Efficient network flooding simulation with glossy

Bruno Miguel Silva Neiva (201106893) Ília Marlene Pereira Azevedo (201303893) Nuno André Teixeira Moreira (201305040)

Mestrado Integrado em Engenharia Eletrotécnica e computadores

Faculdade de Engenharia da Universidade do Porto
14 de Janeiro de 2018

I. Resumo

Uma rede de sensores sem fios designa a interligação de um determinado números de equipamentos com sensores de forma a observar o meio físico. O mecanismo baseia-se na recolha, transmissão e receção de dados através de sensores.
Uma arquitetura para redes de sensores sem fio é a Glossy[1], que explora interferências construtivas dos símbolos IEEE 802.15.4, uma vez que ocorrem transmissões simultâneas do mesmo pacote, permitindo que os recetores recebam, com alta probabilidade, os pacotes transmitidos. Assim o Glossy alcança uma confiabilidade acima dos 99.99% e melhora a latência de propagação associada à densidade dos nós e ao tamanho da rede. Além disso, o Glossy fornece sincronização temporal em toda a rede, uma vez que é necessária a sincronização de todos os nós à medida que o pacote se propaga na rede. Esta arquitetura é formada por dois tipos de nós, o iniciador que inicia a transmissão do pacote, e os recetores, que recebem o pacote e de seguida transmitem para os recetores vizinhos.
Assim este trabalho tem como principal objetivo a simulação desta arquitetura.

II. Arquitetura Glossy

A. *Topologia geral da rede*

A rede é formada principalmente por dois tipos de nós, o iniciador que inicia a transmissão do pacote, e os recetores que recebem os pacotes transmitidos e retransmitem aos seus nós vizinhos. Os nós estão dispostos por camadas, e como se pode observar na figura 1, os nós com a mesma cor, do mesmo nível, recebem o pacote dos nós da camada anterior, e de seguida transmitem para os nós da camada seguinte. É importante realçar que os nós da mesma camada não transmitem o pacote entre si. Esta camada é atribuída tendo em conta o alcance (raio de transmissão de cada nó).

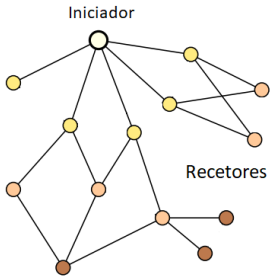

Figure 1: Topologia da rede

B. *Estados do glossy durante a execução*

Na figura 2, apresenta-se um diagrama de estados, onde se pode analisar o conjunto de operações que cada nó tem de desempenhar ao longo da execução da rede.
Se o nó for o iniciador, o seu estado inicial é o **Transmit**, pois apenas irá transmitir o pacote. Se o nó for um recetor, o seu estado inicial é o **Wait**, onde irá abrir um *socket* e esperar por um pacote. Quando um pacote é recebido, o estado muda para o **Receive**, aqui o pacote é descodificado. De seguida o nó irá transmitir, passando assim ao estado **Transmit** é incrementado um contador, *c*, que nos dá informação acerca do nível do nó, o pacote é codificado e transmitido por um *socket multicast* UDP.

