Tesis de Grado de Ingeniería Electrónica

Fragmentación y Mezcla de redes ad hoc utilizando el protocolo ANTop sobre IPv6

Autor: Pablo Damián Torrado
Director: Dr. Ing. J.Ignacio Alvarez-Hamelin

FI.UBA – CoNexDat
http://cnet.fi.uba.ar/

Marzo 2018

Temario

Temario

Motivación

- Estudio de fragmentación y mezcla de redes *ad hoc* utilizando el protocolo ANTop.

Motivación

- Estudio de fragmentación y mezcla de redes *ad hoc* utilizando el protocolo ANTop.

ANTop

- Protocolo de ruteo para redes *ad hoc*.
- Basado en el trabajo de J.I. Alvarez-Hamelin , A.C. Viana y M.D. Amorín "DHT-based functionalities using hypercubes".
- Desarrollado en FIUBA.
  - Tesis de Alejandro Marcu → Diseño y simulación de ANTop de ruteo reactivo.
  - Tesis de Gaston Tejia → Diseño y simulación de ANTop de ruteo proactivo.
  - Tesis de Matias Campolo → Implementación de ANTop de ruteo reactivo sobre un sistema operativo de distribución Debian.

Temario

Principales características

- Direccionamiento → dependiente de la topología.

Principales características

- Direccionamiento → dependiente de la topología.
- Espacios de direcciones basado en hipercubo.

Principales características

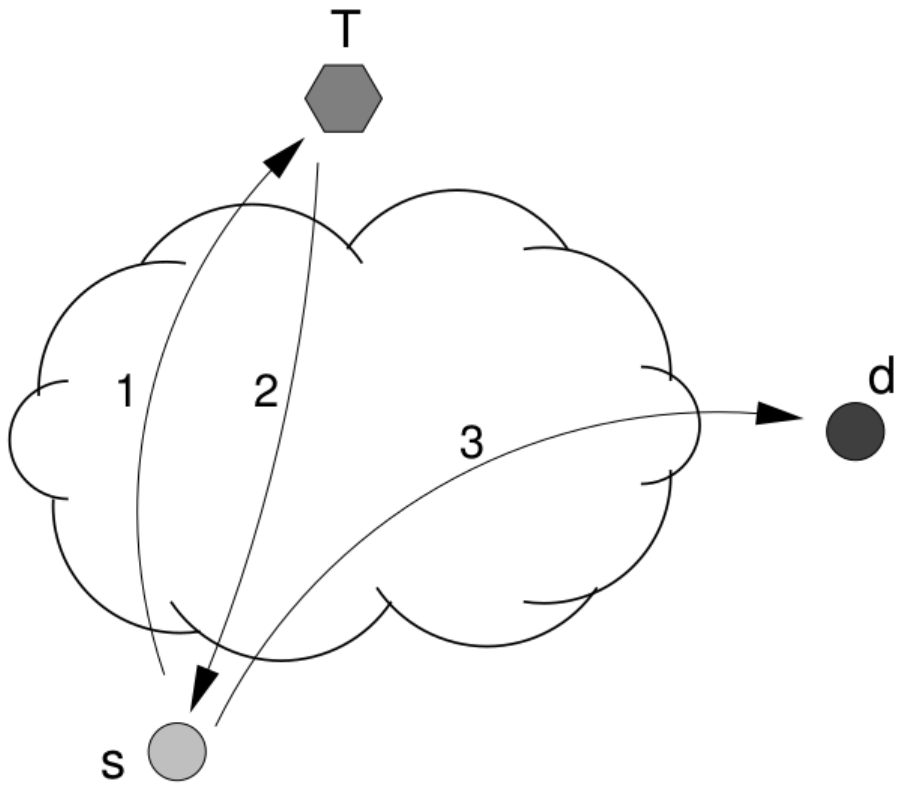
- Direccionamiento → dependiente de la topología.
- Espacios de direcciones basado en hipercubo.
- Servicio de resolución de nombres *Rendez Vous* → ruteo indirecto utilizando DHT (Tabla de Hash Distribuida).

Ruteo Indirecto

- Identificador Universal **U** → Conocido para conectarse.
- Dirección Virtual **V** → traducción **U** *(Rendez Vous)*.
- Dirección Relativa **R** → Cambia si el nodo se mueve.

Ruteo Indirecto

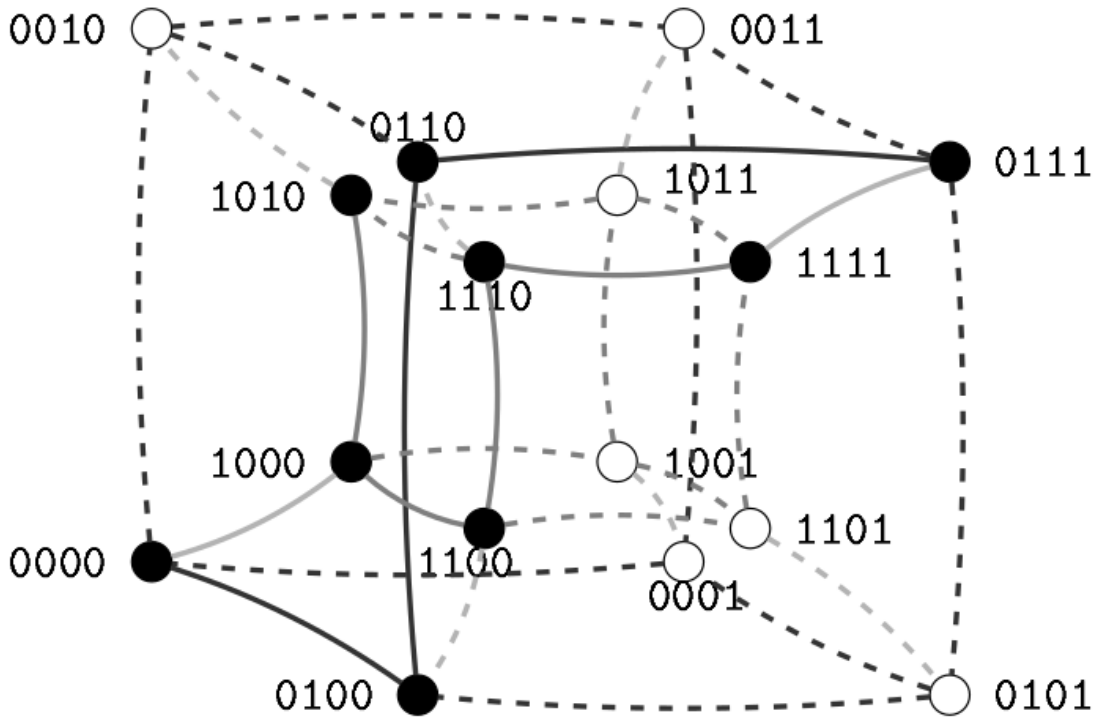
- Identificador Universal **U** → Conocido para conectarse.
- Dirección Virtual **V** → traducción **U** *(Rendez Vous)*.
- Dirección Relativa **R** → Cambia si el nodo se mueve.



1. El nodo **s** pide a **T** por la dirección relativa **Rd**.
2. **T** envía la direcciones relativas **Rd** a **s**.
3. Usando la dirección **Rd**, **s** ahora puede comunicarse con el nodo **d**.

