Proyecto final Programación de Sistemas 2018-2T

El proyecto final consiste en la creación de una aplicación cliente – servidor para monitorear remotamente el estado de una computadora.

Programa servidor

El programa servidor debe poder mantener varias conexiones de clientes simultáneamente y tener el siguiente uso, el cuál debe mostrarse al usar la opción –h (mostrar ayuda en inglés):

```
./monitord –h
monitord broadcasts the host's usage and performance data to all
connected clients.

Usage:
  monitord [-c] [<port>]
  monitord [-l <log_file>] [<port>]
  monitord -h

Options:
  -h             Help, show this screen.
  -c             Console mode.
  -l <log_file>  Log file to use [default: log.txt].
```

Por defecto, *monitord* se ejecuta como *daemon* en el *background* y escucha conexiones de clientes en el puerto 8000. Eventos como una conexión nueva o algún mensaje de error deben ser registrados en la bitácora general de Linux usando la interface de la librería *syslog*. El usuario puede además especificar que se guarden los logs en un archivo local con la opción –l. En este caso, cada evento en el archivo de log debe tener un *timestamp* y si el evento es una conexión o desconexión debe registrarse la IP del cliente. El comportamiento por defecto puede ser modificado usando las opciones, por ejemplo:

```
./monitord –l /var/log/monitor.txt 7070
```

En este caso el programa guarda los eventos en un archivo *monitor.txt* en el directorio */var/log* y escucha conexiones de clientes en el puerto 7070. No usa syslog.

En modo consola, el programa no se ejecuta como *daemon* y los eventos son enviados a STDIN en lugar de un archivo log o syslog, por ejemplo:

```
./monitord –c 7070
[Jan 9 22:29:02] Conexión establecida con 192.168.10.25
```

Al establecerse una conexión, *monitord* debe enviar al cliente la siguiente información, cada segundo:

- Versión del sistema operativo

- Uptime
- Promedio carga en 1 minuto, 5 minutos y 15 minutos
- Número de procesadores
- Porcentaje de uso de cada procesador
- Memoria total
- Memoria libre
- Número de procesos
- Número de procesos en ejecución

La información debe ser serializada y enviada a través del socket usando el formato *Protobuf* (Protocol Buffers) de Google el cual debe ser interpretado correctamente por el cliente. Para usar Protocol Buffers en C se pueden usar los siguientes compiladores:

- https://github.com/protobuf-c/protobuf-c
- https://github.com/nanopb/nanopb

Protocolo de comunicación

El archivo .proto a usar contiene:

```
syntax = "proto3";

message SystemInfo {
	string system_name = 1;
	string version = 2;
	int32 num_processors = 3;
	int32 mem_total = 4; //en KB
}

message PerformanceInfo {
	float uptime = 1;
	repeated int32 processor_usage = 2; //en porcentaje, un valor
por procesador
	int32 mem_free = 3; //en KB
	int32 num_process = 4;
	int32 num_process_running = 5;

	message LoadAvg {
		float min5 = 1;
		float min10 = 2;
		float min15 = 3;
	}
	LoadAvg loadavg = 6;
}
```

Por lo tanto existen dos tipos de mensaje: `SystemInfo` y `PerformanceInfo`.

El mensaje `SystemInfo` es enviado una única vez al establecerse la conexión con el cliente. `PerformanceInfo` es enviado luego cada segundo tal como se muestra en Ilustración 1. Antes de enviar un mensaje, es necesario enviar su tamaño usando dos bytes (entero tipo short en *Little endian*).

