Eleição de líder

Descrição do problema

O problema de eleição de líder se baseia na necessidade da escolha de um processo como sendo o responsável por coordenar os outros, o chamado de líder, e com isso garantir a integridade do sistema distribuído por ter mais responsabilidades de liderar os processos (nós) subordinados. Para isso, é necessário que seja implementado um algoritmo que garanta que os próprios processos possuam uma lógica de escolha de um líder, e caso algum líder pare de responder, é necessário que outro líder seja escolhido pelos processos restantes.

Detalhes da implementação

Para resolver o problema, foi escolhido o algoritmo de eleição de líder chamado de “Algoritmo do valentão”, ou “bully”, que consiste na eleição do novo líder de acordo com o seu número de processo, ou id. Na nossa abordagem foi utilizado o número da porta do servidor (processo) para determinar o líder do sistema.

As portas dos servidores foram definidas da seguinte forma:

```
let nodes = {
 9121: { state: null },
 9122: { state: null },
 9123: { state: null },
 9124: { state: null },
 9125: { state: null },
};
```

Através de uma requisição http, pode-se dar início a eleição em qualquer que seja o processo, através da seguinte chamada:

```
 await sendElection();
 res.json({
   message: "Election sent",
 });
});
```

Quando o processo recebe o “sendElection()”, ele mapeia todos os processos que possuem o número de porta maior que o dele, e envia também uma eleição para cada um. Além disso, ele solicita o número da porta de cada um (com o método “election/port”, e se obtiver resposta, significa que o processo que respondeu é maior, e portanto é seu novo líder. Caso não haja

resposta de nenhum processo, ele mesmo é o líder. Dessa forma, os nós conseguem se comunicar entre si para determinar um líder de acordo com o número de processo de cada um.

```
const sendElection = async () => {
 Object.keys(nodes)
   .filter((node) => parseInt(node, 10) > PORT)
   .map(async (node) => {
     const SERVICE = axios.create({
       baseURL: `http://localhost:${node}`,
     });
     await SERVICE.get("/send-election");
     await SERVICE.get(`/election/${PORT}`).then((res) => {
       if (res.data.message === "OK") {
         nodes = {
           ...nodes,
           [node]: {
             state: "OK",
           },
         };
         leader = Math.max(leader, parseInt(node, 10));
       }
     });
   });
};
```

O método “election/port” é definido de forma a responder ao servidor solicitante caso sua porta seja maior que a porta do solicitante, de forma que o solicitante possa definir seu novo líder.

```
APP.get("/election/:port", (req, res) => {
 const { port } = req.params;
 if (PORT > port) leader = parseInt(PORT, 10);
 else leader = parseInt(port, 10);
```

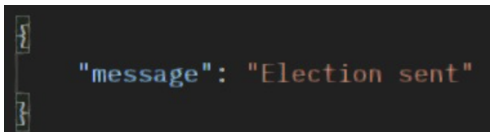
Ao final, é possível solicitar para cada um dos processos qual é seu líder, e caso a eleição esteja terminada, todos devem responder o mesmo número de processo.

```
APP.get("/leader", (req, res) => {
 res.json({ port: PORT, leader });
});
```

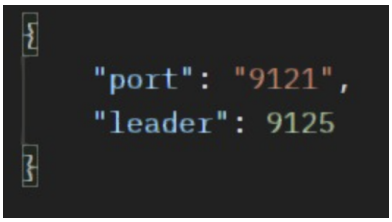
Ainda foi possível demonstrar o caso em que um servidor é derrubado e um novo líder é escolhido, porém seus detalhes foram mostrados na seção de resultados abaixo.

Resultados e comentários

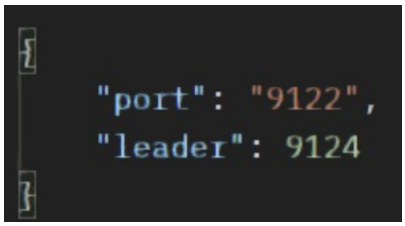
Inicialmente todos os servidores não possuem líder, e uma eleição é iniciada a partir do request em algum dos servidores. No nosso caso usamos o que possui a porta 9121



Após a eleição ser iniciada, ao checar qual o líder do 9121, temos a resposta:



Para todas as portas teremos a mesma resposta. Porém para testarmos “desligamos” a porta 9125, e fizemos o mesmo procedimento. Dessa forma, o novo líder passa a ser o 9124



Dessa forma, temos todos os nós ligados participando da eleição e elegendo sempre o que possui maior porta como o líder.

Um comentário pertinente que pode ser elaborado é que poderia haver uma rotina de sempre que um servidor “cair”, os outros automaticamente elegem um novo líder sem que haja uma eleição sendo ativada de forma manual, porém a lógica por trás da eleição foi feita de maneira correta.

Outro comentário é que poderia haver um número aleatório para cada nó em cada eleição, para sempre haver um líder diferente, porém com a lógica das portas já funciona bem.