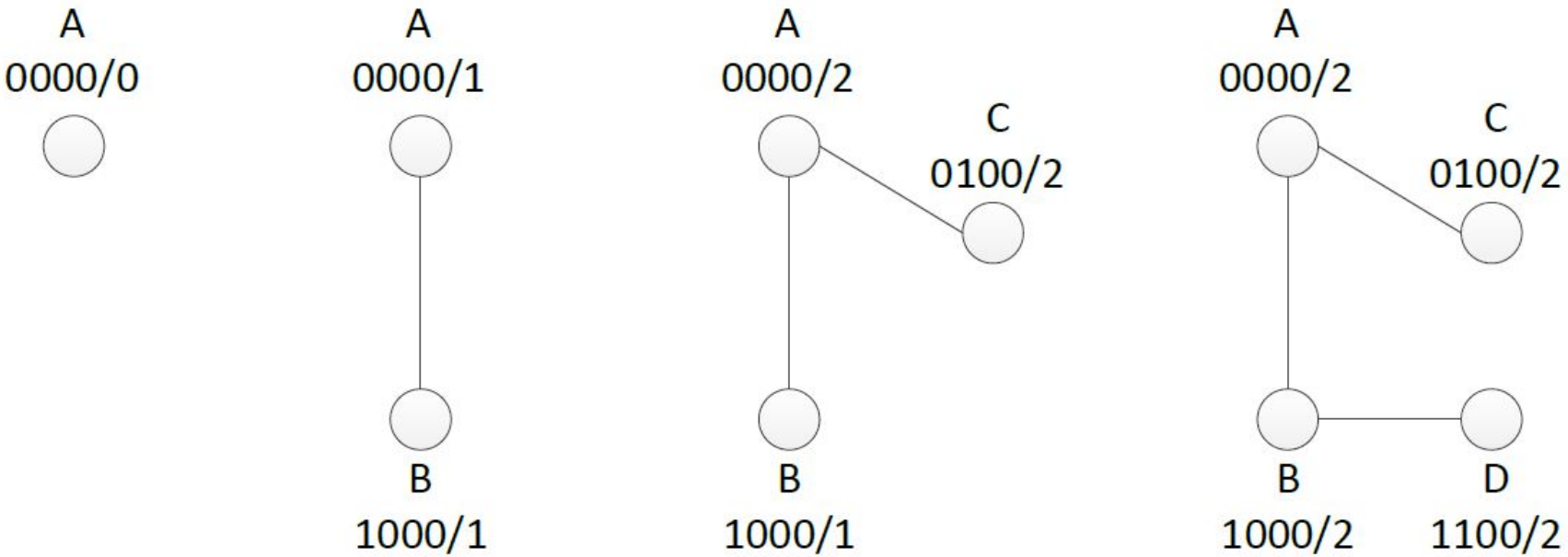
Asignación de direcciones

- Espacio de direcciones → estructura de hipercubo.
- Cada nodo conectado con **d** vecinos.
- Longitud de dirección **R** → **d**.
- Cantidad máxima de direcciones → 2^d.
- Distancia mínima entre nodos → operación XOR entre direcciones **R**.

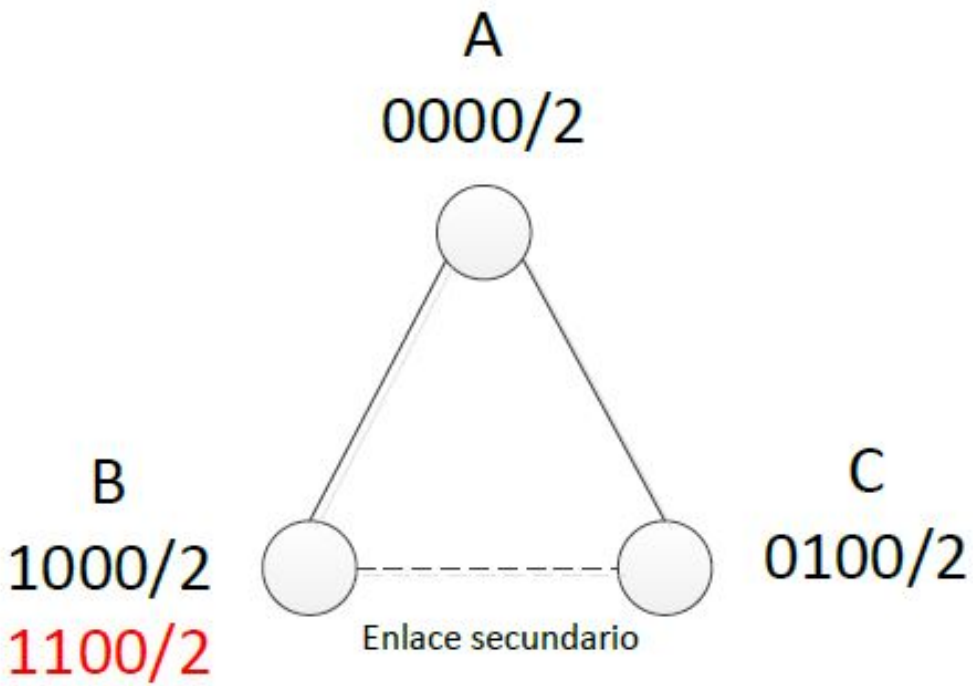


Espacio de direcciones

- Nuevo nodo → Nodo conectado cede mitad de su espacio de direcciones.
- Espacio de direcciones de red → Distribución de árbol.



Dirección secundaria



- **B** se asigna 1100/2 → secundaria.
- (1100 XOR 0100) = 1000 → distancia 1.
- **B** y **C** resultan vecinos.

