deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Distributed Sudoku Solver

Danilo Silva Nº 113384

Tomás Santos Nº 112981

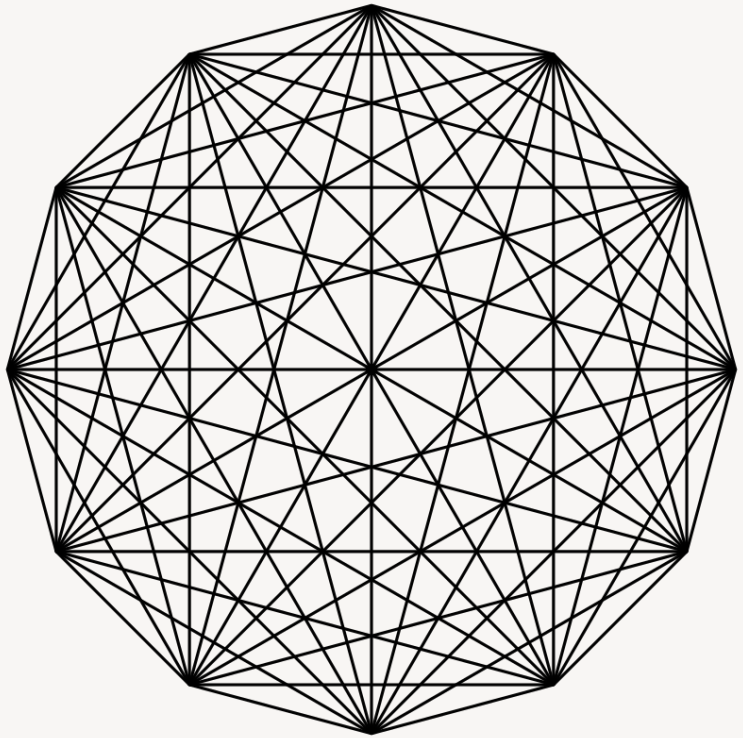
Arquitetura

- **Modelo de arquitetura P2P full mesh:** rede pequena (poucos nós), permite alta resiliência e baixa latência, oferecendo um bom ambiente de resolução global de um mesmo problema; maior redundância e robustez, onde uma falha é facilmente recuperável, sem necessidade de complexidades em termos de escolha e cuidados com um nó primário; simplifica troca de mensagens e organização da rede; benefício de comunicação direta e instantânea sobrepõe-se às desvantagens em termos de custo quanto ao elevado fluxo na rede.

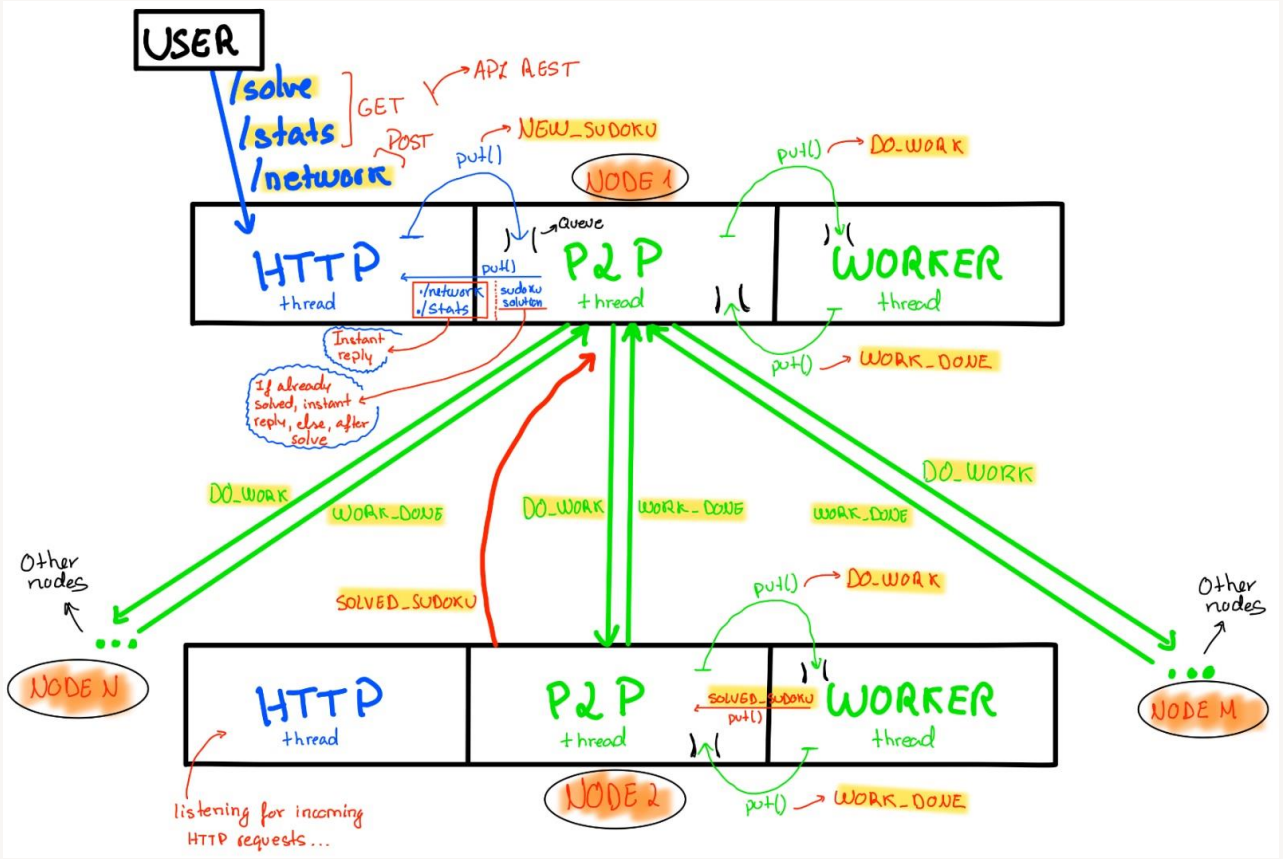
- **Comunicação TCP:** considerámos que o cenário de todos os nós da rede a trabalharem para um mesmo problema exigiria robustez na comunicação; TCP garante a verificação de integridade, a retransmissão de pacotes perdidos (assegura que os pacotes chegam ao destino, mesmo em redes instáveis) e o controlo de fluxo (ordem dos pacotes); deste modo garante que não existe informação perdida ou duplicada (por exemplo, quanto aos números nas estatísticas, ou aos avisos de soluções testadas/validadas).

