

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: 8th, 2021.

Dr. Ing. J. Ignacio Alvarez-Hamelin

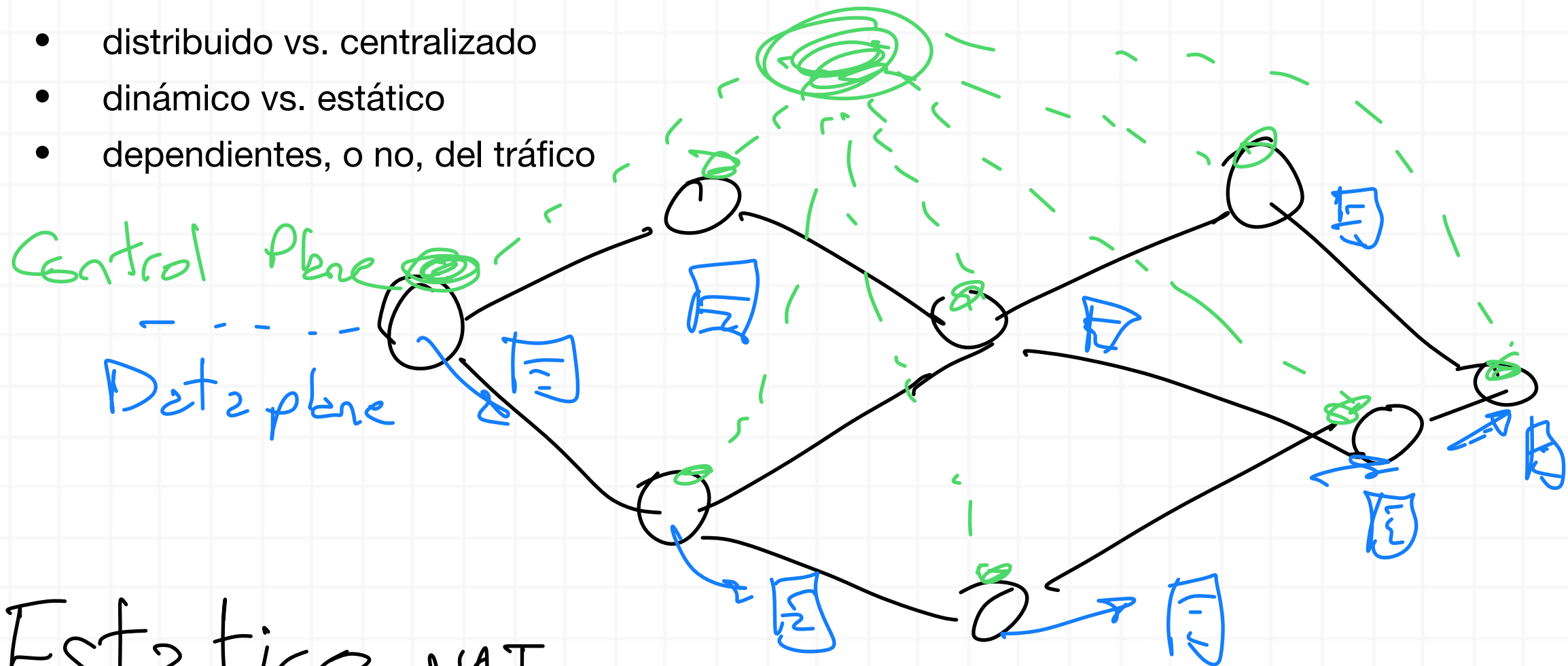
Clase de hoy

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

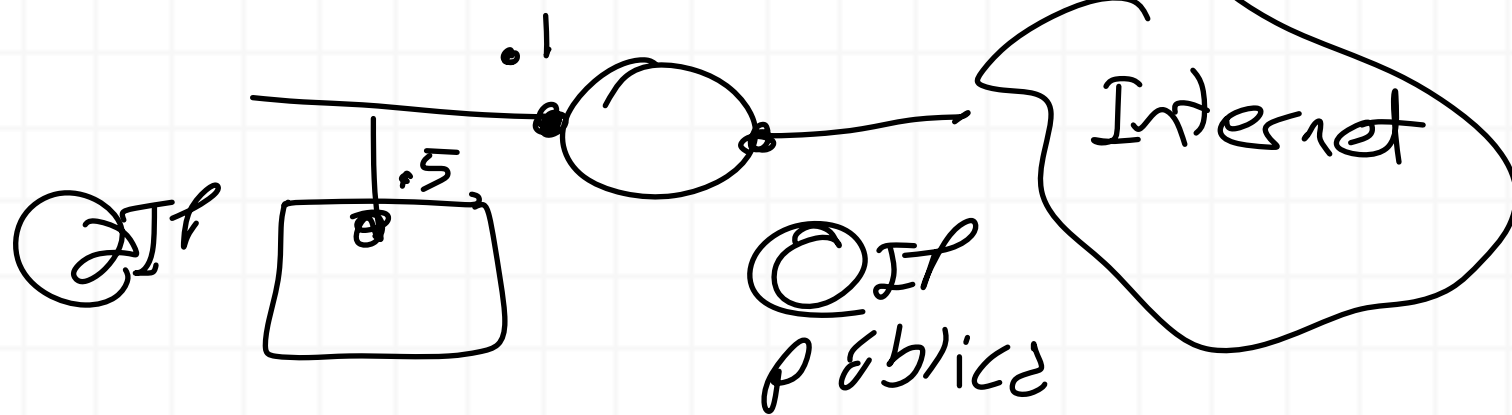
Plano de Control (Control Plane)