Ruteo

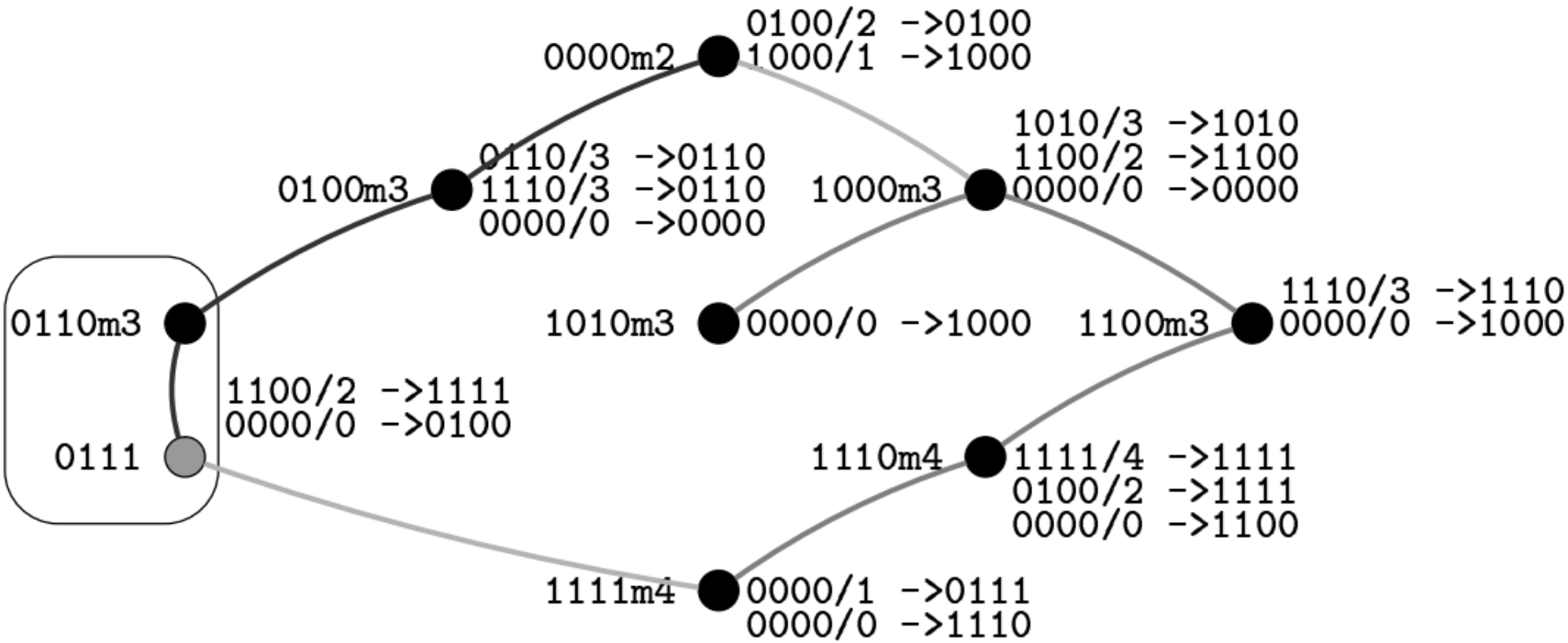
- Ruteo proactivo
  - Crea y mantiene tablas de enrutamiento todo el tiempo.
  - Asegura una ruta para cada nodo.
  - Redes con topologías estables.

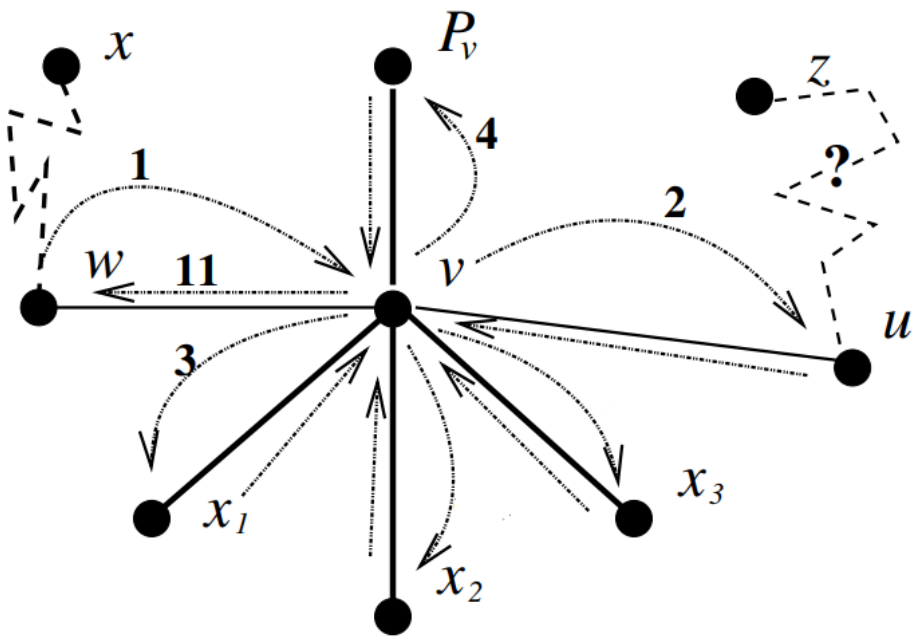
Ruteo

- Ruteo proactivo
  - Crea y mantiene tablas de enrutamiento todo el tiempo.
  - Asegura una ruta para cada nodo.
  - Redes con topologías estables.
- Ruteo reactivo
  - No hay información a priori $\rightarrow$ sin tablas de enrutamiento.
  - Mantiene un seguimiento temporal de las rutas utilizadas.
  - Redes con topologías altamente dinámicas.

Ruteo proactivo

- Utiliza información existente.
- Rutas estables.
- Rutas a *hijos* → Destinatario dentro del espacio de direcciones cedido.
- Ruta default → *padre*.
- Rutas propagadas.





Ruteo reactivo

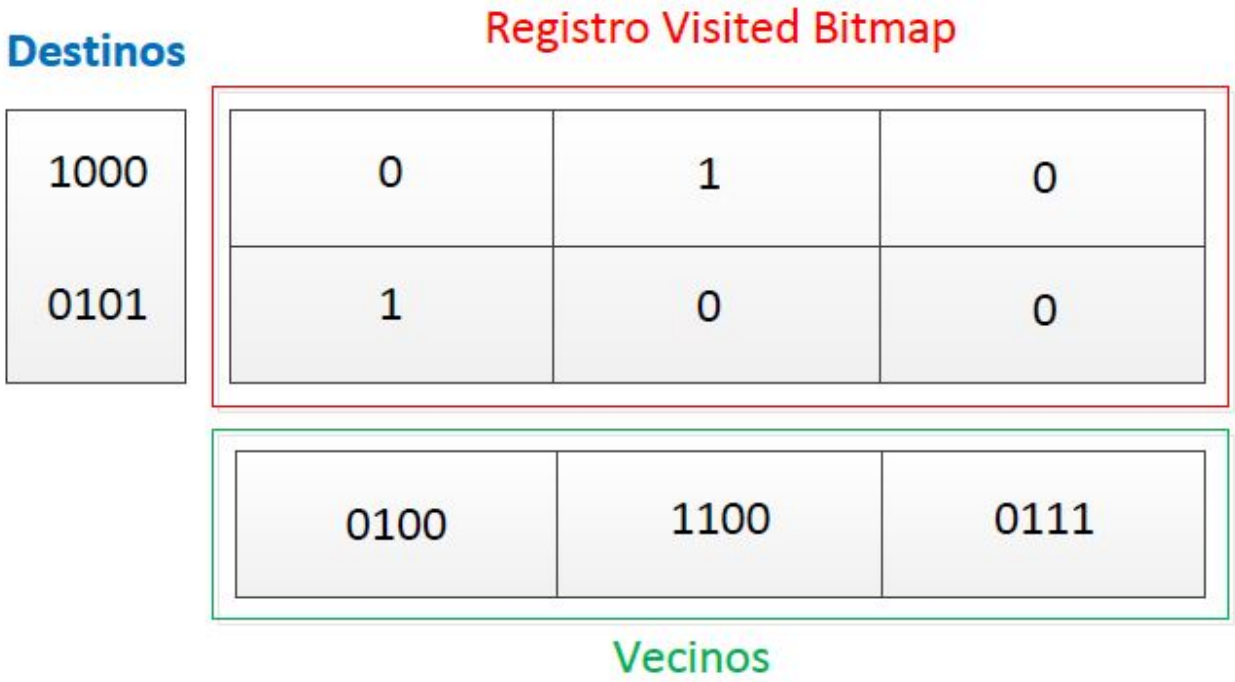
- Sin conocimiento a priori.
- Enrutamiento estándar de hipercubo.
- Almacena rutas no exitosas.
- Enruta mensajes → distancia mínima al destino.