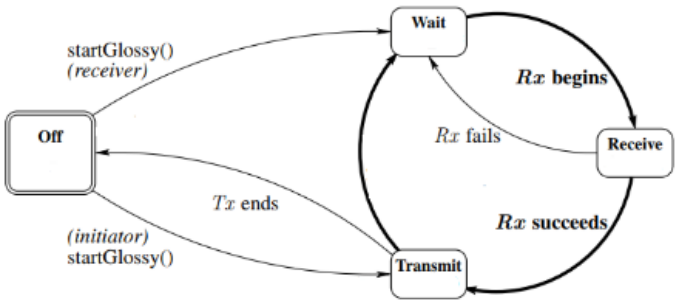

Figura 2: Máquina de estados

C. *Concorrência na transmissão*

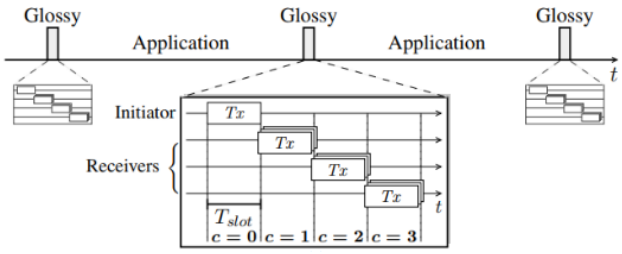

Figura 1: Diagrama temporal de pacotes

Ao longo da transmissão de pacotes entre várias camadas, é necessário garantir que as transmissões ocorram no mesmo instante de tempo, assim é necessário definir um tempo de transmissão, *Tslot* na figura 3. Também é necessário garantir que os nós da mesma camada estejam a receber a transmitir o mesmo pacote. Assim, na informação de cada pacote, existe um contador c, que é incrementado sempre que é transmitido por diferentes camadas, permitindo que o pacote tenha um indentificador por cada camada.

III. SOLUÇÃO

A. *Geração aleatória da topologia*

Inicialmente é gerada uma janela com altura e largura de 500×500px. De seguida, a largura da janela é dividida e gera-se um número aleatório de pontos por cada coluna de forma a obterem-se N pontos. Posto isto, é gerado um nó iniciador e os restantes nós em posições aleatórias com uma distância mínima entre si. Cada nó tem a si associadas coordenadas (x, y) e o um IP. São também definidas as diferentes camadas de rede utilizando um raio previamente definido que simula o alcance de um dado transmissor, permitindo assim associar um grupo de nós a uma camada e criar ligações entre eles.

B. *Formato dos pacotes*

Um pacote é uma trama de 8 bytes, e é formada pela *flag* 0x74 que indica o seu inicio, o contador para indicar em que camada está a ser transmitida, o tamanho dos dados e os seus dados.

Na figura 4 é possível verificar o formato da trama.

Start Flag	Counter	Data lenght	App Data
0x74	1 byte	1byte	0 - 127 bytes

Figura 4: trama

C. *Protocolo de Comunicação*

Optou-se por uma comunicação IP *multicast* para estabelecer comunicações entre um nó e os seus vizinhos.

O IP dos nós vizinhos pode ser obtido através da lista de nós vizinhos parar um determinado nó. Assim esse conjunto de endereços IP é associado a um grupo *multicast*.
Desta forma, o nó transmissor junta-se a esse grupo, e a transmissão ocorre.

D. *Algoritmo*

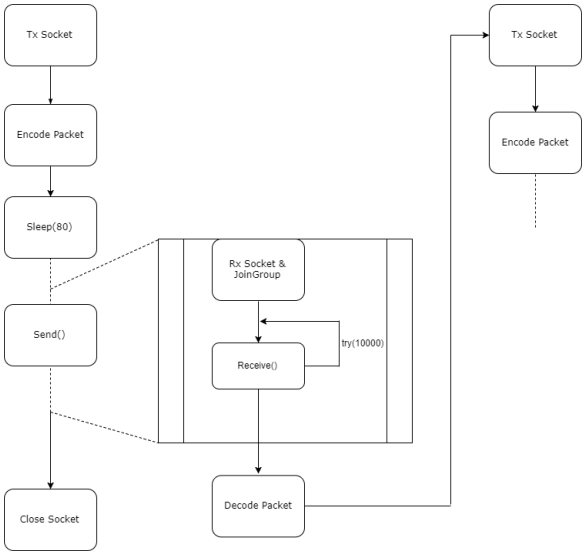

Figura 5: Diagrama de fluxo

Cada nó está associado a um thread de transmissão ou emissão, onde são iniciadas sockets para enviar ou receber informação.

