7ma Clase 26/4/2021

75.43 Introducción a los Sistemas Distribuidos
73.33 Redes y Teleprocesamientos I
TA048 Redes

Tema: Capa de Red (III)

Capítulo 5: desde el inicio hasta el apartado *5.3 Intra-AS Routing in the Internet: OSPF inclusive* de *Computer Networking : A Top-Down Approach. James Kurose and Keith Ross. Publisher: Pearson Edition: 7th, 2016.*

Dr. Ing. J. Ignacio Alvarez-Hamelin

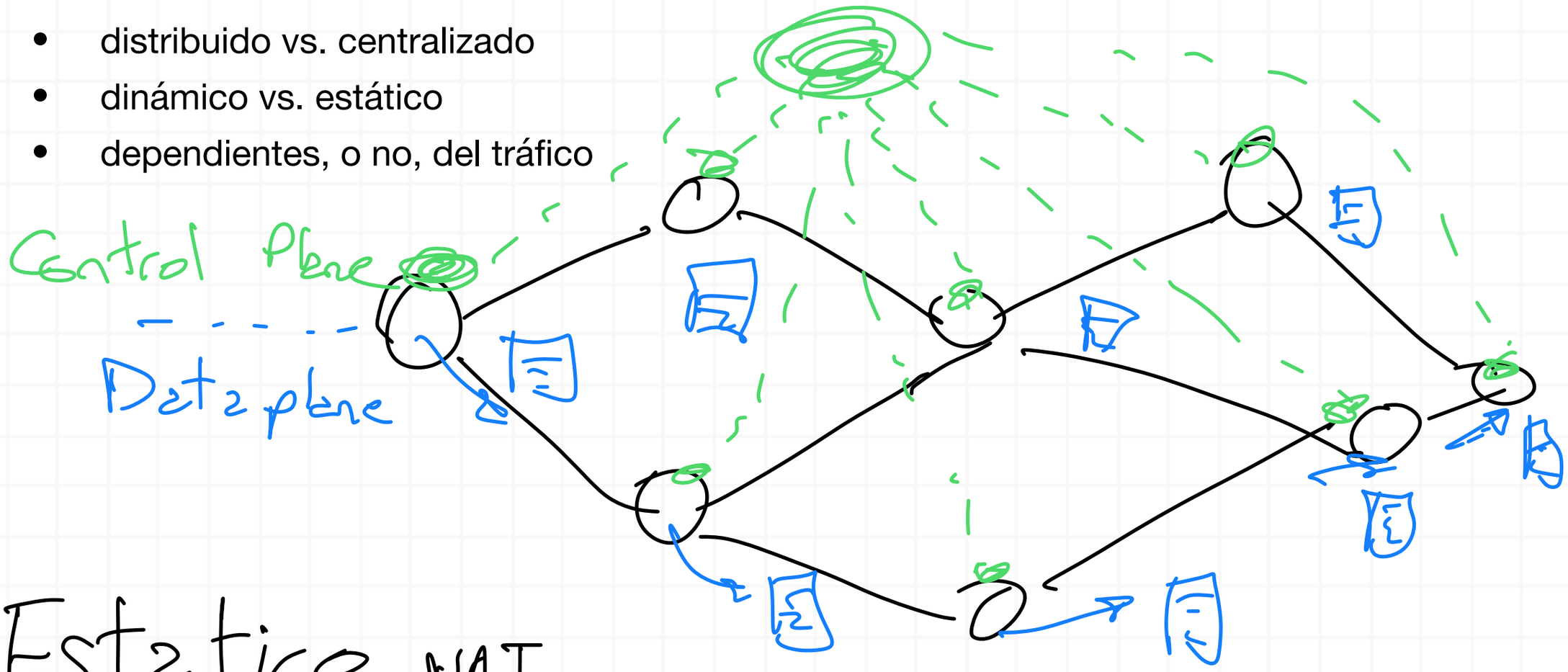
Clase de hoy

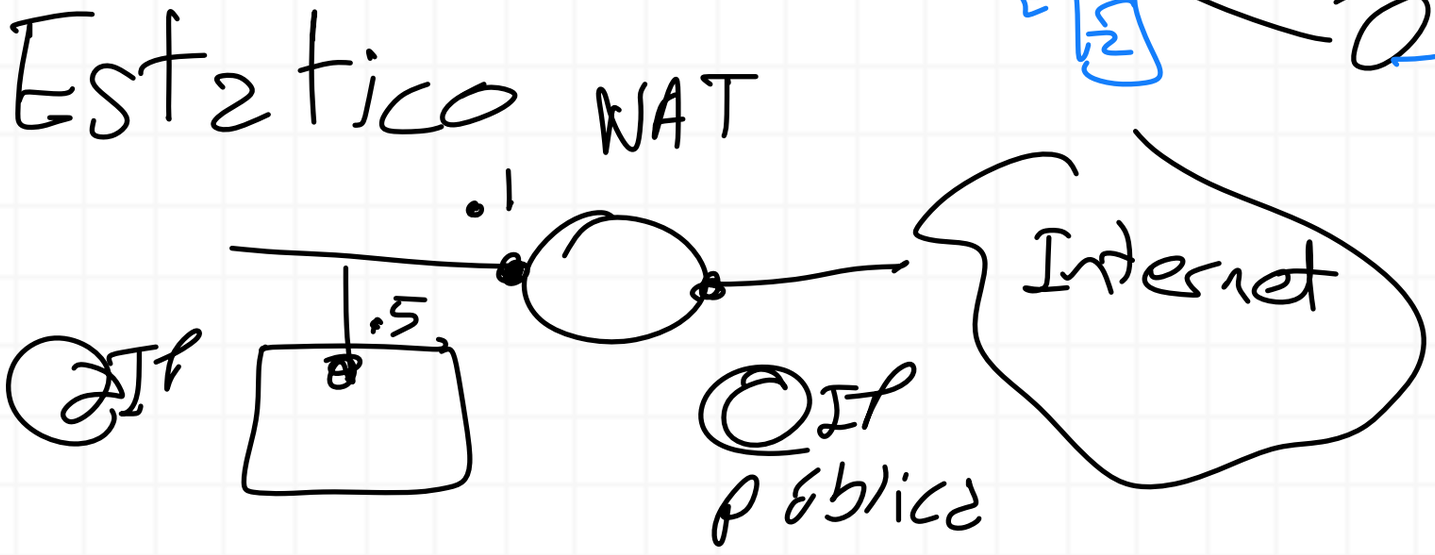
- Plano de Control (*Control Plane*)
- Organización de Internet
- Protocolos de Ruteo

Plano de Control (*Control Plane*)

Introducción:

- distribuido vs. centralizado
- dinámico vs. estático
- dependientes, o no, del tráfico



