Proposal of Fibonacci *Heap* in the *Dijkstra* Algorithm for Low-Power *Ad-Hoc* Mobile Transmissions

T. Silveira, E. Duque, S. Guimarães, H. Marques-Neto, *Member, IEEE,* and H. Freitas, *Member, IEEE*

***Abstract*—Mobile devices have become indispensable to communication over the last decade. In a hypothetical scenario in which conventional forms of connection such as antennas and satellites are not available, other forms of information propagation must be used. *Ad-Hoc* broadcasts are strategies for maintaining communication between devices in these situations, however, they require high transmission power. This article proposes the use of priority queues, such as the Fibonacci *Heap* and the binary *Heap* in the *Dijkstra* algorithm, in order to reduce its computational cost in the search for the smallest network route. From the limitation of the signal strength to reach the nearest device, our results obtained lower transmission power compared to the standard device settings. Simulations show that a fibonacci *Heap* has higher performance than binary *Heap* in networks with higher number of connections. This way, the implementation of the fibonacci *Heap* brings improvements in the computational cost of the algorithm. In addition, we show that the calculation of the smallest route is directly connected to the choice of the path with the lowest transmission power.**

***Index Terms*—*Dijkstra*, Fibonacci heap, *Ad-Hoc* transmissions, Opportunistic routing, Transmission power.**

I. Introdução

REDES sem fio *Ad-Hoc* são redes que permitem a intercomunicação de dispositivos sem a necessidade de um nó centralizador. Por esse motivo, podem ser utilizadas em sistemas de tráfego a fim de receber e compartilhar informações entre veículos e também no cenário militar, permitindo a comunicação de entre unidades e soldados [1] [2]. Em um cenário de desastre em que serviços convencionais de rede estão indisponíveis, o uso de redes *Ad-Hoc* é uma solução para manter a comunicação entre os dispositivos que necessitam de comunicação [3]. O estudo realizado neste trabalho é baseado em um contexto de desastre.

Redes de comunicação em cenários de desastre podem ser muito suscetíveis a alterações em sua estrutura, onde a exclusão de vértices pode significar o isolamento de *Clusters* inteiros, inviabilizando a comunicação dentro da rede. Transmissões *Wi-fi* via dispositivos móveis possuem alto consumo de bateria [4]. Em uma situação em que cada dispositivo desempenha papel crucial na transmissão dos dados pela rede, reduzir ao máximo o consumo energético é importante para maximizar o tempo útil de bateria [5], uma vez que o tempo de permanência ativa de um nó é de extrema importância para a integridade da rede de transmissão.

Algoritmos para determinar a menor rota em grafos visam trazer a solução para um dos problemas mais estudados em teoria dos grafos, o problema do caminho mínimo (*SPP - Shortest Path Problem*). A solução desse problema de forma eficiente trás benefícios para diversas áreas, como transporte, engenharia, telecomunicações e computação [6]. Cormen et al. [7] apresentam três algoritmos para resolver o (*SPP - Shortest Path Problem*), o algoritmo de *Dijkstra*, *Dijkstra Belmman-Ford* e *Floyd-Warshall*. Dentre esses, o algoritmo proposto por *Edsger W. Dijkstra* em 1956 possui a solução com melhor custo computacional para a determinação da menor rota em um grafo [8]. O uso desse tipo de algoritmo em protocolos de rede, pode trazer vantagens em relação a escolha das rotas de transmissão [9], onde a transferência de dados demanda uma menor rota de transmissão a fim de evitar altas latências de comunicação ou prover menor consumo energético.

A estrutura de dados utilizada como fila de prioridades em um algoritmo para cálculo do caminho mínimo afeta diretamente seu custo computacional. Estruturas como *Heap* binário e *Heap* de Fibonacci apresentam características que permitem uma melhoria nas escolhas locais no algoritmo de *Dijkstra* [10]. Utilizando um vetor como estrutura de dados, o algoritmo possui a complexidade de execução para encontrar o caminho mínimo entre dois vértices de um grafo em O(n^2) em que n representa o número de vértices. Alterando sua fila de prioridades, é possível realizar as mesmas operações em O($n$log(n)) para a *Heap*binário e O([m+n]log(n)) com a *Heap* de Fibonacci.

Fredman e Tarjan [10] apresentaram o conceito da *Heap*de Fibonacci em 1984. Cormen et al. [7] evidenciam que, assim como a *Heapbinomial*, uma *Heap* de Fibonacci é uma coleção de árvores ordenadas. Diferentemente da *Heap binária* que apresenta usabilidade para algumas aplicações, a *Heap* de Fibonacci é uma estrutura teórica e com grande complexidade de implementação. Abuaiadh e Kingston, [11] apresentam em seu artigo a complexidade da *Heap* de Fibonacci, enfatizando seu desempenho nas operações realizadas na *Heap*.

Tiago Silveira é mestrando em Informática pelo Programa de Pós-graduação em Informática, PUC Minas, Belo Horizonte, MG, 31980-110 BR e-mail: tbsilveira@sga.pucminas.br.

Ezequiel Duque é Mestre e doutorando em Informática pelo Programa de Pós-graduação em Informática, PUC Minas, Belo Horizonte, MG, 31980-110 BR e-mail: ezequiel.duque@sga.pucminas.br.

Silvio Guimarães é professor do Programa de Pós-graduação em Informática, PUC Minas, Belo Horizonte, MG, 31980-110 BR e-mail: sjamil@pucminas.br.

Humberto Marques-Neto é professor do Programa de Pós-graduação em Informática, PUC Minas, Belo Horizonte, MG, 31980-110 BR e-mail: humberto@pucminas.br.

Henrique Freitas é professor do Programa de Pós-graduação em Informática, PUC Minas, Belo Horizonte, MG, 31980-110 BR e-mail: cota@pucminas.br.

