Memoria Práctica 1
Sistemas Distribuidos
Curso 2024/25
Universidad de Zaragoza

Diego Mateo Lorente - 873338
Carlos Solana Melero - 872815

Introducción

En esta primera práctica de la asignatura 'Sistemas Distribuidos', comenzamos a trabajar en el desarrollo de sistemas distribuidos a través de la utilización de un clúster que proporciona el laboratorio de prácticas. Este clúster actúa como nodo central de un conjunto de Raspberry Pis, que serán empleadas a lo largo de todas las actividades prácticas de la asignatura. El objetivo de la práctica es diseñar e implementar diversas arquitecturas cliente-servidor, además de una barrera distribuida. Todo el desarrollo se lleva a cabo utilizando el lenguaje de programación Go, el cual es especialmente adecuado para este tipo de sistemas distribuidos, dada su capacidad para manejar concurrencia y procesos en red de manera eficiente.

Ejercicio 1

En este apartado se analizarán las prestaciones de red de las Raspberry Pis empleando los protocolos TCP y UDP. Para ello, se han diseñado una serie de experimentos en los que se medirá el tiempo de ejecución de distintas operaciones de red. En primer lugar, se evaluará el comportamiento del protocolo TCP a través de la operación "Dial", tanto cuando falla como cuando se realiza correctamente, comparando los tiempos entre situaciones donde cliente y servidor están en la misma máquina y en máquinas diferentes. En segundo lugar, se medirá el tiempo que tarda en realizarse una transmisión simple entre un cliente y un servidor usando el protocolo UDP, evaluando también las diferencias cuando los dispositivos están en la misma máquina o separadas.

Así, las tablas con todos los tiempos quedan así:

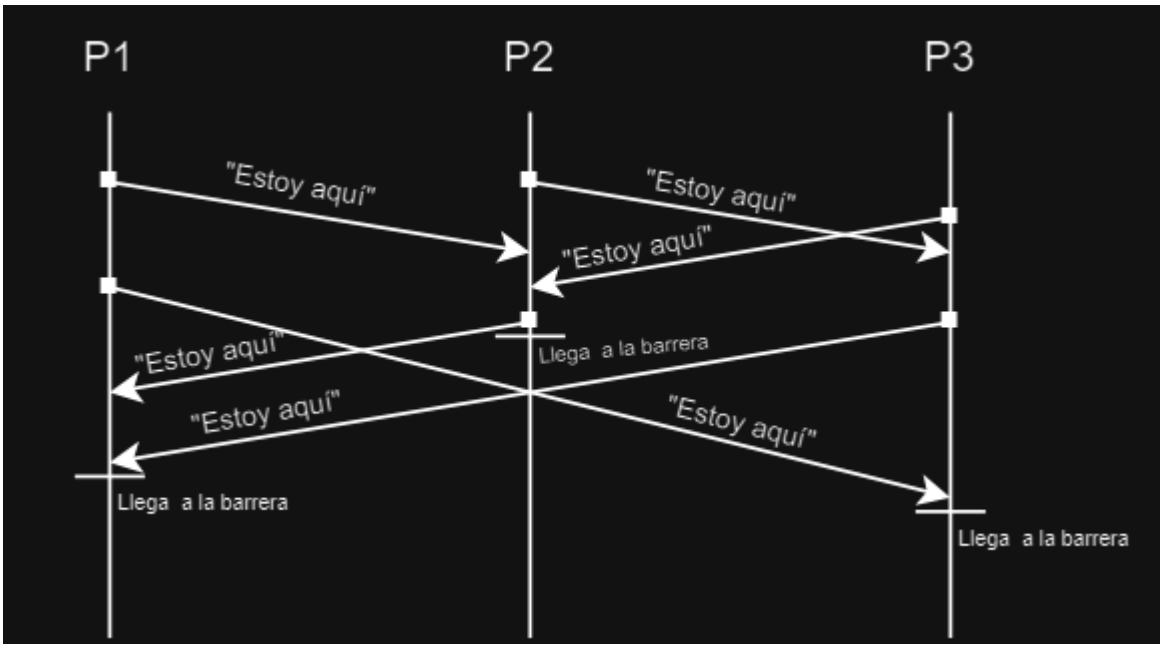
Tipo ejecución	Tiempo fallo medio(primer envío)	Tiempo éxito medio(segundo envío)	Variación
TCP en misma máquina	560.9846 us	1.198 ms	213.55%
TCP en distinta máquina	614.514 us	1.124077 ms	182.92%

Tipo ejecución	Tiempo medio envío
UDP en misma máquina	331.8173 us
UDP en distinta máquina	658.601 us

Ejercicio 2

Para este ejercicio. Se nos pide la realización de una barrera distribuida la cual debe ser traspasada por todos los procesos que la ejecutan. Para el caso del diagrama de secuencia, se ejecuta la barrera para 3 procesos distintos. Debe pasarse como parámetro de la barrera tanto el fichero que contiene todas las máquinas que van a ejecutar dicha barrera con su puerto, así como el número de línea en el que se encuentra la máquina en el fichero.