Ruteo reactivo

- No se puede enrutar mensaje → devolverlo por donde vino.
- Mensaje devuelto → buscar nueva ruta.
- El mensaje se marca como devuelto → evita bucles.
- Rutas utilizadas → Se almacenan como visitado en el registro Bitmap.

Ruteo reactivo

- No se puede enrutar mensaje → devolverlo por donde vino.
- Mensaje devuelto → buscar nueva ruta.
- El mensaje se marca como devuelto → evita bucles.
- Rutas utilizadas → Se almacenan como visitado en el registro Bitmap.



Temario

Objetivos

1. Evaluación de problemas encontrados en la implementación ANTop en materia fragmentación y mezcla.

Objetivos

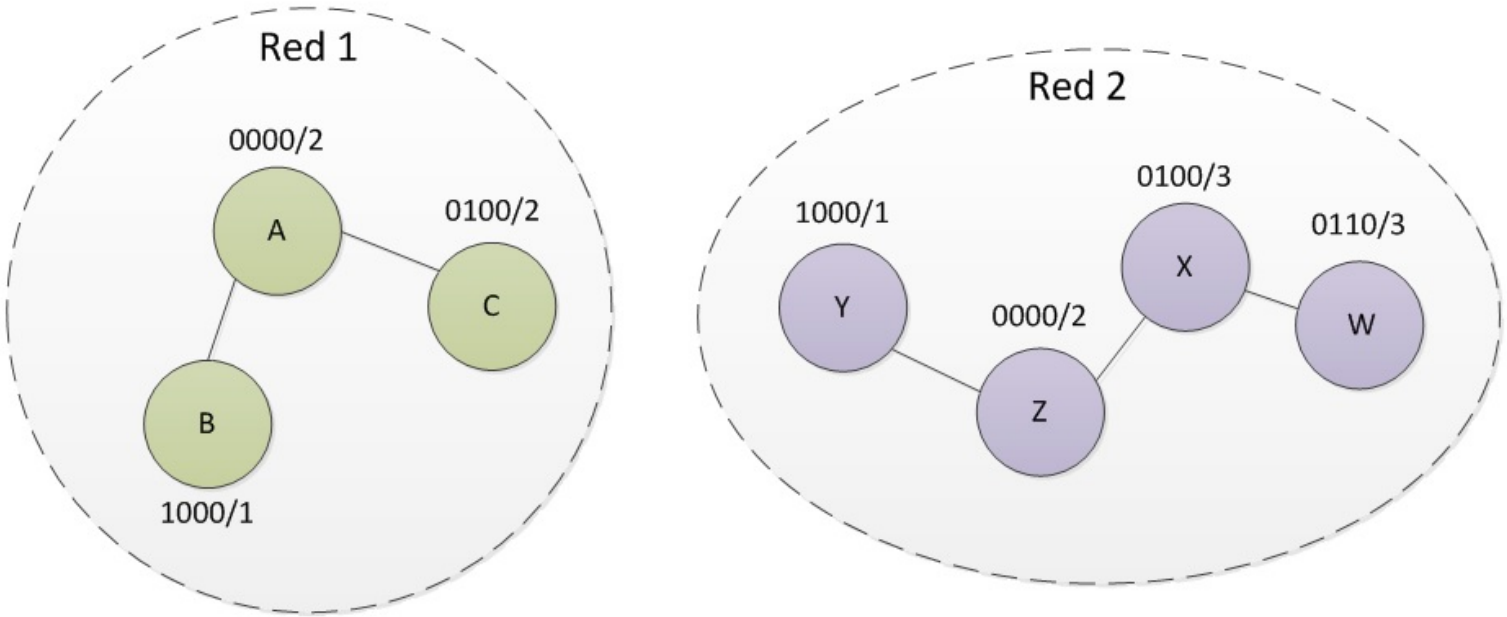
1. Evaluación de problemas encontrados en la implementación ANTop en materia fragmentación y mezcla.

