Bluemob: Algoritmo Dinâmico para Formação de Redes Bluetooth em Aplicações de Sensoriamento Urbano

Giovani Pieri[1], Werner Kraus Jr.[1], and Jean-Marie Farines[1]

[1]Departamento de Controle e Automação – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

pieri@das.ufsc.br, {werner.kraus,j.m.farines}@ufsc.br

Abstract. *We propose a new algorithm for building ad-hoc networks among mobile devices tailored for small and dynamic environments using the Bluetooth interface. The target application is network formation among passengers on urban buses. The proposed algorithm assigns roles to the network participants for reducing the effect of nodes leaving the network upon alighting. Based on estimates of node permanence onboard, an application dependent mechanism assign roles to the nodes indicating those more apt to perform a given function in the network (masters, slaves or gateways). Simmulation studies indicate improvements of up to 75% in the times needed to rebuild the network after nodes have left in comparison to a classic algorithm. Performance levels in relation to the quality of estimates are also evaluated for indication of accuracy of estimation needed for proper application of the algorithm.*

Resumo. *Propomos um algoritmo para formação de redes ad-hoc entre dispositivos móveis em ambientes pequenos e dinâmicos através da interface Bluetooth. A aplicação pretendida é a formação de rede entre passageiros de ônibus urbanos. O algoritmo proposto atribui funções aos integrantes da rede de forma a atenuar o efeito da saída de dispositivos no desembarque. Baseando-se em estimativas de permanência de cada nó a bordo do ônibus, um mecanismo dependente da aplicação atribui papéis aos nós, indicando quais os mais aptos a ocupar uma determinada função na rede (mestre, escravo ou* gateway*). Experimentos de simulação apontam melhora de até 75% nos tempos de reformação após a saída de nós da rede em comparação com algoritmo clássico. Avaliam-se também os níveis de desempenho em relação à qualidade das estimativas, indicando qual a precisão das estimativas necessária para a boa aplicação do algoritmo.*

1. Introdução

Há muitas possibilidade que surgem de redes ad-hoc formadas por smartphones em ambientes fechados. Por exemplo, robôs explorando uma construção podem se comunicar entre si para aperfeiçoar procedimentos de mapeamento; pessoas em um encontro social podem contactar pessoas próximas que estejam dispostas a interagir; e passageiros de ônibus podem, além de interagir entre si através da rede, contribuir para coletar dados sobre o transporte público.

Este trabalho é orientado ao último caso. O objetivo é organizar uma rede que permita o levantamento de informações sobre dados de ônibus para auxílio no planejamento

e operação dos sistemas de transporte público. Tal esquema dispensaria a necessidade de implantação de infraestrutura pública por autoridades e operadores, tanto nos ônibus quanto nas cidades. Duas vantagens da proposta são (1) baixo custo para a cidade e operadores (tanto em infraestrutura quanto em manutenção), importante em situações onde operadores resistem em investir no serviço de transporte público; e (2) a independência e resiliência do sistema de coleta em relação a interesses de agentes que se beneficiariam ocultando informações. Em cenários nos quais estes fatores não são relevantes, o sistema beneficia passageiros interessados em interagir entre si no ônibus através da rede formada.

A delegação da solução de um problema e obtenção de informações sobre o ambiente urbano a voluntários que resulta em um retorno social é conhecido como *urban sensing* [Burke et al. 2006, Campbell et al. 2006]. O sucesso de aplicações como Waze [Waze Mobile] atesta a viabilidade de *urban sensing* usando smartphones para coleta de informações, neste caso sobre o estado do tráfego. Trabalhos anteriores da literatura que fazem uso de smartphones para coleta de dados são: Nericell [Mohan et al. 2008] que usa smartphones para monitorar condições de trânsito e manutenção de rodovias através dos sensores do smartphones para automaticamente detectar frenagens, acelerações, lombadas, buracos e níveis de ruído; [Thiagarajan et al. 2010] descreve um sistema de rastreamento de ônibus e trem para estimar o tempo de chegada do próximo ônibus ou trem na parada; em [Lin et al. 2010] um sistema é proposto para avaliar o nível de conforto do transporte público através de dados dos sensores dos smartphones.

Há trabalhos na literatura acerca da obtenção de dados sobre passageiros em sistemas de transporte público. Por exemplo, [Ben Moshe et al. 2014] propõe o uso de equipamento no interior do ônibus para detectar e identificar dispositivos Bluetooth dos passageiros. Entretanto, equipamentos devem ser instalados no interior do ônibus e smartphones assumem um papel passivo. Baseado no sucesso de *Waze*, *Moovit* [Moovit] provê informação obtida junto aos usuários e operadores, porém usa interface de dados (3G ou similar) com alto consumo energético e não coleta informação sobre viagens realizadas.

Nossa abordagem se apoia na formação de redes espontâneas no interior do ônibus pelos seguintes motivos. Primeiro, é possível manter um registro da vida da rede desde o início da viagem do ônibus até o seu destino sempre que não houver desembarque total em algum ponto ao longo do percurso. Segundo, a transmissão de informações a um servidor para processamento de dados pode ser realizada por um único nó, possivelmente um líder escolhido na rede, economizando tráfego e ajudando na consistência de dados. Terceiro, apesar deste aspecto não ser tratado neste trabalho, é mais fácil confirmar que smartphones estão em um mesmo ônibus quando eles podem trocar informações entre eles (p. ex., através da troca de traços de GPS). O problema de determinar quando um smartphone está de fato embarcado em um ônibus foi abordado anteriormente e mostrado ser de difícil tratamento [Reddy et al. 2010].

Bluetooth é usado como interface de comunicação pois (1) é mais eficiente energeticamente do que WiFi; (2) ônibus são particularmente pequenos (entre 12 m e 18 m) e o maior alcance fornecido pelo WiFi não é necessário para construir uma rede entre passageiros de um ônibus; e (3) Bluetooth tem suporte para criar redes sem infraestrutura.