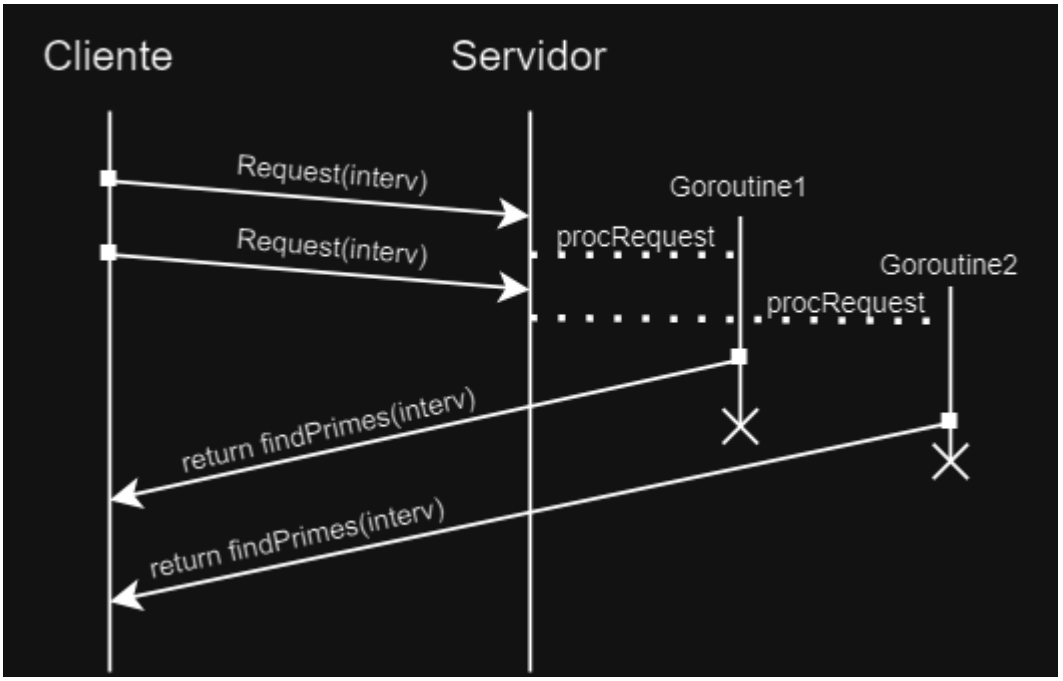
La barrera funciona de la siguiente manera. Para que los procesos alcancen la barrera, cada uno de ellos debe haber enviado un mensaje de notificación al resto de máquinas, así como haber recibido el mensaje de notificación del resto de máquinas también. Es decir, en el caso de 3 procesos, cada uno de ellos debe haber mandado un mensaje de notificación a cada una de las 2 máquinas, además de esperar un mensaje de notificación de cada una de las 2 máquinas. Cabe destacar que todo este envío y recibo de mensajes se realiza de forma distribuida, por lo que la velocidad de envío y recibo de mensajes será mayor. Se ha realizado un diagrama de secuencia del programa para mostrar de forma gráfica el funcionamiento del programa. Queda así:



Ejercicio 3

3.1.- Servidor / Cliente generando una Goroutine por petición

Para este apartado, se nos pide diseñar un cliente y un servidor que funcionen de la siguiente manera: El cliente manda una solicitud al servidor con un intervalo de números (envía el mínimo y el máximo de ese intervalo). Una vez hecho, el servidor crea una Goroutine con la función processRequest, la cual desencripta la solicitud, realiza la función findPrimes y devuelve el resultado directamente al cliente. Una vez hecho esto, la función termina y, por tanto, la Goroutine también. El diagrama de secuencia realizado muestra de forma clara el funcionamiento de este sistema, quedando así:

