Internet of things - Exercício sobre transmissões concorrentes em linha utilizando modelo físico (SINR)

Gabriel Augusto R. dos Reis (16.2.8105)[1]

[1] Departamento de Computação e Sistemas
Universidade Federal de Ouro Preto (UFOP) – João Monlevade, MG

Resumo. *Neste artigo, é realizado um experimento baseado em: signal to noise plus interference (SINR), onde os pacotes podem ser decodificados se o SINR for maior que o β ao chegar no receptor.*

Palavras-chave: SINR

1. Problema abordado

O problema consiste em Utilizando o modelo físico (SINR) descubra a maior sequência de transmissões concorrentes em linha. Pode se variar as potências de transmissão, as distâncias entre eles, e escolher o limiar do rádio β, o expoente de perda de caminho α e o ruído inicial (N).

2. Solução abordada

Foi elaborado um algoritmo em linguagem Java, que gera um combinação dos parâmetros, e verifica se os dispositivos conseguem se comunicar, caso não consiga gera uma nova combinação até que se encontre, caso o usuário deseje parar a execução o programa necessita ser interrompido manualmente. o β é calculado de acordo com a seguinte fórmula:

$$\frac{\frac{Pu}{d(u,v)\alpha}}{N + \sum w \in V \backslash \{u\} \frac{Pw}{d(w,v)^{\alpha}}} \tag{1}$$

. Fórmula para cálculo do SINR

3. Algoritmo

O algoritmo elaborado está disponível em [1] e funciona da seguinte forma: Possui um ArrayList de dispositivos, onde cada dispositivo possui um nome, sua potência de transmissão e distância do dispositivo inicial. O algoritmo gera um α aleatório entre 1 e 3, um beta aleatório entre 1 e 3, gera uma distância aleatória para cada dispositivo da lista sempre crescente e sem repetições, obrigando a estarem alinhados e nunca juntos. É aplicado a fórmula de cálculo do SINR, caso ela seja afirmativa, o programa exibirá como saída cada parâmetro definido, caso contrário gerará novos números aleatórios e tentará novamente.

4. Modificação no algoritmo

Caso se deseje alterar os parâmetros, devem ser feitos os passos a seguir:

- α: Para modificar o α basta modificar a linha 23 do programa e substituir pelo valor desejado.
- β: Para modificar o β basta modificar a linha 24 do programa e substituir pelo valor desejado.
- Ruido: Basta alterar o valor nos parênteses na linha 18.
- Distância: Para modificar o limite de distância basta alterar o número 20 na linha 25.
- Dispositivos: Para adicionar ou remover dispositivos basta acrescentar ou remover linhas no trecho a partir da linha 23 do código, e inserir (para cada dois dispositivos adicionados) ou remover (para cada dois dispositivos adicionados) um laço encaixado na linha 59.

5. Resultados

O algoritmo foi deixa executando por 1hora e 30 minutos de forma estática com a configuração:

- α = 3
- β = 3
- Ruido = 10
- Dispositivos = 6, distribuídos de forma aleatória em relação a potência e distância.

Para dada circunstância foi realizado aproximadamente 170.000 combinações, onde o algoritmo não encontrou uma possível resposta.
A configuração foi modificada mantendo o ruído e todo o restante de forma aleatória, nesta combinação o algoritmo rodou pelo mesmo tempo, realizando aproximadamente 160.000 combinações, onde não encontrou uma possível resposta. Como uma terceira tentativa Foi alterado o limiar β para 1 e α em 2 e o restante dinâmico, o algoritmo após 1137 tentativas encontrou 31 possíveis soluções, demonstradas a seguir. Lembrando que após uma interação com ao menos uma combinação possível, o algoritmo é automaticamente interrompido.

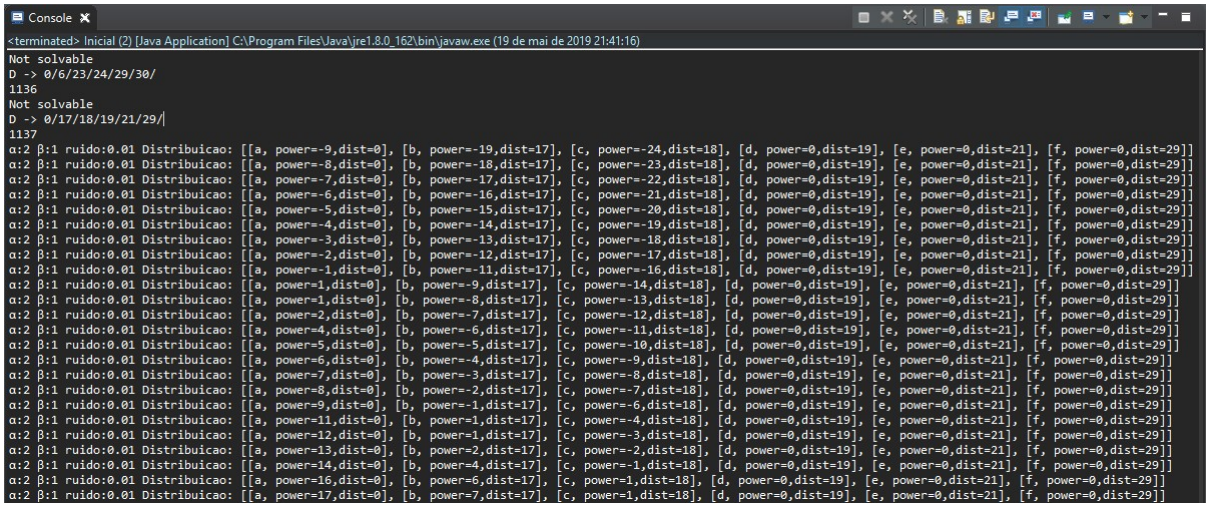

Figura 1. Console para o teste 3

Houve uma tentativa fracassada de executar o algoritmo com 8 dispositivos, mas devido a atual complexidade neste caso (O(n^4)) o algoritmo não responde em tempo hábil. Para testes com 4 dispositivos, o algoritmo encontra responta rápida.

6. Conclusão

Observando os resultados é certo dizer que quanto mais dispositivos na rede, mais difícil fica se comunicar, isto se deve ao fato de que ao ter mais dispositivos o balanceamento entre distância e potência se eleva. Em grandes distâncias os dispositivos das extremidades ao tentarem se comunicar necessitam de uma alta potência para que seu sinal chegue ao receptor, e isto gera uma quantidade muito grande de ruído para os demais, impedindo os demais dispositivos de se comunicarem, no caso oposto, ou seja, em pequenas distâncias gera um ruído muito alto devido a proximidade e consequente ”pouca perda”para o meio. Fica claro que o principal problema abordado se deve a alta interferência entre os dispositivos e não necessariamente aos próprios dispositivos.

Referências

[1] G. Augusto, “Physical model based on sinr - iot.” [Online]. Available: https://github.com/Gabriel-Reis/Physical_Model_based_on_SINR_IOT