Centro de Informática
UFPE

Bacharelado em Sistemas de Informação
IF975 - Redes de Computadores
Prof. Kelvin Lopes Dias

# Projetos de Desenvolvimento com Sockets

- Lançamento: 02 de março de 2020
- Equipe: <= 3 integrantes

1. **Regras**
   - Os projetos devem ser implementados em Python 3 (preferencialmente), Java, C ou C++;
     - Os projetos devem ser multiplataformas ou pelo menos serem capazes de rodarem no Linux;
   - *Frameworks* ou bibliotecas que facilitam a manipulação com sockets não podem ser utilizados;
   - Deve ser elaborado um relatório detalhado para cada projeto desenvolvido;
   - O código-fonte deve ser entregue;
     - O relatório deve conter uma seção com os passos que devem ser seguidos para executar o código fonte de forma satisfatória;
   - Cópias acarretarão em nulidade dos projetos das equipes envolvidas;
     - Um algoritmo de controle de autoria será utilizado para verificar as cópias;

2. **Projeto 1: Quiz Competitivo (desenvolva seu próprio protocolo de comunicação) - UDP**
   - Deverá ser desenvolvido um jogo com competições *online* de perguntas e respostas;
   - Um protocolo da camada de aplicação deverá ser desenvolvido e especificado no relatório;
     - Deverá ter pelo menos uma mensagem de requisição e pelo menos uma mensagem de resposta;
   - O protocolo de transporte UDP deverá ser utilizado;
   - Um servidor UDP deverá gerenciar as competições;
   - Até 5 clientes UDP poderão participar de uma competição;
   - Após a competição ser iniciada, não deverá ser permitido o ingresso de novos participantes;
   - Cada competição terá 5 rodadas de perguntas e respostas;

- O servidor deverá ter um arquivo de texto contendo pelo menos 20 tuplas de perguntas e respostas;
    - As respostas deverão ser compostas por uma única palavra, com caracteres minúsculos;
    - O tema do Quiz ficará a critério da equipe;
- O servidor irá escolher aleatoriamente uma tupla de pergunta/resposta que será utilizada na rodada;
- Uma mesma competição não poderá ter duas tuplas repetidas;
- Cada rodada será encerrada quando algum participante acertar a resposta ou atingir uma duração máxima de 10 segundos;
- Pontuação de cada rodada:
    - Cada resposta errada: -5 pontos
    - Sem resposta: -1 ponto
    - Resposta correta: 25 pontos
- Após uma competição ser encerrada, um *ranking* com a pontuação é divulgado e uma nova competição poderá ser iniciada.

**3. Projeto 2: Servidor Web (implemente o protocolo padronizado HTTP/1.1) - TCP**

- Deverá ser desenvolvido um servidor WEB;
    - Deverá implementar o protocolo HTTP/1.1, apenas o método GET;
- O servidor terá que ser capaz de retornar diversos tipos de arquivos (por ex: html, htm, css, js, png, jpg, svg...);
    - Ou seja, deverá conseguir manipular tanto arquivos de texto, quanto arquivos binários;
- O servidor deverá ser capaz de transmitir arquivos de tamanho muito grande
    - Dessa forma, a leitura de arquivos deverá seguir o princípio de *lazy evaluating* para evitar a sobrecarga da memória RAM;
- Os requisitos mínimos (devem ser implementados obrigatoriamente) são o desenvolvimento das respostas com os códigos de resposta a seguir:
    - 200 OK
        - Requisição bem-sucedida, objeto requisitado será enviado
    - 400 Bad Request
        - Mensagem de requisição não entendida pelo servidor, nesse caso o cliente escreveu a mensagem de requisição com algum erro de sintaxe
    - 404 Not Found
        - Documento requisitado não localizado no servidor
    - 505 HTTP Version Not Supported
        - Versão do HTTP utilizada não é suportada neste servidor
- Com exceção do código 200, o servidor deverá enviar obrigatoriamente um arquivo html personalizado informando o respectivo erro;

- Se a pasta requisitada não contiver um arquivo index.html ou index.htm, o servidor deverá criar uma página html para navegar pelas pastas, semelhante ao que ***apache*** faz (que navega nas pastas de forma semelhante ao windows explorer, nautilus e afins...);
- O servidor Web deverá ler os arquivos de uma pasta específica, caso ela não exista, deverá ser criada automaticamente ao executar o servidor;
- O uso de sockets TCP é obrigatório.
    - Não é permitido o uso de *frameworks* ou bibliotecas que implementem o HTTP e substituam o uso de sockets;
    - Reforçando, não é permitido o uso de APIs que dispensem a implementação de um servidor HTTP. (A ideia é realmente implementar um servidor seguindo um protocolo bem definido pela comunidade de redes...).

**4. Dicas!**

- Não deixe para começar os projetos mais tarde. Comece logo! São apenas 58 dias!
- É (quase) impossível fazer os projetos de "virada"... Mesmo em duas semanas de "viradas" ;-)
- Fazer um cronograma de atividades de desenvolvimento dos projetos;
- Considere épocas de provas e o desenvolvimento de outros projetos em outras disciplinas;
- Não se esqueçam de dar atenção ao relatório!
- Marcaremos alguns dias de acompanhamento dos projetos;
    - Entretanto, a responsabilidade do desenvolvimento dos projetos é da equipe;

**5. Casos omissos**

- Possíveis casos/requisitos omissos serão esclarecidos quando for oportuno;
- Dessa forma, esse documento poderá ser atualizado com novas informações, se necessário.

Boa diversão! =)

yabls@cin