Introducción:

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



Estático NAT

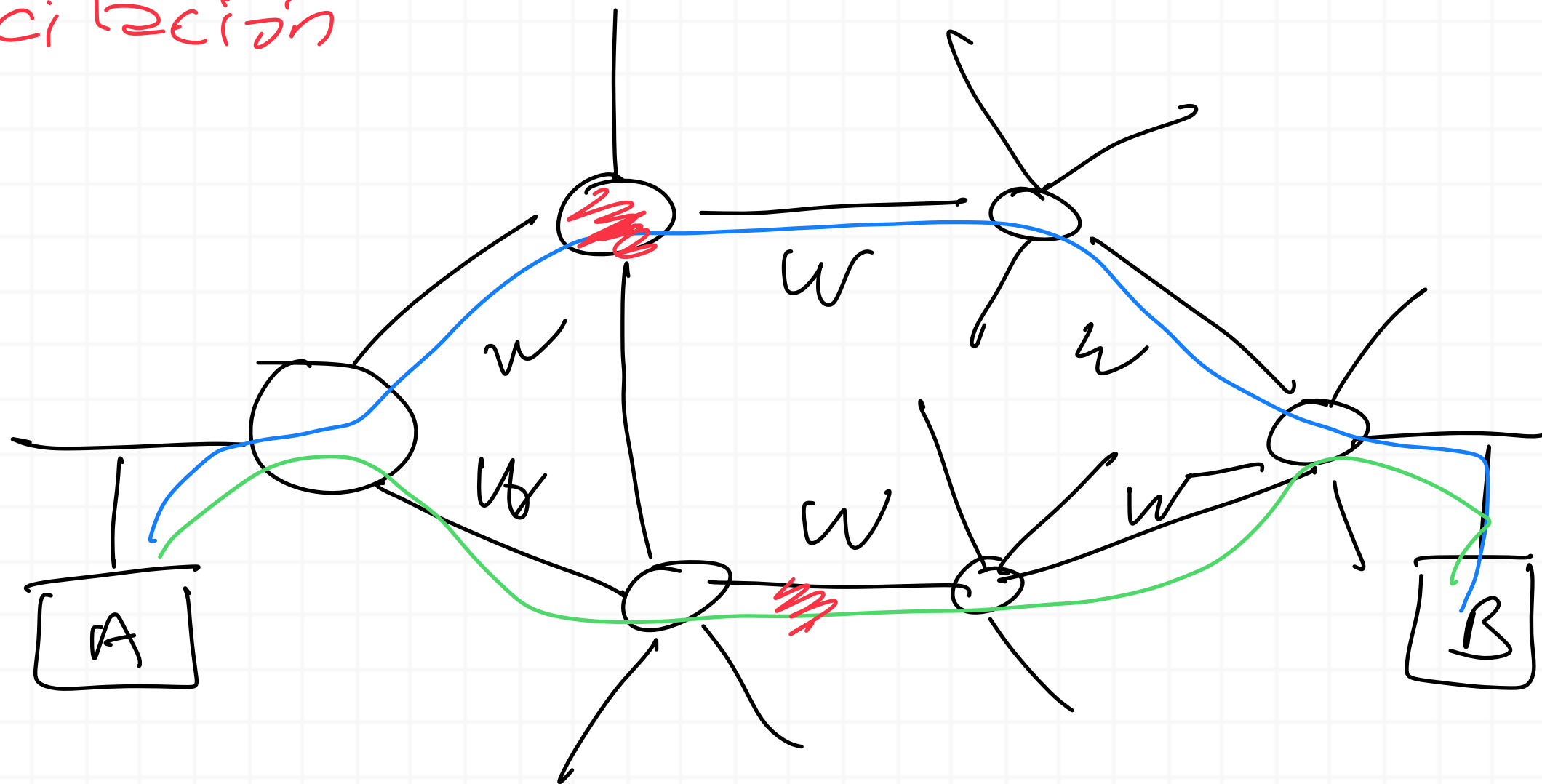