Heaps de Fibonacci e Binária são tipos de filas de prioridade que apresentam ordem de complexidade inferior em certas operações quando comparadas a outros arranjos. Portanto, o presente artigo baseia-se na utilização das *Heaps* de *Fibonacci* e *Binárias* implementadas no algoritmo de *Dijkstra*, desse modo reduzindo o tempo necessário para determinar a menor rota entre dois dispositivos móveis pertencentes a uma rede. Este trabalho apresenta uma nova abordagem baseada na utilização da *Heap* de Fibonacci no algoritmo de *Dijkstra*. A partir da menor rota obtida pela execução do algoritmo, a potência de transmissão dos dispositivos pertencentes ao caminho mínimo é atenuada até o limiar de conexão com o dispositivo mais próximo, evitando o envio e recebimento de pacotes por dispositivos fora da rota e propagando o consumo aos demais nós do caminho, assim estendendo a duração da carga de bateria dos aparelhos na rede.

Portanto, a principal contribuição deste trabalho está na utilização do algoritmo de *Dijkstra* implementado com a *Heap* de Fibonacci como fila de prioridades. Os resultados dos testes apontam essa estrutura com melhor tempo de execução em redes com maior número de conexões quando comparada aos testes realizados com *Vetor* e *Heap Binária* como filas de prioridade. Além disso é importante ressaltar, a adequação da potência de transmissão baseada no resultado do algoritmo para cálculo da rota mínima.

Este artigo está organizado nas seguintes seções: Na Seção II são apresentado os conceitos relevantes para realização dos estudos. A Seção III apresenta os trabalhos trabalhos relacionados. A Seção IV apresenta o método e a proposta deste artigo. A Seção V expõe os resultados obtidos e a Seção VI apresenta a conclusão da pesquisa.

II. Referencial Teórico

A. Heap Binária

Apresentada por Williams [12] como uma estrutura de dados para *Heapsort*, uma *Heap* Binária é uma estrutura de dados desenvolvida a partir de uma árvore binária. São estruturas comuns em utilização em filas de prioridade. Uma *Heap* Binária é definida como uma árvore binária com duas limitações adicionais:

1) Propriedade da Forma - A estrutura da árvore deve estar completa até sua penúltima camada. Caso seu último nível não esteja completo, todos os nós pertencentes ao mesmo deverão estar agrupados à esquerda.
2) *Heap* máxima e mínima - Em uma *Heap* máxima o valor de todos os nós é menor do que o valor do nó pai, a raiz possui o maior elemento. Em uma *Heap* mínima o valor de todos os nós deve ser maior do que o valor do nó pai, a raiz possui o menor elemento.

A Figura 1 apresenta uma *Heap* máxima com o último nível incompleto (propriedade da forma).

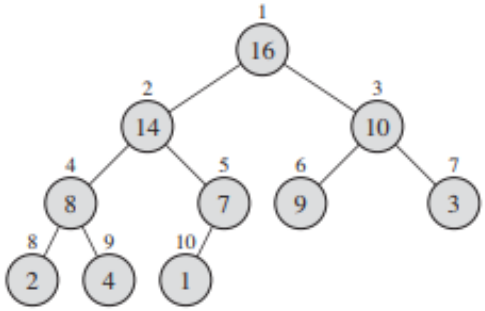

Fig. 1. *Heap* Binária. - [7].

B. Heap de Fibonacci

Cormen et al. [7] afirmam que uma *Heap* de Fibonacci é uma extensão das *Heaps* Binomiais. *Heaps* Binomiais são formadas por uma lista ligada de árvores binomiais, diferente de uma *Heap* Binária, que tem forma de uma única árvore. *Heaps* de Fibonacci contém uma estrutura mais distendida que permite um limite de tempo assintótico melhor, fazendo uso dessa propriedade principalmente quando o problema apresenta valores altos para execução, por exemplo *extract-min* e *delete*. Em situações onde operações de remoção são constantes, *Heaps* de Fibonacci possuem desempenho superior em relação a *Heaps* binárias.

A Figura 2 apresenta uma *Heap* de Fibonacci com cinco árvores ordenadas como *Heaps* mínimos e 14 nós. A linha pontilhada indica a lista de raízes e o nó mínimo do *Heap* é o nó que contém a chave 3.

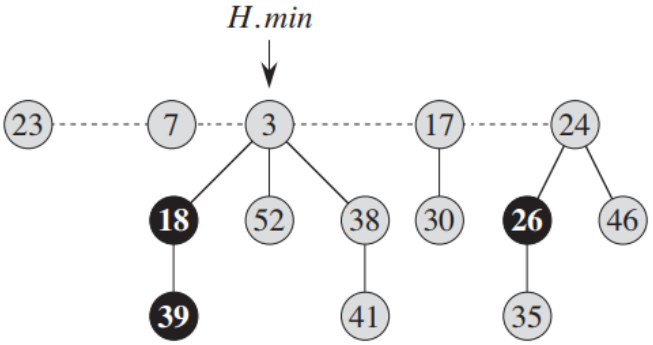

Fig. 2. *Heap* de Fibonacci - [7].

Identificar as capacidades e características de uma *Heap* de Fibonacci é importante para poder definir sua aplicabilidade em um cenário real. A Tabela I apresenta as diferenças entre a ordem de complexidade de uma *Heap* binária e uma *Heap* de Fibonacci, onde n representa o número de vértices. Conforme a Tabela I, a *Heap* de Fibonacci possui melhor ordem de complexidade em operações de inserção e remoção de valores.

Entender a ordem de complexidade das duas estruturas e suas operações além do impacto na solução é de fundamental para a escolha de qual *Heap* usar.

