6. Vimos que sockets TCP da Internet tratam os dados que estão sendo enviados como uma cadeia de bytes, mas que sockets UDP reconhecem fronteiras de mensagens. Cite uma vantagem e uma desvantagem da API orientada para bytes em relação à API que reconhece e preserva explicitamente as fronteiras das mensagens definidas por aplicações.
7. O que é o servidor Web Apache? Quanto custa? Que funcionalidade tem atualmente?
8. Muitos clientes BitTorrent utilizam DHTs para criar um rastreador distribuído. Para esses DHTs, qual é a "chave" e qual é o "valor"?
9. Imagine que as organizações responsáveis pela padronização da Web decidam modificar a convenção de nomeação de modo que cada objeto seja nomeado e referenciado por um nome exclusivo que independa de localização (um URN). Discuta algumas questões que envolveriam tal modificação.
10. Há empresas distribuindo transmissões televisivas ao vivo por meio da Internet hoje? Se sim, essas empresas estão usando arquiteturas cliente-servidor e P2P?
11. As empresas, hoje, estão oferecendo um serviço de vídeo ao vivo através da Internet usando uma arquitetura P2P?
12. Como o Skype provê um serviço PC para telefone a vários países de destino?
13. Quais são os clientes mais populares do BitTorrent atualmente?

Tarefas de programação de sockets

Tarefa 1: servidor Web multithread

Ao final desta tarefa de programação, você terá desenvolvido, em Java, um servidor Web multithread, que seja capaz de atender várias requisições em paralelo. Você implementará a versão 1.0 do HTTP como definida no RFC 1945.

O HTTP/1.0 cria uma conexão TCP separada para cada par requisição/resposta. Cada uma dessas conexões será manipulada por um thread. Haverá também um thread principal, no qual o servidor ficará à escuta de clientes que quiserem estabelecer conexões. Para simplificar o trabalho de programação, desenvolveremos a codificação em dois estágios. No primeiro estágio, você escreverá um servidor multithread que simplesmente apresenta o conteúdo da mensagem de requisição HTTP que recebe. Depois que esse programa estiver executando normalmente, você adicionará a codificação necessária para gerar uma resposta apropriada.

Ao desenvolver a codificação, você poderá testar seu servidor com um browser Web. Mas lembre-se de que você não estará atendendo através da porta padrão 80, portanto, precisará especificar o número de porta dentro do URL que der a seu browser. Por exemplo, se o nome de seu hospedeiro for `host.someschool.edu`, seu servidor estiver à escuta na porta 6789 e você quiser obter o arquivo `index.html`, então deverá especificar o seguinte URL dentro do browser:

```
http://host.someschool.edu:6789/index.html
```

Quando seu servidor encontrar um erro, deverá enviar uma mensagem de resposta com uma fonte HTML adequada, de modo que a informação de erro seja apresentada na janela do browser. Você pode encontrar mais detalhes sobre esta tarefa, assim como trechos importantes em código java no site http://www.aw.com/kurose.br

Tarefa 2: cliente de correio

Nesta tarefa, você desenvolverá um agente de usuário de correio em Java com as seguintes características:

- Que provê uma interface gráfica para o remetente com campos para o servidor de correio local, para o endereço de e-mail do remetente, para o endereço de e-mail do destinatário, para o assunto da mensagem e para a própria mensagem.
- Que estabelece uma conexão TCP entre o cliente de correio e o servidor de correio local. Que envia comandos SMTP para o servidor de correio local. Que recebe e processa comandos SMTP do servidor de correio local.

Esta será a aparência de sua interface: