Ver o ficheiro adxunto da práctica 2 para información sobre as funciones e como crear programas UDP básicos.

Obxectivo da práctica
Crear un programa ''servidor'' e un programa ''cliente'' en UDP e entender o seu funcionamento.

1. Envío/recibo básicos. Sigue os pasos do documento adxunto para crear unha conexión UDP envío/recibo na que se envíe unha mensaxe de saudo.

i.-Programas básicos que envían e reciben datos usando UDP e entender o seu funcionamento:
a. Programa que recibe unha mensaxe:
- Debe escoitar por todas as súas interfaces usando un número de porto indicado como parámetro na liña de comandos
- Debe imprimir en pantalla a IP e o porto do que lle envía a mensaxe, además da mensaxe recibida
b. Programa que envía unha mensaxe:
-Debe permitir indicar o porto propio, a IP e o porto do destinatario da mensaxe como parámetros na liña de comandos
- Debe imprimir o número de bytes enviados
c. Comproba que ocorre se a función recvfrom() le menos datos dos que enviou a función sendto(). É posíbel recuperar os datos restantes cunha nova chamada a recvfrom()? Describe as modificacións introducidas nos códigos e o resultado obtido.
d. Modificar os programas para que, en vez de transmitir cadeas de texto, transmitase un array de números de tipo float (números reais). O número de datos enviados debese especificar únicamente no programa que envía. É dicir, o programa que recibe debe ser capaz de recibir todos os datos enviados e determinar dito número de datos. Describe as modificacións introducidas nos códigos e o resultado obtido. En Python pódese usar a librería "struct" e as súas funcións "pack" e "unpack" para obter os bytes a partires dos floats. (Python usa o double por defecto para os números reais, comprobar se hai erros ao empaquetar en float e modificade o programa para usar double). (chamadeos "f-recibe-udp-1d.py" e "f-envia-udp-1d"):

2. Servidor UDP de maiúsculas. Modificar os programas anteriores para convertilos nun "cliente" e "servidor" de maiúsculas. Facer que "cliente" lea un arquivo de texto de entrada (pasándolle o nome en liña de comandos), e se llo envíe liña a liña ao "servidor" usando a mesma conexión. O "servidor" debe pasar os caracteres de cada liña a maiúsculas e devolverlle ao cliente a liña convertida. Por último, o cliente vai recibindo as liñas e as vai escribindo nun arquivo de saída, que debe ter o mesmo nome que o arquivo de entrada pero en maiúsculas. Chamadeos "O funcionamento desde o punto de vista do usuario debe de ser coma en TCP:

a.- o cliente le unha liña do arquivo de entrada e se lla envía ao servidor (imprimir por pantalla o número de bytes enviados).
b.- o servidor se lla devolve ao cliente pasada a maiúsculas (imprimir por pantalla o número de bytes recibidos e enviados, e as direccións).
c.- o cliente a escribe no arquivo de saída (imprimir por pantalla o número de bytes recibidos e enviados).
d.- de volta ao paso a, ata que se termine o arquivo.

Además, o "servidor" debe aceptar como parámetro de entrada o número de porto polo que atende as solicitudes e o "cliente" o seu porto, a IP e porto do servidor además do nome do ficheiro de entrada, nese orden. Do mesmo modo que no exercicio 1, o "servidor" debe imprimir en pantalla a IP e o porto do "cliente". Para que o "servidor" termine correctamente podedes facer que pare tras recibir un número determinado de liñas, como 10 (ou enviar na primeira mensaxe o número de liñas ou enviar unha mensaxe de terminación). Chamadeos "cliente-udp-2.py" e "servidor-udp-2.py"

3. Comprobación de que se poden atender varios clientes SIMULTÁNEAMENTE. No lazo do cliente, onde se van lendo as liñas do arquivo, introducir un sleep() para que de tempo de lanzar un segundo cliente noutra terminal, describe o que ocorre. Aumentade o número de liñas a recibir no servidor (ou cambiade o necesario para que funcione se escollichedes outro método de terminación).

Requisitos mínimos:
Os programas poden programarse en C ou Python (o recomendado)

Débese facer uso das funcións vistas na clase.
Toda chamada a unha función do sistema debe ter o seu correspondente chequeo de erro.
Toda función debe sair co mensaxe de erro e o código apropiado en caso de erro (se o hai).
O código non pode fallar aínda qu se usen datos de entrada incorrectos.
Toda memoria reservada dinámicamente debe liberarse correctamente (so para C).
O código debe de estar adecuadamente comentado, indicándose de forma clara qué se fai en todas as funcións definidas, xunto coa explicación dos parámetros de entrada e saída das mesmas.
O código deberá estar correctamente formateado e tabulado.
O código debe compilar nun sistema Linux con gcc (para C). Se se usa Windows, débese convertir o ficheiro co código a formato Unix.

?

Os programas deben funcionar aínda que o cliente e o servidor se executen en computadores diferentes
Os parámetros de entrada necesarios deben ser proporcionados como argumentos do main.
Penalizarase que aparezan mensaxes de Warning na compilación (coa opción -Wall)(so C)

Entrega. Un arquivo comprimido zip con:
- Os códigos (non os executables) dos puntos 1.d (programas envío/recibo modificados para envío e recibo do floats) e 2 (servidor e cliente de maiúsculas usando UDP).

- Pódense enviar tamén os códigos dos 1.a (programa básico que recibe) e 1.b (programa básico que envía) para evaluar no caso de que as versión do punto 1.d (versión con floats) teña erros.
- Informe sobre os puntos 1.c, 1.d e 3.
- (Non obrigatorio) Unha captura de pantalla na que se poida ver que o cliente e servidor de maiúsculas se están executando en distinto ordenador (ou máquina virtual).

Última modificación: luns, 14 de novembro de 2022, 09:49