# Laboratorio 8

## Comunicación por Sockets

### Votación Maestro/Esclavo

**Objetivos**

En este laboratorio, los estudiantes aprenderán cómo comunicar y sincronizar procesos en un ambiente distribuido. El objetivo se logrará:

- Creando y usando sockets para la comunicación.
- Creando una configuración de Maestro/Esclavo (Master/Slave)

**Procedimiento**

En una configuración maestro/esclavo, un dispositivo tiene control sobre uno o más dispositivos adicionales. Un ejemplo es una red de computadoras, donde una de ellas (el maestro) recibe tareas y las asigna a otras computadoras (esclavos) basado en ciertos criterios. Usualmente, si no hay dispositivo maestro presente, los dispositivos existentes hacen una elección para determinar un nuevo maestro.

En este laboratorio, cada estudiante implementará un servidor que correrá en una computadora del laboratorio o Raspberry Pi (dispositivo). Cada dispositivo empezará con estatus de esclavo. Un programa cliente (implementado por el instructor) puede preguntarles a todos los dispositivos quién de ellos es el maestro, enviando el mensaje "QUIEN ES". Si ningún dispositivo responde que es el maestro, el programa cliente puede pedirles a todos que voten para determinar al nuevo maestro, enviando el mensaje "VOTE". Para votar, cada dispositivo enviará un mensaje broadcast a todos los demás dispositivos. Dicho mensaje comenzará con un símbolo # y contendrá la dirección IP del dispositivo, seguido de un espacio en blanco, seguido de un número entero aleatorio que el programa genere (ejemplos: "# 192.168.1.2 4" o "# 10.0.0.23 10", según la red).

Luego de recibir los votos, cada programa servidor deberá decidir si el dispositivo se convertirá en el nuevo maestro o no. Esto se hará comparando su propio voto con los demás votos recibidos. El voto más alto gana. Si se diera un empate, el dispositivo con la dirección IP más baja gana. Si el programa cliente envía otro mensaje "QUIEN ES", el dispositivo que se convirtió en el maestro deberá enviar un mensaje de vuelta únicamente al programa cliente (NO por broadcast) indicando que es el maestro. Este mensaje debe incluir su nombre y la dirección IP del dispositivo (ej. "Pedro en 192.168.1.2 es el master" o "María en 10.0.0.23 es la master"). **Asegúrense que las cadenas que envíen al cliente tengan un máximo de 60 caracteres. Esta es una limitación impuesta por el cliente, no por el protocolo TCP/IP.**
El servidor deberá ignorar cualquier mensaje inválido.

**Especificaciones:**

1.) Los mensajes son cadenas de 60 caracteres o menos.
2.) Los votos son números enteros aleatorios en el intervalo [1, 15].
3.) El número de puerto para la comunicación debe ser un argumento de su programa, es decir, cuando se corra el programa, debe ser posible proporcionar el número de puerto. Por defecto, el número de puerto será el 2000.
4.) Pueden usar el programa server_udp_broadcast.cpp como punto de partida para sus programas.
5.) La dirección IP del servidor puede ser escrita directamente en el código.

Es posible que el programa determine la dirección IP del dispositivo automáticamente. **Esto es opcional, aunque se darán puntos extra si lo logran. Asegúrense de completar todos los requisitos obligatorios antes de intentar lo opcional.**

Inicialmente, pueden probar sus programas servidor corriendo en Cygwin, y el cliente en la RPi. Si están en la misma red que sus compañeros, se sugiere que se pongan de acuerdo para no usar todos el mismo puerto cuando estén probando/depurando sus programas. También pueden pedirle al instructor que corra el cliente, para probar así sus servidores. Esto último es posible tanto en las Raspberry Pis como en las computadoras del laboratorio. Luego de sus pruebas iniciales, deberán probar sus servidores con los servidores de otros compañeros corriendo al mismo tiempo (usando el mismo puerto). En la evaluación final de este laboratorio, el instructor correrá el cliente y todos deberán correr sus servidores al mismo tiempo.

**Funciones útiles:**

A continuación, se listan algunas funciones adicionales a las usadas en los ejemplos cliente-servidor que les pueden ser útiles en sus programas. Estudien, compilen y corran el programa de ejemplo funciones_cadenas.cpp, en el que se ilustra el uso de las funciones (algunas ya las conocen de laboratorios y ejemplos anteriores). Se recomienda leer la descripción de las funciones, su uso y sus argumentos. Hay muchas otras funciones para manejo de cadenas que son útiles. **Siempre agregar 'std::' antes de llamar a estas funciones.**

**strcmp, strncmp, strcpy, strtok, sprintf, atoi, srand, rand**