Tarea 3.2

Un protocolo de red en capa de aplicación

Plazo límite: lunes 26 de noviembre a las 23:59 hrs., por Git.

1. Objetivo

El objetivo de esta tarea es implementar una aplicación de red que permita buscar y compartir archivos en internet, incorporando arquitecturas cliente-servidor y *peer-to-peer*.

2. Descripción

Desde fines de los años 90 comenzaron a popularizarse en Internet sistemas para compartir archivos entre los usuarios de la red, sin necesidad de que los archivos estuviesen alojados en servidores de disponibilidad permanente. Si bien los (malos) usos de esta tecnología desataron la crisis de la industria discográfica mundial, desde el punto de vista técnico es buen caso de estudio para un curso introductorio de redes como el nuestro.

En esta tarea deberás diseñar y luego implementar (en la parte 2) un protocolo básico para buscar y compartir archivos entre usuarios en Internet. Las restricciones básicas para diseñar el protocolo son las siguientes:

- Debe funcionar en la Internet pública, y permitir que usuarios en distintas redes IP puedan buscar y transferir archivos.
- Puede existir interacción con uno o varios hosts que actúen como *trackers* en la red y permitan descubrir hosts disponibles al realizar búsquedas de archivos.
- Una búsqueda de archivo debe requerir el string con el nombre de archivo (o parte de éste) a buscar, y puede ser respondida por trackers y/o hosts terminales de la red.
- Un archivo debe ser descargable desde un sólo host que lo contenga. Si varios hosts contienen un determinado archivo, queda libre el criterio bajo el cual elegir el host desde donde descargarlo.

2.1. Requisitos

La segunda parte de la tarea consiste en implementar aplicaciones de red que hagan uso del protocolo definido en la primera parte y que permitan demostrar su funcionamiento. La implementación podrá realizarse utilizando lenguajes Python versiones 2 ó 3, o Microsoft .NET. Para esto

será necesario que estudie las APIs de programación de red disponibles para estas plataformas (ver referencias), junto con las APIs para programación con *threads*.

Es necesario (obligatorio) que las aplicaciones de red que desarrolle permitan configurar flexiblemente las direcciones de red y puertos de los hosts permanentes (p.ej., *trackers*), como también el (o los) directorio(s) en donde residen los archivos a compartir. Se recomienda que su aplicación lea un archivo de configuración desde donde pueda obtener estos parámetros, en formatos JSON, YAML, XML, o incluso, texto de formato arbitrario.

3. Evaluación

La evaluación de la primera parte de la tarea se realizará mediante rúbrica que contemplará los siguientes criterios:

40 % Función de búsqueda de archivos.

40 % Función de descarga de archivo p2p.

20 % Fidelidad de la implementación con el protocolo diseñado en la primera parte de la tarea.

Cada grupo deberá realizar una demostración de sus aplicaciones de red en el laboratorio FICA-COM. Habrá un conjunto de PCs habilitados para realizar la demostración (posiblemente tres peers y un tracker), por lo que cada grupo debe traer por lo menos un pendrive con las aplicaciones listas para ejecutar en Windows 10, o Ubuntu 16.04 LTS. Las direcciones IPs (o nombres de dominio) de los hosts en donde deberá realizarse la demostración serán informados con días de anticipación a la entrega.

La reserva de horarios para demostración en FICA-COM se realizará a través de Doodle, días antes de la fecha de entrega.

3.1. Modalidad de Trabajo

La tarea debe ser desarrollada en grupos de dos (parejas). Cada integrante debe contar con una cuenta de usuario en GitHub. El desarrollo debe realizarse por ambos integrantes, y deben quedar claramente registradas sus operaciones de *commit* en el sistema, con comentarios descriptivos en cada una de estas operaciones.

Los grupos podrán continuar trabajando en el repositorio en donde desarrollaron la primera parte de la tarea.

3.2. Referencias

- Python Network Programming. Disponible en `https://www.tutorialspoint.com/python/python_networking.htm`

- Sockets en Python 3.7. Disponible en `ttps://docs.python.org/3.7/library/socket.html`
- Multithreaded Programming (en Python). Disponible en `https://www.tutorialspoint.com/python3/python_multithreading.htm`
- Programación con Sockets en Framework .NET: `https://docs.microsoft.com/en-us/dotnet/framework/network-programming/how-to-create-a-socket`
- Programación concurrente con threads in .NET (C#): `https://www.tutorialspoint.com/csharp/csharp_multithreading.htm`