192.168.1.3

Estático
Default Gateway

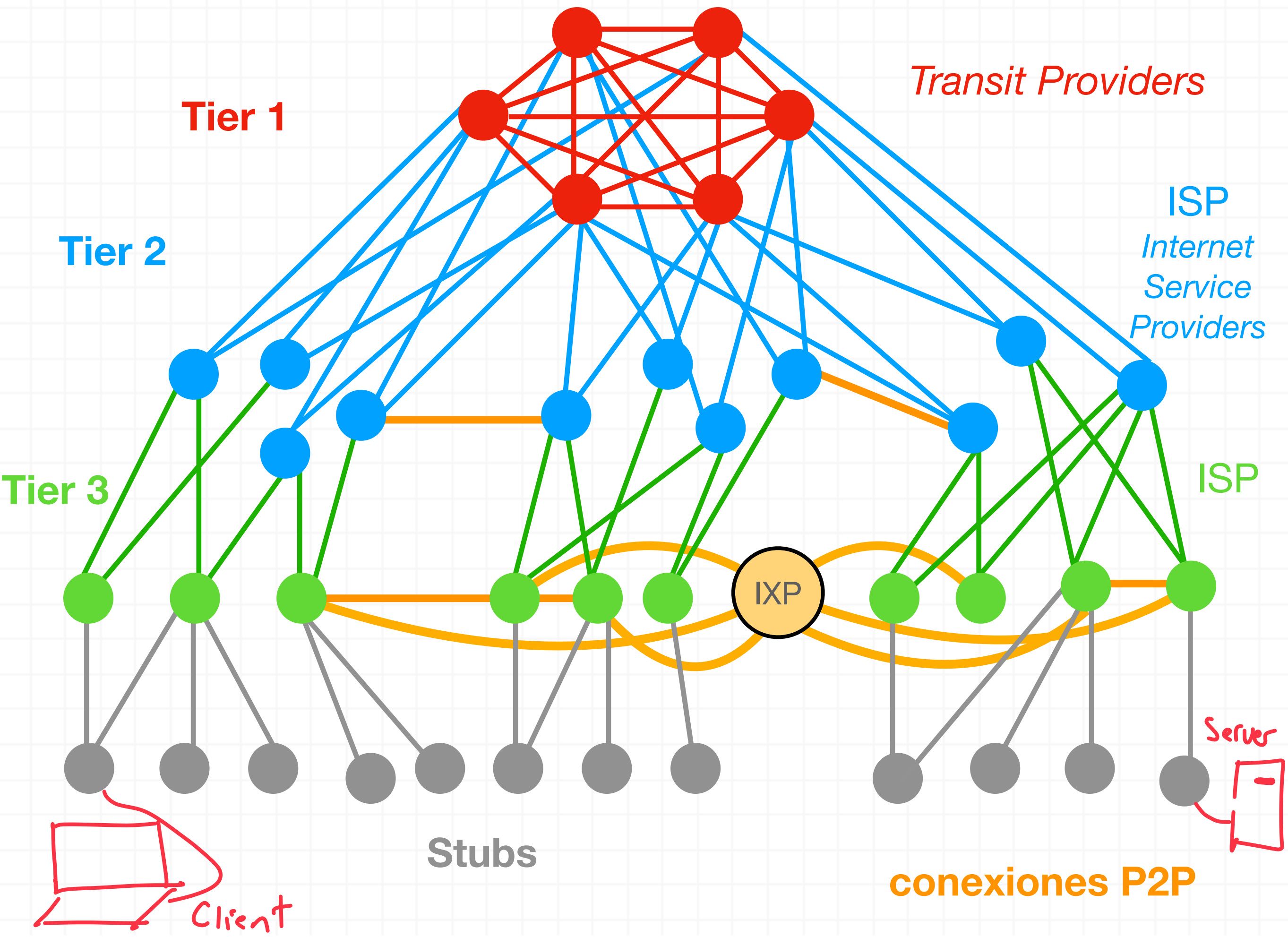
Oscilación

