed Algorithms Java Simulator

A classe `BluetoothFrame` estende a classe `Message` do Daj e é usado para simular a mensagem *Bluetooth* em todos seus estados. Nesta classe é definido por meio das variáveis

- `state`, assume nas mensagens os valores correspondentes ao tipo correspondente aos estados até a fase de *connected* do *Bluetooth*, `INQUIRY=1`, `INQUIRYSCAN=2`, `PAGE=3`, `PAGESCAN=4`, `CONNECTED=5`, `ERRO=666`;
- `channel`, recebe o numero do canal em que a mensagem foi transmitida;
- `mac`, carrega a identificação individual do nó;

A classe é descrita em resumo por `bluetoothFrame(state, FH, mac)`, simulando o `frame` do *Bluetooth*, e sendo responsável pela mensagem que cada nó envia para controle de erro, casamento de canal, troca de estado, e nome do dispositivo.

A classe `Prog` é onde está implementado o algoritmo distribuído, onde o código manipula e toma decisões de acordo com o conteúdo das mensagens que entram e saem dos nós. Segue o corpo do algoritmo que simula a máquina de estado do *Bluetooth*.

`PROG(Node)`

```

1  channelSet
2  state // INQUIRY=1,
3  // INQUIRYSCAN=2, PAGE=3,
4  // PAGESCAN=4, CONNECTED=5
5  FH
6  bluetoothFrame = bt
7  mac
8  Disc(Node)
9  while state == 1
10     FH = ChannelRand(1..79)
11     for i = 0 to TotalChannelsOut
12         OUT.SEND(BLUEOOTHFRAme(1, FH, mac))
13
14     while state < CONNECTED
15         RECEIVEALL()

```

Foram simulados três cenários e colhidos dados de volume de mensagens e estados dos nós para os experimentos.

O primeiro cenário, o caso base, foi inicializado com 8 nós para simular o comportamento de uma *piconet*. Para simular a *piconet* foi iniciado o Nó 0 como único mestre da rede, inicializado com a variável `state` em 1, ou seja em estado de *inquiry*. Os outros 7 nós simulando nós escravos foram iniciados com variável `state` em 2, *inquiry scan*.

O nó 0 então inicializa a variável `fh` como o numero correspondente ao salto de frequência, e em seu papel de mestre envia para os canais de saída uma mensagem contendo o `bluetoothFrame(state, FH, mac)`, por exemplo `frame Bluetooth` enviando estado `state1` com salto de frequência `FH 32` e `macaddress 4`. Este procedimento é tratado como um *broadcast* e é recebido por todos os outros 7 nós escravos, que neste cenário são inicializados como `state = 2` ou *inquiry scan*.

Após o envio das mensagens de *inquiry* pelo mestre, os nós escravos em *inquiry scan* recebem em seus canais de entrada a mensagem de *inquiry*, e verificam se o salto enviado na mensagem é correspondente ao salto local `fh`.

Assim que algum dos nós escravos em estado de *page scan* coincidem o salto de frequência definida pela variável `fh` com a mensagem de *inquiry* muda para o novo estado *page scan* e envia uma mensagem de confirmação de volta para o mestre.

O mestre em estado de *inquiry* ao receber a mensagem de *page*, troca o estado para *page*, enviando uma mensagem com salto de frequência definido.

Ao receber a mensagem de *page*, o escravo em *page scan* troca seu `state` para *connected*, e envia a mensagem *page scan* de volta para o mestre. O mestre em *page* troca seu `state` e assim estabelece a conexão.

VI. EXPERIMENTOS

Como descrito na seção IV, o estudo da variável aleatória envolvida no processo de descoberta do *Bluetooth* afeta a

Tabela I
TEMPO DE EXECUÇÃO EM CICLOS DO DAJ - MELHOR CASO

| Experimento | Média | Desvio Padrão | Máximo | Mínimo |
|---------------------------|-------|---------------|--------|--------|
| Distribuição Normal (79) | 1025 | 11061 | 4712 | 35 |
| Distribuição Normal (32) | 614 | 603 | 2482 | 23 |
| Distribuição Poisson (79) | 997 | 1408 | 6755 | 18 |

Tabela II
TEMPO DE EXECUÇÃO EM CICLOS DO DAJ - CASO MÉDIO

| Experimento | Média | Variância | Máximo | Mínimo |
|---------------------------|-------|-----------|--------|--------|
| Distribuição Normal (79) | 460 | 330 | 1339 | 36 |
| Distribuição Normal (32) | 342 | 356 | 1643 | 27 |
| Distribuição Poisson (79) | 579 | 477 | 2092 | 42 |

Tabela III
NÚMERO DE CONEXÕES - MELHOR CASO

| Cenário | Número de Conexões (média) | Quantidade de Pacotes | Quantidade de Mestres | Quantidade de Escravos |
|---------------------------|----------------------------|-----------------------|-----------------------|------------------------|
| Distribuição Normal (79) | 1.03 | 2193 | 1 | 7 |
| Distribuição Normal (32) | 1.03 | 1436 | 1 | 7 |
| Distribuição Poisson (79) | 1 | 2254.06 | 1 | 7 |