C. Protocolo Ad-Hoc

Segundo Royer e Toh [13], uma rede *Ad-Hoc* é uma coleção de nós que estão dinamicamente e arbitrariamente localizados, de tal modo que interconexões entre os nós são capazes de mudar regularmente. Loo et al. [14] afirmam que uma rede *Ad-Hoc* sem fio é uma coleção de dois ou mais dispositivos

TABELA I
TABELA DE COMPLEXIDADE - [7]

Procedimento	Binary Heap (pior-caso)	Fibonacci Heap (Amortizada)
Criar-*Heap*	$\Theta(1)$	$\Theta(1)$
Inserir	$\Theta(\log n)$	$\Theta(1)$
Minimo	$\Theta(1)$	$\Theta(1)$
Extrair-Min	$\Theta(\log n)$	$\Theta(\log n)$
União	$\Theta(n)$	$\Theta(1)$
Diminuir-Key	$\Theta(\log n)$	$\Theta(1)$
Deletar	$\Theta(\log n)$	$\Theta(\log n)$

sem fio com a capacidade de se comunicarem sem o auxílio de uma nó centralizador. Um protocolo de rede *Ad-Hoc* é uma convenção ou padrão que controla como os nós decidem o direcionamento dos pacotes entre os dispositivos na rede. Exemplos de protocolos *Ad-Hoc* são:

1) *Table-driven* - Royer e Toh [13] afirmam que o protocolo Table-driven mantém uma lista atualizada de destinos e suas rotas, distribuindo periodicamente tabelas através da rede. Algumas desvantagens desse protocolo são a reação lenta a reestruturações e falhas na rede além da grande quantidade de dados transmitidos para manutenção.
2) *On-demand* - Esse protocolo encontra suas rotas através de transmissões contínuas com pacotes de requisição. Suas desvantagens são a alta latência quando há uma falha de roteamento e a possibilidade de causar um gargalo na rede devido a quantidade de transmissões simultâneas [15].

A Tabela II apresenta as características dos protocolos *On-demand* e *Table-Driven*.

TABELA II
COMPARAÇÃO DE PARÂMETROS - [13]

Parâmetros	On-Demand	Table Driven
Informações de roteamento	Disponível quando necessário	Sempre disponível
Atualizações Periódicas	Não necessário	Necessário
Mobilidade	Descoberta de rede localizada	Tabela de roteamento consistente disponível

D. Algoritmo de Dijkstra

O algoritmo de *Dijkstra*, proposto pelo matemático *Edsger Dijkstra* na década de 60, tem como objetivo resolver o problema do caminho mínimo entre vértices.

A partir de um grafo não direcionado G=(n,m), onde n representa o número de vértices e m o número e arestas, o algoritmo em questão mantém um conjunto *S* de vértices cujos pesos finais de caminhos mínimos já partem da fonte *s*. O algoritmo seleciona repetidamente o vértice $u \in V-S$, que tem a mínima estimativa do caminho mais curto, adiciona *u* a *S* e relaxa todas as arestas que saem de *u*. Na implementação apresentada no Algoritmo 1, usa-se filas de prioridade mínimas *Q*, contendo vértices cujas chaves são valores de *d*, sendo *d* as distâncias de *s* até cada vértice *v*.

Algoritmo 1: Pseudo código do algoritmo de *Dijkstra*

```
1  Dijkstra(G, s)
2  INITIALIZE-SINGLE-SOURCE(G, s)
3  S = 0
4  Q = G.V
5  while Q ≠ ∅ do
6  |  u = EXTRACT-MIN(Q)
7  |  S = S ∪ u
8  |  foreach vertex v ∈ G.Adj[u] do
9  |  |  RELAX(u, v, w)
10 |  end
11 end
```

Segundo Cormen et al. [7], a linha 2 inicializa os valores de *d* e π do modo usual, onde π identifica o vértice de onde se origina uma conexão até *v* de maneira a formar o caminho mínimo. Na linha 3 é inicializado o conjunto *S* como vazio ($\emptyset$). O algoritmo mantém o invariante $Q = V - S$ no início de cada iteração do laço *while* da linhas 5 a 11. Na linha 4 é inicializada a fila de prioridade mínimas *Q* para conter todos os vértices em *V*, e, tendo em vista que $S = \emptyset$ nesse momento, o invariante é verdadeiro após a linha 4. Em cada passagem pelo laço *while* da linha 5 a 11, na linha 6 é extraído um vértice *u* de *Q = V - S* e na linha 7 é adicionado ao conjunto *S*, mantendo assim o invariante. Portanto, o vértice *u* tem a menor estimativa de caminhos mínimos em comparação com qualquer vértice em *V - S*. Então nas linhas 8 e 9, relaxa-se cada aresta *(u, v)* que sai de *u*, atualizando assim a estimativa de *v* encontrada até este ponto. Observe-se que, com este comportamento, o algoritmo nunca insere vértices em *Q* após a linha 4 e que cada vértice é extraído de *Q* e adicionado a *S* exatamente uma vez, de modo que o laço *while* itera exatamente $|V|$ vezes. Tendo *n* como a quantidade de vértices e *m* como a quantidade de arestas, o algoritmo de *Dijkstra* executa em $O(m + n \log n)$, quando utilizado com a estrutura *Heap* de Fibonacci, $O(m \log n)$ com a *Heap* Binária e $O(n^2)$ quando usado um vetor. Cormen et al. [7] destacam que o algoritmo utiliza uma estratégia gulosa, pois sempre escolhe o vértice de menor peso em *V-S* para adicionar ao conjunto *S*.

III. TRABALHOS RELACIONADOS

Algoritmos para calcular a menor rota ou caminho mínimo tem diversas aplicabilidades. O algoritmo de *Dijkstra* é um dos mais famosos e utilizados para esse problema, sendo alvo de diversos estudos, como apresentado a seguir.

Bozyiğit et al. [16] apresentam uma proposta de utilização do algoritmo de *Dijkstra* implementado com um sistema de

penalidades, visando encontrar uma solução para o planejamento de rotas no transporte público. O algoritmo testado pela equipe de pesquisa apresentou bons resultados em relação ao algoritmo original, levando em consideração, número de baldeações, distancia e distancia total proposta.

Dai, Q. e Jie Wu [17] utilizam a *Heap* de Fibonacci no algoritmo de *Prim* para obter um cálculo de potência mínima para transmissão de uma rede sem fio *Ad-Hoc*, igualando potência de transmissão, mantendo a conectividade de rede.

