UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

Relatório

UCMP – UDP Command Protocol

Disciplina: Redes de Computadores – 2017.2

Equipe: Cicero Anderson Fernandes Holanda 388714

Quixadá - CE.

1. **Visão geral serviço do remoto**

   Essa aplicação é como um canivete suíço, tem capacidade para ter qualquer funcionalidade. A ideia é que depois de feito o servidor você não precisara mais alterá-lo incluindo comandos diretamente no código poluindo mais ainda ele. Se você quiser adicionar novos comandos basta criá-los na classe CommandList seguindo o padrão e através do uso da API reflection o server saberá qual método o cliente esta tentando executar e se o número de parâmetros é valido ou se existe esse método na classe.

2. **Descrição dos Métodos Remotos**

   Primeiro o cliente deve utilizar um comando (**open** ip[:port]) interno para que quando ele envie dados o socketServer já esteja carregado em uma variável.
   O serviço aceita qualquer método que esteja na classe CommandList e que esse metodo siga o padrão de receber parâmetros com o tipo String e retornar String, O programador poderia criar funções sofisticadas, não somente realizar cálculos simples, como por exemplo, acessar a api do blockchain ou qualquer outra e pegar dados dela como nesse caso o valor atual do bitcoin, podendo passar como parâmetro o dia, hora ou mês, ano, etc.
   ----------| UCMP v1.0 |----------
   > **open** 127.0.0.1
   > **add** 1 2
   3
   > **fibonacci** a
   Comando inválido!
   >

3. **Descrição do Protocolo**

   Sintaxe (formato da mensagem) e Semântica (significado de cada campo da mensagem)
   <metodo> <param> …
   A sintaxe da mensagem é simples, a primeira palavra digitada é o nome do método/comando na classe CommandList e é dado espaço para separar o nome do método e parâmetros, também tem espaço entre os parâmetros.
   Ex: **fibonacci** 10
   add 1 2

   <metodo> = Simplesmente o nome do método contido na classe CommandList.
   <param> … = Lista de paramentos separados por espaços. Conteúdo do parâmetro depende do método/comando usado, mesmo sendo

enviado e processado como String, uma hora ele pode ser convertido para inteiro ou qualquer outro e talvez gerar uma exceção e o servidor avisará que o comando é invalido.

4. **Descrever o modelo de falhas**

    1. O usuário tentar enviar comandos para um servidor que não existe ou a porta está invalida. É gerado uma exception que é tratada e o cliente continua funcionando, o usuário pode inclusive usar o comando open como o ip de outro server e conectar.
    2. O comando enviado está invalido, com o número de parâmetros diferente do método e o server tem um exception no reflection que é tratado e enviado a mensagem para o cliente avisando que o comando está invalido. O comando pode também ser chamado com o nome errado (comando em case sensitive) o server também avisa que o comando está invalido.
    3. Exceções lançadas nos comandos personalizados da classe CommandList são tratados e o servidor avisa o cliente. Pode ocorrer do usuário enviar um parâmetro que não pode ser convertido para inteiro então será lançado um NumberFormatException que será tratado e o server avisara o cliente. Como exemplo: **add** abc 2.

    O servidor sempre tratará as exceptions e ira avisar o cliente que o comando está invalido.

    Observação:
    O uso da thread no processamento do método está correto o erro que deu em sala foi devido ao cmd por motivos desconhecidos mas parece que tinha travado aquele cmd que estava rodando o UDPServer. Rodando no terminal do eclipse mesmo quando ele travava a thread com o comando fibonacci 100 ainda dava pra enviar comandos, pois uma thread nova é criada toda vida que o server recebe alguma coisa. Só que a thread só vai rodar no comando enviado pelo cliente para que a thread do server seja liberada e ele possa receber outras requisições.