Considerando que a abordagem se apoia em redes Bluetooth, a questão da formação de tais redes é tratada no restante deste trabalho.

A principal restrição para o algoritmo de formação de redes é temporal: o tempo para formação da rede deve ser menor que o tempo de viagem entre dois pontos de paradas. Tempos típicos entre paradas são de 1 min, apesar de haver tempos menores em situações com paradas pouco espaçadas. P. ex., considerando-se aceleração, velocidade de cruzeiro e frenagem, velocidades médias de 30 km/h (8.3 m/s) são comumente observadas, resultando em intervalos de 30 s para paradas espaçadas de 250 m.

Neste artigo, propomos *Bluemob*, um algoritmo projetado para esta aplicação. Baseado em algoritmos de formação previamente propostos na literatura, adicionam-se pesos relativos à estimativa de permanência a bordo de forma a posicionar nas bordas da rede aqueles nós com maior propensão a desembarcar. Desta forma, busca-se evitar o particionamento da rede durante os desembarques, provendo a reconstrução da rede de forma mais rápida. Resultados de simulação indicam que os tempos envolvidos para a reformação da rede são menores do que o tempo típico de viagem entre paradas.

A topologia resultante de Bluemob reflete a propensão dos nós em saírem da rede. Salientamos que este mesmo conceito pode ser usado em cenários com restrições diferentes das temporais. P. ex., a energia dos nós pode ser mais importante na comunicação multi-robôs e ambientes sociais. Nestes casos, Bluemob deve ser adaptado para que a topologia da rede reflita estimativas energéticas tais como a carga e capacidade da bateria, e consumo previsto considerando o histórico do uso de cada dispositivo.

O artigo é organizado da forma seguinte. Na seção 2 são apresentados algoritmos para construção de redes Bluetooth existentes na literatura. Na seção 3 apresentamos os detalhes de Bluemob, destacando suas propriedades. A avaliação de Bluemob e resultados de simulação são apresentados na seção 4. Por fim, na seção 5 retomamos as principais contribuições do artigo e apontamos perspectivas futuras.

2. Algoritmos para construção de redes Bluetooth

Bluetooth é uma tecnologia de comunicação para curtas distâncias. *Smartphones* e outros dispositivos de uso pessoal o utilizam devido a seu baixo custo e consumo energético. O protocolo Bluetooth é baseado em comunicação mestre-escravos e mecanismo de salto de frequência na faixa de 2.4 GHz. São usados 79 canais com 1 MHz de diferença entre eles.

O processo de estabelecimento de uma conexão Bluetooth se dá através do processo INQUIRY e INQUIRY SCAN. Um dispositivo realiza buscas através de INQUIRY, enviando mensagens em uma sequência de 32 canais pré-definidos. Dispositivos usam o processo INQUIRY SCAN para serem descobertos, no qual ouvem mensagens enviadas nos mesmos 32 canais usados pelo INQUIRY. Ao receber um pacote enviado por INQUIRY, o dispositivo INQUIRY SCAN estabelece uma conexão. O dispositivo em INQUIRY se torna o mestre do dispositivo em INQUIRY SCAN.

Um mestre pode se conectar a no máximo sete escravos ativos. Esta estrutura é denominada *Piconet*. Toda a comunicação se dá entre o escravo e seu mestre. Não há comunicação direta entre dois mestres ou dois escravos. Dispositivos podem participar de mais de uma piconet ao mesmo tempo, denominados pontes. Escravos que participam de apenas uma piconet são denominados escravos não compartilhados. Redes formadas por piconets interconectadas são chamadas de *Scatternets*.

A formação de *scatternets* (ou *Bluetooth Scatternet Formation* - BSF) consiste em

definir como as piconets são formadas a partir de um conjunto de dispositivos Bluetooth que se conhecem entre si e como estas são topologicamente organizadas [Jedda et al. 2013]. Os algoritmos BSF devem atribuir o papel de mestre, escravo ou ponte a cada um dos participantes da rede e assegurar a conectividade desta, respeitando ainda as restrições impostas pela tecnologia Bluetooth, como o limite de sete escravos por mestre.

Os algoritmos BSF tentam resolver o problema de formação da *scatternet* otimizando diferentes métricas. Dentre as métricas, destacam-se: o tempo necessário para formar a *scatternet*; o número de *piconets*; quantidade de pontes mestre-escravo (ME) e escravo-escravo (EE), priorizando pontes EE sobre pontes ME; e número de papéis por nó, ou seja, a quantas piconets cada nó pertence em média [Jedda et al. 2013].

Os algoritmos BSF podem ser classificados em [Stojmenovic and Zaguia 2006]:

- Algoritmos *Single-hop* vs. *multi-hop*: algoritmos *single-hop* pressupõem que todos os dispositivos encontram-se dentro do alcance de comunicação Bluetooth, permitindo que quaisquer dois destes se conectem. Os algoritmos *Multi-hop* correspondem ao caso da existência de dispositivos fora do alcance de comunicação um do outro, necessitando o roteamento de mensagens.
- Algoritmos centralizados vs distribuídos: no caso centralizado, o algoritmo BSF é executado numa unidade central de processamento, como um servidor externo ou um líder. BTCP [Salonidis et al. 2005] é um exemplo de algoritmo desta classe. No caso distribuído, o processamento é local, dados são trocados entre vizinhos e a decisão da topologia final da rede é realizada de forma distribuída.
- Descoberta dinâmica vs descoberta estática: em algoritmos com descoberta estática, existe uma fase preliminar de descoberta responsável por identificar todos os dispositivos Bluetooth próximos. BlueStar [Petrioli et al. 2003], BlueMesh [Petrioli et al. 2004], BlueNet [Wang et al. 2002], BlueMIS [Zaguia et al. 2008] são algoritmos com descoberta estática. Os algoritmos com descoberta dinâmica não possuem uma etapa preliminar de descoberta, sendo este procedimento realizado junto ao algoritmo de formação. Portanto, algoritmo com descoberta dinâmica são melhores adaptados a cenários onde o conjunto de dispositivos não é conhecido a priori, ou quando novos dispositivos surgem durante o processo de formação de scatternet.