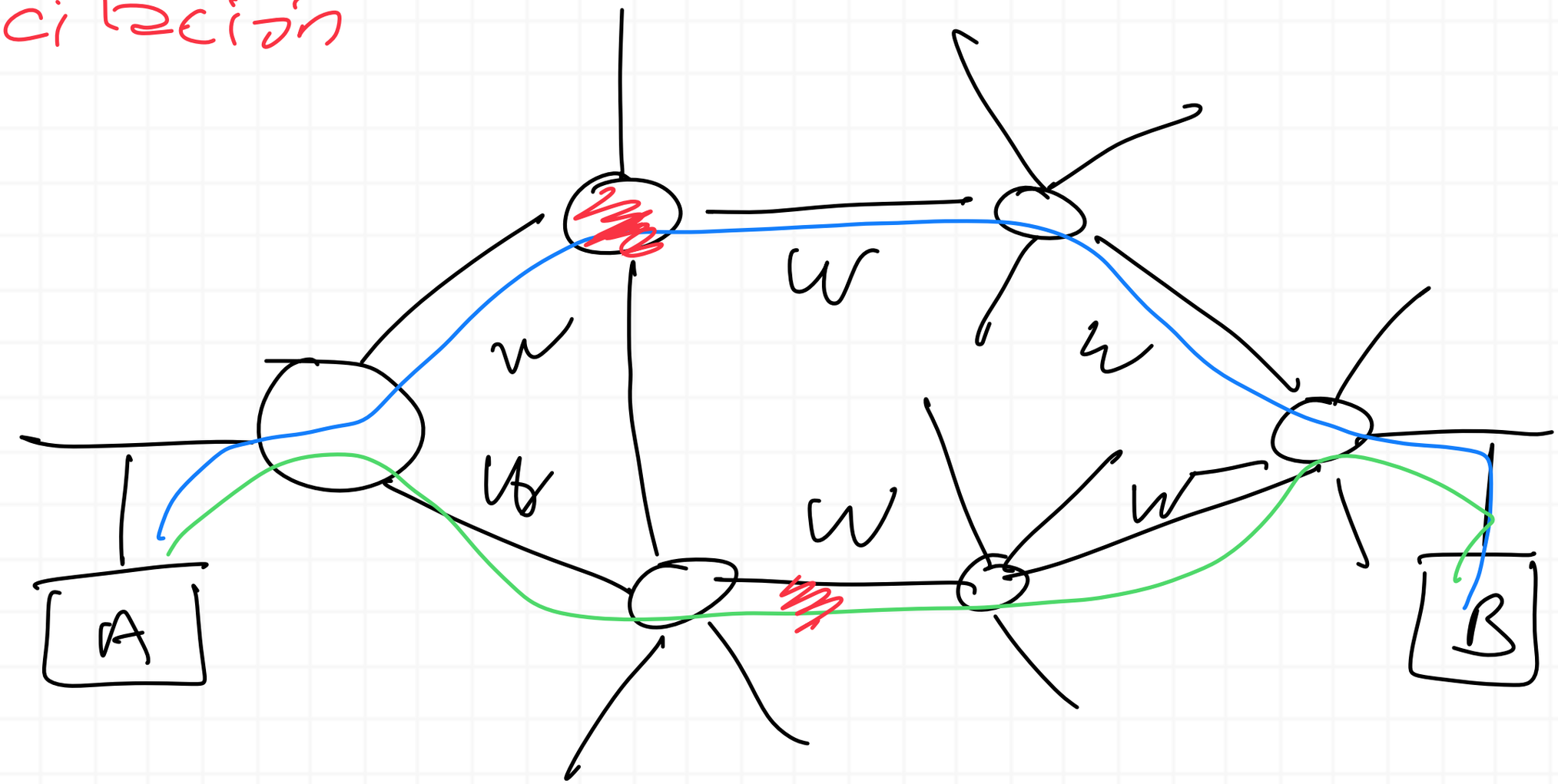


Estatica
Default Gateway

192.168.1.3

Oscilación

QoS



Organización de Internet

- ASes
- ISP
- Jerarquía y relaciones

Tier 1

Transit Providers

Tier 2

ISP

Internet Service Providers

Tier 3

ISP

IXP

Stubs

conexiones P2P

Client

Server

Tier 1

Tier 2

Tier 3

