Universidad José Antonio Paez.
Facultad de Ingeniería.
Escuela de Ingeniería en Computación.

Simulador de Red de Dispositivos (LAN) — CLI Estilo Router

Programar un simulador de una red de área local (LAN) que funcione íntegramente desde una interfaz de línea de comandos (CLI), inspirada en la consola de configuración de routers reales, como el sistema IOS de Cisco.

El simulador permitirá crear, conectar y administrar distintos tipos de dispositivos virtuales —como routers, switches, computadoras y firewalls— con el objetivo de simular el envío de paquetes, su procesamiento, propagación y análisis del comportamiento de la red.

Módulo 1: Dispositivos y Red (4 ptos)

En este módulo se define el corazón del simulador: las entidades que conforman la red y la propia topología. Cada dispositivo (router, switch, host o firewall) se modela como una instancia de la clase Device, capaz de mantener un nombre, un tipo y un conjunto de interfaces —por ejemplo g0/0 o eth0— sobre las cuales se pueden asignar direcciones IP simuladas y activar o desactivar su estado (“shutdown” / “no shutdown”). Para representar las conexiones físicas, cada interfaz crea vínculos con las de otros dispositivos y aquellos enlaces se almacenan internamente mediante una lista enlazada, lo que facilita recorrer vecindarios y descubrir rutas.

La clase Network asume la responsabilidad de orquestar el conjunto de dispositivos, permitiendo agregar nuevos nodos, establecer o quitar conexiones y consultar rápidamente qué dispositivos existen en la red. Cuando un nuevo dispositivo se une o se conecta, Network actualiza la topología y se asegura de que todos los vecinos sean perfectamente consistentes. De esta manera, este módulo prepara la base sobre la que se construyen todas las operaciones de envío de paquetes y configuración remota.

Comandos relevantes

1. enable:Al iniciar sesión en nuestro simulador, el prompt primero muestra algo como: Router1>

   Ese símbolo de mayor (>) indica **modo usuario**, con capacidad de ejecutar únicamente comandos muy básicos (por ejemplo, send, ping). Para acceder a las operaciones de configuración de la red y administración de dispositivos, primero debes elevar tu privilegio al **modo privilegiado**.

2. configure terminal: Ya en modo privilegiado (Router1#), este comando te permite entrar al **modo de configuración global**, donde podrás definir nombre del dispositivo, interfaces, IPs, etc
3. hostname <device_name>: Dentro de **configure terminal**, este comando asigna o modifica el nombre lógico del dispositivo

4. interface <iface_name>: Aún en modo configuración global, con interface entras al **modo de configuración de esa interfaz** particular. Por ejemplo:
5. ip address <ip>: Dentro del submodo config-if, este comando asigna la dirección IP simulada a la interfaz activa:
6. shutdown / no shutdown: Con shutdown desactivas la interfaz; no shutdown la reactiva. Esto modela el comportamiento real de un puerto de red que puede estar “down” (inactivo) o “up” (activo)
7. exit: Cada vez que finalices un submodo de configuración, usa exit para **volver** al nivel superior de prompt:
8. connect <iface1> <device2> <iface2>: Una vez configuradas las interfaces y asegurado su estado “up”, este comando **crea la conexión física** entre dos puertos de dispositivos distintos:
9. disconnect <iface1> <device2> <iface2>: Este comando revierte la operación de connect, **eliminando** el vínculo entre dos interfaces y actualizando la estructura interna de la red:
10. set_device_status <device> <online|offline>: Permite **activar o desactivar** por completo un dispositivo, sin afectar la configuración de sus interfaces. Cuando un nodo está en estado offline, no podrá enviar ni recibir paquetes
11. list_devices: Para obtener un **listado rápido** de todos los nodos presentes en la red, su nombre y su estado (online/offline)

Ejemplo de sesión

```
Router1> enable
Router1# configure terminal
Router1(config)# hostname Router1
Router1(config)# interface g0/0
Router1(config-if)# ip address 192.168.1.1
Router1(config-if)# no shutdown
Router1(config-if)# exit
Router1(config)# exit
Router1# connect g0/0 Switch1 g0/1
Router1# list_devices
Devices in network:
  - Router1 (online)
  - Switch1 (online)
```

Módulo 2: Paquetes y Comunicación (4 ptos)

Aquí definimos la estructura interna de un paquete de red virtual y las rutinas que simulan su tránsito. La clase Packet encapsula el origen y destino (ambos en formato de dirección IP), el contenido de texto que transporta, un contador de saltos (TTL) que disminuye en cada salto, y una lista de nodos por los que ha pasado para permitir el trazado posterior.

Cuando el usuario emite un comando de envío, el paquete se coloca en la cola de salida de la interfaz correspondiente. Al invocar el comando tick (o su alias process), el

simulador recorre todas las colas de cada dispositivo; extrae los paquetes, decrementa su TTL y los reenvía al siguiente salto según la topología vigente. Si la TTL llega a cero antes de alcanzar el destino, el paquete se descarta; si llega con éxito, el dispositivo destino lo añade a su pila de historial de recepción.