Um algoritmo BSF para nossa aplicação deve possuir as seguintes características:

- *Single-hop*: nos cenários considerados, o alcance máximo dos dispositivos Bluetooth, é suficiente para cobrir toda a área onde a rede opera. Dispositivos modernos possuem rádios Bluetooth Classe 2 capazes de estabelecer canais de comunicação a distâncias de até 20 metros [Pei et al. 2010], suficientes para cobrir todo o ônibus, dispensando o uso de algoritmos *multi-hop* e sua complexidade.
- Descoberta Dinâmica: nos cenários considerados de redes formadas entre passageiros de ônibus, há entrada e saída constante de nós da rede, correspondente aos embarques e desembarques do ônibus. Consequentemente, um algoritmo BSF deve ser capaz de incorporar novos dispositivos a *scatternet*. Este cenário é mais favorável a algoritmos com descoberta dinâmica.

Os algoritmos Law [Law et al. 2003] e TSF [Tan et al. 2002] são os dois principais algoritmos BSF *single-hop* com descoberta dinâmica.

Law e TSF constroem componentes, e expandem a scatternet através da união (*merge*) destes. As principais diferenças entre Law e TSF são: (i) a organização dos componentes numa árvore para o algoritmo TSF, enquanto o algoritmo Law cria redes *mesh*; (ii) pequenas diferenças no processo de busca por novos nós, apesar da similaridade entre os dois.

Neste trabalho escolhemos o algoritmo Law como ponto de partida para a proposição de um novo algoritmo BSF devido ao uso da topologia *mesh*. Isso permite maior liberdade durante a reorganização da scatternet que ocorre durante a entrada e saída de nós da rede, sendo mais apropriada ao nosso objetivo. Entretanto, o algoritmo TSF pode ser adaptado para operar nestes cenários, mas com um *overhead* extra introduzido pela obrigatoriedade da manutenção da topologia em árvore.

3. Algoritmo BSF Dinâmico para Aplicações de Sensoriamento Urbano

Algoritmos BSF dinâmicos propostos na literatura não levam em consideração que alguns nós da rede têm maior propensão a deixar a *scatternet*. Esta situação é intrínseca do ambiente no qual Bluemob opera. Atribui-se a cada nó da rede um número, chamado *peso*, para quantificar a sua propensão em desembarcar. A introdução de um modelo de comportamento na formação possibilita alocar nós com menor propensão de sair da rede em posições de mestre e ponte. Desta forma, durante a saída dos nós da rede haverá um menor número de particionamentos em relação a algoritmos sem modelo de comportamento pois os nós que saem da rede não ocuparão posições estruturantes. Menos particionamentos acarretam tempo menor para restaurar a conectividade da rede.

Possíveis indicadores da propensão de um passageiro para desembarcar são, por exemplo: (i) o tempo que o passageiro está a bordo, considerando que quanto maior este tempo, mais próximo está o seu ponto de desembarque; (ii) a posição do passageiro no interior do ônibus, pois a proximidade dos passageiros às portas de saída indica uma propensão a desembarcar, principalmente em ônibus lotados; (iii) o conhecimento prévio sobre padrões de embarques e desembarques em uma dada rota de ônibus, obtido a partir de dados históricos de origem-destino que fornecem a probabilidade de desembarque em uma parada dado o local onde o embarque ocorreu. Neste trabalho, o modelo de comportamento é traduzido em um número que indica o ponto de ônibus de saída do nó. Embora o processo de obtenção da estimativa não seja discutido, variam-se os níveis de precisão da estimação para avaliar o quão crítico é este processo.

Bluemob é um algoritmo BSF com descoberta dinâmica inédito, diferenciando-se por considerar a aplicação na estruturação da rede através do peso a ele atribuído. A melhoria no desempenho do algoritmo pode ser medida pelo tempo que necessita para reestabelecer a rede após a saída de nós. No cenário em estudo, a rede deve ser sempre reconstruída antes da chegada do ônibus na próxima parada. O tempo de viagem entre paradas, então, impõe um *soft deadline* que precisa ser respeitado para que as aplicações possam identificar os passageiros embarcados que desejam participar da rede.

O peso h do nó i utilizado no algoritmo Bluemob pode assumir quaisquer valores, desde que exista uma relação de ordem total definida sobre os valores possíveis de $h(i)$.

Um componente é definido como sendo: (1) um nó desconectado; (2) uma piconet; ou (3) uma *scatternet*. Todo componente possui um *líder* responsável por coordenar as

atividades de buscas por dispositivos. Este papel é desempenhado por um de seus mestres. Caso o componente possua um único nó, então este nó será o líder do componente. Um componente válido deve satisfazer as seguintes propriedades:

(i) o componente possui um único líder;
(ii) o líder deve possuir no máximo seis escravos;
(iii) os mestres devem possuir menos que sete escravos;
(iv) o líder deve possuir um escravo não-compartilhado, ou não possuir nenhum escravo;
(v) todo escravo não-compartilhado possui peso h menor ou igual ao peso h de seu mestre.

No que segue será utilizada a seguinte notação:

- $S(x)$: o conjunto de escravos de x;
- $M(x)$: o conjunto de mestres de x;
- $S_u(x)$: o conjunto de escravos não compartilhados de x;
- $Y_p(u)$: o conjunto de escravos não compartilhados de u que têm h maior que ou igual a todos os outros escravos não-compartilhados de u, i. é: $Y_p(u) \leftarrow \{y \in S_u(u) | \forall k \in S_u(u) : h(y) \geq h(k)\}$

O processo básico de Bluemob é a união de componentes. Este processo transforma dois componentes em um componente válido, ou seja, as propriedades acima são verificadas no novo componente gerado.