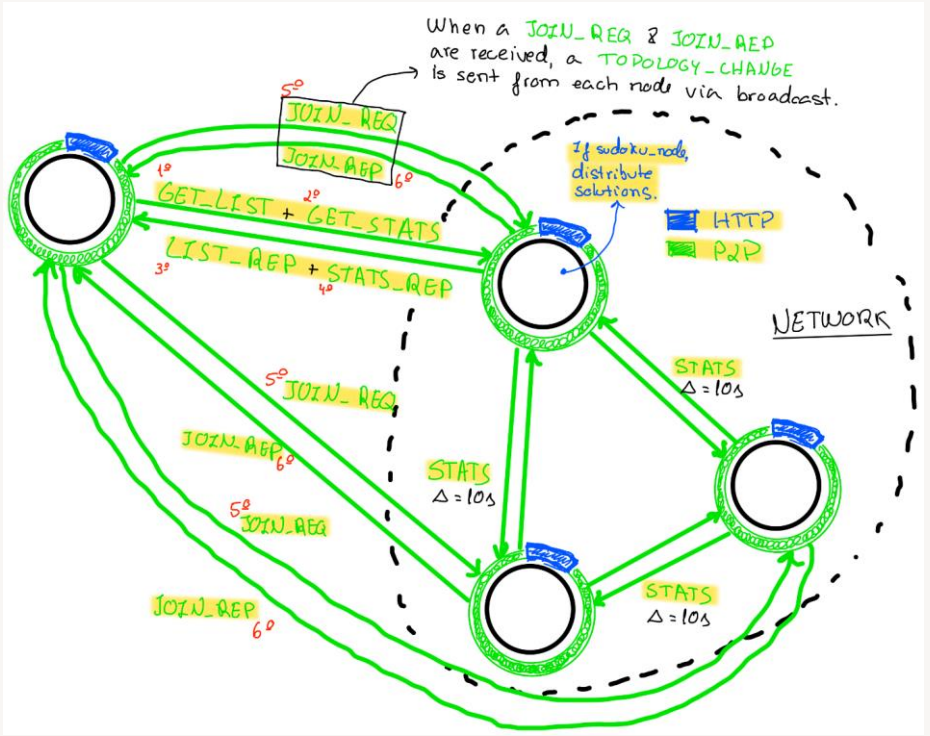
- **Protocolo utilizado:** garante a robustez da rede, com todas as mensagens, e respetivas trocas, necessárias à criação e manutenção da mesma.

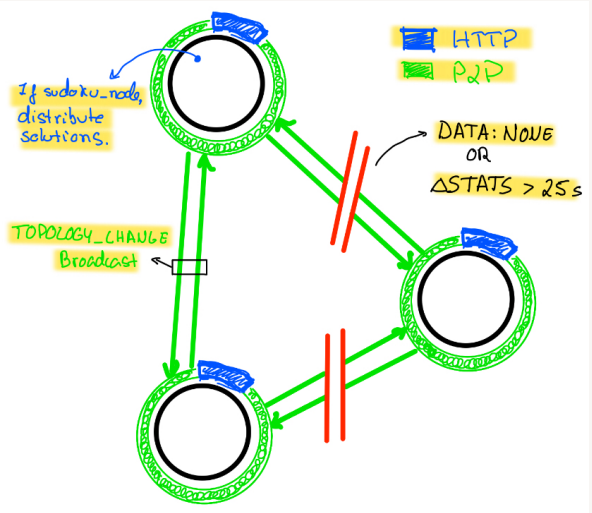
- **Comunicação entre threads:** para além da comunicação entre nós, tivemos também de nos preocupar com a comunicação entre threads num nó; utilizámos a Queue() da biblioteca *queue* do Python para fazer as threads HTTP e Worker esperarem no get() (na vez de busy waiting), mas precisámos de usar a Queue() da biblioteca *multiprocessing* de modo a puder usá-la como file object num selector.