2. Encontrar una solución para mitigarlos.

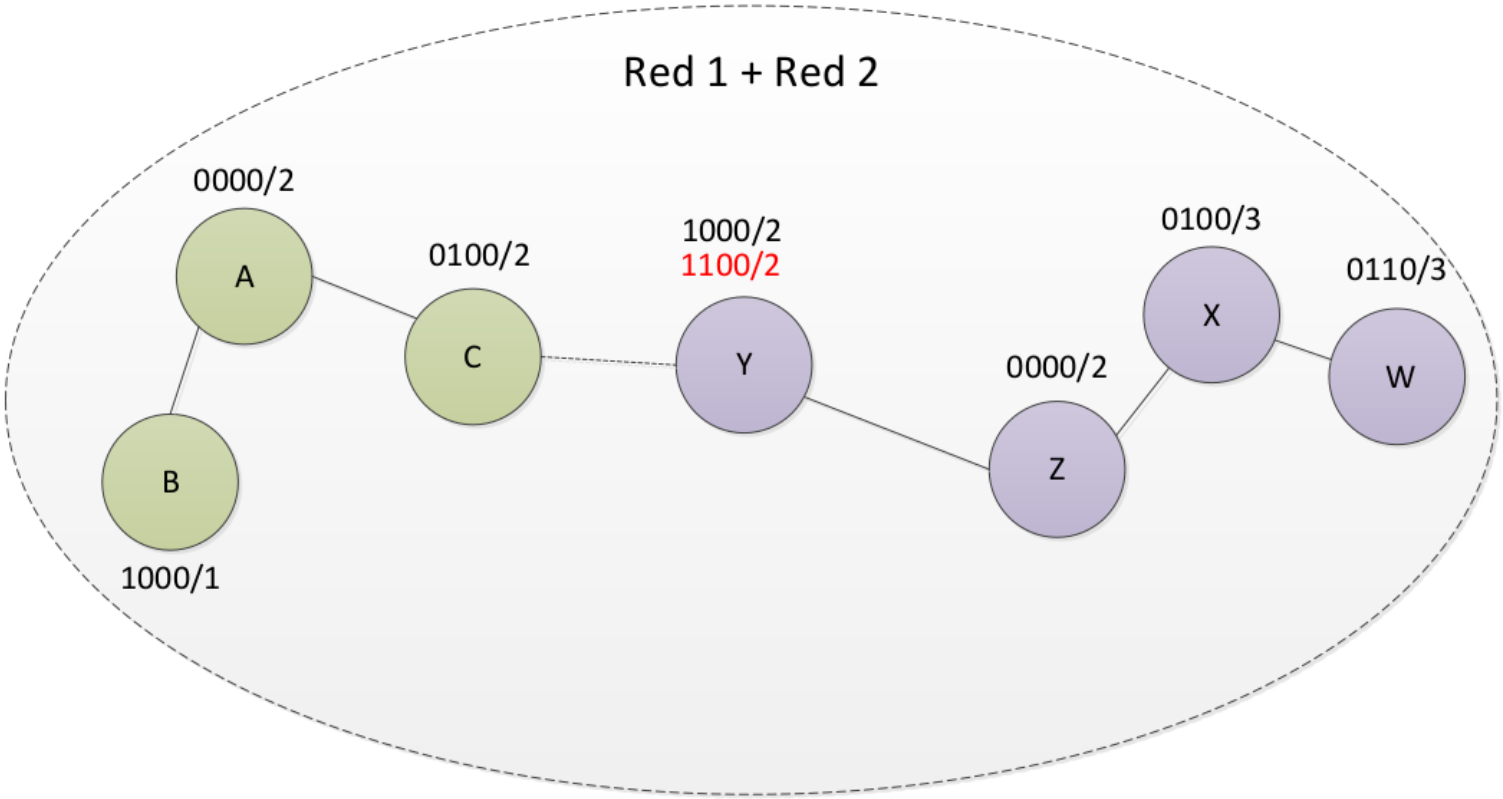
Objetivos

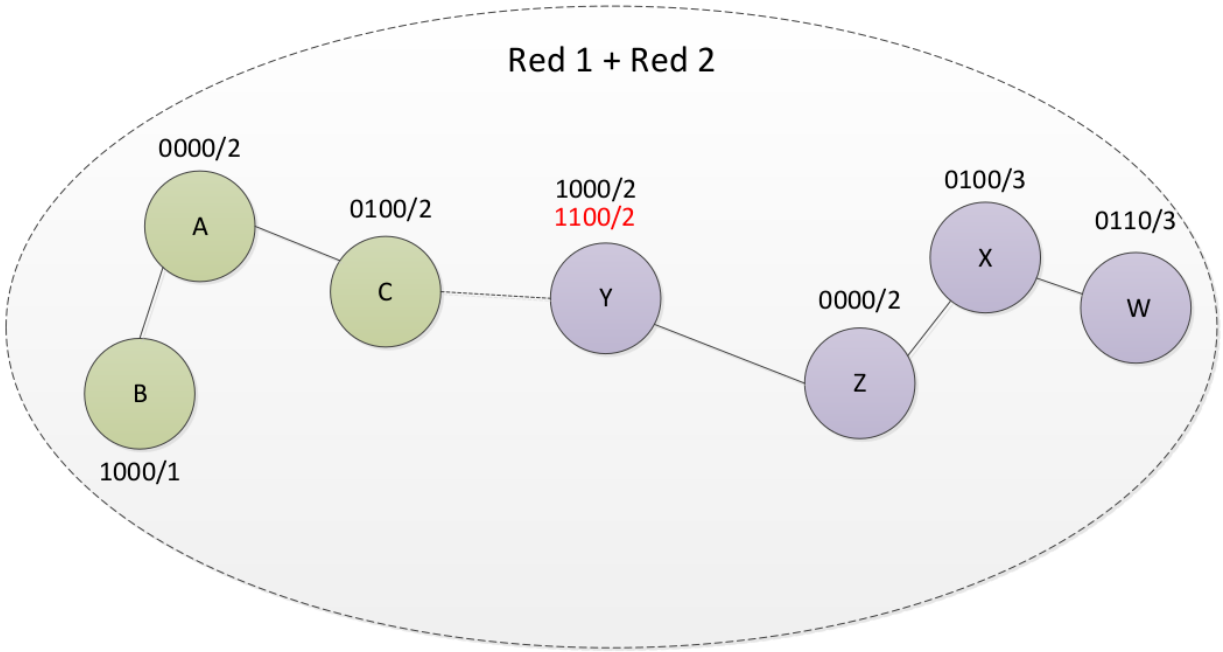
1. Evaluación de problemas encontrados en la implementación ANTop en materia fragmentación y mezcla.
2. Encontrar una solución para mitigarlos.
3. Lograr una nueva implementación del protocolo ANTop que soporte fragmentación y mezcla.

Ejemplo de mezcla



Ejemplo de mezcla





Algunos problemas encontrados

- Direcciones **R** duplicadas.
- El nodo Y pierde comunicación con el nodo X.
- Destruye el direccionamiento por hipercubo de la red.
- El ruteo deja de funcionar.

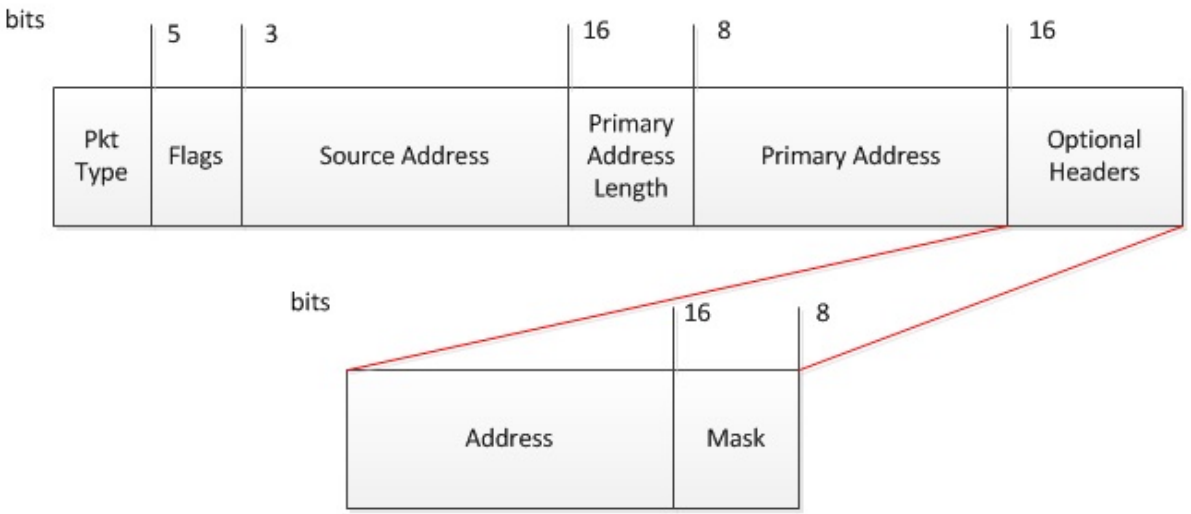
Necesidad

- Identificar las redes administradas independientes.
- Detección de paquetes de control pertenecientes a otra red.

Necesidad

- Identificar las redes administradas independientes.
- Detección de paquetes de control pertenecientes a otra red.