O algoritmo Bluemob opera em rodadas, sendo que cada rodada possui uma duração δ durante a qual o líder tenta realizar a união com outro componente. No início de uma rodada (linha 3 do algoritmo 1) o líder realiza um processo INQUIRY (linha 6) ou requisita a um de seus escravos não-compartilhados que realize INQUIRY SCAN (linha 12). A escolha entre qual dos dois processos (INQUIRY ou INQUIRY SCAN) será realizado é aleatória com probabilidade P de realizar INQUIRY (linha 5). Caso o componente consista de um único nó, o líder realiza o processo INQUIRY SCAN ele mesmo (linha 9). É importante notar que este algoritmo pressupõe que todas as operações da rodada, incluindo o processo de união, sejam concluídas dentro do limite de tempo δ.

Ao estabelecer uma conexão, dados sobre os componentes são trocados e o processo de reorganização da topologia iniciado.

Seja $h(u)$ o valor de h do nó u, e $y(u) \in Y_p(u)$. O mestre em INQUIRY é u; o nó em INQUIRY SCAN é v; e o líder do componente de v é denominado w. O conjunto dos escravos de um nó qualquer x é denominado $S(x)$. Salienta-se que, se v é um escravo de u ($v \in S(u)$) e o componente de v possui mais do que um nó ($v \neq w$), então v também será escravo do líder w ($v \in S(w)$). Em outras palavras, quando o líder u encontra um outro componente, uma conexão é estabelecida entre v e u sendo que u assume o papel de mestre de v. Também, por construção, se v não está sozinho em seu componente, então w será mestre de v. Isto é importante, pois v pertence tanto a $S(w)$ quanto a $S(u)$. Este fato deve ser levado em consideração durante o processo de reorganização.

Quando o líder em INQUIRY descobre outro componente, uma reorganização é feita de forma que o componente resultante respeite as propriedades estabelecidas. Bluemob escolhe uma dentre treze reorganizações, dependendo da estrutura dos componentes:

1. Se $v = w$, o componente descoberto é composto por um nó. Pois, um líder w só entra em INQUIRY SCAN neste caso. Para este cenário, temos cinco casos:

Algorithm 1 Bluemob executando no nó i

```
1: P ← (0, 1); δ ← duração da rodada
2: isLeader_i ← true; h_i ← h(i)
3: procedure START ROUND
4:     p ← random number between 0 and 1
5:     if p < P then
6:         INQUIRY
7:     else
8:         if S(i) = ∅ then
9:             INQUIRYSCAN(i)
10:        else
11:            y ← any element ∈ Y_p(i)
12:            INQUIRYSCAN(y)
13:        end if
14:    end if
15: end procedure
16: procedure WHEN CONNECT(to)
17:    u ← i; v ← to
18:    if M(v) = {u} then
19:        w ← v
20:    else
21:        w ← M(v) − {u}
22:    end if
23:    if Y_p(i) = ∅ then
24:        y ←⊥
25:    else
26:        y ← qualquer elemento ∈ Y_p(i)
27:    end if
28:    REORGANIZE(u, v, w, y)
29: end procedure
30: procedure BECOME LEADER(u)
31:    leader_u ← true
32:    Agendar a execução da tarefa periódica de Start Round com período δ
33: end procedure
34: procedure RETIRE(u)
35:    leader_u ← false
36:    Parar a execução da tarefa periódica de Start Round com período δ
37: end procedure
```

(a) Caso 1: $|S(u)| < 7$ e $h(v) \leq h(u)$
(b) Caso 2: $|S(u)| < 7$ e $h(v) > h(u)$
(c) Caso 3: $|S(u)| = 7$ e $h(u) \geq h(v)$ e $h(y) \leq h(v)$
(d) Caso 4: $|S(u)| = 7$ e $h(u) \geq h(v)$ e $h(y) > h(v)$
(e) Caso 5: $|S(u)| = 7$ e $h(u) < h(v)$

2. Se $v \neq w$, o componente descoberto é uma piconet ou scatternet. Neste cenário, temos os seguintes casos:
   (a) Caso 6: $|S(u)| + |S(w)| < 7$ e $h(u) \geq h(w)$
   (b) Caso 7: $|S(u)| + |S(w)| < 7$ e $h(u) < h(w)$
   (c) Caso 8: $|S(u)| = 1$ e $h(v) \geq h(u)$
   (d) Caso 9: $|S(u)| = 1$ e $h(v) < h(u)$
   (e) Caso 10: $|S(u)| + |S(w)| == 7$ e $h(v) \geq h(w)$
   (f) Caso 11: $|S(u)| + |S(w)| == 7$ e $h(v) < h(w)$
   (g) Caso 12: $h(w) \geq h(u)$
   (h) Caso 13: $h(w) < h(u)$

Para cada caso, Bluemob define os papéis dos nós das piconets dos líderes (u e w), refazendo conexões e determinando quem será o líder do novo componente.

Devido a restrições de espaço, não é possível detalhar as treze reorganizações. A descrição completa do algoritmo e suas reorganização podem ser encontradas no relatório

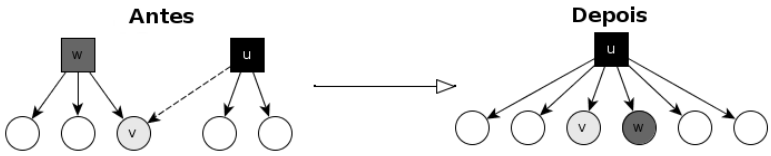


Figura 1. Procedimento de reorganização após união de componentes (Caso 7).

técnico [Pieri 2014]. Para ilustrar, a Figura 1 mostra o caso 7, no qual u realiza o processo de INQUIRY, descobre o nó v e estabelece a conexão entre u e v representada pela linha tracejada. O mestre de v é w, líder do componente. Neste exemplo, o peso de u é menor que o de w ($h(u) < h(w)$). Dado o número de escravos na piconet de cada um dos líderes, $|S(u)| + |S(w)| < 7$, os nós de ambas as piconets podem ser unificadas em uma única piconet sem violar o limite superior de escravos do líder (neste exemplo, seis). A questão pendente em relação à reorganização é a seguinte: deve u se tornar escravo de w e w se tornar o novo líder, ou deve w se tornar escravo de u e u assumir o papel de novo líder?