Thomas et al. [18] apresentam uma proposta de utilização do algoritmo de *Dijkstra* para diminuir o *delay* de operação causado por um sistema de *Sleep-wake* em redes de sensores sem fio. O objetivo da implementação é estender a duração da carga de bateria de estações de transmissão através da utilização da menor rota.

Fredman et al. [10] afirmam que *Heaps* possuem grande variedade de aplicações em problemas de otimização de redes. O mesmo apresentara o conceito de *Heap* de Fibonacci ou *F-Heaps* com a intenção de obter melhor complexidade de execução no algoritmo de *Dijkstra*.

O trabalho elaborado neste artigo utiliza o algoritmo proposto por *Dijkstra* de maneira dinâmica, atualizando a cada alteração na rede a tabela com as informações sobre as rotas. De maneira distinta ao trabalho apresentado por Dai, Q. e Jie Wu [17] onde a potência de transmissão é ajustada igualmente para todos os nós, o estudo realizado neste artigo visa adequar individualmente a potência de transmissão dos dispositivos. Visando dessa forma a menor sobreposição possível entre os sinais, resultando assim em menor consumo energético na rede.

IV. Metodologia

A. Cenário Utilizando o Protocolo Table-driven

O cenário escolhido para os testes simula uma localidade de desastre onde equipes de busca necessitam de comunicação. Entretanto, as redes móveis convencionais que fazem uso de antenas e satélites estão indisponíveis, como em um possível cenário descrito por Lu et al. [3]. A Figura 3 ilustra o cenário. Dessa forma, a transmissão de pacotes é realizada por um dispositivo móvel e direcionada a um roteador em uma rede estática. Os dados são coletados e a rede é modificada alterando as distâncias e impedimentos físicos à transmissão. A partir disso, os resultados são comparados.

O protocolo *Table-driven* apresenta características favoráveis para ser utilizado no modelo proposto. Características como tabelas de roteamento com informações sobre a rede e a não demanda de transmissões contínuas com pacotes de requisição como exigido no protocolo *On-demand* permitem respectivamente: Armazenamento das informações referentes ao caminho mínimo na rede e economia de energética. A escolha do protocolo *Table-Driven* do tipo *Destination-Sequenced Distance-Vector Routing* (DSDV) deve-se ao fato de possuir tabelas de roteamento com o endereço do nó de destino, a quantidade de nós subsequentes para alcançar o nó de destino e a sequência dos nós. Dessa forma, os resultados do cálculo da menor rota obtidos pelo algoritmo de *Dijkstra* são inseridos na tabela de roteamento, informando a qual nó o transmissor deve alcançar. A Tabela III apresenta os cenários de teste.

TABELA III
Cenários de simulação

Distância	Potência	Barreira
14 Metros	14 dBm	Não
14 Metros	15 dBm	Sim
5 Metros	19 dBm	Não
10 Metros	19 dBm	Não

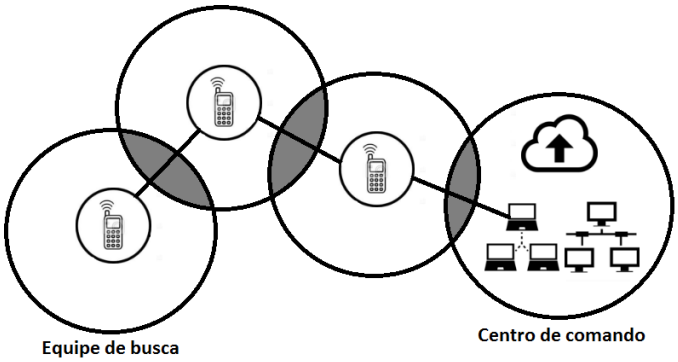

Fig. 3. Cenário da simulação.

B. Custo e Escolha da Rota

Cada nó, composto por um dispositivo móvel apresenta um custo de transmissão para seus nós adjacentes. Esse cálculo deve levar em consideração diversos fatores, uma vez que os mesmos influenciam na propagação do sinal entre o nó transmissor e o nó receptor. Um dos fatores a ser considerado para o cálculo da potência aplicável, é obtido pela fórmula da atenuação no espaço livre, uma das formas mais utilizadas para mensurar o efeito da atenuação de sinal na cobertura de sistemas sem fio [19]. A Equação 1 representa essa fórmula, em que c é uma constante que depende da unidade de medida f e p. Onde f representa a frequência do sinal e p a potência em *dB*.

$$d = 10^{((c-(20*\log_{10}(f))+p)/20} \quad (1)$$

A partir da Equação 1 é possível determinar a potência necessária para a transmissão entre os nós, já considerando possíveis bloqueios gerados por barreiras físicas e atenuação do sinal pela distância. De acordo com o protocolo *table-driven*, uma tabela é gerada com as informações dos nós disponíveis na rede. Dessa forma, a métrica usada como custo entre os vértices do grafo diz respeito a potência de transmissão necessária para alcançar o nó adjacente. A partir dessa tabela é possível modelar um grafo não direcionado que representa o cenário como apresentado na Figura 4. Os vértices do grafo gerado representam os dispositivos móveis, e as arestas a potência necessária para a transmissão entre os mesmos.

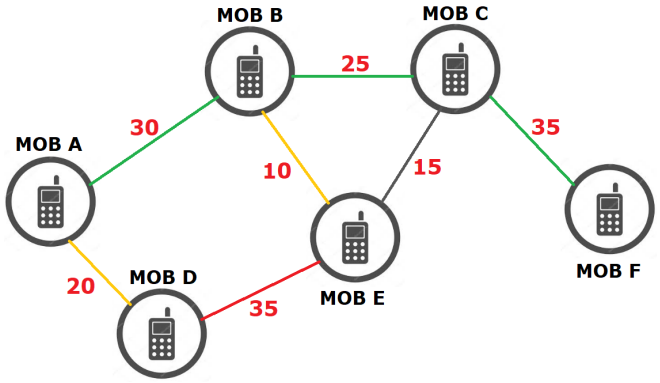

Fig. 4. Distância entre os nós.

