

Laboratorio 3

Algoritmos de enrutamiento

Descripción

El laboratorio 3, Algoritmos de enrutamiento, consistió en conocer e implementar algunos de los algoritmos actuales para el enrutamiento de Internet. En este laboratorio implementamos flooding, distance vector routing (Bellman-Ford) y link state routing (Dijkstra). Para poder implementar los algoritmos también se tuvo que aprender para qué y cómo funcionan las tablas de enrutamiento. Sabiendo que es importante que estas tablas sean dinámicas aprendimos acerca de cómo hacer que estas se actualicen para que se puedan ir acomodando con los cambios hechos en la infraestructura.

Las tablas de enrutamiento es una tabla o base de datos que contiene las direcciones de los nodos a los que se pueden enviar paquetes o mensajes en el caso del laboratorio. Cuando un mensaje debe de ser enviado de un nodo a otro se consulta la tabla de enrutamiento con el fin de encontrar la mejor ruta para transferir el mensaje. Este proceso que ayuda a determinar la ruta óptima es conocido como el enrutamiento dinámico o adaptativo.

El algoritmo de flooding, consiste en enviar el mensaje o paquete a cada nodo excepto al nodo que emite el mensaje. Cuando un nodo recibe un mensaje este lo envía a sus vecinos siempre y cuando este no sea el receptor del mensaje.

El algoritmo de link state routing, Dijkstra, nos permite encontrar el camino más corto entre nodos de un grafo. Este algoritmo utiliza un nodo inicial y un nodo terminal. En el nodo inicial determina cual es el camino de menor costo desde un vértice “x” a un vértice “y”. Este es un algoritmo greedy ya que en cada etapa toma la mejor solución sin considerar consecuencias futuras. Es importante notar que este algoritmo no funciona con grafos que tienen pesos negativos.

El algoritmo de distance vector routing, Bellman-Ford, funciona al enviar a sus nodos vecinos su listado de vectores de distancia y a la vez recibir de sus nodos vecinos sus vectores de distancia. Con los vectores de distancia de los vecinos este va optimizando sus propios distance vectors. Utilizando la fórmula de la Figura 1., se decide a qué nodo reenviar el mensaje. A diferencia de Dijkstra, este algoritmo sí funciona con grafos de pesos negativos.

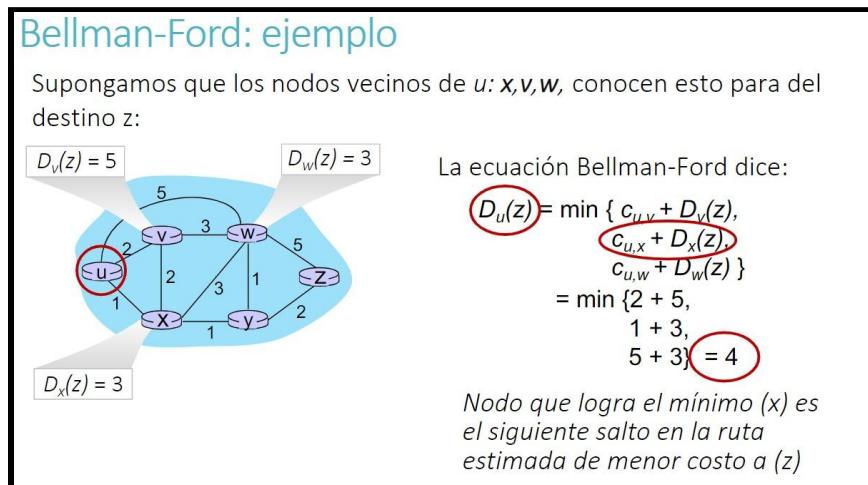


Figura 1. Bellman-Ford (tomado de presentación en clase)

Resultados

Flooding:

```
C:\ Command Prompt - python flooding_client.py
C:\Users\Odi\Desktop\U\2021 segundo semestre\Redes\lab3\Lab3_Redes>python flooding_client.py
Bienvenido por favor ingrese que desea hacer:
    1.login
    2.signup
    3.salir
Ingrese una opción para poder continuar: 1
Usuario: odirtry@alumchat.xyz
Contraseña: odi
1. Ingresar un nodo vecino
2. Continuar
1
Ingrese el nombre de usuario del vecino: pepity
1. Ingresar un nodo vecino
2. Continuar
```

Figura 2. Ingresando nodos vecinos al programa de flooding.

