Um Algoritmo Guloso para Configuração e Reconfiguração de *Scatternets* Bluetooth

Wanderley Ravagnani Junior, Jacques Wainer, Edmundo R. M. Madeira

[1]IC - Instituto de Computação
UNICAMP - Universidade Estadual de Campinas
CEP 6176 13083-970 Campinas, SP, Brasil
{wanderley.junior, wainer, edmundo}@ic.unicamp.br

Resumo. *Bluetooth é uma tecnologia de rádio que permite dispositivos eletrônicos comunicarem a curta distância através de redes ad-hoc sem fio. Este artigo apresenta um algoritmo para configurar e reconfigurar scatternets Bluetooth, na tentativa de satisfazer as necessidades específicas de comunicação entre cada par de dispositivos da rede. O artigo também mostra algumas simulações para testar a eficiência deste algoritmo.*

1. Introdução

A tecnologia Bluetooth é uma especificação de um padrão aberto para comunicação sem fio, de curto alcance e baixo custo entre dispositivos, através de conexões de rádio. Esta tecnologia visa a criação de um sistema de rádio, onde os dispositivos se conectam formando uma pequena rede denominada ***piconet***, na qual podem existir oito (8) dispositivos se comunicando a uma taxa máxima de 1 Mbit/s. Além disso, podemos encontrar em ambientes Bluetooth, várias *piconets* sobrepostas, comunicando entre si, formando uma rede *ad-hoc* chamada ***scatternet***. Numa *scatternet*, as *piconets* se comunicam compartilhando um ou mais dispositivos que "saltam"de uma *piconet* para outra. Esses dispositivos são denominados **pontes**. Já a taxa de comunicação entre *piconets* depende do número de pontes e de uma política de escalonamento que define o tempo que uma ponte deve estar ativa em uma *piconet* e quando essa ponte deve "saltar"para a outra *piconet*. Alguns trabalhos mostraram que esta taxa é aproximadamente de até 100 kbits/s para cada ponte.

O problema da formação de *scatternets* Bluetooth é um assunto muito importante a ser abordado, já que a especificação da tecnologia Bluetooth não trata diretamente deste assunto, ou seja, não diz qual é o protocolo utilizado na criação deste tipo de rede. Além disso, o tempo gasto e o modo como esta rede é formada pode afetar diretamente o desempenho da rede. Na literatura são encontrados alguns trabalhos abordando este assunto [5, 7, 9, 10], onde as *scatternets* são criadas de forma distribuída. Entretanto, estes trabalhos estão preocupados somente com o tempo de formação da *scatternet* e com o número de mensagens trocadas. Porém, estes não levam em consideração a necessidade de comunicação entre os dispositivos que formarão a rede. Isso pode resultar numa perda significativa de desempenho da rede, já que desta forma podemos ter numa mesma *piconet* dispositivos que queiram se comunicar muito pouco e, em *piconets* diferentes, distantes umas das outras, dispositivos que queiram se comunicar a uma alta taxa, como por exemplo, dois *laptops* que compartilham uma aplicação distribuída. Tendo em vista que a taxa

de comunicação interna de uma *piconet* (1 Mbit/s) pode ser quase dez (10) vezes maior que a taxa de comunicação entre duas *piconets* (100 kbit/s), seria muito interessante que os dispositivos, que necessitam se comunicar a uma alta taxa, ficassem numa mesma *piconet* ou em *piconets* próximas para gerar menos *overhead* de comunicação. Sendo assim, a proposta deste trabalho é configurar uma *scatternet* Bluetooth na tentativa de satisfazer as necessidades específicas de comunicação entre cada par de dispositivos da rede, ou seja, dispor estes dispositivos na rede de maneira que não falte banda de comunicação entre eles. Para tanto, criamos um **Algoritmo Guloso** que tem como objetivo dispor os dispositivos que mais necessitam se comunicar, o mais próximo possível na rede.