Os dados armazenados nas tabelas de roteamento permitem que o cálculo da menor rota no grafo seja realizado. O algoritmo de *Dijkstra* é então executado, retornando a menor rota entre o vértice de origem e o de destino. Ao realizar esse cálculo, o nó de origem pode propagar a informação sobre a rota aos nós subsequentes, evitando o cálculo desnecessário pelos outros nós. Com a tabela de roteamento atualizada é possível então atenuar o sinal de transmissão. Essa estratégia permite que os nós reduzam a potência do sinal de maneira a alcançar apenas o nó destino para que a mensagem seja retransmitida via roteamento oportunístico. A definição da menor rota é de extrema importância, pois a mesma implica na rota com o menor consumo de bateria. Para a simulação, os testes realizados no grafo da Figura 4 utilizaram a ferramenta proposta por Leonardo et al. [20]. O menor caminho pode ser definido pelas conexões entre *MOB F, MOB C, MOB B, MOB A*, definidas com as conexões na cor verde, que representam arestas processadas, pertencentes ao caminho mínimo entre o nó de origem e nó de destino. As conexões *Mob E, Mob B* e *Mob D, Mob A*, definidas na cor amarela, representam arestas processadas em um possível caminho mínimo. A conexão *Mob E, Mob D*, definida na cor vermelha, representa uma aresta processada, não pertencente ao caminho mínimo. A conexão *Mob C, Mob E*, Definida na cor preta, representa uma aresta não processada.

Pela análise do grafo gerado pelo simulador é possível identificar a menor rota entre o vértice de origem e destino. O cálculo da menor rota permite que a mensagem seja propagada pelos nós onde irá demandar menor potência de transmissão.

C. Calculo de Potência

A simulação da transmissão de dados utilizou um aparelho celular *OnePlus 3T* como transmissor. Acessando as configurações do terminal do aparelho é possível alterar o parâmetro *TX Power*, que controla a potência de transmissão *wi-fi* do dispositivo. Por padrão, o parâmetro *TX Power* configurado pelo sistema é determinado com o valor de 19dBm, o que pela Equação 2 reflete em um consumo de aproximadamente *79,4mW*.

$$P = 1mW * 10^{p/10} \quad (2)$$

A Equação 2 apresenta o cálculo da potência de transmissão. Onde P representa a potência em miliwatts e p a potência em dBm. Em segundo momento, utilizou-se o valor de 0 para o parâmetro *TX Power* via terminal do dispositivo. Esse valor inabilita a transmissão de pacotes, resultando em um *(Bit Rate:0 kb/s)*. Para a realização dos testes de potência de transmissão, utilizou-se como base o valor de 19dBm predefinido pelo sistema. Realizou-se a redução de potência de transmissão gradualmente até o limiar de conexão, mantendo a mesma distância de aresta entre os dispositivos. Manteve-se o envio de pacotes de dados com o parâmetro *TX Power* em 14dBm. Valores inferiores ao referido resultaram em perda de pacotes e conexão. Entretanto, Lu et al. [3] afirmam que o cenário físico onde os nós se encontram pode sofrer alterações. Barreiras físicas podem alterar os parâmetros, sendo necessário a realização de novos cálculos para definir a potência ideal de transmissão. A simulação desse cenário teve a adição de uma barreira física entre os dispositivos. O sinal então teve de ser ajustado para alcançar o nó, com o valor de 15dBm. Todavia, permanecendo ainda com valor inferior ao valor padrão ajustado pelo sistema.

D. Abordagem Proposta

Nesta Seção é proposta uma nova abordagem para a transmissão de dados em redes *Ad-Hoc*. A abordagem tem o objetivo de reduzir o custo energético de transmissões *wi-fi* em dispositivos móveis, utilizando adequação de potência de transmissão.

A proposta leva em consideração alterações na estrutura da rede, como movimentação de nós, alterações nas distâncias entre os dispositivos e barreiras físicas entre os nós, recalculando a rota de transmissão periodicamente, como descrito a seguir:

- Cada nó mantém uma tabela contendo informações de roteamento disponíveis para os demais nós da rede, Figura 5 passo 1;
- Cada nó atualiza periodicamente sua tabela de roteamento com as informações de seus vizinhos através da coleta realizada por um sinal de requisição, Figura 5 passo 1;
- Após a coleta de dados, cada nó adéqua a potência de transmissão afim de enviar pacotes a seus vizinhos utilizando a menor potência de sinal possível, Figura 5 passo 2;
- Caso um novo nó passe a ser o vizinho mais próximo de um vértice, as tabelas de roteamento são adequadas e a potência recalculada, Figura 5 passo 3 e 2;
- O nó que deseja transmitir utiliza os dados das tabelas de roteamento, e através do algoritmo de Djikstra, calcula a menor rota de transmissão. Envia juntamente com os dados a serem transmitidos a tabela de roteamento atualizada contendo a rota, Figura 5 passo 4;
- Caso a conexão entre nós seja interrompida devido a alterações nas distâncias ou a perda de um nó, na etapa de requisição, o vértice aumenta a potência de sinal até alcançar novamente seu vizinho ou um novo nó mais próximo, Figura 5 passo 1 e 2;
- Os dados são transmitidos através da rede com a menor potência de transmissão possível.

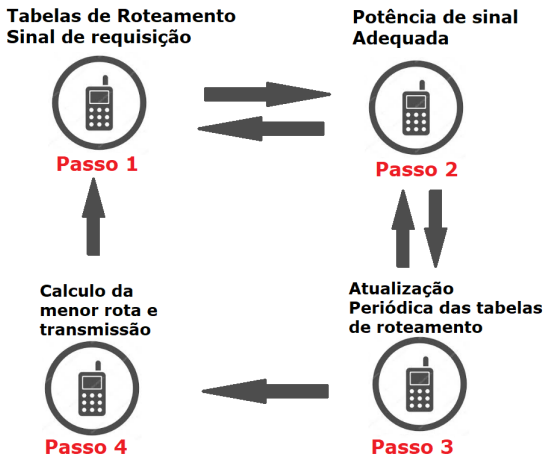

Fig. 5. Nova Abordagem.