Para que o novo componente seja válido, as propriedades devem ser respeitadas após a reorganização. Como $h(u) < h(w)$, u não pode ser mestre de w, então, u se torna escravo de w. Os escravos de u tornam-se escravos de w pois a terceira propriedade diz que todo escravo não compartilhado deve possuir h menor ou igual ao h de seu mestre. Logo, todos os escravos não-compartilhados de u possuem um h menor que u, e, por transitividade, menor que w. Consequentemente, o algoritmo pode concluir que w é o novo líder, e todos os escravos de u se tornam escravos de w sem que nenhuma propriedade seja violada, conforme ilustrado na figura 1.

Os demais casos seguem processos similares para gerar componentes válidos. Dois resultados garantem a preservação das propriedades por Bluemob e, consequentemente, sua vivacidade e corretude, conforme segue.

Lema. *Dado que dois componentes A e B respeitam as propriedades antes da união, segue que: (1) o componente resultante possui um único líder; (2) o líder do componente resultante possui no máximo seis escravos; (3) não existe nenhum mestre com mais de sete escravos no componente resultante; (4) o líder do componente resultante possui pelo menos um escravo não-compartilhado; (5) todos os escravos do componente resultante que não são pontes possuem h menor ou igual ao h de seu mestre.*

Teorema. *Todo componente respeita as seguintes propriedades: (1) o componente possui um único líder; (2) o líder deve possuir no máximo seis escravos; (3) os mestres devem possuir menos que sete escravos; (4) o líder deve possuir um escravo não-compartilhado ou nenhum escravo; (5) todo escravo que não é uma ponte possui h menor ou igual ao h de seu mestre.*

As provas do lema e do teorema podem ser encontradas no relatório técnico [Pieri 2014].

4. Avaliação de Bluemob

O algoritmo Bluemob deve ter desempenho suficiente para reconstruir a rede de nós a bordo de um ônibus após o desembarque de passageiros num tempo menor que o tempo

típico de viagem entre pontos de paradas. Também, pode-se avaliar a melhoria de Bluemob em relação ao algoritmo de Law usado como base. Esta seção realiza tal avaliação usando como métrica comparativa o número de rodadas para formação e reformação da rede. São avaliados os efeitos de (1) iniciar a formação de uma rede a partir de componentes individuais; (2) passageiros deixando a rede e possivelmente particionando-a; e (3) precisão das estimativas do peso h.

Os experimentos usam um simulador especialmente desenvolvido em Python. Este simulador não leva em consideração detalhes da pilha Bluetooth. Isto é possível pois os algoritmos simulados baseiam-se em tarefa periódica (ou *rodada*) de período igual a δ. Logo, o tempo para a união de todos os componentes (i. é, para obtenção de conectividade total da rede) é δ multiplicado pelo número de rodadas.

Naturalmente, é preciso conhecer o valor de δ para concluir algo sobre os tempos envolvidos na formação da rede. O limite inferior de δ é o tempo necessário para concluir a união de quaisquer dois componentes desde o INQUIRY ou INQUIRY SCAN do líder.

Através de experimentos no simulador UCBT [Wang and Agrawal], principal simulador Bluetooth de código aberto, observou-se que o maior tempo para reorganizar dois componentes foi de 7.75 s. Este tempo foi medido através da simulação de todos os treze casos possíveis listados na Seção 3, variando o tamanho das piconets e os pesos h de u, w, y, v de tal forma que todas as ordens relativas entre os pesos fossem contempladas.

Nota-se que o tempo da rodada do algoritmo de Law é pelo menos da mesma magnitude daqueles verificados para Bluemob. Isto porque a reorganização mais demorada (número 6) de Bluemob cujo tempo máximo é 7.75 s é idêntica a uma das reorganizações usadas no algoritmo Law. Logo, δ de Law tem limite inferior igual a 7.75 s. Deste ponto em diante, escolhemos que $\delta = 8$ s.

4.1. Primeiro experimento: formação a partir de uma rede desconectada

Neste experimento é avaliado o comportamento de Bluemob para formar uma rede a partir de nós desconectados. Este cenário equivale à formação da rede no ônibus entre seu ponto de origem até o primeiro ponto de parada. A simulação é executada até ser formada uma scatternet conectada com todos os nós. O número de rodadas necessárias para formar a *scatternet* é mostrado na Figura 2) para quantidades de nós variando entre 2 e 128.

Observa-se que o tempo para formação da rede está entre $\delta \times 10 = 80$ s e $\delta \times 11 = 88$ s com 128 passageiros. Dado que o tempo típico entre paradas é 60 s, com 128 passageiros não se consegue formar a rede até a próxima parada. Entretanto, esta situação não é usual durante a trajetória do ônibus, sendo observada normalmente apenas na origem do itinerário. Neste ponto, os desembarques não costumam ocorrer durante as primeiras paradas, permitindo ultrapassagem da restrição.

O número de rodadas aumenta logaritmicamente em relação ao número de componentes, não havendo aumentos expressivos no tempo de formação observado com o aumento de passageiros.