Tabela IV
NÚMERO DE CONEXÕES - CASO MÉDIO

| Cenário | Número de Conexões | Quantidade de Pacotes | Quantidade de Mestres | Quantidade de Escravos |
|---------------------------|--------------------|-----------------------|-----------------------|------------------------|
| Distribuição Normal (79) | 1.16 | 2280 | 4.33 | 3.13 |
| Distribuição Normal (32) | 1 | 1856.93 | 3.96 | 4.03 |
| Distribuição Poisson (79) | 1.13 | 3247 | 3.93 | 4.06 |

complexidade das mensagens enviadas. Iremos considerar três tipos de distribuições. A primeira, segue uma distribuição normal, que escolhe um dos 79 canais de frequência. A segunda, segue uma distribuição normal, seguindo a definição do *Bluetooth*, descrita na seção III e uma terceira, na qual foi experimentado a distribuição de Poisson, sobre os 79 canais.

Para cada um dos cenários foram enumeradas três possibilidades:

- Sem conexão: uma situação na qual todos os dispositivos são mestres ou todos escravos. Como este caso nunca acontecerá conexão, não será testado, apenas identificado;
- Chance de uma conexão: um dispositivo está configurado como mestre, os outros como escravos. Neste caso, existe a certeza de pelo menos uma conexão;
- Possibilidade de conexão: situação na qual os dispositivos podem ser mestres e escravos. A escolha é feita aleatoriamente. Este caso pode ser considerado próximo da realidade, uma vez que não existe uma regra que define qual dispositivo será mestre ou escravo.

Como situação de convergência, serão contabilizados

quantos ciclos foram necessários para que a conexão acontecesse. Os testes foram rodados 30 vezes, gerando um intervalo de confiança de 90%. A Tabela I mostra a quantidade de ciclos necessários para a convergência no melhor caso. Em termos de melhor convergência, a implementação do *Bluetooth* foi melhor do que os dois casos, como pode ser visto pela menor média e menor desvio padrão. Usando somente a distribuição normal, a convergência foi pior, com maior média e maior desvio padrão. O uso da distribuição poisson mostrou um resultado melhor para a convergência.

A tabela II mostra os tempos de execução em situações onde existe possibilidade de conexão, na qual a quantidade de conexão é proporcional a quantidade de mestres e escravos. Neste caso, a distribuição normal apresentou melhor resultado que a poisson.

Nas tabelas III e IV temos os números de conexões estabelecidas. No melhor caso, o uso da distribuição normal mostrou a melhor quantidade de pacotes e conexões.

VII. CONCLUSÃO

Neste trabalho apresentamos uma formalização e caracterização do processo de descoberta e conexão do *Bluetooth*. A rede *Bluetooth* foi modelada como um grafo dinâmico, o

qual tem regras de construção em suas arestas. A partir disso, foi desenvolvido um modelo de análise de complexidade da troca de mensagens no processo de conexão. Foi observado o impacto da variável aleatória no cálculo da complexidade de troca de mensagens.

Uma possível conclusão é que o uso de estratégias que minimizem essa variável aleatória, como por exemplo IrDa e RFID, a quantidade de mensagens usadas na rede diminuíra.

REFERÊNCIAS

- [1] R. Woodings, D. Joos, T. Clifton, and C. Knutson, "Rapid heterogeneous ad hoc connection establishment: accelerating bluetooth inquiry using irda," in *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, vol. 1, mar 2002, pp. 342 – 349.
- [2] T. Salminen, S. Hosio, and J. Riekk, "Enhancing bluetooth connectivity with rfid," In *proceedings of IEEE International Conference on Pervasive Computing and Communications*, vol. 0, pp. 36–41, 2006.
- [3] W. Schreiner, "Daj - a toolkit for the simulation of distributed algorithms in java," <http://www.risc.jku.at/software/daj>.
- [4] S. M. Farinaz Edalat, Ganesh Gopal, "Bluetooth technology," <http://pt.scribd.com/doc/48331364/16645836-Bluetooth>.
- [5] A. Pettarin, A. Pietracaprina, and G. Pucci, "On the expansion and diameter of bluetooth-like topologies," in *Algorithms - ESA 2009*, ser. Lecture Notes in Computer Science, A. Fiat and P. Sanders, Eds. Springer Berlin / Heidelberg, 2009, vol. 5757, pp. 528–539.
- [6] A. Panconesi and J. Radhakrishnan, "Expansion properties of (secure) wireless networks," in *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, ser. SPAA '04. New York, NY, USA: ACM, 2004, pp. 281–285. [Online]. Available: <http://doi.acm.org/10.1145/1007912.1007959>
- [7] D. Dubhashi, C. Johansson, O. Häggström, A. Panconesi, and M. Sozio, "Irrigating ad hoc networks in constant time," in *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, ser. SPAA '05. New York, NY, USA: ACM, 2005, pp. 106–115. [Online]. Available: <http://doi.acm.org/10.1145/1073970.1073986>
- [8] V. C. Barbosa, *An introduction to distributed algorithms*. Cambridge, MA, USA: MIT Press, 1996.