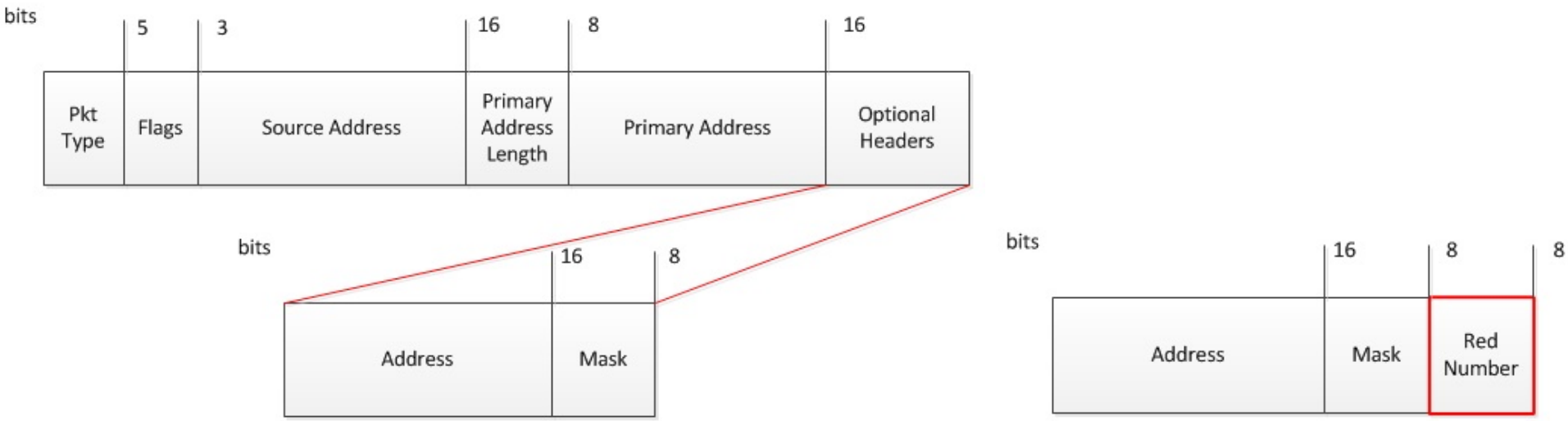
Campo *número de red*



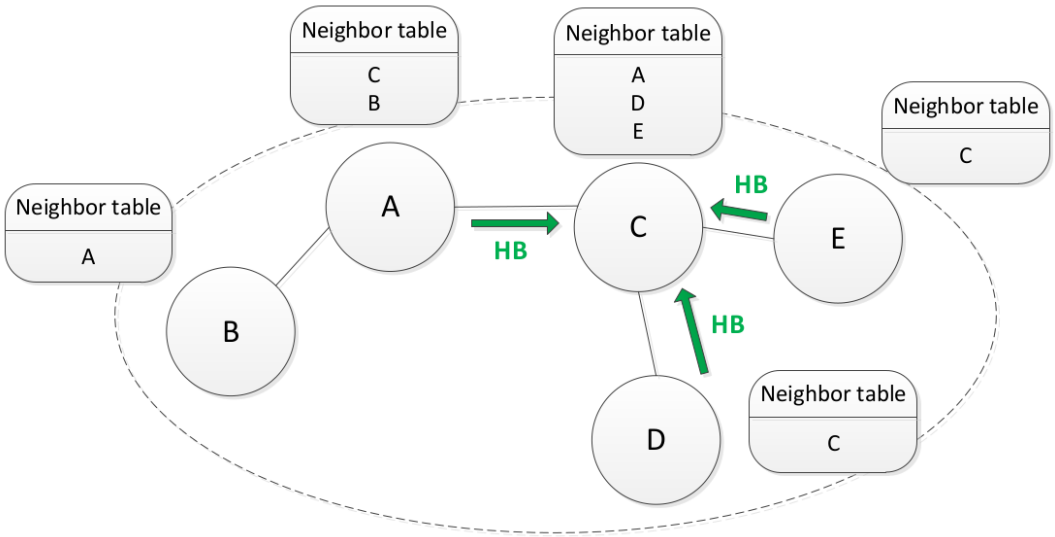
Necesidad

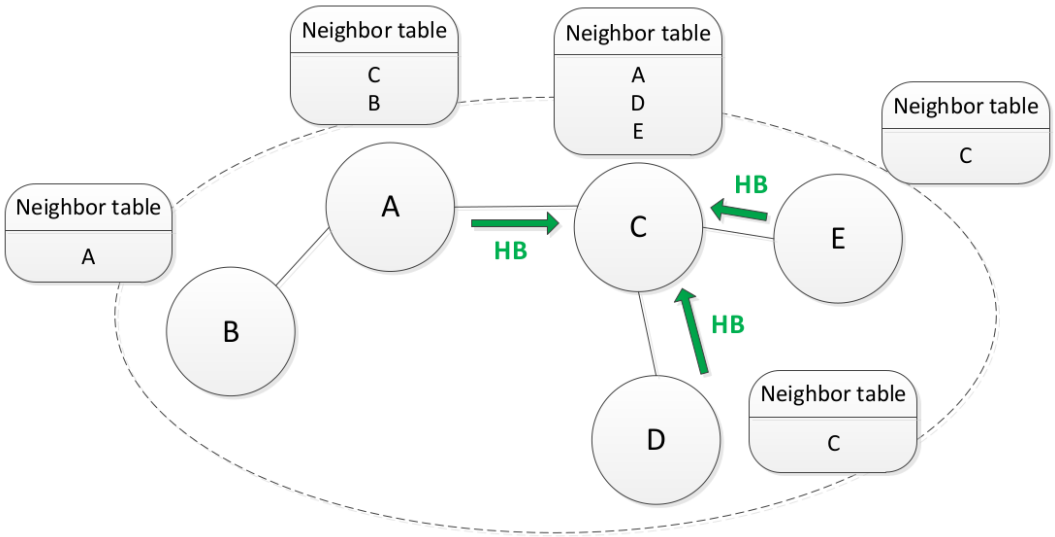
- Identificar las redes administradas independientes.
- Detección de paquetes de control pertenecientes a otra red.