Universidad del Valle de Guatemala

Oliver Graf 17190

Kristen Brandt 171482

Redes

The screenshot shows two separate Command Prompt windows. Both windows have identical code at the top:

```
TypeError: exception() takes no arguments (1 given)
He entrado al chat exitosamente :)
```

The first window (Sender) has the following output:

```
pepitry
1. Chat
2.Salir
3.Send Flood
4.Listen
Que opción deseas: 3
Sending Flood message
Message: Hola Mundo!
Receiver: kristry
Enviado a kristry@alumchat.xyz
Enviado a kristry@alumchat.xyz
```

The second window (Recipient) has the following output:

```
kristry
1. Chat
2.Salir
3.Send Flood
4.Listen
Que opción deseas: 4
I am listening!
Mensaje Flood recibido exitosamente!
```

A pink arrow points from the "Sender" window's "Message" line to the "Receiver" window's "Mensaje Flood recibido exitosamente!" line.

Figura 3. A punto de enviar un flood desde el usuario pepity hasta el usuario kristry.

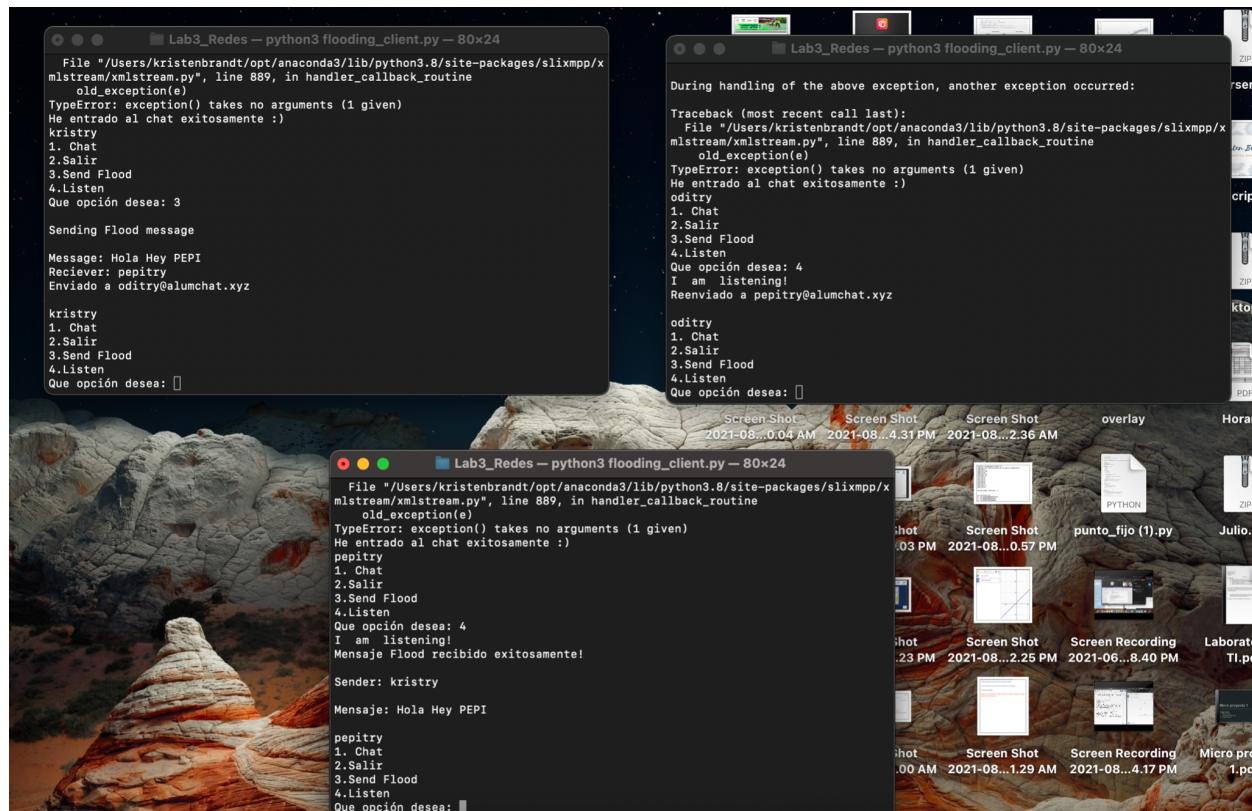


Figura 4. Mensaje enviado de Kristry a Pepity pasando por Oditory

Universidad del Valle de Guatemala

Oliver Graf 17190

Kristen Brandt 171482

Redes

Dijkstra:

The image shows two terminal windows side-by-side. Both windows have a title bar 'Lab3_Redes — python3 dijkstra_client.py — 80x24'. The left window shows the initial state of the algorithm with options 1 through 4 listed. The right window shows the algorithm in progress, where 'kristry' has sent a message to 'pepitry', which then forwarded it to 'oditry'. The messages exchanged are: 'Que loco esto we', 'Receiving from pepity', 'Enviado a pepity@alumchat.xyz', 'Mensaje Dijkstra reenviado a oditry@alumchat.xyz', and 'Mensaje: Que loco esto we'.

```
old_exception(e)
TypeError: exception() takes no arguments (1 given)
He entrado al chat exitosamente :
kristry
1. Chat
2.Salir
3.Send Dijkstra
4.Listen
Que opción deseas: 3

Sending Dijkstra message

Message: Que loco esto we
Receiver: oditry
v
kristry -> pepity -> oditry
Enviado a pepity@alumchat.xyz

kristry
