Cesar Kallas **RA: 02099224**
cesarkallas (at) gmx.net - http://cesarkallas.soulivre.org
Pontificia Universidade Catolica - Campinas

Atividade 3

---

1. Estude o texto acima e traga as suas dúvidas para discussão em sala: ok
2. Estude o código fornecido e responda:

a. Para que serve a variável global *portbase* no arquivo *passivesock.c*?

Serve para dizer qual é a porta mais baixa no qual o usuário comum pode rodar sua aplicação, pois, o orgão IANA (Internet Assigned Numbers Authority) especifica que portas abaixo de 1024 são chamadas de portas de baixa numeração, que só podem ser executadas ou abertas pelo administrador do sistema (root).

A idéia é que os clientes que conectam em portas abaixo da 1024 estejam conectando em um serviço padrão do sistema e não em uma aplicação feita por um usuário comum que esteja usando a máquina.

A utilidade dessa variável é setar o número da primeira porta no qual o cliente pode rodar a aplicação servidor, para que o sistema libere a escuta na porta. Especificamente nessa aplicação (UDPtimed), deve-se setar a porta 1024 - 37, porque há uma soma na hora de mapear a porta de escuta (pse->s_port + portbase).

b. O que indica e para que serve a constante UNIXEPOCH, em *UDPtimed.c?*

Essa constante significa a quantidade de segundos equiavalentes a data até Wed Dec 31 21:00:00 1969, no qual o sistema Unix pega como referência para datas, ou seja, do primeiro segundo do sistema de tempo atual até o dia 31/Dez/1969 às 21:00h.

Sendo assim, toda medição de tempo feita em um Unix tem como resultado quantos segundos se passaram desde a data UNIXEPOCH, a soma da medição feita no momento mais a data de partida do Unix (UNIXEPOCH) equivale a data atual.

3. O servidor *UDPtimed* necessita de algum privilégio especial para rodar em seu sistema local? Para rodar o cliente TIME é necessário algum privilégio? Justifique.

Para rodar o servidor UDPtimed não é necessário nenhum privilégio especial, desde que, o usuário rode em uma porta acima da 1023, para rodar em portas abaixo da 1024, é necessário ser administrador no sistema (uid 0). Veja mais detalhes na pergunta 2a.

Para rodar o cliente time não é necessário nenhum privilégio, mesmo que o servidor esteja rodando em uma porta abaixo da 1024, pois ele não escuta em nenhuma porta abaixo da 1024.

4. Como o servidor UDPtimed obtém o endereço do cliente para lhe enviar a resposta?

O servidor quando espera por uma mensagem do cliente, envia para o protocolo adjacente uma estrutura para que esse protocolo preencha com os dados do cliente conectado ao socket. Nessa implementação a struct passada é a fsin (sockaddr_in fsin).

Para saber o IP do cliente basta imprimir a string resultado da função ***inet_ntoa(fsin.sin_addr).***

5. O código do exemplo UDPtimed.c especifica um buffer de 1 byte quando chama recvfrom(). O que aconteceria se um buffer de 0 (zero) byte fosse especificado? Teste esta alteração e explique o resultado.

Não houve problemas nos testes realizados com buffer de 0 bytes. Isso porque a implentação de sockets no Unix diz que, se a mensagem recebida for maior que a capacidade do buffer de receber, os excessos de bytes serão descartados.

Como essa é uma implentação UDP, que só necessita receber o pedido, que é o próprio sinal do

socket cliente e não dados em um buffer em si, o servidor funcionará perfeitamente.

6. Conduza um experimento e determine o que acontece se N clientes enviarem pedidos simultâneos para o servidor UDPtimed. Varie o valor de N (número de clientes) e S (tamanho do datagrama enviado). Explique como foi realizado o experimento e mostre e explique os resultados obtidos.

O cliente UDPtime.c foi alterado de modo a lançar várias consultas no servidor UDPtimed_m usando threads.

O teste realizado foi com a criação seguidas de 500 threads, assim, o cliente lança sequencialmente 500 requisições de time ao servidor, de modo a testar a resposta.

Também foi alterado o tamanho da MSG enviada (buffer) para requerer o time.

Não houve nenhum problema no teste realizado com as threads, todas as respostas foram recebidas e confirmadas atráves de um contador no cliente, levando em consideração que em um computador com um único processador é impossível criar N requisições ao mesmo tempo (N > 1).

Como o UDP trabalha sem uma conexão, a explicação mais plausivel para todas as requisições atendidas pelo servidor é o uso de algum tipo de cache no protocolo UDP.