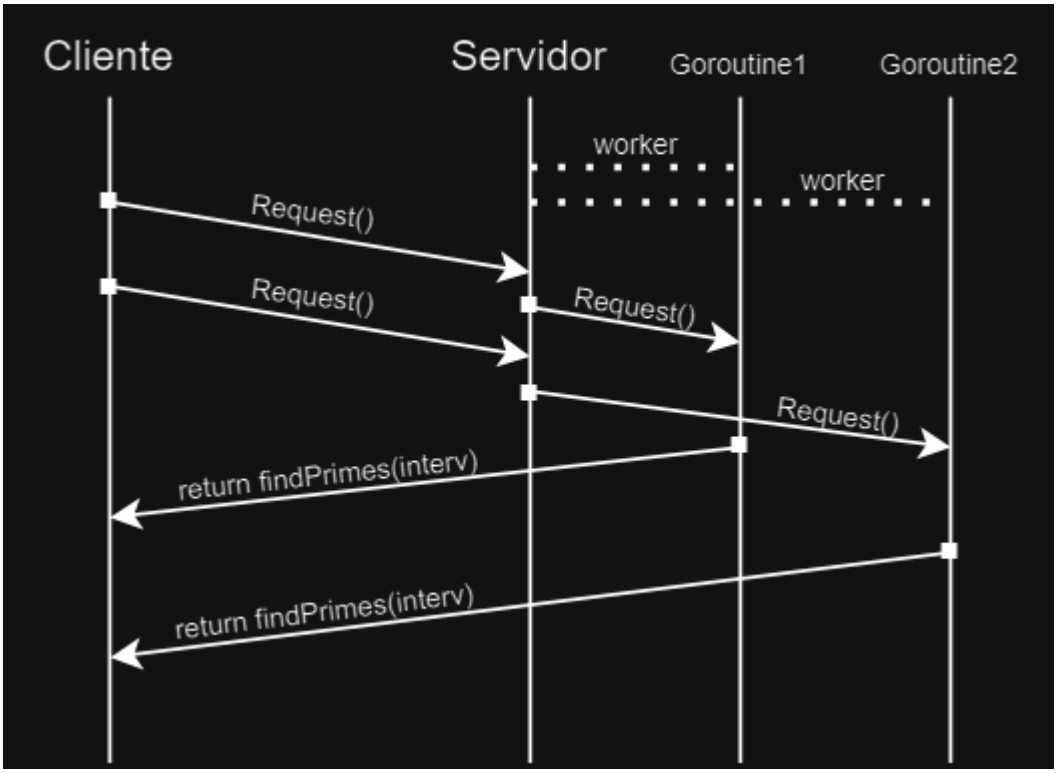


3.2.- Servidor / Cliente con pool fija de Goroutines

En este apartado se nos pide diseñar un sistema que funcione de forma similar al anterior, en el sentido de que el cliente solicita al servidor que devuelva todos los primos de un intervalo de números. La diferencia está en que en vez de generar una Goroutine por petición que encuentre la solución, el servidor nada más ser lanzado lanza una pool fija de Goroutines, cuyo tamaño es definido como parámetro de la función que lanza dicho servidor.

Dichas Goroutines son lanzadas con la función worker(), la cual lo que hace es recoger la tarea de un canal compartido entre el servidor y la pool de Goroutines. Una vez recogida esa tarea, al igual que en el apartado anterior, se desencripta la tarea, se buscan todos los primos y se devuelve el resultado al cliente.

En el caso del servidor, este se pone a la espera de que el cliente envíe la solicitud y, una vez recibida, este la sube a un canal de tareas para ser recogido por uno de los worker.
Una vez definido el sistema, se ha realizado un diagrama de secuencia que trata de simular el funcionamiento de este sistema con una pool de 2 Goroutines fijas, quedando así:



3.- Master / Worker

Para este último apartado, se pide algo un poco más peculiar. En este caso, se nos pide desarrollar un master, el cual será el encargado de lanzar una pool fija de worker, que son definidos en un fichero de texto que tendrá el master. El master va a lanzar dichos worker mediante ssh. Una vez hecho eso, define una pool de tantas Goroutines como worker haya. La idea es asignar una Goroutine por worker.

Entonces, una vez hecho esto, comienza el funcionamiento del sistema: El cliente, como en todas las arquitecturas, manda una solicitud con un intervalo de números esperando que se le devuelva el resultado. El servidor, una vez recibe esa petición y la mete en un canal de tareas para que una de las Goroutines la recoja y la desencripte.

Una vez hecho, la Goroutine manda dicha solicitud a la dirección del worker, el cual desencripta el mensaje, busca todos los primos del intervalo y devolverá el resultado de nuevo a la Goroutine. Finalmente, una vez la Goroutine reciba el resultado, se dispone a devolverlo al cliente.

El diagrama de secuencia que represena la arquitectura queda así: