Proyecto I

Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Redes (IC 7602)
Primer Semestre 2025

TEC | Tecnológico de Costa Rica

---

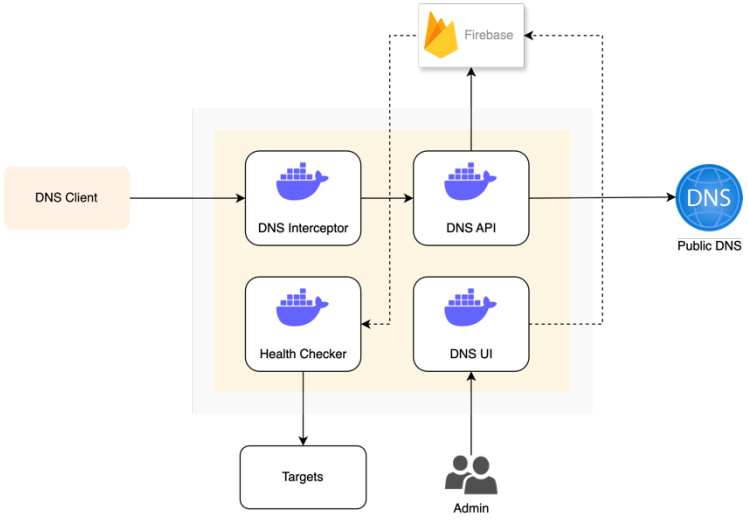
1. **Objetivo General**
   - ➔ Implementar servicios de capa de aplicación sobre protocolos de transporte UDP y TCP.
2. **Objetivos Específicos**
   - Desarrollar servidores UDP y TCP en lenguajes de programación C y Python respectivamente.
   - Implementar el procesamiento de algunas peticiones DNS especificadas en RFC-2929.
   - Desarrollar una aplicación que transporte un servicio no orientado a conexión sobre un servicio orientado a conexión.
   - Interactuar con documentación formal de redes.
   - Implementar un REST API en Python.
   - Automatizar una solución mediante el uso de Docker, Helm Charts, Terraform, Docker Compose, Kubernetes o Cloud Providers.
   - Implementar health checks para protocolos TCP y HTTP.

3. **Datos Generales**
   - ➔ El valor del proyecto: 30%
   - ➔ La tarea debe ser implementada de forma individual o en grupos de máximo 5 personas.
   - ➔ La **fecha de entrega** es 04 de Mayo del 2025 antes de las 11:59 pm.
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo con el reglamento. La copia incluye código/configuraciones que se puede encontrar en Internet y que sea utilizado parcial o totalmente sin el debido reconocimiento al autor.
   - ➔ La revisión es realizada de forma virtual mediante citas en las cuáles todos los y las integrantes del grupo deben estar presentes, el proyecto debe estar completamente automatizado con Docker compose/Terraform o Helm, el profesor decide en que computadora probar.
   - ➔ Se debe incluir documentación con instrucciones claras para ejecutar el proyecto.
   - ➔ Se espera que todos y todas las integrantes del grupo entiendan la implementación suministrada.
   - ➔ Se deben seguir buenas prácticas de programación en el caso de que apliquen. Por ejemplo, documentación interna y externa, estándares de código, diagramas de arquitectura, diagramas de flujo, pruebas unitarias son algunas de las buenas prácticas que se esperan de un estudiante de Ingeniería en Computación en sus cursos finales.
   - ➔ Los sitios web de prueba pueden ser simples páginas web estáticas.
   - ➔ Toda documentación debe ser implementada en Markdown.
   - ➔ Si la instalación de servicios no se encuentra automatizado Chef Solo/Ansible (o similar) se obtendrá una nota de 0. Si el proyecto no se entrega completo, la porción que sea entregada debe estar completamente automatizada, de lo contrario se obtendrá una nota de 0.

- ➔ Si la creación de las redes no se encuentra automatizado con Terraform se obtendrá una nota de 0.
- ➔ En caso de no entregar documentación se obtendrá una nota de 0.
- ➔ El email de entrega debe contener una copia del proyecto en formato tar.gz y un link al repositorio dónde se encuentra almacenado, debe seguir los lineamientos en el programa de curso.

4. **Descripción**

El proyecto I busca profundizar los temas de capa de aplicación y capa de transporte mediante la implementación de aplicaciones que utilicen protocolos TCP y UDP. Como parte de este ejercicio, se deberá implementar un servicio de DNS con algunas de las características ofrecidas por el RFC1035



Se deberán crear los siguientes componentes:

- **DNS Interceptor**: Este es un Docker container, que ejecuta una pequeña aplicación escrita en lenguaje de programación GNU C, la misma escuchará en el puerto UDP/53, esta aplicación recibirá paquetes del protocolo DNS, al recibir estos, deberá examinar el paquete siguiendo la especificación oficial en el RFC2929 y tomar alguna de las siguientes acciones:
    1. En el caso de recibir un paquete diferente a *query estándar* (QR = 0 y OPCODE = 0), deberá codificar la solicitud a BASE64 y enviarla vía HTTPS (POST /api/dns_resolver) al componente llamado DNS API, cuando se recibe la respuesta esta vendrá en BASE64, la misma debe ser decodificada y enviada al cliente solicitante.
    2. En el caso de recibir un paquete de tipo *query estándar* (QR = 0 y OPCODE = 0), deberá examinarlo detenidamente e identificar el host que se está tratando de resolver, una vez que el host ha sido identificado, deberá buscar en Firebase a través del DNS API si existe un registro para este host y tomar alguna de las siguientes acciones:

        - Si existe, deberá extraer la información de esta base de datos y retornar la información, la información que retorna, debe cumplir al 100% con la especificación de los protocolos en RFC2929 y RFC1035 para asegurar que los clientes existentes como nslookup y los sistemas de resolución de nombres de Windows/Linux pueden comunicarse con su componente sin ningún problema. Se pueden dar los siguientes casos al obtener el valor de Firebase:
            - El tipo de registro es de tipo "single", en este caso solo se tiene un IP y se retorna este.
            - El tipo de registro es "multi", en este caso se tienen múltiples IPs, se espera que las personas estudiantes puedan idear una forma creativa de implementar un round-robin, de esta forma diferentes peticiones, retornaran IPs diferentes.
            - El tipo de registro es "round-trip" se deberá retornar el IP con menor latencia dependiendo de la cercanía con el usuario.
            - El tipo de registro es "weight", en este caso se tendrá múltiples IPs con un peso, el que tiene el peso más grande, sirve más peticiones.
            - El tipo de registro es "geo", en este caso se tienen IPs asignados por país, se toma el "source IP" de la petición original y se busca en la base de dato IP to Country (esta base de datos se debe cargar en Firebase), este permitirá identificar a que país pertenece el IP, con esto se retornara el valor adecuado para el país, en caso de no existir un registro para el país se retorna alguno de los IPs de forma aleatoria.
        - En caso de no existir o que el IP que se debe asignar se encuentre unhealthy, la solicitud se trata como el primer caso.

    ** Esta aplicación deberá permitir múltiples solicitudes de forma simultánea.

- DNS API: Un container que ejecuta un REST API implementado en Python, que ejecuta los métodos necesarios para implementar la aplicación, al menos son necesarios dos métodos (/api/dns_resolver y /api/exists) con verbo HTTP (POST) y HTTP (GET) respectivamente.
    - /api/exists: Este método se usará para verificar si un dominio existe en Firebase.
    - /api/dns_resolver: Este método recibe en el data un paquete DNS (RFC2929) codificado en BASE64, el mismo deberá ser decodificado y enviado a un servidor DNS remoto para su resolución, esto implica lo siguiente:
        - Esta aplicación deberá implementar un REST API (pueden usar cualquier framework) pero también deberá implementar un cliente UDP/DNS para enviar las solicitudes hacia el servidor DNS remoto.
        - Esta aplicación deberá recibir como parámetro/archivo de configuración el IP de un servidor DNS remoto.
        - Esta aplicación deberá soportar múltiples peticiones al mismo tiempo.