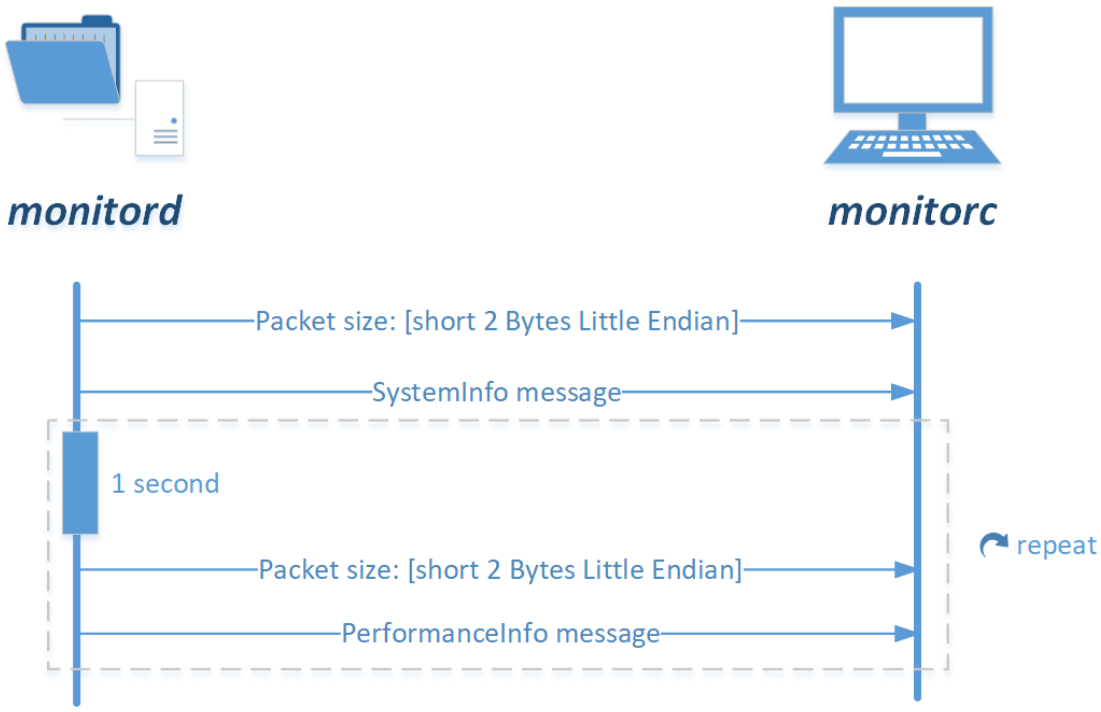

Ilustración 1: Diagrama de comunicación

Programa cliente

El programa cliente tiene el siguiente uso:

```
./monitorc –h
monitorc connects to a remote monitord service and display the
server’s usage and performance data.

Usage:
  monitorc [<ip>] [<port>]
  monitorc -h

Options:
  -h             Help, show this screen.
```

Por defecto la IP del servidor es 127.0.0.1 y el puerto 8000. Por ejemplo, asumiendo que *monitord* está corriendo en una máquina remota con la IP: 192.168.100.12 en el puerto 7070:

```
./monitorc 192.168.100.12 7070
Sistema: Ubuntu 16, Linux version 4.10.0-42-generic
Uptime: 1h 32m
Promedio carga (1-min, 5-min, 15-min): 1.98, 2.15, 2.21
Cores: 2
Core 1: 80%
Core 2: 20%
Memoria total: 2045804 kB
Memoria libre: 250900 kB
Número de procesos: 25976
Número de procesos en ejecución: 2
```

La salida de *monitorc* debe ser refrescada cada vez que llegue información nueva del servidor. Los datos se deben mantener en pantalla de manera similar al comando top. Se puede presionar 'q' para salir, en ese momento *monitorc* debe cerrar la conexión y terminar.

Restricciones

El programa servidor debe obtener la información del sistema de archivos */proc* (por ejemplo */proc/stat, /proc/meminfo*, etc.). El programa cliente debe recibir esta información y mostrarla en formato legible por un humano: Uptime debe estar en horas, minutos y segundos; el uso del procesador/es en porcentajes. En la Versión del Sistema Operativo como mínimo se debe mostrar la versión del *kernel* y la distribución. El programa *monitord* debe ser multi-hilo con cada conexión en un hilo diferente.