Cada paquete debe contener: Identificador único, Origen, Destino, Contenido, TTL (Time To Live), Traza de ruta,

Comandos relevantes

```
send <source_ip> <destination_ip> <message> [ttl]
tick              # avanza 1 paso de simulación (procesa todas las colas)
process           # alias para tick
```

Ejemplo de uso

```
console PC1

PC1> send 10.0.0.2 "Hello, World!" 5
Message queued for delivery.

PC1> tick
[Tick] PC1 → Router1: packet received (TTL=4)

Router1> tick
[Tick] Router1 → PC2: packet forwarded (TTL=3)

PC2> show history
Packet from 10.0.0.1 to 10.0.0.2: "Hello, World!"
TTL expired? No | Hops: PC1 → Router1 → PC2
```

Módulo 3: Estructuras de Datos (3 ptos)

Este módulo implica implementar desde cero las estructuras esenciales: listas enlazadas, colas y pilas. La lista enlazada se utiliza para almacenar de manera dinámica los vecinos de cada interfaz, la cola gestiona los paquetes entrantes y salientes de cada dispositivo, y la pila registra el historial de mensajes recibidos, de modo que el último paquete almacenado sea el primero en consultarse.

Módulo 4: Interfaz de Línea de Comandos (3 ptos)

Imitando la experiencia de la CLI de un router profesional, este módulo desarrolla un parser capaz de distinguir varios niveles de contexto: el modo usuario (Device>), el modo

privilegiado (Device#), el modo de configuración global (Device(config)#) y el modo de configuración de interfaz (Device(config-if)#). Cada nivel ofrece un conjunto concreto de comandos (por ejemplo, enable, show, configure terminal, interface, exit, end) y el parser debe traducir cada instrucción en llamadas a las clases Device y Network que realizan la acción solicitada. La gestión de prompts, la validación de sintaxis y el manejo de errores de entrada también forman parte de este módulo.

El CLI debe ser programado utilizando el patrón comandos.

enable
disable
exit # retrocede un nivel de modo
end # sale a modo privilegiado
show <subcommand>
save running-config
load config <filename>

Módulo 5: Estadísticas y Reportes (3 ptos)

Más allá de la mera transmisión de paquetes, el simulador debe ofrecer al usuario información detallada sobre el estado de la red. Con comandos show history <device>, show queue <device> o show interfaces, cada dispositivo informa sobre los mensajes que ha recibido, los paquetes pendientes y la configuración de sus puertos. A nivel global, el comando show statistics presenta métricas como el número total de paquetes enviados, los entregados con éxito, los descartados por TTL o bloqueados por el firewall, el promedio de saltos por paquete e incluso cuál ha sido el dispositivo con más actividad. Estas consultas refuerzan la comprensión de cómo las estructuras de datos almacenan y recuperan información en tiempo real.

Comandos relevantes
show history <device>
show queue <device>
show statistics

Ejemplos de salida

Historial de un dispositivo:

Router1# show history
1) From 10.0.0.1 to 10.0.0.2: "Hello" | TTL at arrival: 3 | Path: PC1 → Router1 → PC2

Estadísticas de red:

Router1# show statistics
Total packets sent: 15
Delivered: 13
Dropped (TTL): 1
Average hops: 2.1

Top talker: Router1 (processed 20 packets)

Módulo 6: Persistencia de Configuración (3ptos)

Para que el simulador sea realmente útil en escenarios de uso continuado, debe ofrecer la capacidad de **guardar** y **cargar** su estado completo. El comando save running-config volca en un archivo de texto (running-config.txt por defecto) toda la información de dispositivos, interfaces, direcciones IP y conexiones, usando una sintaxis muy similar al lenguaje de la CLI. Más adelante, load config <filename> lee ese mismo formato y reconstruye todos los objetos internos del simulador de forma automática. Esto permite pausar y retomar prácticas o desplegar configuraciones complejas sin tener que reescribir cada comando manualmente.

Ejemplo de archivo network.cfg

```
hostname Router1
interface g0/0
  ip address 192.168.1.1
  no shutdown
exit
interface g0/1
  ip address 10.0.0.1
exit

hostname Switch1
interface g0/1
  no shutdown
exit

connect Router1 g0/0 Switch1 g0/1
connect Router1 g0/1 PC1 eth0
```

Comandos

```
Router1# save running-config
Configuration saved to running-config.txt

Router1# load config network.cfg
Configuration loaded successfully.
Devices and connections restored.
```

Pautas de Evaluación.

1. La evaluación es individual o en equipo de un máximo de tres personas.
2. Usar paradigma de programación orientada a objetos
3. Asistencia obligatoria
4. Utilizar repositorio tiene un valor de 2 ptos.
5. Los códigos iguales tendrán una penalización de puntos menos.

6. La entrega y defensa se realizará de forma presencial en hora de clases.
7. Realizar validaciones de datos introducidos por el usuario en los comandos y datos cargados.
8. Los datos deben ser guardados como archivos.json y cargados al momento de iniciar el programa.
9. El código deberá estar comentado.
10. Tener datos por defectos para tomarlos como prueba.
11. En cada módulo se evaluará los siguiente:
    - Funcionamiento del módulo(errores, resultados correctos, independencia).
    - Legibilidad del código(nombres de variables, código comentado correctamente)
    - El alumno aplicó elementos conceptuales en la programación del módulo
    - Módulo entregado puntualmente.