Inicialmente o nó iniciador prepara-se para transmitir, codificando o pacote que pretende enviar e abrindo um *socket* para comunicação *multicast*, procedendo a uma pausa na execuçãodo seu *thread*, permitindo assim que os nós recetores abram as respetivas *sockets* para receção do pacote. Este tempo de 80ms foi obtido de forma empírica. Após este tempo o pacote é enviado. No caso da receção existem dois tipos diferentes de nós: apenas recetores, se forem extremos da rede, ou recetores seguidos de emissores. No primeiro caso o nó apenas recebe o pacote e descodifica-o tendo acesso à mensagem enviada; no segundo caso o nó recebe o pacote, descodifica-o, incrementa o contador *c*, abre *socket* para transmitir e pausa a sua execução para permitir que os seus recetores tenham tempo de abrir as suas *sockets*. O algoritmo é repetido até toda a rede receber o mesmo pacote. A figura 5 esquematiza o fluxo do programa.

E. *Interface*

De forma a visualizar todo o funcionamento da rede, foi implementada uma interface.

A figura 5, representa a janela inicial, onde é apresentada uma topologia *default* de 100 nós e com alcance de 75px, que foi criada aleatoriamente, mas seguindo algumas regras, nomeadamente garantir que existe uma distância mínima entre

diferentes nós de forma a que não exista quebra frequente da rede. Permite que o utilizador altere a mensagem a ser transmitida ao longo da rede, o número de nós, e o raio de cada camada. É possível, ainda, iniciar a simulação com a topologia *default* ou com a definida, e também é permitida a observação de um nó à escolha do utilizador. Na figura x verifica-se um exemplo de observação do nó, onde é possível ver os nós que estão a transmitir, e os que não receber do nó escolhido.

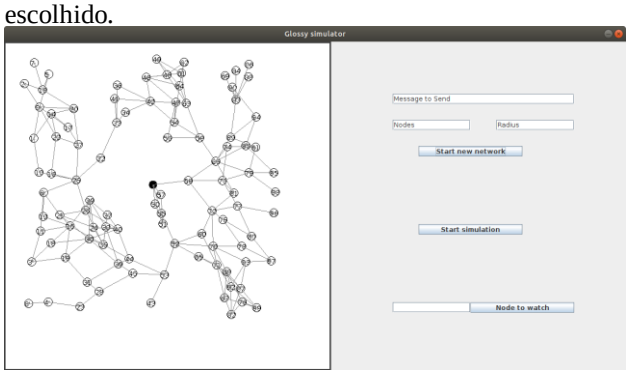

Figura 5: Janela principal

Quando uma topologia é simulada, é possível observar os nós e as suas respetivas camadas, como nos mostra a figura 7.

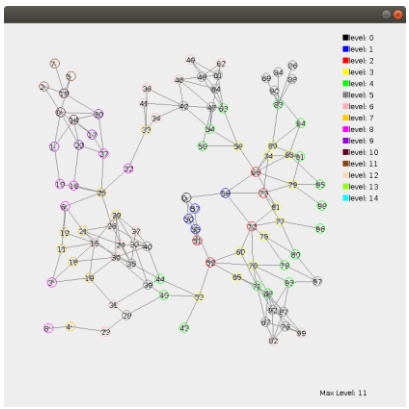

Figura 7: Topologia de rede

Além disso, é possível verificar a propagação dos nós por cada camada, clicando no botão "next level". Comparando as figuras 8 e 9 é possível observar como se propaga a informação ao longo da rede.