V. Resultados

O protocolo *table-driven* atualiza as informações contidas na tabela de roteamento periodicamente. Sempre que a tabela for atualizada com novas informações o algoritmo de *Dijkstra* realiza o cálculo da menor rota entre os vértices do grafo. As tabelas então são atualizadas com as novas informações sobre as rotas. Dessa forma, a propagação das mensagens será mais eficiente energeticamente, propagando na menor potência necessária para alcançar o nó adjacente. Executar o algoritmo de Djikstra no menor tempo possível é importante, pois possibilita uma menor latência na rede e um cálculo mais rápido das tabelas de roteamento. No cenário simulado, é possível atingir uma diferença de potência de transmissão de aproximadamente 41% entre o valor de 19dBm automaticamente configurado pelo sistema, e o valor atenuado de 8dBm.

Ao aplicar a Equação 2 é possível chegar a um resultado de 6.3mW para transmissões em 8dBm de potência e de 79.4mW para a transmissões em 19dBm. O que representa uma redução de 92% em relação ao consumo energético demandado. A Figura 6 expõe a curva de aumento da potência em *miliwatts* exigida para a transmissão em relação a potência em *dBm*.

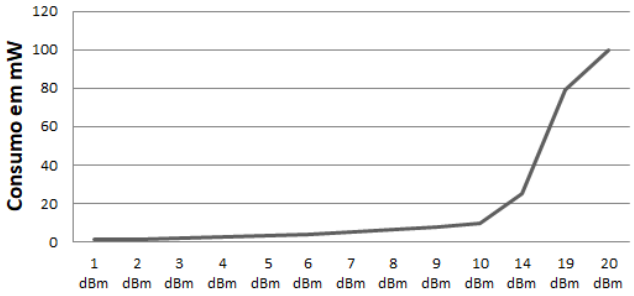

Fig. 6. Consumo de transmissão por dBm.

Ao utilizar a menor rota para a transmissão dos dados, consequentemente a rota com o menor consumo de bateria é escolhida, uma vez que os nós necessários para a retransmissão dos dados irão transmitir em potência reduzida. Entretanto, caso a transmissão seja interrompida, é necessário realizar novamente o cálculo da menor rota.

O algoritmo de *Dijkstra* usa um vetor como fila de prioridades, por esse motivo, o tempo de execução do mesmo é fortemente relacionado a como a estrutura do vetor é implementada. Considerando um vetor não ordenado como fila de prioridades do algoritmo, o tempo de execução para encontrar todos os caminhos mínimos partindo de um vértice de origem é O(n^2). Ao realizar o cálculo do caminho mínimo usando como fila de prioridades a *Heap* de Fibonacci a ordem de complexidade do algoritmo é de O($[m+n]$log(n)). Ao realizar a simulação utilizando a *Binária* a ordem de complexidade para a execução é de O($n$log(n)).

Nas Tabelas IV e V é possível ver um comparativo entre as estruturas de dados testadas e o custo para cada operação.

TABELA IV
COMPLEXIDADE PARA INSERÇÃO E REMOÇÃO

	Vetor	***Heap* de Fibonacci**	***Heap* Binária**
inserção	O(1)	O(1)	O(log
Remover Mínimo	O(n)	O(log n)	O(log n)

TABELA V
COMPLEXIDADE DE EXECUÇÃO PARA CAMINHO MÍNIMO

	G=(n,m)
Vetor	O(n^2)
***Heap* de Fibonacci**	O(m+n)*log n
***Heap* Binária**	O((m+n)*log n)

Ao escolher a fila de prioridades para a implementação no algoritmo de *Dijkstra*, é importante levar em consideração o tipo grafo a ser analisado. Ao considerar os testes realizados com o grafo proposto na Figura 4 da Seção IV-B, onde o mesmo possui 6 vértices e 7 arestas os resultados apontam melhor desempenho da *Heap* Binária. A Figura 7 apresenta os resultados obtidos nas simulações. É possível notar que a estrutura da *Heap* de Fibonacci obteve um resultado melhor em relação a fila de prioridades com o vetor. Entretanto, um resultado inferior a *Heap* Binária, onde a mesma apresenta resultados melhores se comparado com as outras estruturas testadas.

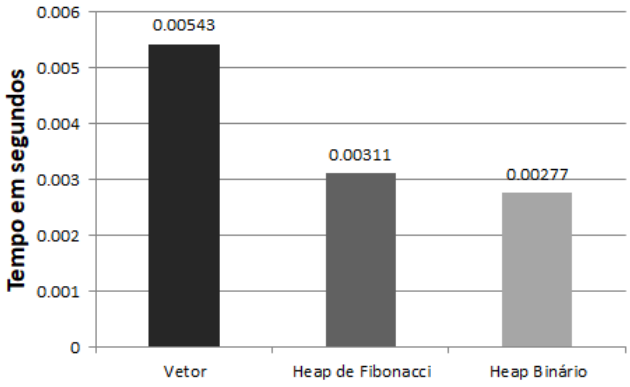

Fig. 7. Tempo de execução das filas de prioridade em um grafo de tamanho total igual a 131.

As duas estruturas propostas são opções viáveis para reduzir o tempo de execução do algoritmo de Djikstra. Entretanto, a *Heap* Binária apresenta melhores resultados em grafos esparsos, tal estrutura representando o tipo de rede mais comum no estudo de caso. A Figura 8 apresenta os dados dos testes realizados com diferentes grafos. Os resultados obtidos apresentam os cenários onde cada estrutura obtém vantagem. É possível notar a *Heap* de Fibonacci com melhor tempo de execução em grafos mais densos a partir de tamanho igual a 226, e a *Heap* Binária em grafos esparsos com tamanhos inferiores a 226 no estudo realizado.

