

# Redes de Computadoras

## Obligatorio 2 – Instructivo Parte 2

Facultad de Ingeniería  
Instituto de Computación  
Departamento de Arquitectura de Sistemas

### Nota previa - IMPORTANTE

Se debe cumplir íntegramente el “Reglamento del Instituto de Computación ante Instancias de No Individualidad en los Laboratorios”, disponible en el EVA.

En particular está prohibido utilizar documentación de otros estudiantes, de otros años, de cualquier índole, o hacer público código a través de cualquier medio (EVA, news, correo, papeles sobre la mesa, etc.).

## Introducción

A continuación presentamos un instructivo más detallado de como deberá implementar la segunda parte del Obligatorio 2. Esto no sustituye la letra sino que la complementa, por lo que sugerimos leer ambos documentos.

Recordemos que en esta parte el objetivo es desarrollar la función de enrutamiento de un *router*, es decir, encontrar de forma dinámica y autónoma las rutas a los distintos destinos de la red. Se implementará un protocolo de enrutamiento dinámico simple, llamado PWOSPF, para que el router pueda generar su tabla de reenvío automáticamente basándose en las rutas anunciadas por otros routers de la red.

## PWOSPF

El protocolo de enrutamiento que implementará es un protocolo de estado de enlace que se basa vagamente en OSPFv2. Se encuentra disponible la especificación completa de PWOSPF [aquí](#). Recomendamos estudiar detalladamente la especificación antes de comenzar a implementar.

Como verá en la especificación, PWOSPF basa su funcionamiento en dos tipos de mensajes: mensajes HELLO y mensajes LSU (de Link State Updates). Con estos mensajes cada router de la red puede conocer el resto de routers y armar una visión local de la topología de toda la red. Su tarea consistirá en implementar la gestión (envío y recepción) de estos dos tipos de mensajes. No deberá implementar el mantenimiento de la topología ni el armado de la tabla de enrutamiento. Tampoco deberá implementar el algoritmo de búsqueda del camino mas corto (Dijkstra).

## Descripción del código

Para la implementación de esta segunda parte tiene a su disposición código modificado que agrega varias de las funcionalidades necesarias para su implementación.

El archivo principal donde deberá trabajar es `sr_pwospf.c`. En ese archivo se encuentran las funciones principales para la gestión de los mensajes PWOSPF. Usted deberá implementar las funciones:

- `void* check_neighbors_life(void*);`
- `void* check_topology_entries_age(void*);`
- `void* send_hellos(void*);`
- `void* send_hello_packet(void*);`
- `void* send_all_lsu(void*);`
- `void* send_lsu(void*);`
- `void sr_handle_pwospf_hello_packet(struct sr_instance*, uint8_t*, unsigned int, struct sr_if*);`
- `void* sr_handle_pwospf_lsu_packet(void*);`

El archivo ya incluye implementada la función de inicialización que inicia el hilo principal de ejecución así como hilos separados para todas las tareas periódicas: envío de HELLOs, envío de LSUs, chequeo de vecinos y chequeo de topología. Además, se incluyen funciones para la mutua exclusión de los hilos.

Se incluye además el archivo `pwospf_protocol.h` con una serie de definiciones de estructuras y constantes que le serán muy útiles en su implementación.

Por otro lado, también se incluyen archivos que ya resuelven la gestión de los vecinos (`pwospf_neighbors.h/.c`), la gestión de la topología (`pwospf_topology.h/.c`) y la implementación del algoritmo de Dijkstra (`dijkstra.h/.c`). Estudie las funciones que le proveen estos módulos ya que serán de ayuda para su implementación.

Por último, se incluyen pequeñas modificaciones de varios otros archivos, entre ellas los archivos `router.c` y `router.h`, donde se agrega un atributo a `sr_instance` y se agrega la llamada para iniciar el hilo que gestiona PWOSPF. Usted deberá agregar su código de la primera parte aquí teniendo en cuenta estas modificaciones.

Además, dado que los mensajes PWOSPF viajan dentro de un paquete IP, deberá hacer las modificaciones necesarias para aceptar estos paquetes y llamar a la función `sr_handle_pwospf_packet` cuando corresponda.

## Algunas consideraciones a tener en cuenta

### Destino de los paquetes PWOSPF

Como verá en la especificación del protocolo, los mensajes HELLO se deben enviar a una dirección IP de multicast específica. Esta dirección ya se encuentra definida en la constante `OSPF_ALLSPFRouters`. Pero además, dado que para esta dirección IP también es necesario contar con una dirección MAC correspondiente, para esto se encuentra ya definida en `sr_router.c` y `sr_pwospf.c` esta dirección.

Para que la recepción de mensajes funcione, es importante que en `sr_router.c` los routers acepten los mensajes con estas direcciones MAC e IP. Además, será necesario aceptar los paquetes IP cuyo tipo sea el correspondiente a `ip_protocol_ospfv2`.

## Rutas estáticas vs. dinámicas

Para la diferenciación entre rutas estáticas y dinámicas, se ha agregado un parámetro en la tabla de enrutamiento, `admin_dst`. Este valor debe ser 1 para rutas directamente conectadas, 0 para rutas estáticas agregadas a mano y mayor a 1 para el resto de las rutas (las agregadas por PWOSPF).

De esta forma usted el router podrá diferenciar las rutas que debe enviar en los mensajes LSU.

## Qué rutas anunciar en los LSU

El router es responsable de anunciar todas sus rutas estáticas junto con las subredes directamente conectadas a cada una de sus interfaces.

No deberá añadir una ruta en su tabla de enrutamiento para subredes conectadas directamente a una de sus interfaces que sean anunciadas por otros routers. Estas rutas ya se incluyeron en la inicialización.

## Mantenimiento de vecinos y periodicidad de mensajes HELLO

Además de la tabla de vecinos que se implementa en `pwosp_neighbors`, se espera que cada interfaz conozca su vecino en el enlace. Para esto se agregaron los atributos correspondientes al struct `sr_if`.

Por otro lado, cada interfaz cuenta ahora también con un contador `helloint` para mantener de forma independiente por interfaz el temporizador para los mensajes HELLO.

## El código entregado

El código de esta parte se encuentra en el subdirectorio enrutamiento de:  
[https://gitlab.fing.edu.uy/mrichart/redes2024\\_ob2](https://gitlab.fing.edu.uy/mrichart/redes2024_ob2)

Para descargarlo puede hacer:

```
> git clone https://gitlab.fing.edu.uy/mrichart/redes2024\_ob2.git
```

O si ya tiene el repositorio descargado, puede hacer lo siguiente para actualizarlo:  
> git pull

Al momento de la ejecución es importante tener en cuenta que para esta segunda parte deberá modificar las tablas de enrutamiento `rtable.vhost` para que estén vacías.