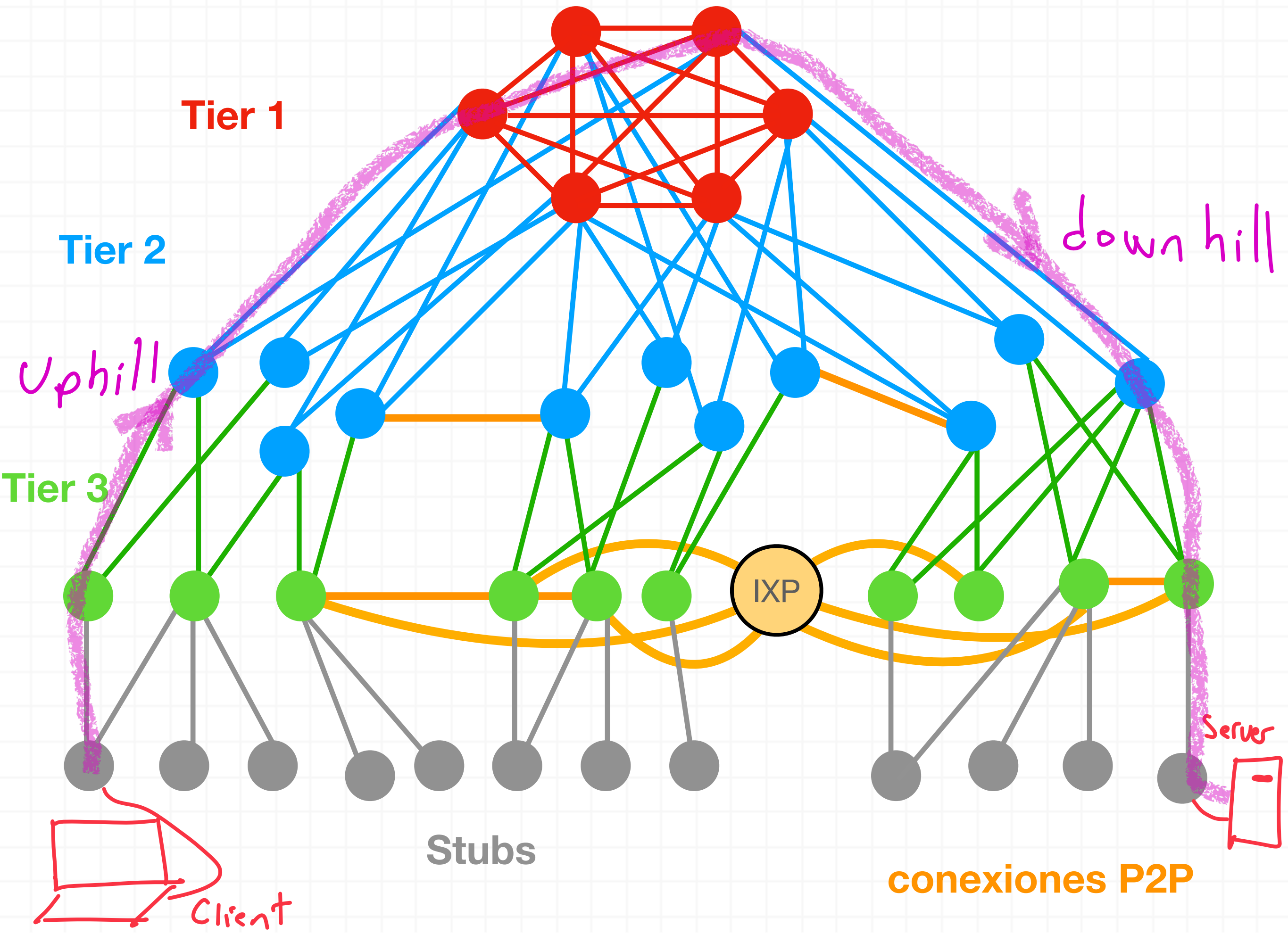
QoS

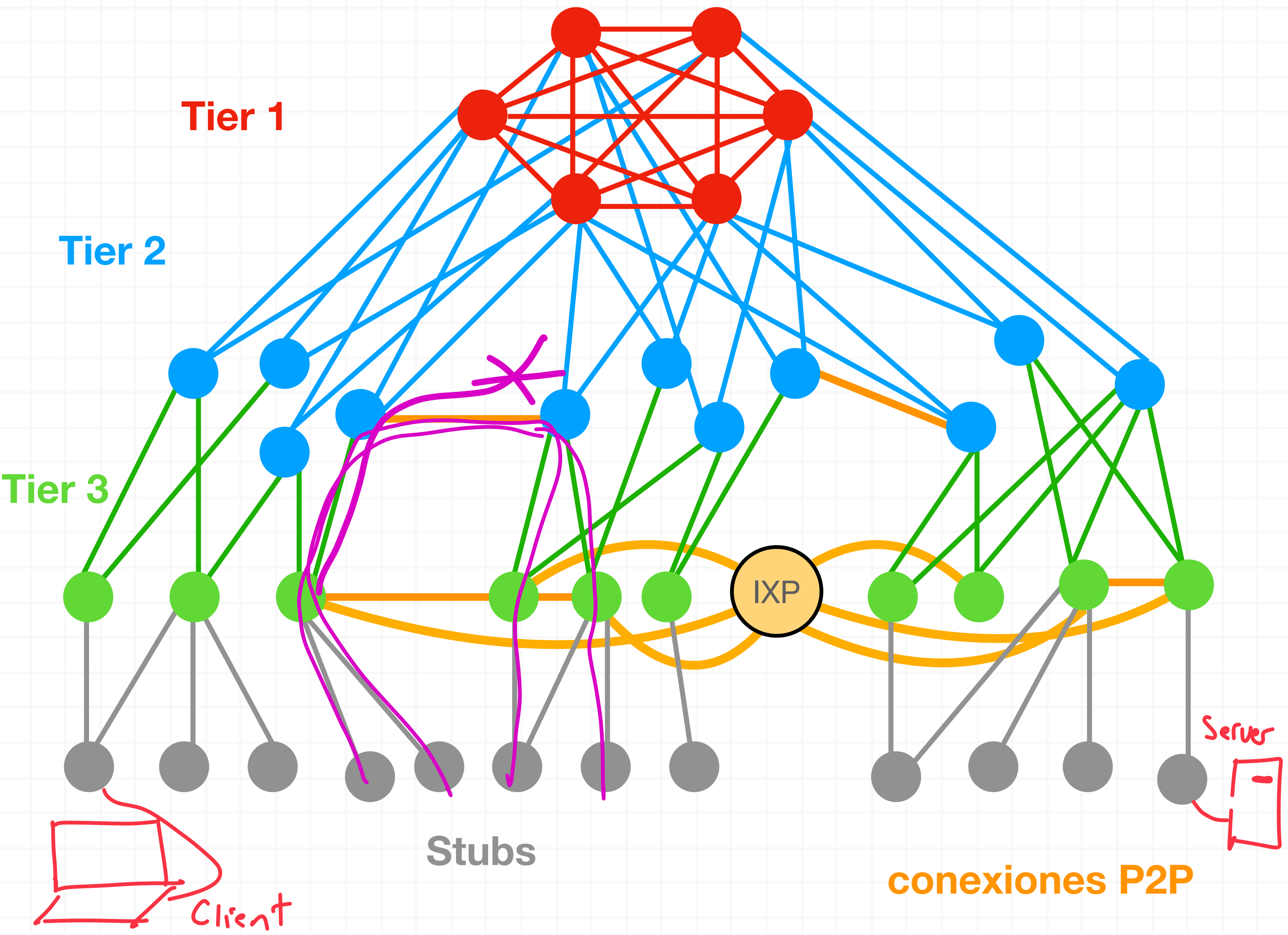


Organización de Internet