Uphill

down hill

IXP

Stubs

Client

Server

conexiones P2P

Tier 1

Tier 2

Tier 3

IXP

Stubs

conexiones P2P

Client

Server

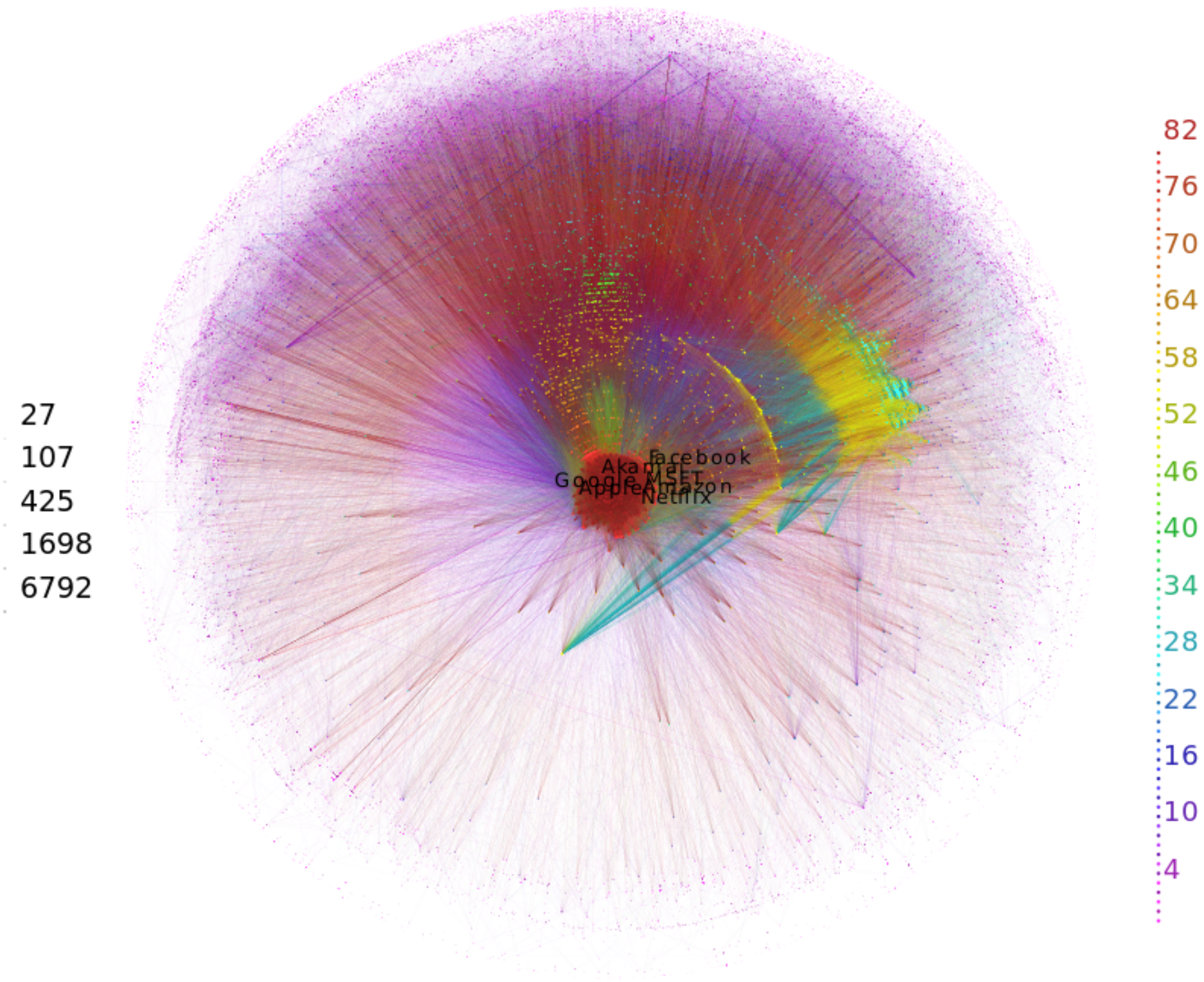


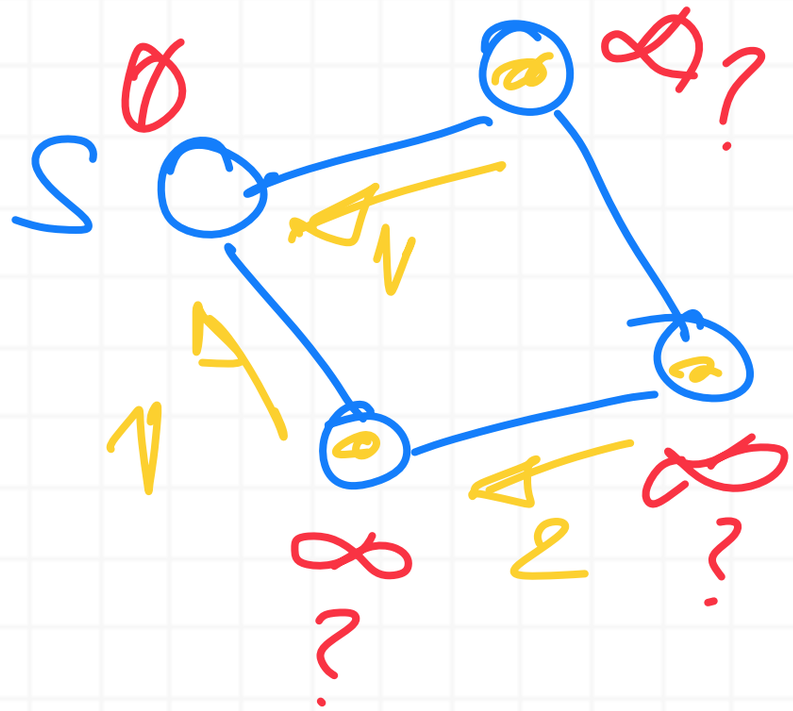
Imagen obtenida de https://cnet.fi.uba.ar/TMA2018/
Para más información: https://cnet.fi.uba.ar/