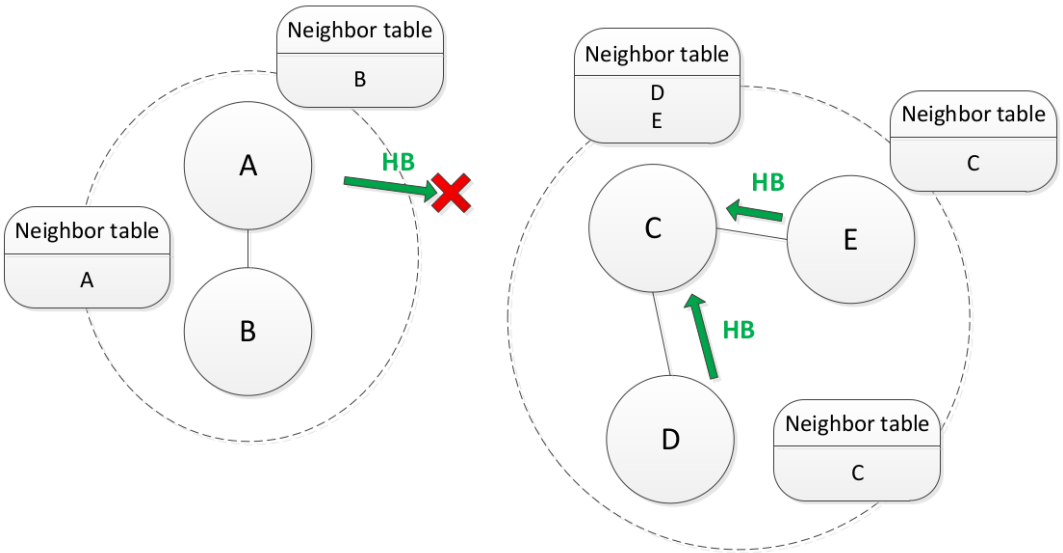
Campo *número de red*

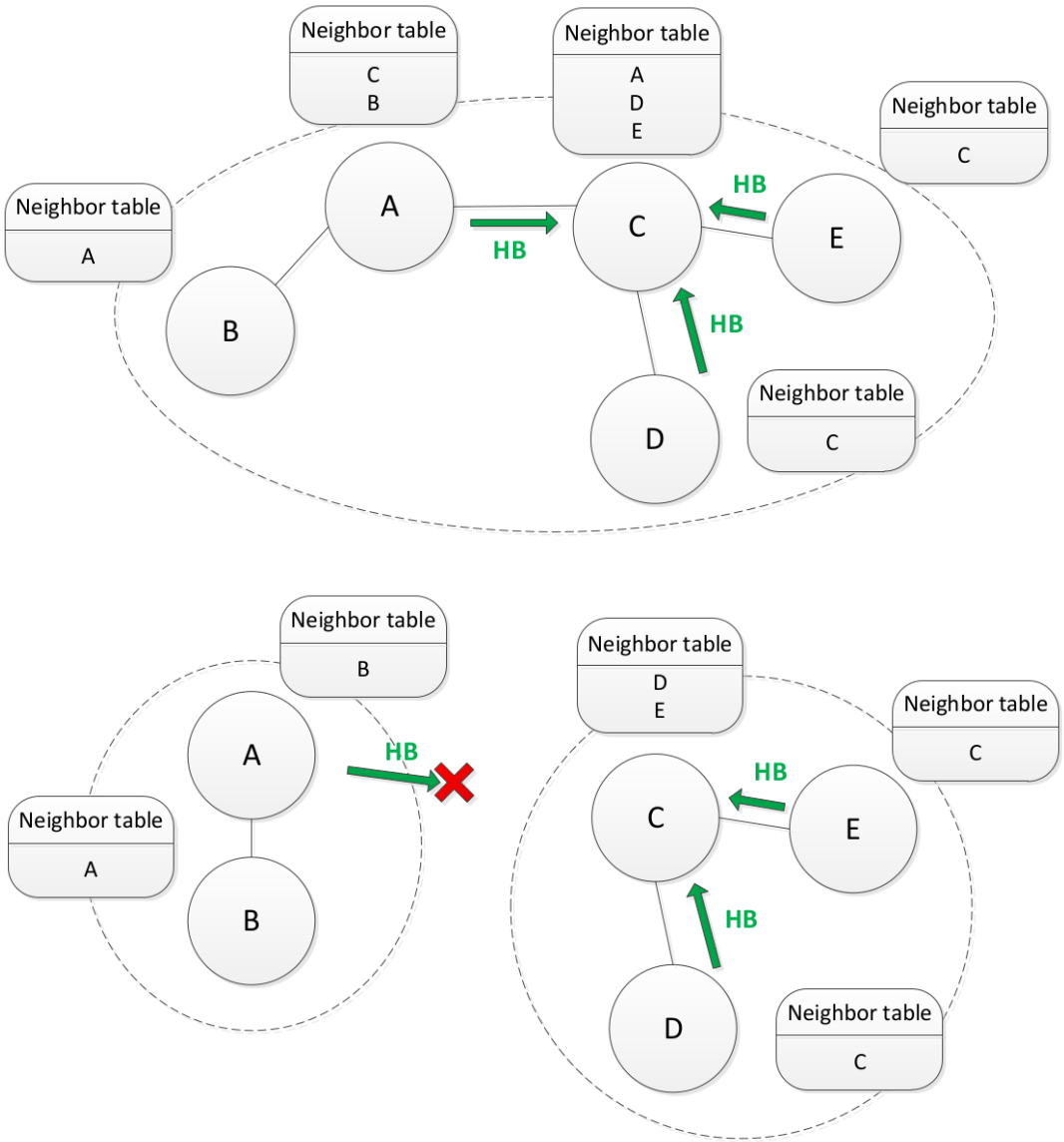


1. Nodo aislado → **4° Byte @ MAC**
2. Conexión de nodo → Campo *número de red* del PAP.
3. No hay direcciones disponibles → NULL.


Neighbor table
C
B
Neighbor table
A
D
E
Neighbor table
C
Neighbor table
A
A
B
C
E
D
HB
HB
HB
Neighbor table
C


Neighbor table
C
B
Neighbor table
A
D
E
Neighbor table
C
Neighbor table
A
A
B
C
E
D
HB
HB
HB
Neighbor table
C


Neighbor table
B
A
HB
Neighbor table
A
B
Neighbor table
D
E
Neighbor table
C
C
E
D
HB
HB
Neighbor table
C

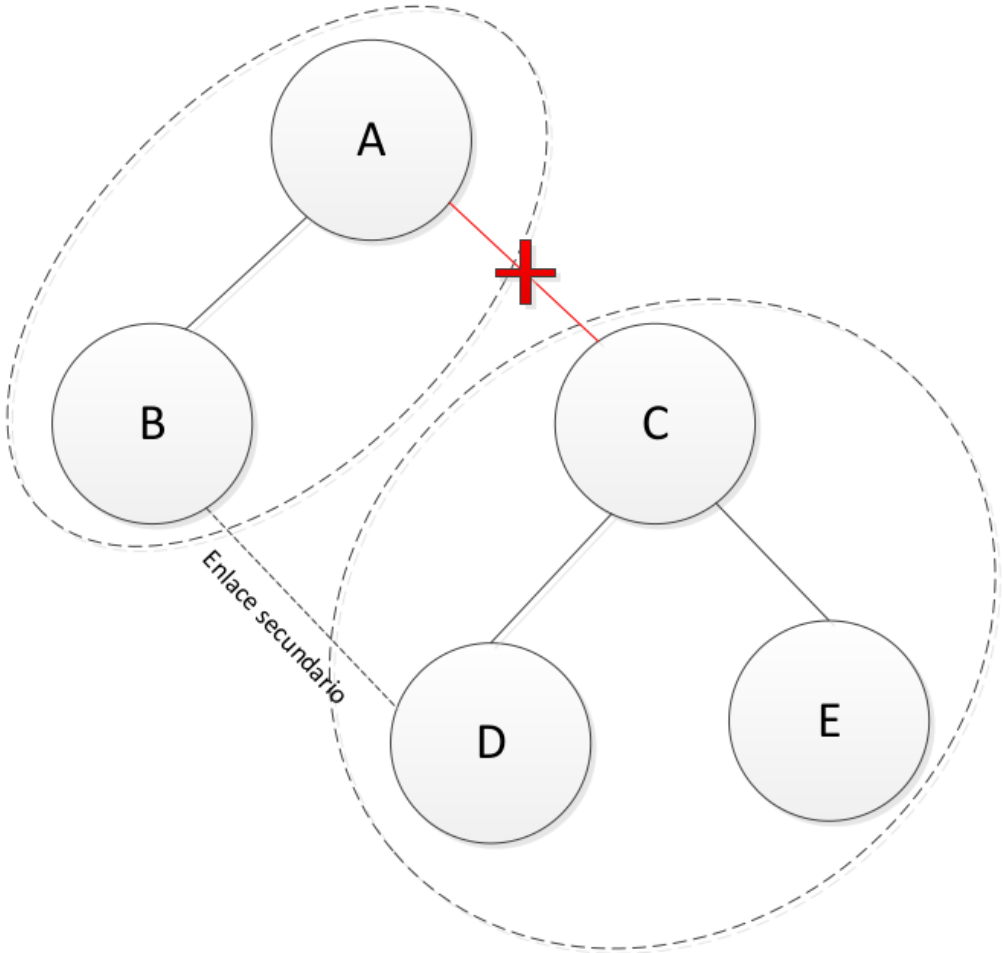


Nodo *padre*:
Es el nodo vecino en el cuál le proporcionó a uno el espacio de direcciones cuando éste se conecto a la red.