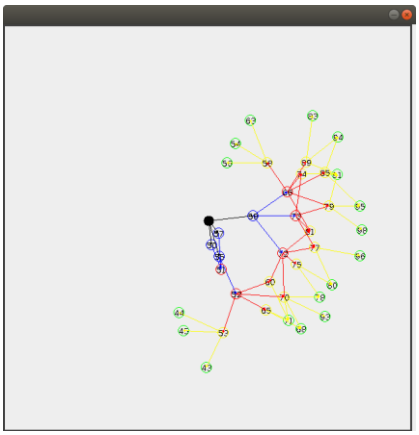

Figura 8: Evolução do fluxo

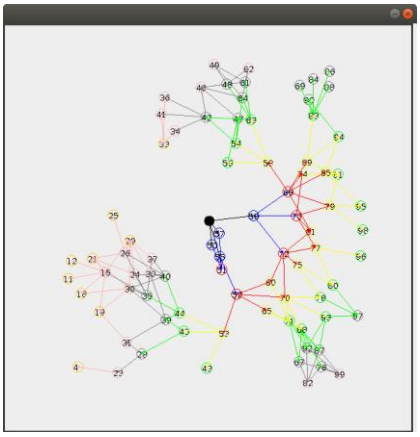

Figura 9: Evolução do fluxo

Na figura 10, é apresentada uma observação de um nó. É possível saber assim que nós estão a transmitir para o dado nó, e que nós recebem o seu pacote.

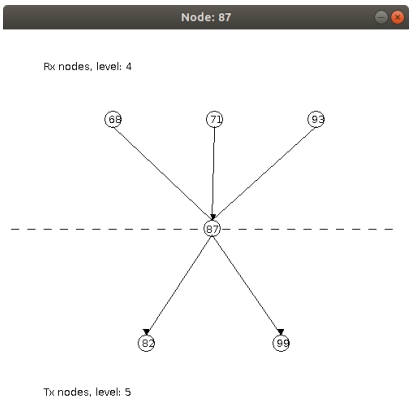

Figura 10: Observação de um nó

IV. RESULTADOS ESTATÍSTICOS

A. *Resultados estatísticos*

Existem algumas propriedades estatísticas interessantes de analisar na rede. Para gerar os resultados da simulação foram feitas 10 simulações, uma topologia com 100 nós, e um raio superior a 75 pixéis para que todos os nós recebessem o pacote. Fez-se uma extração de informação para o software MATLAB obtendo-se assim os diagramas com os resultados.

Na figura 11, verifica-se o estudo do número médio de recetores para um dado nó. Pode-se observar que o nó iniciador teve recetores em todas as simulações, e que alguns nós recetores não receberam o pacote.

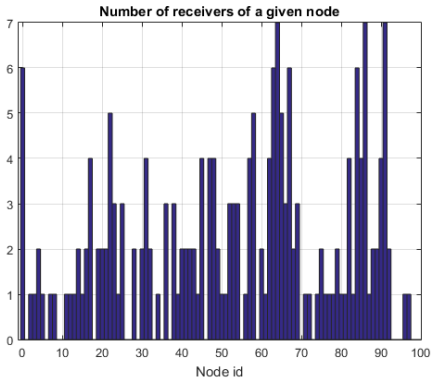

Figura 11: Número de recetores para um dado nó

De seguida, na figura 12, apresenta-se, em média, o número de nós com um dado número de recetores. Conclui-se assim que em média 20 nós não têm recetores, e que nenhum nó teve 8 ou mais recetores.

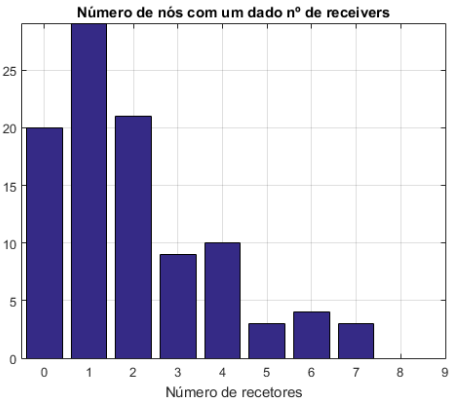

Figura 12: Número de nós para um dado número de recetores

Foi analisado o numero médio de recetores de um dado nó, e o máximo número de saltos ao longo da rede que ocorreram durante a transmissão. Assim, obteve-se um número médio de recetores de 1.9604, e um número máximo de 7 saltos.

Na figura 13, também é apresentado o histograma com os resultados, permitindo assim ver que a média ronda os 2 recetores.