- ASes
- ISP
- Jerarquía y relaciones







27
107
425
1698
6792

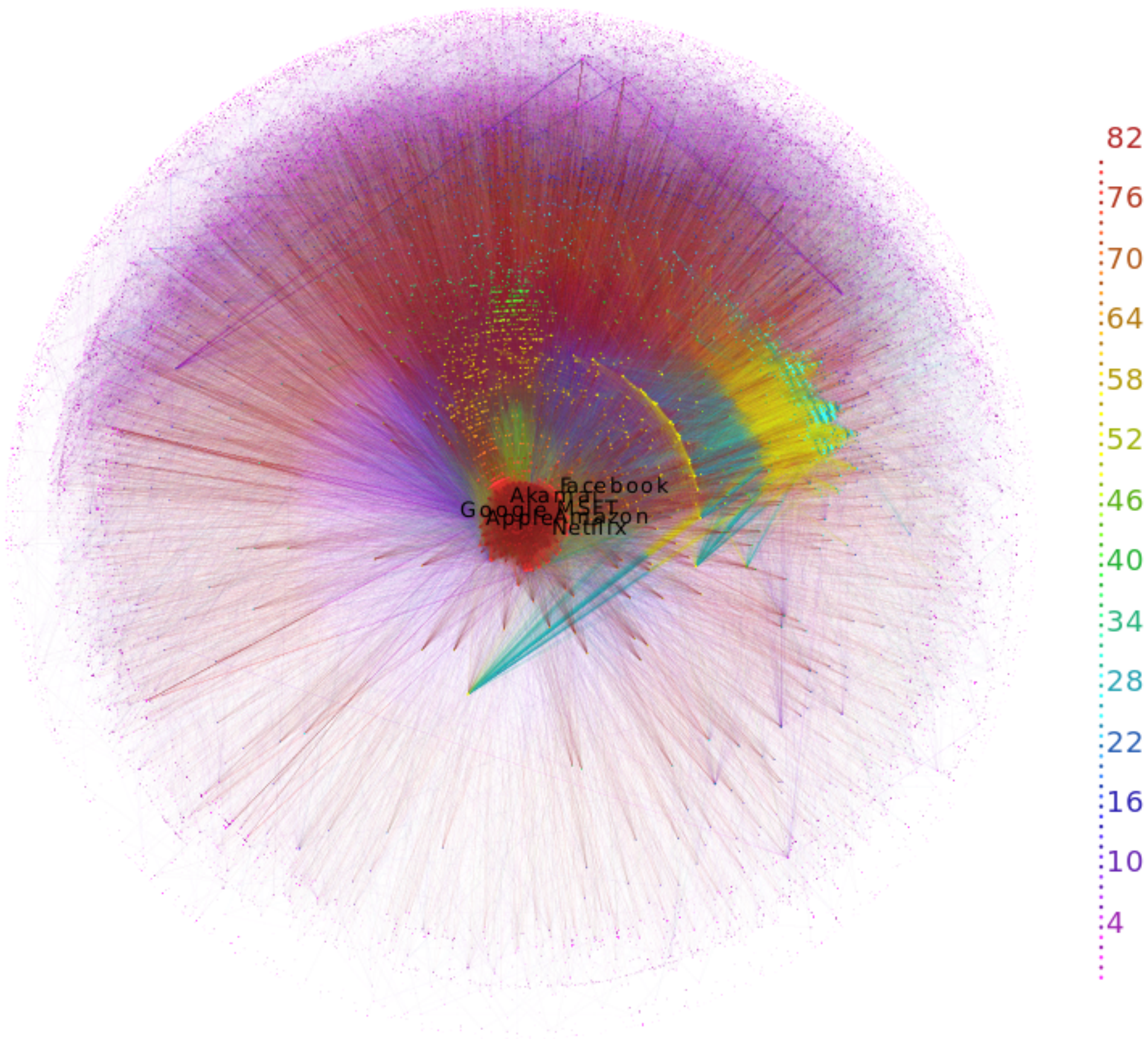