Em comparação ao algoritmo Law, pode-se ver na Figura 2 que não há diferenças em relação ao número de rodadas necessárias para a formação da rede. Isto é, não há penalidade devido à maior complexidade da união de componentes de Bluemob. Este comportamento pode ser explicado pela forma como a busca por componentes é realizada

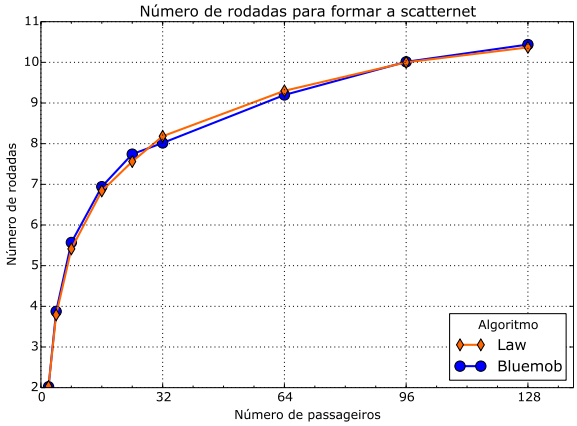
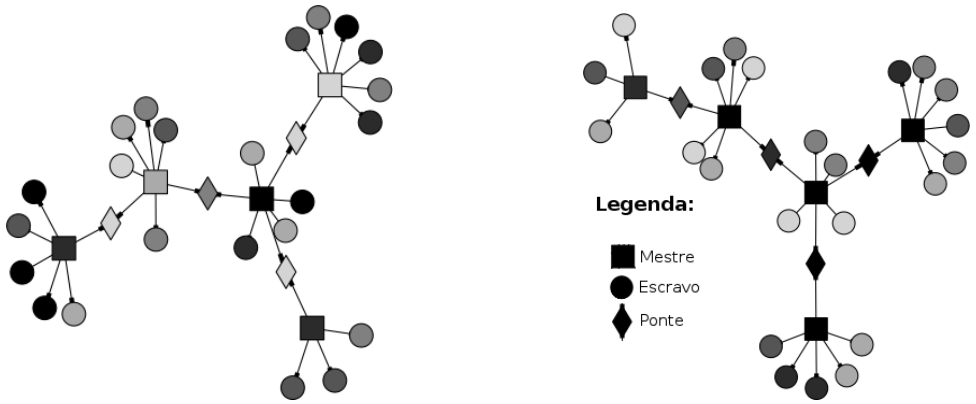


Figura 2. Número de rodadas para formação de rede a partir de uma rede desconectada, quantidade de nós variando de 0 a 128



(a) Law et. al (b) Bluemob

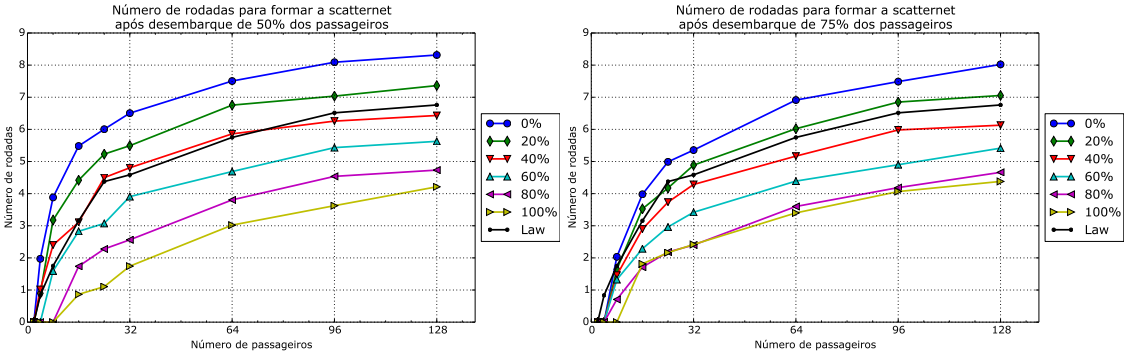
Figura 3. Uma rede Bluetooth formada pelo (a) algoritmo Law, e (b) algoritmo Bluemob; quanto mais clara a tonalidade do nó, maior sua propensão a deixar a rede

em rodadas [Law et al. 2003].

Do ponto de vista da topologia da rede formada, há uma melhora qualitativa no posicionamento dos nós, conforme mostrado na Figura 3, na qual os nós com tonalidade mais clara apresentam maior propensão a deixar a rede. Na rede formada com Bluemob (Figura 3(b)) mestres e pontes tem tonalidade mais escura que os escravos. Logo, quando um nó deixa a rede nesta topologia, é geralmente um escravo e não ocorrem particionamentos. Já na rede formada pelo algoritmo Law (Figura 3(a)) vários mestres e pontes possuem tonalidade clara, indicando maior chance de particionamento da rede por conta de desembarques.

4.2. Segundo experimento: tempo de formação após desembarque de nós

O objetivo do segundo experimento é avaliar o efeito do arranjo de nós quando passageiros deixam a rede, mostrando as melhoras obtidas com a introdução do peso h no procedimento de reorganização durante o desembarque. Este experimento consiste em, dado n nós no ponto 0 do itinerários, construir uma scatternet. No momento em que a scatternet se torna conectada, uma fração dos nós é removida da rede, particionando-a para simular



(a) saída de 50% dos nós da rede (b) saída de 75% dos nós da rede

Figura 4. Experimento reconstruindo scatternet após desembarque

desembarques no ponto 1. A partir deste instante, mede-se o número de rodadas para que a rede seja reconectada no trajeto entre pontos 1 e 2.

O método de atribuição de pesos h é o seguinte: aos nós que deixarão a rede é atribuído o valor 1 a h, enquanto aos nós que permanecerão é atribuído valor 2 a h. A precisão de $x\%$ indica a porcentagem de passageiros com h correto. Assim, $100\% - x\%$ dos nós que deixam a rede tem seu peso h modificado de 1 para 2, enquanto $100\% - x\%$ dos que permanecem na rede tem seu peso h modificado de 2 para 1.