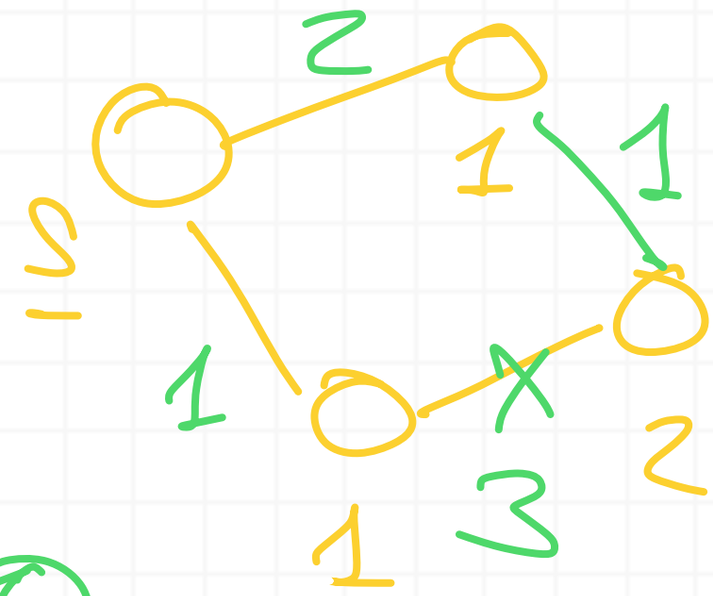
Protocolos de ruteo

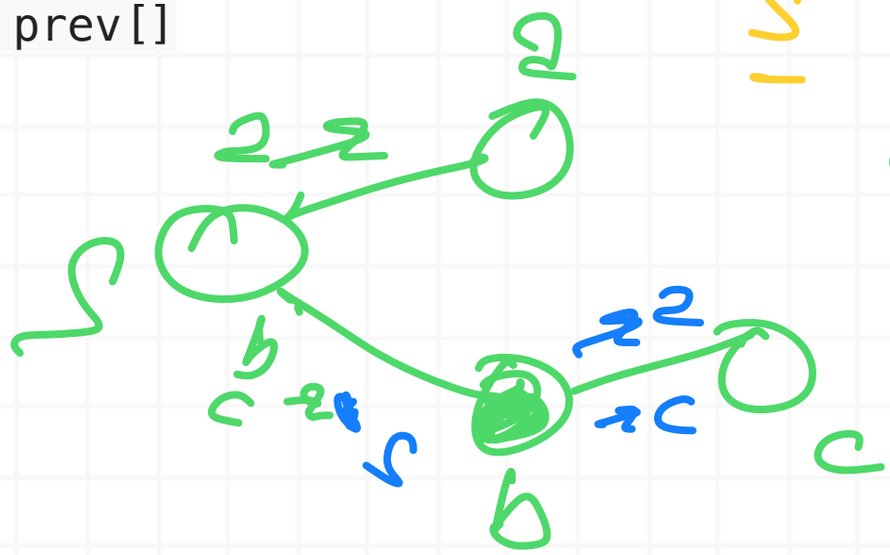
- Distintos puntos de vista: link-state, distance-vector
- OSPF y RIP

Algoritmo de Dijkstra[1]

```
 1  function Dijkstra(Graph, source):
 2
 3      for each vertex v in Graph.Vertices:
 4          dist[v] ← INFINITY
 5          prev[v] ← UNDEFINED
 6          add v to Q
 7      dist[source] ← 0
 8
 9      while Q is not empty:
10          u ← vertex in Q with min dist[u]
11          remove u from Q
12
13          for each neighbor v of u still in Q:
14              alt ← dist[u] + Graph.Edges(u, v)
15              if alt < dist[v]:
16                  dist[v] ← alt
17                  prev[v] ← u
18
19      return dist[], prev[]
```