Devido a complexidade destes problemas, algumas restrições foram feitas para modelar nossa proposta de configuração das redes Bluetooth de forma a satisfazer as necessidades de comunicação descritas anteriormente:

- A *scatternet* possui N dispositivos, e os N dispositivos estão próximos entre si, de forma que quaisquer dois dispositivos possam se comunicar;
- A *scatternet* é **conexa**, ou seja, existe pelo menos um caminho ligando quaisquer duas *piconets* da *scatternet* através de pontes;
- Uma ponte deve conectar somente duas *piconets*, pois assim ela não fica sobrecarregada;
- Como o *throughput* entre *piconets* ainda é uma questão aberta, nós assumimos que o *throughput* suportado por uma ponte que conecta quaisquer duas *piconets* é no máximo de 100 kbits/s, pois os trabalhos [8, 4] mostraram que para a maioria das políticas de escalonamento por eles testadas, esse valor é de aproximadamente 100 kbits/s.

Numa primeira versão deste trabalho ([3]) usamos uma modelagem mais simples onde as *scatternets* configuradas seriam totalmente conexas (ou seja, todas as *piconets* estariam conectadas entre si), e a taxa máxima de comunicação entre duas *piconets* era de no máximo 100 kbits/s (pois não levamos em consideração o número de pontes que conectavam estas *piconets*), mas o algoritmo guloso é fundamentalmente o mesmo. O leitor pode consultar esta referência para obter mais detalhes.

Para testar a eficiência do algoritmo guloso construímos um gerador aleatório de redes Bluetooth onde as redes geradas são de alguma forma factíveis (ou seja, cada rede possui pelo menos uma solução) em relação às necessidades de comunicação entre os dispositivos da rede. A idéia deste gerador é construir uma rede com uma determinada topologia e gerar tráfego aleatório entre os dispositivos que formam a rede de maneira que não falte banda de comunicação. A saída do gerador é uma matriz triangular inferior de *throughputs* (MT) que representa quanto cada dispositivo deseja se comunicar com os demais dispositivos que formam a rede.

Três topologias diferentes de rede foram utilizadas: Anel, Aleatória e Triângulo-Aresta. Na topologia Aleatória, a rede gerada pode ter qualquer topologia. Entretanto assumimos que nesta topologia Aleatória, cada *piconet* possuirá no mínimo uma e no máximo três pontes, pois utilizando mais que três pontes por *piconet*, degradaríamos o seu desempenho ([1, 2]). Já na topologia Triângulo-Aresta a rede gerada possuirá primeiramente um triângulo de *piconets*, seguido de uma aresta, seguido de um triângulo, seguido de uma aresta, e assim consecutivamente. Esta topologia de rede foi utilizada, pois sua estrutura assemelha-se com a idéia gulosa do algoritmo conectando as *piconets* que mais

precisam se comunicar (*piconets* geradas consecutivamente) de forma não-linear. Ela também respeita o limite máximo de três pontes por *piconet*.

Este artigo está estruturado da seguinte maneira: A Seção 2 descreve o algoritmo guloso, a Seção 3 apresenta os resultados obtidos pelo algoritmo na configuração e reconfiguração de *scatternets* Bluetooth, e finalmente a Seção 4 traz a conclusão e trabalhos futuros.

2. Algoritmo Guloso

A idéia do nosso algoritmo guloso é colocar os dispositivos que mais precisam se comunicar, dentro de uma mesma *piconet*, pois o canal *intra-piconet* (comunicação entre os dispositivos de uma mesma *piconet*) possui o maior *throughput* (1 Mbit/s). Se isso não for possível, esses dispositivos são colocados em *piconets* próximas, sendo que quanto mais próximas eles estiverem na rede, menos tráfego entre *piconets* será gerado, já que as pontes tendem a ser os gargalos da rede, pois essas ficam alternando de uma *piconet* para outra em intervalos de tempo. O algoritmo recebe como entrada o número de dispositivos (N) e a matriz de *throughputs* desejados (MT).

Devido à extensão do algoritmo não foi possível descrevê-lo neste artigo. Portanto, o leitor deve consultar as referências [1, 2] para obter uma descrição detalhada deste.