Os experimentos foram simulados 400 vezes para cada uma das situações. O número médio de rodadas para reconectar a rede é mostrado na Figura 4. Dois experimentos foram feitos: com 50%, Fig. 4(a), e 75%, Fig. 4(b), dos nós deixando a rede. Pelas figuras, observa-se novamente o aumento logarítmico do número de rodadas com o número de nós. Em relação à precisão das estimativas, fica claro que quanto melhor for estimada a saída, melhor Bluemob se comporta. Nota-se que no pior caso, com precisão 0% e 128 passageiros, o tempo para reconstrução da rede ultrapassa $60s$. Com o aumento da precisão de h obtém-se tempos na faixa $30s$, atingindo o objetivo de reconstrução da rede entre paradas. Observa-se que a partir do momento em que precisão de h ultrapassa 50%, Bluemob supera o desempenho de Law. Em alguns casos observa-se melhora de até 75% do tempo necessário para reconstruir a rede após saída de parte dos nós.

Quando h possui precisão menor que 50%, Bluemob coloca como mestres e pontes passageiros com alta propensão de desembarque. Com isso, há uma perda de desempenho em relação a Law, que posiciona os nós de forma aleatória na rede.

4.3. Terceiro experimento: tempos de formação durante trajetória

Este experimento tem como objetivo mostrar o comportamento do algoritmo ao longo da trajetória de um ônibus com uma série de embarques e desembarques. Para tanto, assume-se uma trajetória do ônibus conforme mostrado na Fig. 5. A trajetória do ônibus ocorre ao longo de 17 paradas, numeradas sequencialmente sendo a primeira parada a de número 0. Na parada zero ocorre o embarque de 64 passageiros. Nas paradas ímpares, há o desembarque de 16 passageiros e nas paradas pares embarcam novos 16. Os passageiros que desembarcam são aqueles embarcados a mais tempo, e o peso h é igual ao número do ponto estimado para desembarque. A precisão de h indica a porcentagem de passageiros com h correto.

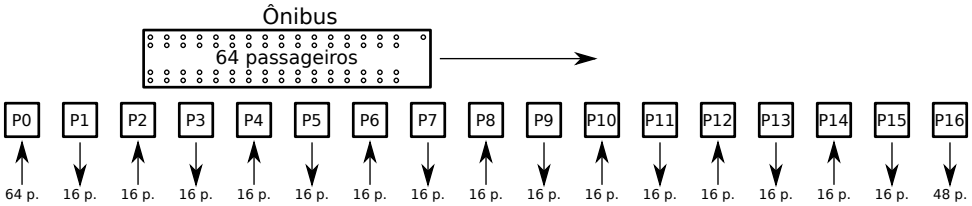


Figura 5. Trajetória de ônibus para teste de reformação da rede em itinerário

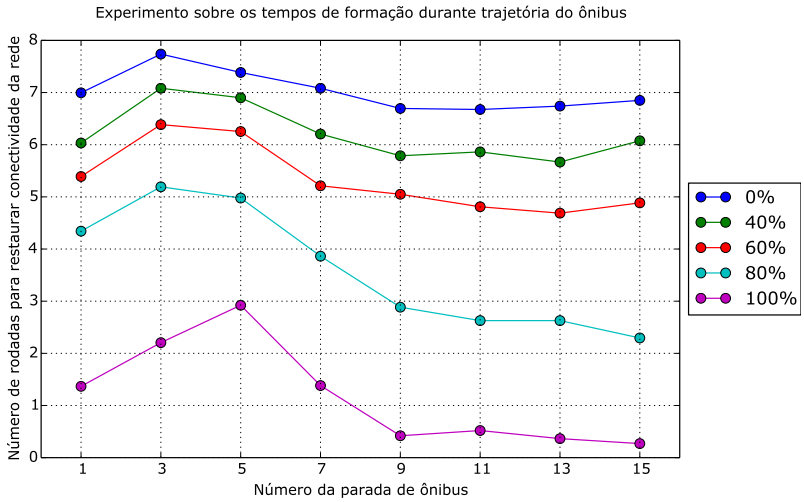


Figura 6. Efeito da saída de passageiros durante trajetória com 64 passageiros 16 desembarques seguidos de 16 embarques alternadamente

Para Bluemob, o importante para definir a corretude de h é a corretude das relações de ordem entre os pesos h. Portanto, o h do passageiro que desembarca em um ponto de parada está correto caso seja menor que o pesos dos passageiros que irão desembarcar posteriormente. Logo, atribui-se aos passageiros com h errados o número de parada menos o número da sua parada de embarque. Por exemplo, passageiros com h errado que embarcaram no ponto de parada 0 tem $h = 17$. Já passageiros que embarcam em 2, terão $h = 17 - 2 = 15$

Na figura 6 é mostrado o efeito da saída de nós da rede quando há uma sequência de entradas e saídas de nós na rede e o efeito em Bluemob. Observa-se que neste experimento os tempos para as reorganizações diminuem durante a trajetória do ônibus. Isto ocorre devido ao modo de operação de Bluemob. Há uma tendência a formação de subredes compostas por passageiros antigos já que dois componentes se unem a partir das piconets dos líderes. Caso haja poucos particionamentos da rede, aglomerados de passageiros antigos são formados. Ao desembarcarem, estes aglomerados de passageiros antigos tendem a desembarcar todos juntos, diminuindo assim o número de partições causadas pelo desembarque.

Após os desembarques iniciais, com os passageiros mais antigos desembarcando primeiro, há uma tendência a estabilização do número de rodadas para reestabelecer a conectividade. Após quatro rodadas de desembarques, praticamente não há necessidade de reorganizações para reconectar a rede quando precisão de h é 100%.

5. Conclusão e Perspectivas

Este trabalho apresenta um novo algoritmo BSF, denominado Bluemob, especificamente projetado para construir redes Bluetooh ad-hoc entre passageiros no interior de ônibus.

O algoritmo proposto melhora algoritmos encontrados na literatura levando em consideração a propensão de um passageiro a deixar a rede, usando esta informação para influenciar a topologia da rede. Passageiros propensos a deixar a rede são posicionados nas bordas da scatternet de tal forma que o seu desembarque não cause o particionamento da rede.