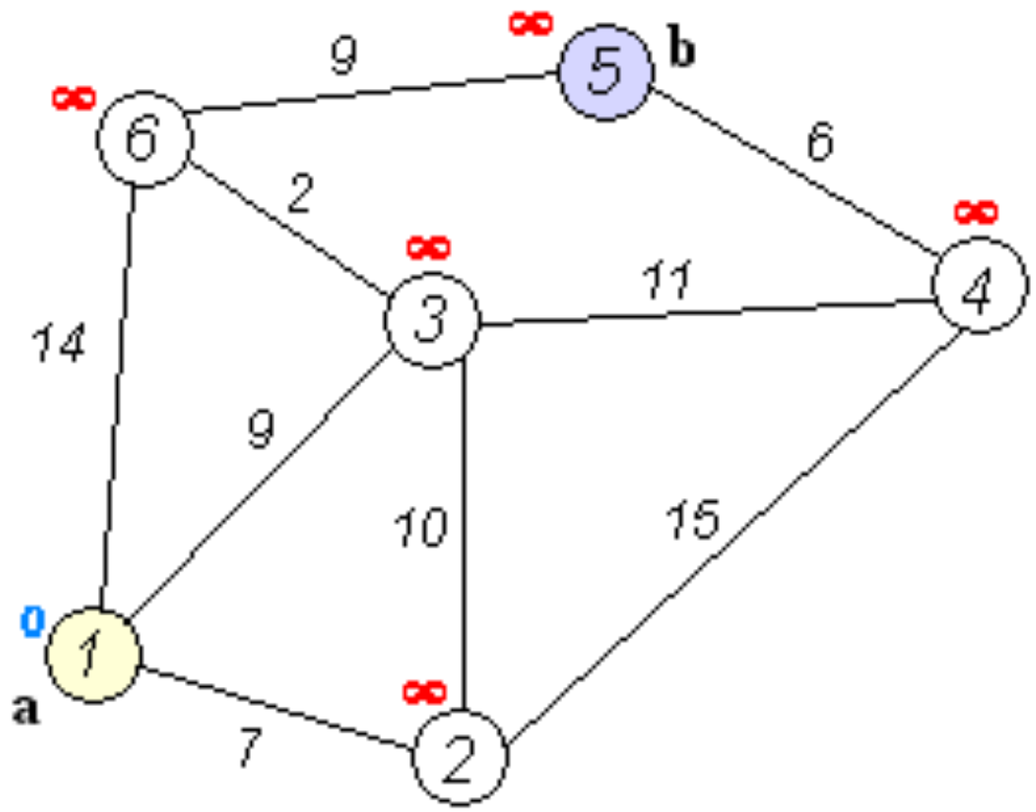






1_ Versión de https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Algoritmo de Dijkstra [1]



