Marque a opção do tipo de trabalho que está inscrevendo:

(X) Resumo **() Relato de Caso**

Configuração Remota Para a Plataforma Embarcada do Protegemed

AUTOR PRINCIPAL: Kelvin M. Klann
ORIENTADOR: Marcelo Trindade Rebonatto
UNIVERSIDADE: Universidade de Passo Fundo

INTRODUÇÃO

O projeto Protegemed busca detectar, em equipamentos eletromédicos (EEMs), variações no consumo da energia elétrica que podem por em risco a vida de pacientes durante procedimentos cirúrgicos. A detecção é feita utilizando uma plataforma embarcada conectada nas tomadas de uma sala de cirurgia. A plataforma possui sensores que medem valores da corrente elétrica e os enviam para um servidor. O servidor analisa os dados recebidos e alerta em caso de riscos (SPALDING, CARPES JR. e BATISTELA, 2009).

Cada EEM possui um perfil de uso de energia diferente, que deve ser considerado ao medir a corrente elétrica. Porém, no momento, o perfil utilizado em cada tomada é fixo para todos os equipamentos conectados. Para cada atualização do perfil, é necessário estar presente para alterar os valores e gravar o resultado no sistema embarcado. O objetivo do trabalho é permitir que essa atualização seja feita remotamente usando a internet como meio de conexão.

DESENVOLVIMENTO:

Para uma comunicação bidirecional, o sistema utiliza o protocolo WebSocket, que é baseado no protocolo HTTP. Comparado ao HTTP, ele é um protocolo de mais baixo nível e sua principal vantagem é permitir a comunicação bidirecional sem a necessidade de abrir múltiplas conexões, resultando em um *overhead* menor (FETTE e

MELNIKOV, 2011) (SCHMITZ, 2017). Além disso, o WebSocket não especifica um padrão de serialização, nem o formato das mensagens a serem enviadas, deixando essa decisão a quem for implementar um programa que o utiliza.

Visando facilitar o reuso e a manutenção, buscou-se utilizar um formato simples para a troca de mensagens, com uma semântica bem definida e que já fosse conhecido. Foi decidido implementar a semântica e o formato do HTTP em um outro protocolo sobre o WebSocket. Esse protocolo, denominado MHTTP (nome provisório), é um *subset* do HTTP relativamente compatível. Porém, certas funcionalidades não se aplicam, como, por exemplo, os cabeçalhos do HTTP relacionados a *caching*, conexão e *cookies*. Ambos trabalham no modelo cliente-servidor, em que um cliente envia uma requisição a um servidor, que a processa e retorna uma resposta. O formato da primeira linha da requisição é o seguinte[4]:

{MÉTODO} {RECURSO} {PROTOCOLO}/{VERSÃO}

Os métodos do HTTP suportados são os seguintes: GET, HEAD, POST e OPTIONS.

GET solicita informações sobre o recurso.

HEAD é equivalente ao método GET, porém, o corpo da mensagem não deve ser retornado na resposta.

POST solicita que o recurso seja processado remotamente. Por exemplo, pode ser utilizado para alterar a representação de um recurso remoto.

OPTIONS solicita informações sobre quais métodos podem ser executados no recurso, entre outras informações.

O recurso identifica uma entidade remota, representado por uma URI (*Uniform Resource Identifier*), que é o “alvo” do método. Exemplos: O caminho de um arquivo no servidor ou a interface de um comando.

O formato da primeira linha da resposta é o seguinte[4]:

{PROTOCOLO}/{VERSÃO}{CÓDIGO_DE_ESTADO {DESCRIÇÃO_DO_ESTADO}

O código de estado é um número inteiro que determina o resultado da requisição e a descrição é uma frase correspondente ao número. Um código de 200 a 299 indica que a requisição foi processada com sucesso. Um código de 400 a 499 indica que houve um erro no cliente (ex.: por causa de um erro sintático ou semântico na requisição). Um

código de 500 e 599 indica que houve um erro interno no servidor que o impediu de completar a requisição. Os intervalos citados são semanticamente equivalentes aos do HTTP.

Opcionalmente, tanto a requisição quanto a resposta podem conter um ou mais cabeçalhos com seus respectivos valores (um conjunto por linha) e também o corpo da mensagem, este antecedido por uma linha em branco. Os cabeçalhos relevantes ao projeto são “Accept” e “Content-Type”, que definem o tipo de conteúdo aceito e a ser enviado no corpo da mensagem, respectivamente. Valores comuns são “text/plain” e “application/json”. A figura 1 exemplifica os diferentes formatos das mensagens do protocolo.

CONSIDERAÇÕES FINAIS:

Com uma conexão WebSocket, o protocolo permite a obtenção e atualização dos valores configurados na placa, além da execução de comandos através da rede, sem a necessidade presencial de um agente especializado. Além disso, o cabeçalho “Content-Type” permite enviar e receber qualquer tipo de conteúdo que pode ser representado como texto (ou seja, não-binário), o que facilita a extensibilidade.

REFERÊNCIAS

RFC Editor. **The WebSocket Protocol**. 2011. FETTE, I.; MELNIKOV, A. Disponível em: <http://www.rfc-editor.org/rfc/rfc6455.txt>. Acesso em: 16 jul. 2018.

SPALDING, L. E. S; CARPES, W. P.; BATISTELA, N. J. A method to detect the microshock risk during a surgical procedure. **IEEE Transactions on Instrumentation and Measurement**, v. 58, p. 2335–2342, 2009.

SCHMITZ, M. A. **Comunicação bidirectional para plataforma embarcada do Protegemed**. 2017. Dissertação (Mestrado em Ciência da Computação) — Universidade de Passo Fundo - Instituto de Ciências Exatas e Geociências – Programa de Pós-Graduação em Computação Aplicada, 2017.

NÚMERO DA APROVAÇÃO CEP OU CEUA (para trabalhos de pesquisa): N/A

ANEXOS

3.4.2 GET

3.4.2.1 Exemplo (texto).

Request:

```
GET /config/serverIp MHTTP/1.0
```

Response:

```
MHTTP/1.0 200 OK

192.168.103.101
```

3.4.2.2 Exemplo (JSON).

Request:

```
GET /config MHTTP/1.0
Accept: application/json
Content-Type: application/json
```

Response:

```
MHTTP/1.0 200 OK
Content-Type: application/json

{ "serverIp" : "192.168.103.101", "abc" : "123" }
```

3.4.4 POST (configure)

3.4.4.1 Exemplo (texto).

Request:

```
POST /config/server MHTTP/1.0

192.168.103.101
```

Response:

```
MHTTP/1.0 204 No Content
```

3.4.4.2 Exemplo (JSON).

Request:

```
POST /config MHTTP/1.0
Content-Type: application/json

{ "server" : "192.168.103.101" }
```

Response:

```
MHTTP/1.0 200 OK

{ "server" : "192.168.103.101" }
```

3.4.3 POST (command)

3.4.3.1 Exemplo (texto).

Request:

```
POST /cmd/config MHTTP/1.0
```

Response:

```
MHTTP/1.0 204 No Content
```

3.4.3.2 Exemplo (JSON).

Request:

```
POST /cmd MHTTP/1.0

{ "cmd" : "reset" }
```

Response:

```
MHTTP/1.0 204 No Content
```

Illustration 1: Exemplo utilizando o protocolo