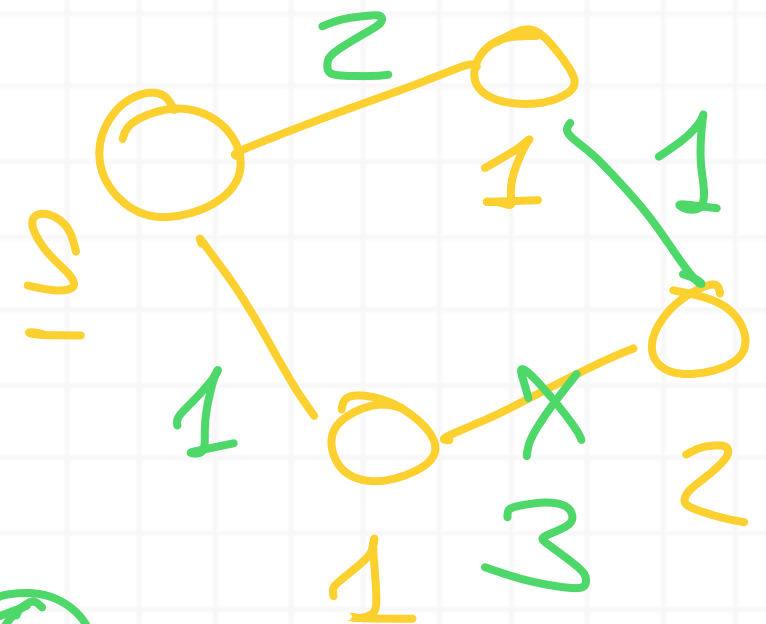
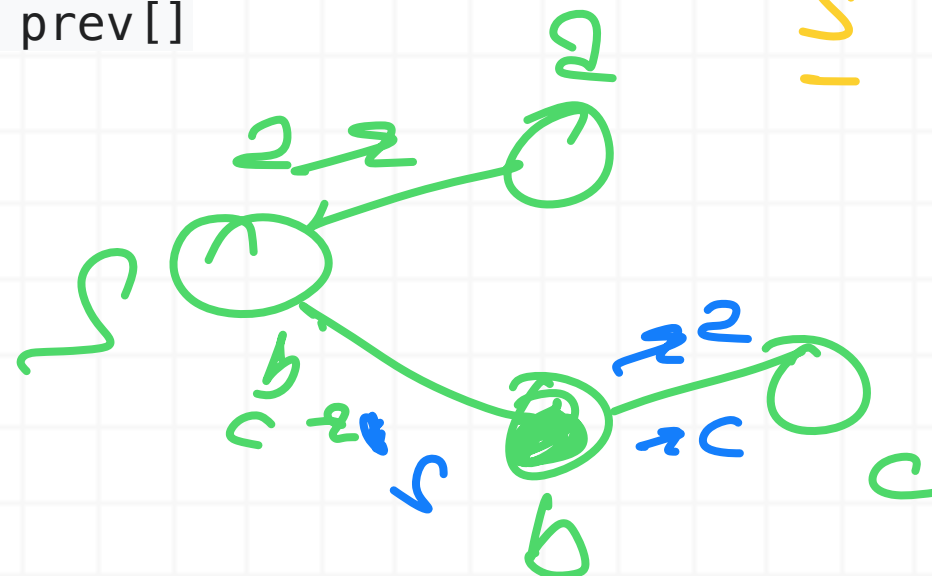
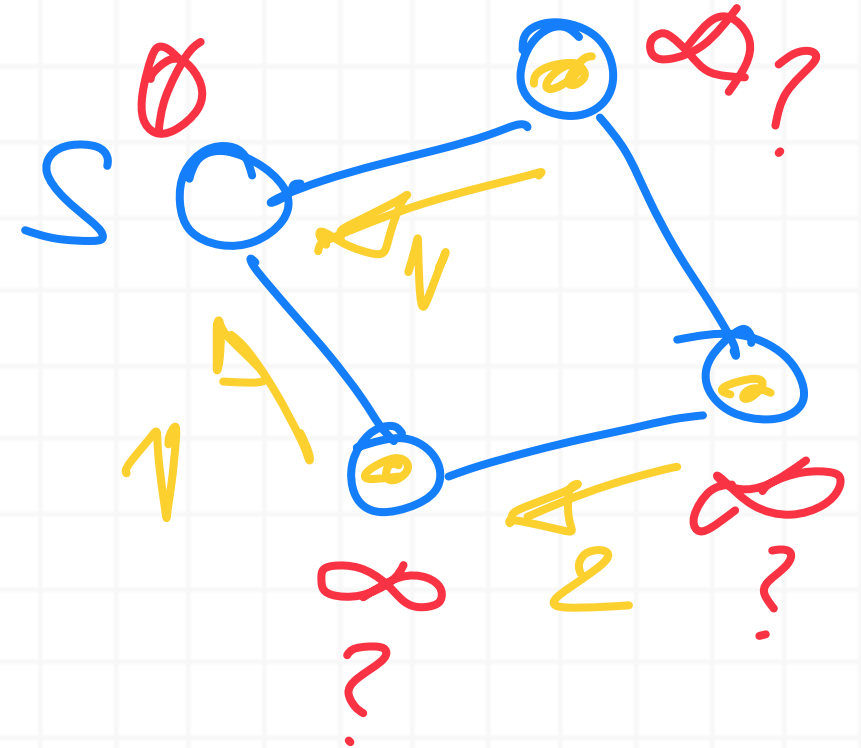
Imagen obtenida de <https://cnet.fi.uba.ar/TMA2018/>
Para más información: <https://cnet.fi.uba.ar/>