Complejidad: $O(|E| + |V| \log |V|)$

E es el número de aristas, ejes o conexiones

V es el número de vértices, nodos o participantes

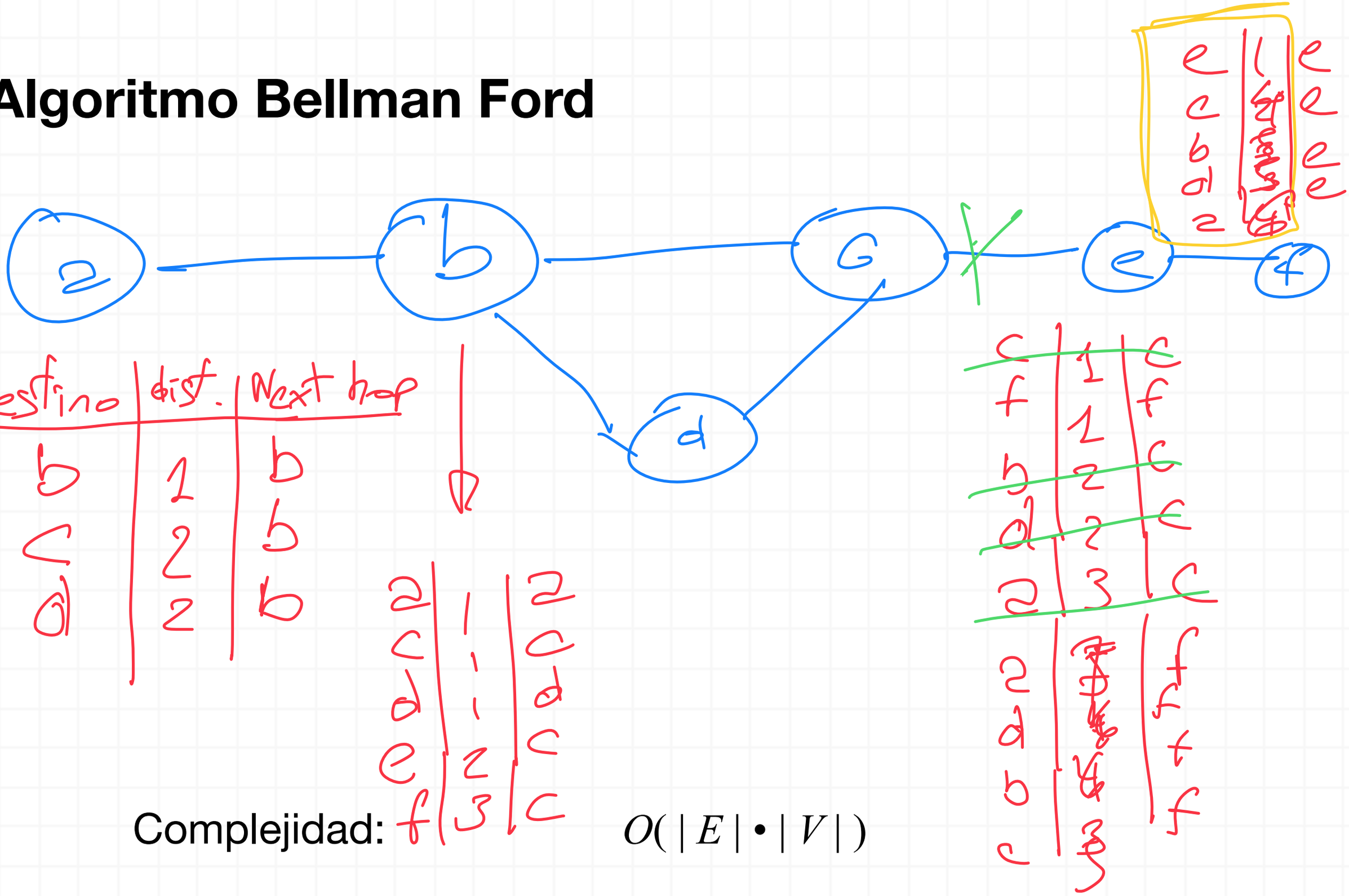
1_ Versión de https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

Algoritmo Bellman Ford[1]

```
 1 function bellmanFord(G, S)
 2   for each vertex V in G
 3     distance[V] <- infinite
 4         previous[V] <- NULL
 5   distance[S] <- 0
 6
 7   for each vertex V in G
 9     for each edge (U,V) in G
10         tempDistance <- distance[U] + edge_weight(U, V)
11         if tempDistance < distance[V]
12             distance[V] <- tempDistance
13             previous[V] <- U
14
15   for each edge (U,V) in G
16     If distance[U] + edge_weight(U, V) < distance[V}
17         Error: Negative Cycle Exists
18
19   return distance[], previous[]
```

1_ Versión de https://www.programiz.com/dsa/bellman-ford-algorithm

Algoritmo Bellman Ford



Complejidad: $O(|E| \bullet |V|)$

E es el número de aristas, ejes o conexiones

V es el número de vértices, nodos o participantes

OSPF : Dijkstra

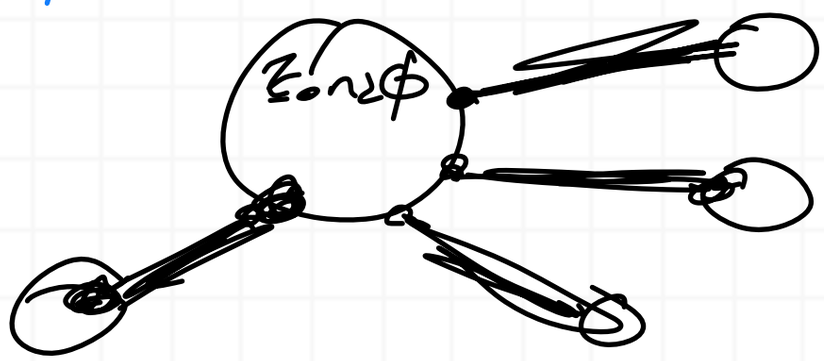
Pesos —— Capacidades

Priocidades

Autenticación

Zonas

Multicast



RIP (distance-Vector) autenticación

OSPF: RFC 2328

RIP: RFC 2453

Lectura para la próxima clase:
Completar la lectura del **Capítulo 5**