Relógio de Lamport

Descrição do problema

O problema resolvido pelo algoritmo do relógio de Lamport consiste na sincronização entre diferentes servidores para que todos garantam a comunicação entre si sem que haja disparidade do ciclo de clock local de cada um.

Detalhes da implementação

Para resolver o problema, foi implementado o algoritmo de Lamport, que consiste no recebimento da mensagem entre os servidores e a comparação do clock local entre o receptor e o mensageiro. Para que fosse simulado o problema com os servidores instanciados, foi definido um timer local para cada um de acordo com a respectiva porta. Quando algum envio é realizado, é feito uma comparação entre o timer do receptor e do mensageiro e a partir da comparação, quando o timer do receptor for menor que o mensageiro, o servidor atualiza seu clock para o recebido e acrescenta em uma unidade quando outro envio é realizado. Do contrário, nada é feito com o clock do receptor. Dessa forma os relógios ficam sincronizados entre si e não permitem que um tempo de servidor mensageiro esteja menor do que o tempo do receptor.

O seguinte trecho de código é uma chamada HTTP GET para saber o timer do servidor desejado:

```
APP.get("/get-timer", (req, res) => {
 console.log(`Timer ${PORT}: ${timer}`);
 console.log("");
 res.json({});
});
```

O seguinte trecho de código é um envio manual a cargo de exemplo para o envio da mensagem para o servidor desejado. Pelo Postman é feita a requisição para um servidor para este mandar uma mensagem para outro servidor informado no parâmetro *to*.

```
APP.get("/send-message/:to", (req, res) => {
 timer += 1;

 const SERVICE = axios.create({
   baseURL: `http://localhost:${req.params.to}`,
 });
 SERVICE.get(`/message/${PORT}/${timer}`).then(() => {
   console.log(`${PORT} sends a message to ${req.params.to}`);
```

```
  console.log(`Timer ${PORT}: ${timer}`);
  console.log("");
 });
 res.json({});
});
```

E por fim, o seguinte trecho de código é um endpoint que recebe a mensagem enviada e realiza a comparação entre o timer dos servidores e atualiza o timer local caso o timer local seja menor que o timer recebido.

```
APP.get("/message/:from/:timer", (req, res) => {
 if (parseInt(timer, 10) < parseInt(req.params.timer, 10))
  timer = parseInt(req.params.timer, 10);
 console.log(`${PORT} receives a message from ${req.params.from}`);
 console.log(`Timer ${PORT}: ${timer}`);
 console.log("");O
 res.json({});
});
```

Resultados e comentários

Esses são os resultados apresentados para o relógio, onde quando se tem uma diferença de clock entre o receptor e o server de envio, e o receptor possui um clock menor, é necessário que ele atualize seu valor local de timer.

```
[0] Timer 9123: 21
[0]
[1] Timer 9124: 28
[1]
[2] Timer 9125: 30
[2]
[0] 9123 sends a message to 9124
[0] Timer 9123: 40
[0]
[1] 9124 receives a message from 9123
[1] Current timer 9124: 40
[1]
[1] ########### Message sent ###########
[1]
[2] 9125 sends a message to 9123
[2] Timer 9125: 41
[2]
[0] 9123 receives a message from 9125
[0] Current timer 9123: 43
[0]
[0] ########### Message sent ###########
[0]
[1] 9124 sends a message to 9123
[1] Timer 9124: 49
[1]
[0] 9123 receives a message from 9124
[0] Current timer 9123: 49
[0]
[0] ########### Message sent ###########
[0]
[1] 9124 sends a message to 9125
[1] Timer 9124: 50
[1]
[2] 9125 receives a message from 9124
[2] Current timer 9125: 51
[2]
[2] ########### Message sent ###########
[2]
[2] 9125 sends a message to 9123
[2] Timer 9125: 57
[2]
[0] 9123 receives a message from 9125
[0] Previous timer 9123: 55
[0] Current timer 9123: 57
[0]
[0] ########### Message sent ###########
[0]
[2] 9125 sends a message to 9124
[2] Timer 9125: 58
```