Actividad Fragmentación

1. Consideraciones

A continuación se presentan las decisiones de diseño que se tomaron a la hora de implementar métodos y manejo de datos en datagramas y tablas de rutas para la presente actividad:

- Se consideró que las tablas de rutas comienzan con el nuevo formato definido en la actividad, es decir el formato:

```
[Red destino (CIDR)] [ruta ASN] [IP_siguiente_salto] [Puerto_siguiente_salto] [MTU]
```

2. `router.py`

Esta módulo contiene la clase `Router`, la cual representa un router con las funcionalidades para manejar y parsear los paquetes IP, asimismo maneja el forwarding de estos paquetes chequeando las rutas de la tabla de rutas otorgada según el nuevo formato de BGP. Finalmente también maneja el fragmentado y posterior re-ensamblaje de paquetes en caso de que la ruta posea un MTU menor al tamaño del paquete. Esta clase se encuentra totalmente documentada por lo que no requiere entrar en detalle respecto a los métodos implementados.

Adicionalmente, contiene la clase `BGP` encargada de realizar el algoritmo BGP para compartir rutas entre los distintos AS. Ambas clases se encuentran totalmente documentadas por lo que no requiere entrar en detalle respecto a los métodos implementados.

3. `router_creator.py`

Este código crea un router con dirección IP según los parámetros entregados en la ejecución de este, como se muestra abajo, adicionalmente se encarga de manejar un diccionario con los paquetes/fragmentos que tengan destino la dirección asociada, en tal caso se realiza el manejo adecuado de estos mediante los correspondientes métodos de la clase `Router` y `BGP`.

```
$ python3 router_test.py router_IP router_puerto router_rutas.txt
```

4. Pruebas

- **Primero ponga a correr su algoritmo BGP en cada router y verifique que sus tablas de ruta convergen. Observe las tablas de ruta impresas y compruebe que cada router tiene una ruta a cada uno de los otros 6 routers dentro de la configuración.**

  Como se observa de las tablas de rutas generadas al enviar el paquete `START_BGP` en la sección Anexo, para cada router se crean rutas hacia los 6 routers que están conectados en la red.

- **Pruebe sus nuevas tablas de rutas usando `netcat`. Para ello pruebe enviando mensajes desde R7 a R1, de R7 a R4, y de R7 a R3. ¿Quién imprime el mensaje? Recuerde que puede usar `netcat` como se muestra a continuación. Añada sus observaciones al informe.**

  En todos los casos el mensaje es impreso solo por el router receptor del mensaje, esto se realiza debido a que como se consideró desde un principio el nuevo formato de rutas acorde BGP, al

hacer el chequeo de rutas en la tabla de rutas de un router en caso de poseer como ASN de destino el puerto asociado al router entonces se envía directamente a este router el mensaje.

- **Ahora suponga que inicialmente solo tiene los primeros 6 routers (R1 a R6) tal que echa a correr BGP por primera vez considerando solo a estos 6 routers. Suponga que R7 se une a la red en un momento posterior a esta llamada inicial de BGP y que R7 quiere ser invisible para R4 ¿Cómo puede modificar la tabla de rutas inicial de R7 para que, luego de ejecutar `run_BGP()`, R4 no pueda alcanzar a R7? Pruebe que su respuesta funciona utilizando su código. Añada su respuesta y observaciones al informe.**

  Una forma de hacer que R7 sea invisible es creando un campo adicional en la clase `BGP` que se encargue de llevar un registro de los ASN con los cuales no quiere tener comunicación, seria un análogo a las políticas de ruteo de los AS, esto se puede lograr creando una lista con los ASN no permitidos, luego cuando llegue un paquete `BGP_ROUTES` se tendrá que verificar que no exista un ASN de la lista en la ruta ASN.

5. Anexo

A continuación se presentan las tablas de rutas para los 7 routers luego de aplicar el algoritmo BGP:

```
# rutas_R1_BGP.txt
127.0.0.1 8882 8881 127.0.0.1 8882 1000
127.0.0.1 8883 8882 8881 127.0.0.1 8882 1000
127.0.0.1 8884 8882 8881 127.0.0.1 8882 1000
127.0.0.1 8885 8884 8882 8881 127.0.0.1 8882 1000
127.0.0.1 8886 8885 8883 8882 8881 127.0.0.1 8882 1000
127.0.0.1 8887 8886 8885 8883 8882 8881 127.0.0.1 8882 1000

# rutas_R2_BGP.txt
127.0.0.1 8881 8882 127.0.0.1 8881 1000
127.0.0.1 8883 8882 127.0.0.1 8883 1000
127.0.0.1 8884 8882 127.0.0.1 8884 1000
127.0.0.1 8885 8884 8882 127.0.0.1 8884 1000
127.0.0.1 8886 8885 8883 8882 127.0.0.1 8883 1000
127.0.0.1 8887 8886 8885 8883 8882 127.0.0.1 8883 1000

# rutas_R3_BGP.txt
127.0.0.1 8882 8883 127.0.0.1 8882 1000
127.0.0.1 8885 8883 127.0.0.1 8885 1000
127.0.0.1 8881 8882 8883 127.0.0.1 8882 1000
127.0.0.1 8884 8882 8883 127.0.0.1 8882 1000
127.0.0.1 8886 8885 8883 127.0.0.1 8885 1000
127.0.0.1 8887 8886 8885 8883 127.0.0.1 8885 1000

# rutas_R4_BGP.txt
127.0.0.1 8882 8884 127.0.0.1 8882 1000
127.0.0.1 8885 8884 127.0.0.1 8885 1000
127.0.0.1 8881 8882 8884 127.0.0.1 8882 1000
127.0.0.1 8883 8882 8884 127.0.0.1 8882 1000
127.0.0.1 8886 8885 8884 127.0.0.1 8885 1000
127.0.0.1 8887 8886 8885 8884 127.0.0.1 8885 1000

# rutas_R5_BGP.txt
127.0.0.1 8883 8885 127.0.0.1 8883 1000
127.0.0.1 8884 8885 127.0.0.1 8884 1000
127.0.0.1 8886 8885 127.0.0.1 8886 1000
```

```
127.0.0.1 8882 8884 8885 127.0.0.1 8884 1000
127.0.0.1 8881 8882 8883 8885 127.0.0.1 8883 1000
127.0.0.1 8887 8886 8885 127.0.0.1 8886 1000

# rutas_R6_BGP.txt
127.0.0.1 8885 8886 127.0.0.1 8885 1000
127.0.0.1 8887 8886 127.0.0.1 8887 1000
127.0.0.1 8883 8885 8886 127.0.0.1 8885 1000
127.0.0.1 8884 8885 8886 127.0.0.1 8885 1000
127.0.0.1 8882 8884 8885 8886 127.0.0.1 8885 1000
127.0.0.1 8881 8882 8883 8885 8886 127.0.0.1 8885 1000

# rutas_R7_BGP.txt
127.0.0.1 8886 8887 127.0.0.1 8886 1000
127.0.0.1 8885 8886 8887 127.0.0.1 8886 1000
127.0.0.1 8883 8885 8886 8887 127.0.0.1 8886 1000
127.0.0.1 8884 8885 8886 8887 127.0.0.1 8886 1000
127.0.0.1 8882 8884 8885 8886 8887 127.0.0.1 8886 1000
127.0.0.1 8881 8882 8883 8885 8886 8887 127.0.0.1 8886 1000
```