Resultados experimentais obtidos através de simulações são apresentados. Estas simulações mostram que Bluemob melhora algoritmos existentes, desde que provido com um mecanismo para estimar a propensão de desembarques suficientemente preciso. A distribuição dos nós na rede resultante é melhorado, o que é desejável em uma rede formada por passageiros em transporte público. Isto acarreta melhora no tempo necessário para reconstrução da rede após desembarques. Mais importante, o tempo para construir redes encontram-se dentro da faixa aceitável de menos de 1 minuto, tempo típico para viagem de um ônibus entre paradas.

Como trabalhos futuros e área de melhora, pode-se citar os seguintes aspectos: (1) aperfeiçoar o algoritmo para que a topologia evolua para uma distribuição ótima de nós dentro da scatternet, (2) investigar e avaliar métodos para atribuição e comparação da propensão de permanência no ônibus; (3) investigar outras áreas de aplicação, como cenários com restrições energéticas nos quais nós com menor energia possam ser colocados nas bordas da scatternet; (4) desenvolver estudos e comparativos usando dispositivos reais.

Agradecimentos

Os autores agradecem o apoio do CNPq e do projeto SPACeS (Sistema PArticipativo de Gestão de Cidades e Serviços Públicos) do CTIC-RNP.

Referências

Ben Moshe, B., Hadas, Y., and Levi, H. (2014). Energy-Efficient Framework for Indoor and Outdoor Tracking of Public Transit Passengers Using Bluetooth-Enabled Devices. In *Transportation Research Board 93rd Annual Meeting*, pp. 16–30, Washington DC.

Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., and Srivastava, M. B. (2006). Participatory sensing. In *Proc. Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications*, pp. 117–134.

Campbell, A. T., Eisenman, S. B., Lane, N. D., Miluzzo, E., and Peterson, R. A. (2006). People-centric urban sensing. In *Proc. 2nd Annual International Wireless Internet Conference (WICON '06)*, pp. 2–5, New York, NY, USA. ACM.

Jedda, A., Casteigts, A., Jourdan, G.-V., and Mouftah, H. T. (2013). Bluetooth scatternet formation from a time-efficiency perspective. *Wireless Networks*, 20(5):1133–1156.

Law, C., Mehta, A. K., and Siu, K. Y. (2003). A New Bluetooth Scatternet Formation Protocol. *Mobile Networks and Applications*, 8:485–498.

Lin, C.-Y., Chen, L.-J., Chen, Y.-Y., and Lee, W.-C. (2010). A comfort measuring system for public transportation systems using participatory phone sensing. In *Proc. of First Int. Workshop on Sensing for App Phones (PhoneSense'10)*, Zurich, Switzerland.

Mohan, P., Padmanabhan, V. N., and Ramjee, R. (2008). Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proc. 6th ACM Conf. on Embedded Networked Sensor Systems (SenSys '08)*, pp. 323–336, New York, NY, USA.

Moovit. Moovit. `http://www.moovitapp.com/`. Acessado em: 2012-12-12.

Pei, L., Chen, R., Liu, J., Kuusniemi, H., Tenhunen, T., and Chen, Y. (2010). Using inquiry-based bluetooth rssi probability distributions for indoor positioning. *Journal of Global Positioning Systems*, 9(2):122–130.

Petrioli, C., Basagni, S., and Chlamtac, I. (2003). Configuring bluestars: multihop scatternet formation for bluetooth networks. *IEEE Trans. on Computers*, 52(6):779–790.

Petrioli, C., Basagni, S., and Chlamtac, I. (2004). BlueMesh: Degree-Constrained Multi-Hop Scatternet Formation for Bluetooth Networks. *Mobile Netw. and Applications*, 9:33–47.

Pieri, G. (2014). Bluemob: Description of a BSF Algorithm for Building Bus Riders Bluetooth ad-hoc networks. Technical report, Departamento de Controle e Automação, Universidade Federal de Santa Catarina, Florianópolis, Santa Catarina. `http://www.das.ufsc.br/~werner/workInProgress/bluemob/TR-algorithmDescription.pdf`.

Reddy, S., Mun, M., Burke, J., Estrin, D., Hansen, M., and Srivastava, M. (2010). Using mobile phones to determine transportation modes. *ACM Trans. Sen. Netw.*, 6(2):13:1–13:27.

Salonidis, T., Bhagwat, P., Tassiulas, L., and LaMaire, R. (2005). Distributed topology construction of Bluetooth wireless personal area networks. *IEEE Journal on Selected Areas in Communications*, 23:633–643.

Stojmenovic, I. and Zaguia, N. (2006). Bluetooth scatternet formation in ad hoc wireless networks. *Performance Modeling and Analysis of Bluetooth Networks*, pp. 147–171.

Tan, G., Miu, A., Guttag, J., and Balakrishnan, H. (2002). An efficient scatternet formation algorithm for dynamic environments. In *Proceedings of the IASTED Communications and Computer Networks (CCN)*, vol. 1. Citeseer.

Thiagarajan, A., Biagioni, J., Gerlich, T., and Eriksson, J. (2010). Cooperative transit tracking using smart-phones. In *Proc. 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*, p. 85–98, New York, NY, USA.

Wang, Q. and Agrawal, D. Ucbt- bluetooth extension for ns2 at the university of cincinnati. `http://www.ececs.uc.edu/~cdmc/ucbt/`. Accessed: 2014-03-30.

Wang, Z. W. Z., Thomas, R., and Haas, Z. (2002). Bluenet - a new scatternet formation scheme. *Proc. of the 35th Annual Hawaii Int. Conf. on System Sciences*, 00(c):1–9.

Waze Mobile. Waze. `https://www.waze.com/`. Acessado em: 2012-12-12.

Zaguia, N., Daadaa, Y., and Stojmenovic, I. (2008). Simplified bluetooth scatternet formation using maximal independent sets. *Proceedings - CISIS 2008: 2nd International Conference on Complex, Intelligent and Software Intensive Systems*, pp. 443–448.