Prueba Corta # 5 y 6 - (pc34)

Esteban Ignacio Durán Vargas - 2020388144

IC - 7602 - 2023 I Semestre

Fecha de entrega: 03/05/23

Respuestas

1. **¿Porqué no es posible que cada host en Internet ejecute el algoritmo de Dijkstra para encontrar la ruta mas corta hacia cualquier host en Internet?**

   Esto es por múltiples razones. El internet es una red muy grande, de muchos hosts y además altamente dinámica. En esta constantemente hay rutas y hosts que están siendo conectadas y desconectadas constantemente. Por esto habría que estar realizando este algoritmo constantemente (sin mencionar que hacer Dijkstra por cada **host** es un proceso sumamente costoso si nunca va a llegar a un resultado fijo). Además, al hacer Dijkstra, no se tendrían en cuenta algunas redes privada y que por lo general no está disponible afuera en internet.

2. **Explique el concepto de Hierarchical Routing**

   En el enrutamiento jerárquico se tienen asignaciones de rangos de ips por zona geográfica. Es decir, con esto se sabe si algún servicio está dentro de esta zona geográfica (como un conntinente). Gracias a este, la tabla de routeo sabe por dónde enviar el tráfico para que llegue a un backbone, que sabe a qué router enviar la información según la jerarquía de ips.

3. **Si tiene la siguiente subred 10.0.0.0/8 y ocupa crear 20 subredes de mínimo 160 hosts, ¿Cual máscara utilizaría? Explique en detalle**

   En primer lugar se tienen que se necesitan 160 hosts, por lo que se necesita una potencia de 2 que cubra todos esos hosts (con el gateway y broadcast). Se escoge 8, 256, que desperdicia unos cuantos hosts (sin contar el broadcast y gateway quedan sin usar 94) pero incluye los 160 hosts como mínimo.

   Luego hay que buscar una potencia que incluya las 20 subnets, que sería 5 ya que 2^5 es 32. Por lo que se dedican 5 bits a las subredes.

   Al ser 10.0.0.0/8 una red clase A su máscara de red por defecto es:

   255.248.0.0

   o

   11111111.11111000.00000000.00000000

   Pero esta de /13 daría más hosts de los necesitados, 524286 cuando solo se requiere un mínimo de 160, desperdiciando 524,126. Entonces ignorando el tipo de la ip de origen también tenemos la siguiente máscara de red, donde los últimos bits son los hosts disponibles y se usarían los dos octetos del medio como identificador para las subredes:

255.255.255.0

o

11111111.11111111.11111111.00000000

Se escogería esta (/24) si importara más el número de hosts antes que el número de subredes. Pero esto generaría 65536 subredes pero desperdiciando 65516

Por lo tanto se escogería primordialmente la primera de /13, que tendría muchos hosts extra pero el enunciado solo pide un mínimo, mientras que da un número específco de subredes, y este la cumple.

4. **El mecanismo de inundación es utilizado para distribuir paquetes de findings/updates (hallazgos) con los vecinos, comente acerca de los mecanismos para evitar que los paquetes se queden por siempre en la red.**

   Existen varios mecanismos para evitar que estos paquetes se queden por largo tiempo en la red:

   - Primero es asignarle un tiempo de vida (TTL) o una cantidad de saltos máximos que disminuye con cada salto y hace que el paquete se descarte cuando el contador llegue a 0.
   - Otro método es asingar un número de secuencia a cada paquete. De esta manera el enrutador tiene una lista con la que sabe cuáles paquetes han pasado por este y no difundirlo si llega uno que ya estaba incluído.
   - Otro método es no enviar cada paquete por todas las líneas, sino solo por aquellas que se vayan acercando a la dirección correcta.

5. **Para el IP 10.0.39.25/27, calcule los siguientes parametros:**

- **Máscara de subred (ambas notaciones)**
  - 255.255.255.22
  - 11111111.11111111.11111111.11100000
- **Número de red**

  11111111.11111111.11111111.11100000

  AND

  00001010.00000000.00100111.00011001

  =

  00001010.00000000.00100111.00000000

  Número de red: 10.0.39.0
- **Broadcast Address**

  10.0.39.31 (Máximo de hosts 30 + 1)
- **Lista de hosts de la red**

10.0.39.1 (Gateway) 10.0.39.2 10.0.39.3 10.0.39.4 10.0.39.5 10.0.39.6 10.0.39.7 10.0.39.8 10.0.39.9 10.0.39.10 10.0.39.12 10.0.39.13 10.0.39.13 10.0.39.14 10.0.39.15 10.0.39.16 10.0.39.17 10.0.39.18 10.0.39.19 10.0.39.20 10.0.39.21 10.0.39.22 10.0.39.23 10.0.39.24 10.0.39.25 10.0.39.26 10.0.39.27 10.0.39.28 10.0.39.29 10.0.39.30