A complexidade total do algoritmo guloso é exponencial. Entretanto, como será visto na próxima seção, limitaremos o seu tempo de execução em 1,5 segundos, uma vez que um usuário geralmente está disposto a esperar um tempo maior que este.

É importante observar que este é um algoritmo de satisfação que utiliza uma heurística gulosa para tentar satisfazer a necessidade de comunicação da rede.

3. Resultados Obtidos

Os testes foram feitos num computador com processador Intel Pentium II 300, com 192 Mbytes de RAM, usando sistema operacional Windows 98, e os algoritmos foram implementados na linguagem C.

Na configuração, supomos que a necessidade de comunicação entre os dispositivos da rede é previamente conhecida. Para isso, utilizamos o nosso gerador de redes aleatórias Bluetooth para construir redes com as topologias Anel, Aleatória e Triângulo-Aresta. Para cada uma delas geramos 4000 redes, sendo 500 para cada tamanho de rede em número de *piconets*, de 3 a 10. Os dados foram calculados para um tempo máximo de execução do algoritmo de 1,5 segundos, que pode ser considerado um tempo razoável para diversos tipos de aplicações Bluetooth.

Os resultados obtidos na configuração são apresentados a seguir (para obter mais detalhes sobre os resultados obtidos consulte as referências [1, 2]):

Para as redes geradas e resolvidas com a topologia Anel, o algoritmo guloso conseguiu satisfazer 99,48% das redes. Já para as redes geradas com a topologia Aleatória e resolvidas com a topologia Anel, o algoritmo guloso conseguiu satisfazer 75,75% das redes testadas.

Na tentativa de melhorar este índice de satisfação utilizamos a topologia Triângulo-Aresta que assemelha-se com a idéia gulosa do algoritmo. Sendo assim, geramos as redes com a topologia Aleatória e resolvemos com a Triângulo-Aresta. Desta forma o algoritmo conseguiu satisfazer 83,13% das redes.

Finalmente, para as redes geradas e resolvidas com a topologia Triângulo-Aresta, o algoritmo guloso conseguiu satisfazer 91,5% das redes testadas.

Para simular uma reconfiguração, construímos as redes através do nosso gerador (redes factíveis) e em seguida aumentamos o valor da necessidade de comunicação de apenas um enlace que conecta quaisquer dois dispositivos da rede, de forma a extrapolar o *throughput intra-piconet* ou *extra-piconet* máximo. A intenção aqui é simular uma rede já configurada, que depois de um certo tempo, passa a não suportar a necessidade de comunicação. É importante ressaltar que quando ocorre essa mudança na rede, não sabemos se essa nova rede é factível, pois a rede antiga, apesar de ser construída aleatoriamente, possui uma alta taxa de comunicação.

Para um efeito comparativo ao nosso algoritmo guloso, construímos um algoritmo de busca local, que utiliza a estrutura (topologia) da rede atual como ponto de partida para tentar resolver o problema, enquanto o nosso algoritmo cria uma rede totalmente nova sem saber a topologia antiga.

Na reconfiguração, geramos o mesmo número de redes da configuração, entretanto utilizamos as topologias Aleatória e Triângulo-Aresta, pois esta última foi a que obteve o melhor resultado na configuração de redes geradas com uma topologia Aleatória. O tempo de execução máximo foi mantido em 1,5 segundos.

Os resultados obtidos na reconfiguração são apresentados a seguir:

Para as redes geradas e resolvidas com a topologia Triângulo-Aresta, o algoritmo guloso conseguiu satisfazer 39,93% das redes. Já o algoritmo de busca local conseguiu satisfazer apenas 12,78% das redes testadas.

Para as redes geradas com a topologia Aleatória e resolvidas com a Triângulo-Aresta, o algoritmo guloso conseguiu satisfazer 36,78% das redes testadas. O algoritmo de busca local conseguiu satisfazer apenas 11,35% das redes.