1. Chat
2.Salir
3.Send Dijkstra
4.Listen
Que opción deseas: 4

pepitry
1. Chat
2.Salir
3.Send Dijkstra
4.Listen
Que opción deseas: 4

I am listening!
['oditry', 'kristry']

kristry -> pepity -> oditry
Mensaje Dijkstra reenviado a oditry@alumchat.xyz

Mensaje: Que loco esto we

oditry
1. Chat
2.Salir
3.Send Dijkstra
4.Listen
Que opción deseas: 
```

```
old_exception(e)
TypeError: exception() takes no arguments (1 given)
He entrado al chat exitosamente :
oditry
1. Chat
2.Salir
3.Send Dijkstra
4.Listen
Que opción deseas: 4
I am listening!
['oditry', 'kristry']

kristry -> pepity -> oditry
Mensaje Dijkstra recibido exitosamente!

Mensaje: Que loco esto we

oditry
1. Chat
2.Salir
3.Send Dijkstra
4.Listen
Que opción deseas: 
```

Figura 5. Mensaje enviado de kristry a oditry, pasando por el camino más corto que es de kristry a pepity y luego olitry.

Bellman-Ford:

The image shows two terminal windows side-by-side. Both windows have a title bar 'Lab3_Redes — python3 dinamic_client.py — 80x24'. The left window shows the initial state of the algorithm with options 1 through 6 listed. The right window shows the algorithm in progress, where 'oditry' has sent DVs to 'kristry', which then forwarded them to 'pepitry'. The DVs exchanged are: 'oditry;0 kristry;10 pepity;1' and 'pepitry;0 kristry;1 oditry;1'. The message exchanged is: 'Enviado a pepity@alumchat.xyz'.

```
oditry
1. Chat
2.Salir
3.Send DVs
4.Send Bellman-Ford message
5.Listen
6.Print DVs
Que opción deseas: 3

Sending updated DVs

oditry;0 kristry;10 pepity;1
Enviado a kristry@alumchat.xyz

Enviado a pepity@alumchat.xyz

oditry
1. Chat
2.Salir
3.Send DVs
4.Send Bellman-Ford message
5.Listen
6.Print DVs
Que opción deseas: 
```

```
Sending updated DVs

pepitry;0 kristry;1 oditry;1
Enviado a kristry@alumchat.xyz

Enviado a oditry@alumchat.xyz

'DVS'

DVs update received from oditry
'DVS'

DVs update received from oditry

pepitry
.. Chat
.. Salir
.. Send DVs
.. Send Bellman-Ford message
.. Listen
.. Print DVs
Que opción deseas: 
```