Esquema de rede

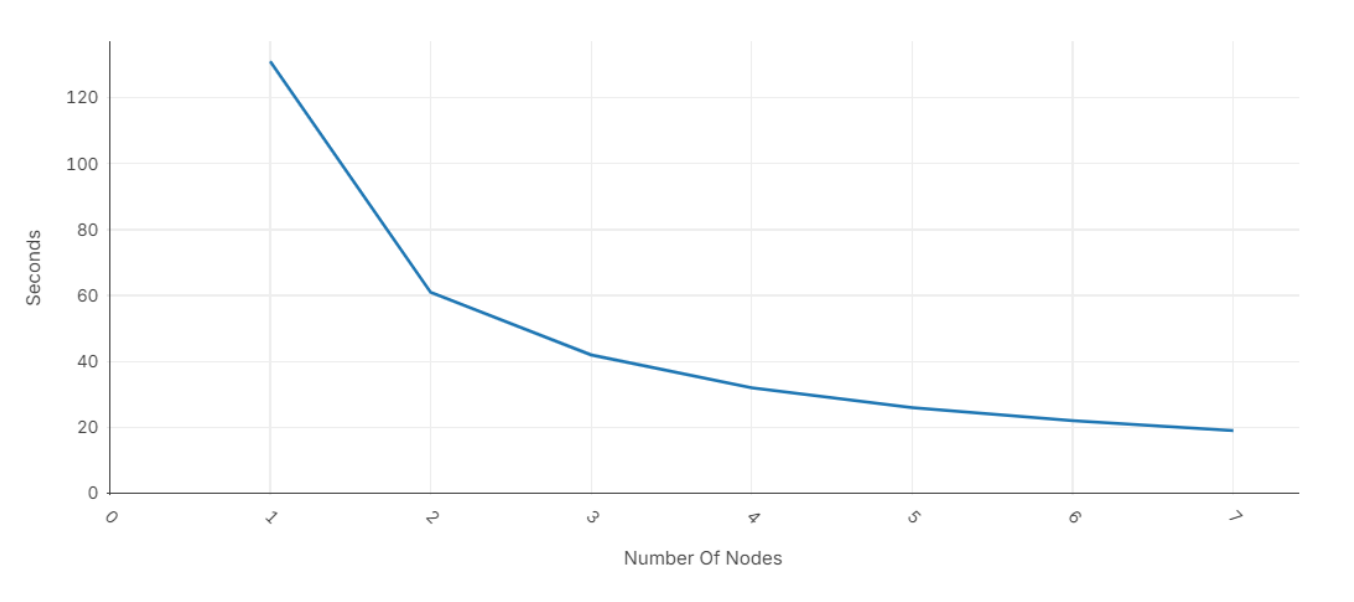






Benchmark

Com o script *./solve3* (*src/scripts*), utilizámos um sudoku com 3 zeros para testar a capacidade de distribuição de trabalho por um determinado número de nós na rede (todos com handicap de 0.01).
O resultado foi o seguinte:



Load Balancer

O mesmo script foi usado para testar a capacidade de distribuição do *Load Balancer* implementado (***Least Response Time***), com diferentes handicaps.
O resultado encontra-se na seguinte tabela:

Nó	-h (handicap)	Validações
1	0.01	51
2	0.01	50
3	0.03	32
4	0.01	51
5	0.01	50
6	0.01	47
7	0.60	11

Alguns Requisitos

- Solução funciona em rede (diferentes computadores);
- Rede dinâmica (entrada e saída de nós a qualquer momento, com reajustamento);
- Suporta-se mais do que um sudoku em resolução na rede;
- Nº de processos arbitrário (tantos quanto o necessário);
- Solução distribuída, respeitando os tempos de atraso de cada nó;
- Multithreaded: todos os nós podem estar atentos a pedidos HTTP, a mensagens na rede P2P e ainda a trabalhar, tudo isto ao mesmo tempo.
- Etc.

Fault Tolerance

Para desenvolver uma rede totalmente tolerante a falhas, sem perdas de informação, implementámos as seguintes funcionalidades:

- Ao perder-se um nó inesperadamente, caso este esteja a trabalhar, o seu trabalho é recuperado e redistribuído na rede;
- Todos os nós estabelecem um consenso quanto ao número total de validações na rede, mantendo a mesma consistente.