4. Conclusão e Trabalhos Futuros

Este trabalho apresenta um algoritmo guloso para configurar e reconfigurar *scatternets* Bluetooth de acordo com as necessidades de comunicação entre os dispositivos, na tentativa de satisfazê-las.

Os resultados mostraram que o algoritmo foi muito eficiente na configuração de redes geradas e resolvidas em Anel (99,48%) e que a melhor topologia a ser utilizada na configuração de redes geradas com uma topologia Aleatória é a Triângulo-Aresta (83,13%), pois sua estrutura assemelha-se com a idéia gulosa do algoritmo. Na reconfiguração, tanto para redes geradas e reconfiguradas em Triângulo-Aresta, quanto para redes geradas com uma topologia Aleatória e reconfiguradas em Triângulo-Aresta, o algoritmo guloso foi três vezes mais eficiente que o algoritmo de busca local, construído para efeito comparativo.

Uma observação importante a ser feita é que a idéia gulosa, utilizada pelo algoritmo, é muito eficiente na resolução deste tipo de problema, colocando os dispositivos que mais necessitam se comunicar próximos na rede, gerando assim, menos tráfego entre *piconets*.

Como trabalhos futuros, pretendemos testar o algoritmo guloso inserindo cenários onde uma ponte pode estar conectada a mais de duas *piconets* e investigar novas topologias de rede que melhorem ainda mais o índice de satisfação das *scatternets* Bluetooth.

Referências

[1] Wanderley Ravagnani Junior, Edmundo R. M. Madeira, and Jacques Wainer. A greedy algorithm to configure and reconfigure bluetooth scatternets. *The Fifth IEEE Conference on Mobile and Wireless Communications Networks (IEEE MWCN'2003)*, Cingapura, Outubro 2003.

[2] Wanderley Ravagnani Junior, Edmundo R. M. Madeira, and Jacques Wainer. Um algoritmo guloso para configuração e reconfiguração de scatternets bluetooth. *Seminário Integrado de Software e Hardware (SEMISH) - SBC*, Campinas, Agosto 2003.

[3] Wanderley Ravagnani Junior, Jacques Wainer, and Edmundo R. M. Madeira. Um algoritmo para configuração de scatternets bluetooth. *IV Workshop de Comunicação sem Fio e Computação Móvel*, pp. 184–190, São Paulo, Outubro 2002.

[4] Manthos Kazantzidis and Mario Gerla. On the impact of inter-piconet scheduling in bluetooth scatternets. *International Conference on Internet Computing*, pp. 37–43, 2002.

[5] Ching Law, Amar K. Mehta, and Kai-Yeung Siu. Perfomance of a new bluetooth scatternet formation protocol. *ACM Symposium on Mobile ad Hoc Networking and Computing*, Outubro 2001.

[6] Werner Priess, José Ferreira de Rezende, Luiz Fernando Rust da C. Carmo, and Luci Pirmez. Bluenets: Uma ferramenta para estudo do desempenho do bluetooth. *IV Workshop de Comunicação sem Fio e Computação Móvel*, pp. 161–171, São Paulo, Outubro 2002.

[7] Lakshmi Ramachandran, Manika Kapoor, and Alok Aggarwal. Clustering algorithms for wireless ad hoc networks. *SIG Mobile*, 2001.

[8] András Rácz, György Miklós, Ferenc Kubinzky, and András Valkó. A pseudo random coordinated scheduling algorithm for bluetooth scatternets. *ACM Symposium on Mobile ad Hoc Networking and Computing*, Outubro 2001.

[9] Theodoros Salonidis, Pravin Bhagwat, Leandros Tassiulas, and Richard LaMaire. Proximity awareness and ad hoc establishment in bluetooth. *Infocom*, 2001.

[10] Sergio Gonzalez Valenzuela, Son T. Vuong, and Victor C. M. Leung. Efficient formation of dynamic bluetooth scatternet via mobile agent processing. *5th International Workshop on Mobile Agents for Telecommunication Applications - MATA'03*, pp. 1–10, Marrakech, Marrocos, LNCS n. 2881, Springer-Verlag, Outubro 2003.