Figura 6. Enviando DVs

Universidad del Valle de Guatemala

Oliver Graf 17190

Kristen Brandt 171482

Redes

The image shows two terminal windows side-by-side, both titled "Lab3_Redes — python3 dinamic_client.py — 80x24".

Top Terminal Window:

```
6.Print DVs
Que opción desea: 6
{'oditry': 0, 'kristry': 10, 'pepitry': 1}
['DVS']

DVs update received from pepitry

Hubo cambio en los DVs, enviando DVs a vecinos

Sending updated DVs

oditry;0 kristry;2 pepitry;1
Enviado a kristry@alumchat.xyz

Enviado a pepitry@alumchat.xyz

oditry
1. Chat
2.Salir
3.Send DVs
4.Send Bellman-Ford message
5.Listen
6.Print DVs
Que opción desea: 
```

Bottom Terminal Window:

```
Vs update received from oditry
'DVS']

Vs update received from oditry

epitry
. Chat
.Salir
.Send DVs
.Send Bellman-Ford message
.Listen
.Print DVs
ue opción desea: 6
'pepitry': 0, 'kristry': 1, 'oditry': 1}
epitry
. Chat
.Salir
.Send DVs
.Send Bellman-Ford message
.Listen
.Print DVs
ue opción desea: 
```

Figura 7. Tablas dinamicas

Universidad del Valle de Guatemala

Oliver Graf 17190

Kristen Brandt 171482

Redes

The screenshot shows two terminal windows side-by-side. Both windows have the title "Lab3_Redes - python3 dinamic_client.py - 80x24".
The left window (node kristry) displays the following interaction:
- It sends updated DVs to node oditry.
- It receives a message from oditry: "Enviado a kristry@alumchat.xyz" and "Enviado a pepity@alumchat.xyz".
- It lists options 1 through 6.
- It receives a message from oditry: "Que opción desea: 6" and a JSON object: {"oditry": 0, "kristry": 2, "pepitry": 1}.
- It sends DVs to oditry.
- It receives a message from oditry: "oditry 1. Chat 2.Salir 3.Send DVs 4.Send Bellman-Ford message 5.Listen 6.Print DVs".
- It lists options 1 through 6 again.
- It receives a message from oditry: "Que opción desea: []".
The right window (node oditry) displays the following interaction:
- It receives a message from kristry: "Enviado a kristry@alumchat.xyz" and "Enviado a pepity@alumchat.xyz".
- It lists options 1 through 6.
- It receives a message from kristry: "Que opción desea: 6" and a JSON object: {"pepitry": 0, "kristry": 1, "oditry": 1}.
- It sends DVs to kristry.
- It receives a message from kristry: "pepitry .. Chat .. Salir .. Send DVs .. Send Bellman-Ford message .. Listen .. Print DVs".
- It lists options 1 through 6 again.
- It receives a message from kristry: "Que opción desea: []".