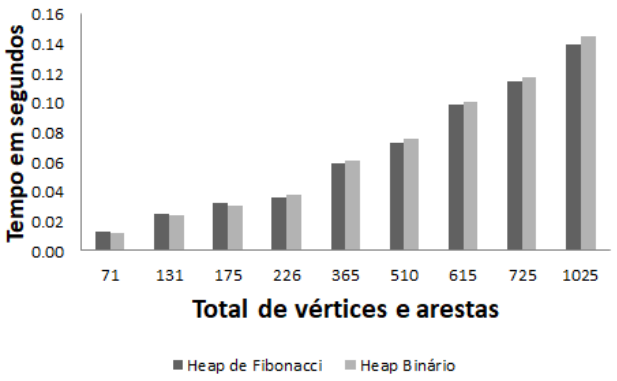

Fig. 8. Tempo de execução em grafos de tamanhos variados.

O gráfico da Figura 9 apresenta a diferença entre o tempo de execução da *Heap* Binária e a *Heap* de Fibonacci. É possível notar o aumento dos valores de acordo com o aumento no tamanho do grafo utilizado na execução, o que indica melhor eficiência da *Heap* de Fibonacci a medida que mais vértices e arestas são adicionados na rede.

No gráfico da Figura 9 é possível observar que a diferença entre o tempo de execução das duas estruturas tende a aumentar a medida que o total de vértices e arestas cresce. Ao realizar a computação em um grafo de tamanho 226, a *Heap* de Fibonacci representa 96,29% do tempo da *Heap* Binária, sendo que em um grafo de tamanho 1025 há uma queda, onde o tempo representa 95,8%. Portanto, os resultados mostram que há uma tendência de melhor escalabilidade do *Heap* de Fibonacci à medida que o tamanho aumenta.

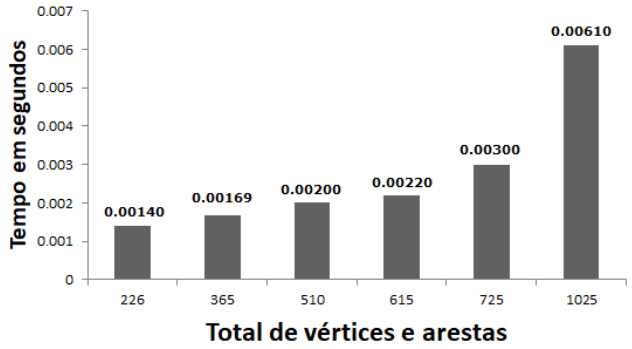

Fig. 9. Aumento de tempo de execução entre as *Heaps*.

As duas estruturas apresentaram tempo de execução menor do que a fila de prioridades utilizando o vetor. Atualizar as tabelas de roteamento com a menor rota de transmissão é crucial, por esse motivo, executar esse passo no menor tempo possível resulta em vantagens, como a redução do *Delay* entre a atualização das tabelas e a diminuição do tempo computacional total. Em um cenário onde a vida útil da bateria deve ser a maior possível, o tempo de cálculo para atualização das tabelas de roteamento influencia no consumo energético. Dessa forma, a redução no tempo de cálculo das rotas é de extrema importância.

VI. Conclusão

Transmissões entre dispositivos móveis demandam grande consumo energético, em cenários de desastre onde a vida útil de bateria destes dispositivos é crucial para manter a comunicação entre os demais aparelhos da rede, manter um nó ativo por maior tempo é fundamental. A escolha da menor rota entre os nós de retransmissão impacta diretamente na escolha do caminho onde os dispositivos transmitirão os dados de maneira energeticamente mais eficiente. A partir da utilização do algoritmo de textitDijkstra é possível determinar a menor rota de transmissão em uma rede. Essa rota representa a menor distância entre o nó de origem e de destino para transmissões de dados. A partir das distâncias entre os nós pertencentes ao caminho mínimo é possível reduzir a potência de transmissão dos mesmos para manter a conexão apenas entre o nó de transmissão e o nó seguinte na caminho mínimo. Os resultados indicam que a adequação de sinal para a redução da potência além da otimização do algoritmo de *Dijkstra* com a *Heap* de Fibonacci impacta diretamente no consumo em *miliwatts* demandado para o cálculo da menor rota e para a transmissão dos dados, dessa forma reduzindo e propagando o consumo de bateria entre os nós do caminho e eliminando a necessidade de transmissão dos nós não pertencentes ao mesmo, alcançando reduções de potência demanda de até 92%. Como trabalhos futuros sugere-se o cálculo da redução total de transmissão entre os nós do caminho mínimo e o cálculo total da vida útil dos nós da rede além de testes com a remoção de nós e reestruturação da rede dinamicamente.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Os autores agradecem ao CNPq, à FAPEMIG e a PUC Minas pelo suporte parcial na execução deste trabalho.

Referências

[1] K. Liu, J. K. Y. Ng, V. C. S. Lee, S. H. Son, and I. Stojmenovic, "Cooperative data scheduling in hybrid vehicular ad hoc networks: Vanet as a software defined network," *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1759–1773, Jun. 2016. [Online]. Available: https://doi.org/10.1109/TNET.2015.2432804

[2] M. M. G. Ghorale and A. Bang, "Wireless ad-hoc networks: Types, applications, security goals," in *National Conference "CONVERGENCE*, vol. 2015, 2015, p. 28.

[3] Z. Lu, G. Cao, and T. La Porta, "Teamphone: Networking smartphones for disaster recovery," *IEEE Transactions on Mobile Computing*, vol. 16, no. 12, pp. 3554–3567, 2017.