Protocolos de ruteo

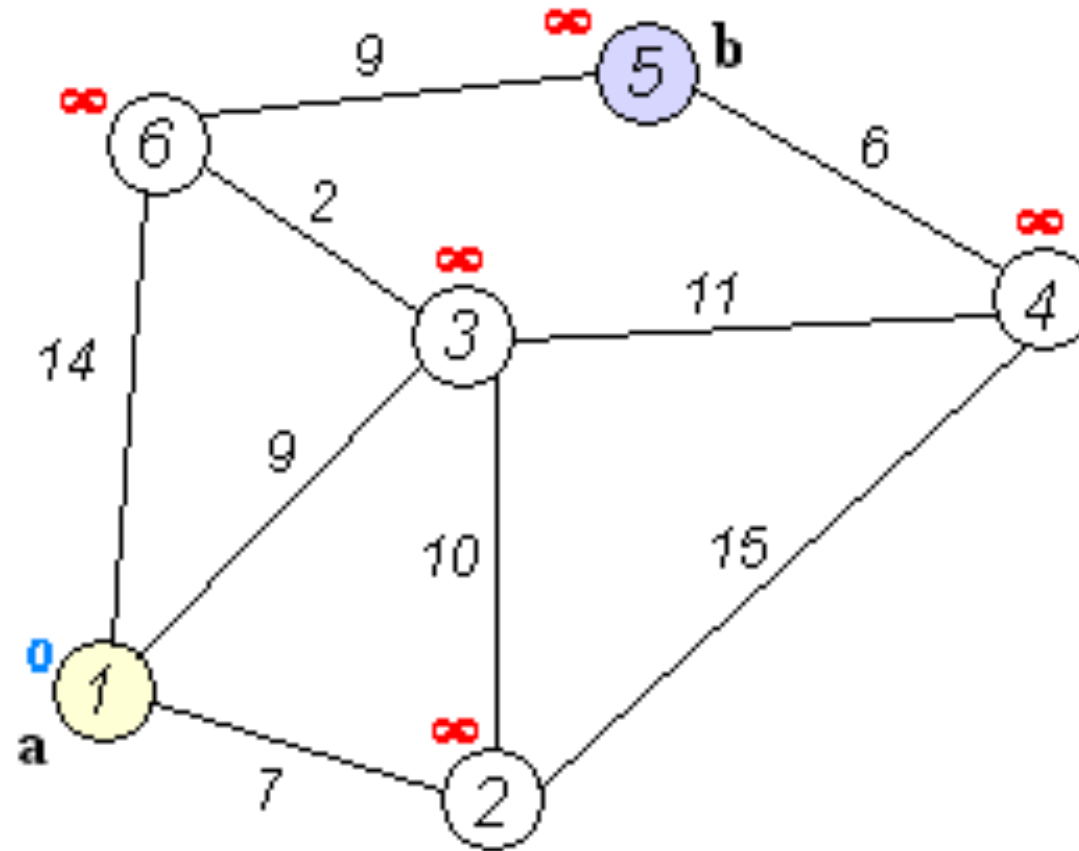
- Distintos puntos de vista: link-state, distance-vector
- OSPF y RIP

Algoritmo de Dijkstra¹

```
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[]
```



Algoritmo de Dijkstra¹



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

E es el número de aristas, ejes o conexiones

V es el número de vértices, nodos o participantes

Algoritmo Bellman Ford¹

```
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
8     for each edge (U,V) in G
9       tempDistance <- distance[U] + edge_weight(U, V)
10      if tempDistance < distance[V]
11        distance[V] <- tempDistance
12        previous[V] <- U
13
14  for each edge (U,V) in G
15    If distance[U] + edge_weight(U, V) < distance[V]
16      Error: Negative Cycle Exists
17
18
19  return distance[], previous[]
```


Algoritmo Bellman Ford



destino	dist.	Next hop
b	1	b
c	2	b
d	2	b

e	2	c
f	3	c

c	1	c
f	1	f
b	2	c
d	2	c
e	3	c
a	2	a
b	2	a
c	2	a
d	2	a
e	2	a
f	2	a

Complejidad:

$$O(|E| \cdot |V|)$$

E es el número de aristas, ejes o conexiones

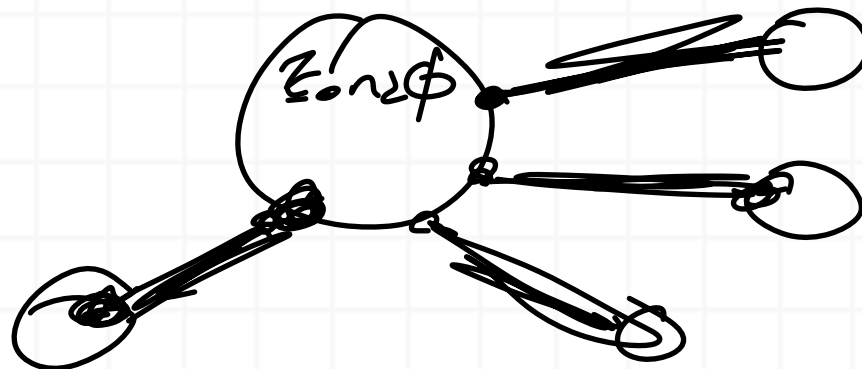
V es el número de vértices, nodos o participantes

© SPF : Dijkstra

Pesos — Capacidades
Prioridades

Autenticación

Zonas
Multicast



RIP (distance-vector) autenticación

OSPF: [RFC 2328](#)

RIP: [RFC 2453](#)

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