

Um fileserver baseado em TCP

Conforme discutido em sala, é possível fazer dois pequenos programas que funcionam como cliente e servidor de arquivos TDP. A função primária é que o cliente possa baixar arquivos do servidor, mas algumas outras operações correlatadas são suportadas.

Sua atividade consiste em desenvolver:

- Um programa servidor de arquivos, escutando em todos os IPs da sua máquina, na porta UDP 20000;
- Um programa cliente do servidor de arquivos anterior.

O cliente vai interagir um usuário, que solicitará alguma das operações disponíveis. O cliente vai solicitar esta operação junto ao servidor e realizar as operações associadas.

O protocolo de comunicação para as operações é:

Usuário	Cliente (envia)	Servidor (envia)
Solicita o <i>download</i> de um arquivo, informando o nome.	<ul style="list-style-type: none">▪ 1 <i>byte</i> de operação (em binário);<ul style="list-style-type: none">▪ Valor: 10▪ 4 bytes (em binário – big <i>endian</i>) com o tamanho do nome do arquivo;▪ Múltiplos <i>bytes</i> contendo o nome do arquivo. A quantidade corresponde ao tamanho informado antes.	<ul style="list-style-type: none">▪ 1 <i>byte</i> de status (em binário);<ul style="list-style-type: none">▪ Valor: 0 (arquivo existe e será enviado a seguir);▪ Valor: 1 (arquivo não existe e nada mais será enviado);▪ 4 bytes (em binário – big <i>endian</i>) com o tamanho do arquivo que será enviado;▪ Múltiplos blocos de 1024 <i>bytes</i> contendo os dados do arquivo, até que todos os bytes (até o tamanho informado antes) sejam enviados.
Solicita a listagem dos arquivos disponíveis no servidor	<ul style="list-style-type: none">▪ 1 <i>byte</i> de operação (em binário);<ul style="list-style-type: none">▪ Valor: 20	<ul style="list-style-type: none">▪ 1 <i>byte</i> de status (em binário);<ul style="list-style-type: none">▪ Valor: 0 (a listagem será enviada a seguir);▪ Valor: 1 (erro não identificado, nada mais será enviado);▪ 4 bytes (em binário – big <i>endian</i>) com o tamanho a seguir;▪ A listagem em formato JSON, com o seguinte formato: [{ 'nome': 'arquivo1.txt', 'tamanho': '12345' }, { 'nome': 'arquivo2.jpg', 'tamanho': '45678' }, ]

Solicita o <i>upload</i> de um arquivo para o servidor, informando o nome.	<ul style="list-style-type: none"> ▪ 1 <i>byte</i> de operação (em binário); <ul style="list-style-type: none"> ▪ Valor: 30 ▪ 4 bytes (em binário – big <i>endian</i>) com o tamanho do nome do arquivo; ▪ Múltiplos <i>bytes</i> contendo o nome do arquivo. A quantidade corresponde ao tamanho informado antes. 	<ul style="list-style-type: none"> ▪ 1 <i>byte</i> de status (em binário); <ul style="list-style-type: none"> ▪ Valor: 0 (o upload foi aceito, os bytes serão esperados); ▪ Valor: 1 (erro não identificado, os bytes não serão esperados);
	<p>Só enviar se o servidor aceitou o <i>upload</i>.</p> <ul style="list-style-type: none"> ▪ 4 bytes (em binário – big <i>endian</i>) com o tamanho do arquivo; ▪ Múltiplos blocos de 1024 <i>bytes</i> contendo os dados do arquivo, até que todo os bytes (até o tamanho informado antes) sejam enviados. 	<ul style="list-style-type: none"> ▪ 1 <i>byte</i> de status (em binário); <ul style="list-style-type: none"> ▪ Valor: 0 (o upload foi bem-sucedido); ▪ Valor: 1 (erro não identificado);
Solicita o <i>download</i> de arquivo a partir de determinada posição, informando o nome, a posição a partir de onde deseja e o MD5 da parte que possui.	<ul style="list-style-type: none"> ▪ 1 <i>byte</i> de operação (em binário); <ul style="list-style-type: none"> ▪ Valor: 40 ▪ 4 bytes (em binário – big <i>endian</i>) com o tamanho do nome do arquivo; ▪ Múltiplos <i>bytes</i> contendo o nome do arquivo. A quantidade corresponde ao tamanho informado antes; ▪ 4 bytes (em binário – big <i>endian</i>) contendo a posição inicial a partir de onde quer baixar o arquivo; ▪ 16 bytes (em binário – big <i>endian</i>) contendo o hash MD5 da parte que possui. 	<ul style="list-style-type: none"> ▪ 1 <i>byte</i> de status (em binário); <ul style="list-style-type: none"> ▪ Valor: 0 (o download foi aceito, os bytes serão esperados); ▪ Valor: 1 (erro não identificado, os bytes não serão esperados);
		<ul style="list-style-type: none"> ▪ 1 <i>byte</i> de status (em binário); <ul style="list-style-type: none"> ▪ Valor: 0 (o upload foi bem-sucedido); ▪ Valor: 10 (o arquivo não existe, nada mais será enviado); ▪ Valor: 20 (o hash não confere, nada mais será enviado); ▪ 4 bytes (em binário – big <i>endian</i>) com o tamanho remanescente do arquivo que será enviado; <ul style="list-style-type: none"> ▪ Múltiplos blocos de 1024 <i>bytes</i> contendo os dados do arquivo, até que todos os bytes (até o tamanho informado antes) sejam enviados.
Solicita o <i>download</i> de múltiplos arquivos, informando uma máscara. Ex: *.jpg	<ul style="list-style-type: none"> ▪ 1 <i>byte</i> de operação (em binário); <ul style="list-style-type: none"> ▪ Valor: 50 ▪ 4 bytes (em binário – big <i>endian</i>) com o tamanho do da máscara a enviar; ▪ Múltiplos <i>bytes</i> contendo o nome do arquivo. A quantidade corresponde ao tamanho informado antes; 	<ul style="list-style-type: none"> ▪ Faça uma proposta de como será a resposta do servidor.

Considere, em adição, os seguintes aspectos:

- Defina uma pasta no cliente e no servidor em que os arquivos serão armazenados; todos os pedidos respostas serão relativos a essas pastas. Não permita escape dessas pastas, ou seja, arquivos fora delas jamais deverão atendidos;
- Trata erros de conexão e congêneres;
- Poda operação se dá em uma conexão;
- Estabeleça um timeout para as operações;
- O cliente deve receber na linha de comando o nome (ou ip) do servidor e a porta à qual conectar, no formato ip:porta. Ex:
`python cliente.py 10.25.2.120:20000`
- O servidor deve escutar em todas os IPs e receber na linha de comando em que porta. Ex:
`python servidor.py 20000`
- O servidor deve informar os IPs e as portas em que está escutando.