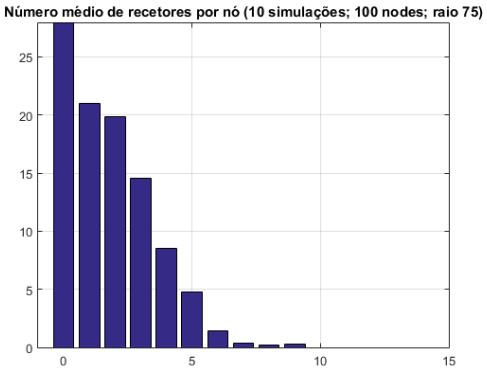

Figura 13: Número medio de rectores por nó

Um resultado importante que se obteve foi a relação entre o tempo de transmissão na rede e o raio que define o tamanho das camadas, independentemente do número de nós na rede. Através de uma aproximação polinomial foi possível concluir que o raio e o tempo são proporcionais. O resultado obtido é representado pela figura 14.

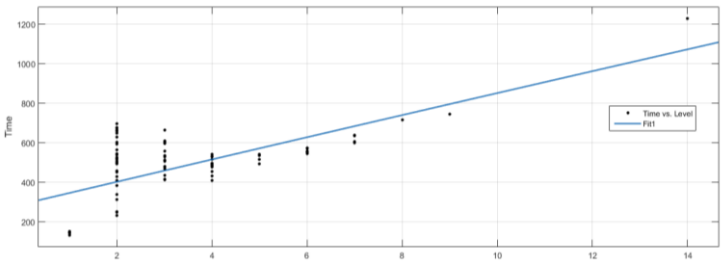

Figura 14: Tempo de transmissão vs raio de alcance

V. ANÁLISE CRITICA

A simulação aqui apresentada tem em conta os aspetos principais do Glossy, que tiveram de ser estudados e planeados de forma a serem executados com sucesso.
O artigo referido foi o guia para a implementação do simulador, contudo este foca-se maioritariamente no aspeto físico da arquitetura, e além disso, estamos perante uma arquitetura que na realidade é bastante complexa e tem em conta aspetos que não foram aqui abordados. Assim implementou-se uma versão bastante simplificada.

Contudo, através deste trabalho foi possível desenvolver uma simulação bastante próxima daquela que os autores do Glossy desenvolveram no *contiki*. Em termos de interface gráfica, foi desenvolvido algo simples e auto explicativo onde se torna evidente a arquitetura da rede assim como ela se encontra disposta. É possível ainda definir vários parâmetros, como o raio de alcance de cada nó da rede, que depois de simulado permitiu extrair informação importante sobre o estado da rede, assim com os requisitos mínimos para disseminar informação a toda a rede.
Apesar dos dados temporais extraídos estarem bastantes desfasados daquilo que seriam os tempos reais numa implementação física, tornou-se claro através desta simulação a relação direta entre o numero de camadas que compõem a rede e o tempo de disseminação de uma mensagem a toda a rede.
Do ponto de vista da unidade curricular, ajudou a ter uma ideia mais clara sobre a perspetiva de sistemas distribuídos, assim como as tecnologias a ela associadas, no nosso caso a API do java para a criação de *sockets*.

VI. CONTRINUIÇÕES

Bruno: Contribuiu para o estudo da arquitetura, desenvolvimento do código e relatório final. A sua contribuição é aproximadamente 35%.

Ília: Contribuiu para o estudo da arquitetura, soluções de problemas, desenvolvimento de código e relatório final. A sua contribuição é aproximadamente 30%.

Nuno: Contribuiu para o estudo da arquitetura, soluções de problemas, desenvolvimento do código e relatório. A sua contribuição é aproximadamente 35%

REFERENCES

[1] Federico Ferrari, Marco Zimmerling, Lothar Thiele, Olga Saukh, “Efficient Network Flooding and Time Synchronization with Glossy”