  Cualquier otro método requerido en el API, deberá ser definido por cada grupo. Toda información nativa de DNS (paquetes) que se intercambian entre el DNS Interceptor y el DNS API se debe codificar en BASE64.

- Health Checker: Este es un Docker container, que ejecuta una pequeña aplicación escrita en lenguaje de programación GNU C, esta aplicación lee los registros que se crean en Firebase y se encarga de realizar el health check respectivo y marcar el registro como healthy o unhealthy.
- DNS UI: Un container que ejecuta una aplicación React que permite:
    - Crear/borrar/modificar registros de tipo “single”, “multi”, “weight”, “round-trip” y “geo”, junto con la definición de sus posibles health checks:
        - TCP: Verifica si una conexión HTTP puede ser abierta, se debe especificar un timeout, retries y un tiempo entre pruebas (por ejemplo, cada 30 segundos).
        - HTTP: Se especifica un path, timeout, retries, un tiempo entre pruebas (por ejemplo, cada 30 segundos) y varios HTTP codes que se esperan recibir de vuelta.
    - Crear/borrar/modificar registros de la base de datos IP to Country.

El proyecto debe correr en Docker, el mismo se puede ejecutar localmente en Kubernetes, así como en una máquina virtual en algún Cloud Provider.

Registro round-trip y Health Checker

Para una implementación satisfactoria de los registros round-trip, es necesario tener múltiples Health Checkers en diferentes ubicaciones geográficas, cada uno tendrá asociado una latitud/longitud/País/Ciudad, cuando un Health Checker realiza una prueba, envía la información a la base de datos Firebase, cuando un cliente solicita la resolución de un nombre que es de tipo round-trip, mediante la base de datos IP to Country podemos conocer su país/ciudad (algunas versiones incluyen la ciudad, si no trabajamos con país) y con esta información podemos conocer su ubicación latitud/longitud/País/Ciudad para realizar una comparación del Health Checker más cercano y que tenga la menor latencia. Debido a la dificultad de implementar este tipo de pruebas, se invita a los grupos a realizar alguna simulación o penalización de los “round-trip” para verificar que su implementación funciona.

Documentación

Se deberá entregar una documentación que al menos debe incluir:

- Instrucciones claras de como ejecutar su proyecto.
- Pruebas realizadas, con pasos para reproducirlas.
- Recomendaciones y conclusiones (al menos 10 de cada una).
- La documentación debe cubrir todos los componentes implementados o instalados/configurados, en caso de que algún componente no se encuentre implementado, no se podrá documentar y tendrá un impacto en la completitud de la documentación.

Imaginen que la documentación está siendo creada para una persona con conocimientos básicos en el área de computación y ustedes esperan que esta persona pueda correr su proyecto y probarlo sin ningún problema, el uso de imágenes, diagramas, code snippets, videos, etc. son recursos que pueden ser útiles.

5. **Recomendaciones**
    - Utilizar nslookup para interactuar con el DNS Resolver.
    - Utilizar postman para interactuar con el DNS API
    - Si su proyecto funciona correctamente, podrían modificar el archivo resolv.conf de sus computadoras, ponerlo a apuntar al IP de su contenedor y estas deberían seguir funcionando sin problemas.
    - Realicen peer-review/checkpoints semanales y no esperen hasta el último momento para integrar su aplicación.
6. **Entregables**

    - Documentación.
    - Todos los archivos/scripts requeridos para ejecutar su proyecto.

7. **Evaluación**

Funcionalidad / Requerimiento	Porcentaje
Documentación	15%
Implementación (*): ● DNS Interceptor (30%) ● DNS API (25%) ● DNS 2.0 UI (15%) ● Health Checker (15%)	85%
	100%

(*) Todo tiene que estar debidamente automatizado, de lo contrario se calificará con una nota de 0.