[4] A. Carroll, G. Heiser *et al.*, "An analysis of power consumption in a smartphone." in *USENIX annual technical conference*, vol. 14. Boston, MA, 2010, pp. 21–21.
[5] A. B. Vicente, J. G. Jimenez, and A. G. Cervero, "Characterization of energy consumption of a 802.11 g device," *IEEE Latin America Transactions*, vol. 13, no. 8, pp. 2495–2499, 2015.
[6] H. Ortega-Arranz, D. R. Llanos, and A. Gonzalez-Escribano, "The shortest-path problem: Analysis and comparison of methods," *Synthesis Lectures on Theoretical Computer Science*, vol. 1, no. 1, pp. 1–87, 2014.
[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.
[8] A. V. Goldberg and R. E. Tarjan, "Expected performance of dijkstra's shortest path algorithm," *NEC Research Institute Report*, 1996.
[9] M. Xu, Y. Liu, Q. Huang, Y. Zhang, and G. Luan, "An improved dijkstra's shortest path algorithm for sparse network," *Applied Mathematics and Computation*, vol. 185, no. 1, pp. 247–254, 2007.
[10] R. E. Fredman, Michael L. e Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. ACM*, vol. 34, no. 3, pp. 596–615, Jul. 1987.
[11] D. Abuaiadh and J. H. Kingston, "Are fibonacci heaps optimal?" in *International Symposium on Algorithms and Computation*. Springer, 1994, pp. 442–450.
[12] J. W. J. Williams, "Algorithm 232: Heapsort," *Communications of the ACM*, vol. 7, p. 347–348, 1964.
[13] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE personal communications*, vol. 6, no. 2, pp. 46–55, 1999.
[14] J. Loo, J. L. Mauri, and J. H. Ortiz, *Mobile ad hoc networks: current status and future trends*. CRC Press, 2016.
[15] N. S. Yadav and R. Yadav, "Performance comparison and analysis of table-driven and on-demand routing protocols for mobile ad-hoc networks," *International Journal of Information Technology*, vol. 4, no. 2, pp. 101–109, 2007.
[16] A. Bozyiğit, G. Alankuş, and E. Nasiboğlu, "Public transport route planning: Modified dijkstra's algorithm," in *Computer Science and Engineering (UBMK), 2017 International Conference on*. IEEE, 2017, pp. 502–505.
[17] Q. Dai and J. Wu, "Computation of minimal uniform transmission power in ad hoc wireless networks," in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, May 2003, pp. 680–684.
[18] S. Thomas, I. Gayathri, and A. Raj, "Joint design of dijkstra's shortest path routing and sleep-wake scheduling in wireless sensor networks," in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. IEEE, 2017.
[19] S. Sun, T. S. Rappaport, S. Rangan, T. A. Thomas, A. Ghosh, I. Z. Kovacs, I. Rodriguez, O. Koymen, A. Partyka, and J. Jarvelainen, "Propagation path loss models for 5g urban micro- and macro-cellular scenarios," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, May 2016, pp. 1–6.
[20] M. R. V. V. B. d. S. Leonardo Beck Prates, Marlon Jordan Francisco. (2015) Dijkstra simulator. [Online]. Available: https://github.com/leobeckp/DijkstraSimulator

Tiago Batista da Silveira Técnico em informática pelo Colégio Técnico Padre De Man (2010), bacharel em Sistemas de Informação (2016) pelo Centro Universitário do Leste de Minas Gerais e mestrando em Informática pela Pontifícia Universidade Católica de Minas Gerais. Possuí interesse de pesquisa em computação de alto desempenho, computação heterogênea, arquiteturas paralelas, programação paralela e computação quântica.

Ezequiel Mendes Duque Atualmente é Doutorando em Informática na Pontifícia Universidade Católica de Minas Gerais. Possui Mestrado em Informática pela Pontifícia Universidade Católica de Minas Gerais (2016), MBA em Gestão Empresarial pelo Centro Universitário do Leste de Minas Gerais (2011) e graduação em Computação - Sistemas de Informação pelo Centro Universitário do Leste de Minas Gerais (2009). Leciona no IGTI para Pós Graduação em Banco de Dados e Arquitetura de Software, no Centro Universitário do Leste de Minas Gerais, leciona para o curso de Sistemas de Informação e nos cursos de Engenharia. É organizador do Congresso Nacional de Profissões em Tecnologia da Informação (CONAPTI). Tem experiência na área de Ciência da Computação, com ênfase em Sistemas de Informação, atuando com projetos sociais na área de TI.

Silvio Jamil Ferzoli Guimarães É doutor pela Universidade Federal de Minas Gerais, Brasil, e Université de Marne-la-Valle, France (2003). É professor do Departamento de Ciência da Computação da Pontifícia Universidade Católica do Minas Gerais (PUC Minas) desde 2002 e pesquisador convidado da ESIEE/Paris desde 2011. Coordena o laboratório de Processamento de Informação Áudio-Visual da PUC Minas. Tem interesse de pesquisa em análise de imagem e vídeo, recuperação de dados multimídia e análise hierárquica de dados.

Humberto Torres Marques-Neto É professor associado do Departamento de Ciência da Computação da Pontifícia Universidade Católica de Minas Gerais (PUC Minas), Brasil. Doutor em Ciência da Computação pela Universidade Federal de Minas Gerais (UFMG) em 2008. É docente do Programa de Pós-Graduação em Informática desde 2009. Seus interesses de pesquisa incluem caracterização e modelagem do comportamento do usuário em sistemas distribuídos de grande porte, engenharia de software, mobilidade humana e questões que podem conduzir soluções para problemas de TIC (Tecnologia da Informação e Comunicação) relacionados à qualidade do serviço de sistemas de computação, como desempenho, confiabilidade, disponibilidade, segurança, custo e questões econômicas. Ele é membro da SBC (Sociedade Brasileira de Computação), IEEE e ACM. Atualmente coordena o Ecossistema de Inovação Tecnológica da PUC Minas.

Henrique Cota de Freitas Possui graduação em Ciência da Computação (2000) e mestrado em Engenharia Elétrica (2003) pela Pontifícia Universidade Católica de Minas Gerais e doutorado (2009) em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Atuou como pesquisador convidado (2015-2016) no grupo de pesquisa CORSE do INRIA Grenoble, França. Atualmente é professor da Pontifícia Universidade Católica de Minas Gerais. Tem interesse de pesquisa em computação de alto desempenho, computação heterogênea, arquiteturas paralelas e programação paralela.