Figura 8. Tablas dinámicas

The screenshot shows two terminal windows side-by-side. Both windows have the title "Lab3_Redes - python3 dinamic_client.py - 80x24".
The left window (node kristry) displays the following interaction:
- It sends DVs to oditry.
- It receives a message from oditry: "Que opción desea: 5" and a JSON object: {"oditry": "I am listening!", "kristry": "1'"}.
- It prints a success message: "Mensaje Bellman-Ford recibido exitosamente!".
- It lists options 1 through 6.
- It receives a message from oditry: "Sender: kristry" and "Jumps: 1".
- It receives a message from oditry: "Mensaje: hola WODO".
The right window (node oditry) displays the following interaction:
- It receives a message from kristry: "Message: hola WODO" and "Receiver: oditry".
- It receives a message from kristry: "Enviado a oditry@alumchat.xyz".
- It lists options 1 through 6.
- It receives a message from kristry: "Que opción desea: []".

Figura 9 . Mensaje enviado

Universidad del Valle de Guatemala

Oliver Graf 17190

Kristen Brandt 171482

Redes

Discusión

Para poder comprender más a fondo los algoritmos de enrutamiento realizamos tres clientes que se conectan a un servidor público. Hay un cliente para cada algoritmo, Flooding, Dijkstra, Bellman-Ford.

Al utilizar el algoritmo de Flooding se le pide al usuario que ingrese sus nodos vecinos. A la hora de mandar un mensaje se pide el mensaje y el nodo receptor, este nodo puede no estar dentro de los vecinos del nodo emisor. El mensaje se envía del nodo emisor a todos sus vecinos y los vecinos lo envían a todos sus vecinos. Este proceso se repite hasta que el mensaje llega al destinatario. Nuestra implementación de este algoritmo permite que el mensaje se enviará de manera exitosa a todos los vecinos y que el mensaje se imprimiera únicamente en el nodo receptor. Nuestra implementación también es capaz de detectar si un mensaje ya había pasado por el nodo previamente. Logramos contar cuántos saltos hizo el mensaje antes de llegar a su destino.

Al utilizar el algoritmo de Dijkstra se lee un archivo .txt que contiene la topología del grafo en forma de las aristas y sus pesos. A la hora de enviar un mensaje se utiliza el grafo para decidir el camino más corto y enviar el mensaje al nodo receptor. En nuestra implementación de este algoritmo se permite que el nodo emisor envíe por el camino más corto un mensaje al nodo receptor. Con nuestra implementación es evidente el camino que se recorrió y la cantidad de saltos que éste contiene, dado que se imprime el camino más corto en cada salto. Además, se puede saber en qué nodo del camino se encuentra.

En nuestra implementación del algoritmo de Bellman-Ford se permite el uso de tablas de enrutamiento dinámicas para poder enviar el mensaje del nodo emisor al nodo receptor. Pudimos observar que las tablas de enrutamiento se actualizan correctamente al mandar los nodos sus vectores de distancia a sus vecinos.

Al hacer el laboratorio logramos entender más a fondo las tablas de enrutamiento. Estas nos dicen cuál es la distancia entre un nodo y el resto. Esta información se va actualizando conforme los nodos vecinos comparten sus tablas de enrutamiento, haciendo que estas tablas sean dinámicas.

Comentario Grupal

Al hacer este laboratorio nos pareció muy interesante aprender de los algoritmos de enrutamiento. Aunque en otras clases ya los habíamos utilizado (solo Dijkstra) fue interesante aprender otra aplicación que estos pueden tener.

Universidad del Valle de Guatemala

Oliver Graf 17190

Kristen Brandt 171482

Redes

Ambos integrantes del grupo nos sentimos felices con el trabajo realizado ya que en el proyecto nos enfocamos mucho en concurrencia y esto nos impidió terminar las funcionalidades. Para este laboratorio le dedicamos bastante tiempo y logramos conseguir el resultado esperado.

Conclusiones

- Las tablas de enrutamiento dinámicas se van modificando con los vectores de distancia de los vecinos.
- Los tres algoritmos cumplen su trabajo de enviar un mensaje de un nodo inicial a un nodo destino, pero los tres algoritmos funcionan mejor en diferentes situaciones.
- El algoritmo de Dijkstra es mejor para topologías que no cambian mucho.
- El algoritmo de Bellman-Ford es mejor para topologías que cambian y que pueden tener aristas con pesos negativos.