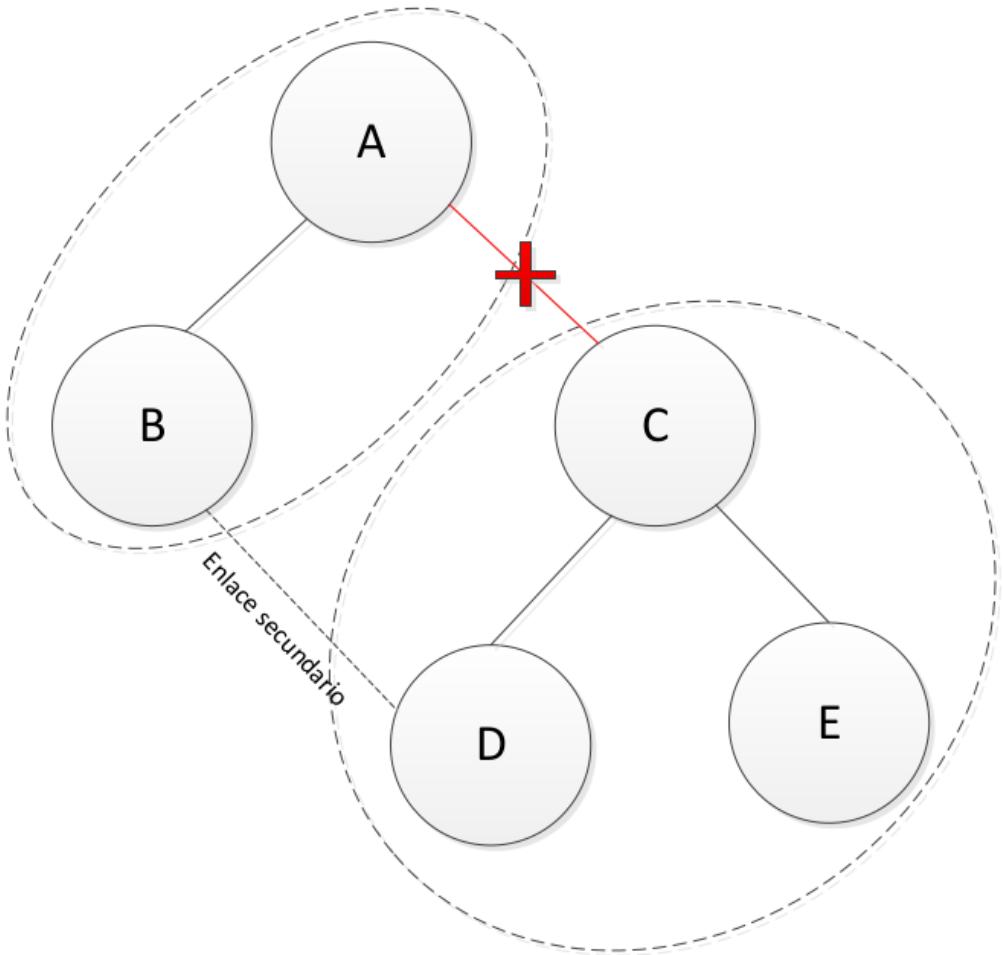
Primera definición

Una red se fragmenta cuando uno de los nodos pierde conexión con su nodo *padre*.

¿Y si hay un enlace secundario?



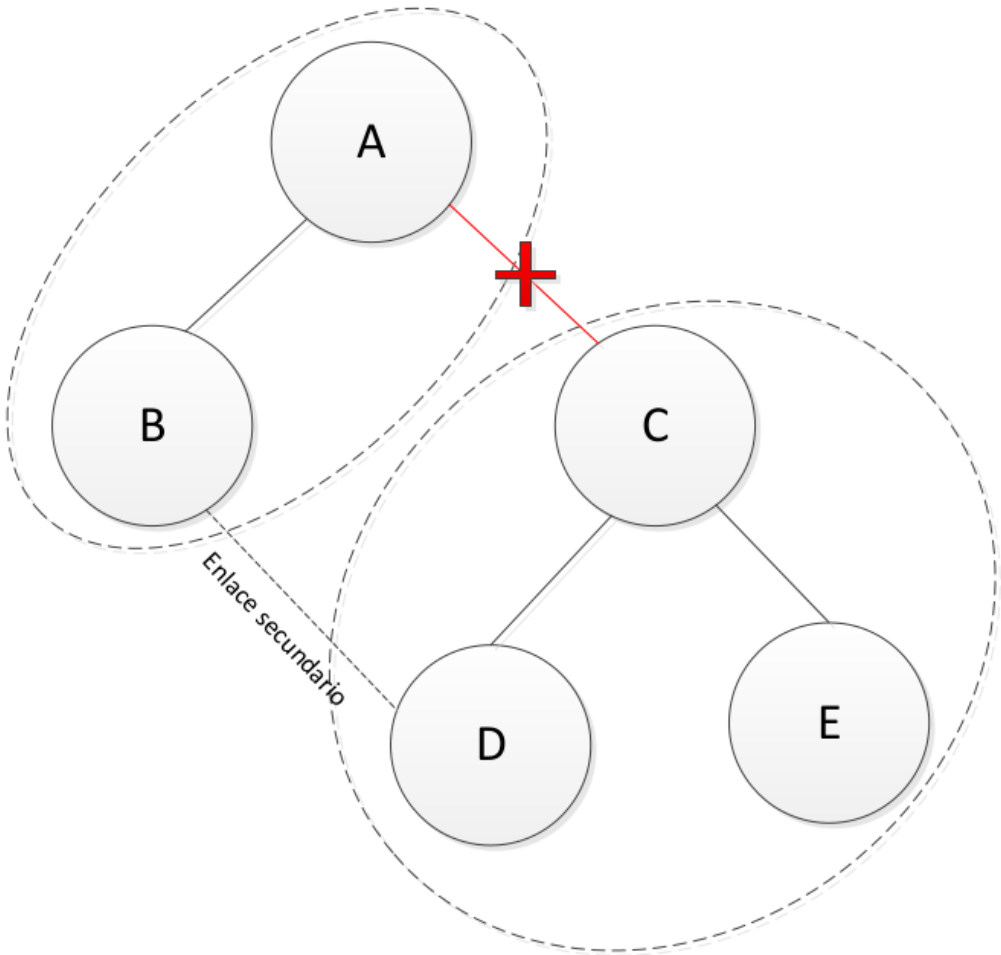
¿Y si hay un enlace secundario?



Segunda definición

Una red se fragmenta cuando uno de los nodos pierde conexión con su nodo *padre*, **y además no existe otro camino por enlace secundario entre las partes.**

¿Y si hay un enlace secundario?



Segunda definición

Una red se fragmenta cuando uno de los nodos pierde conexión con su nodo *padre*, **y además no existe otro camino por enlace secundario entre las partes**.

Detección

1. El nodo *padre* se borra del registro de vecinos.
2. No se tiene respuesta del *ping* al nodo *root*.

Principales acciones a tomar en la red fragmentada

- Asignar nuevo *número de red*.
- Asignar nueva dirección **R**.
- Asignar máscara y máscara inicial (*mi*).

Principales acciones a tomar en la red fragmentada

- Asignar nuevo *número de red*.
- Asignar nueva dirección **R**.
- Asignar máscara y máscara inicial (mi).

Tratamiento

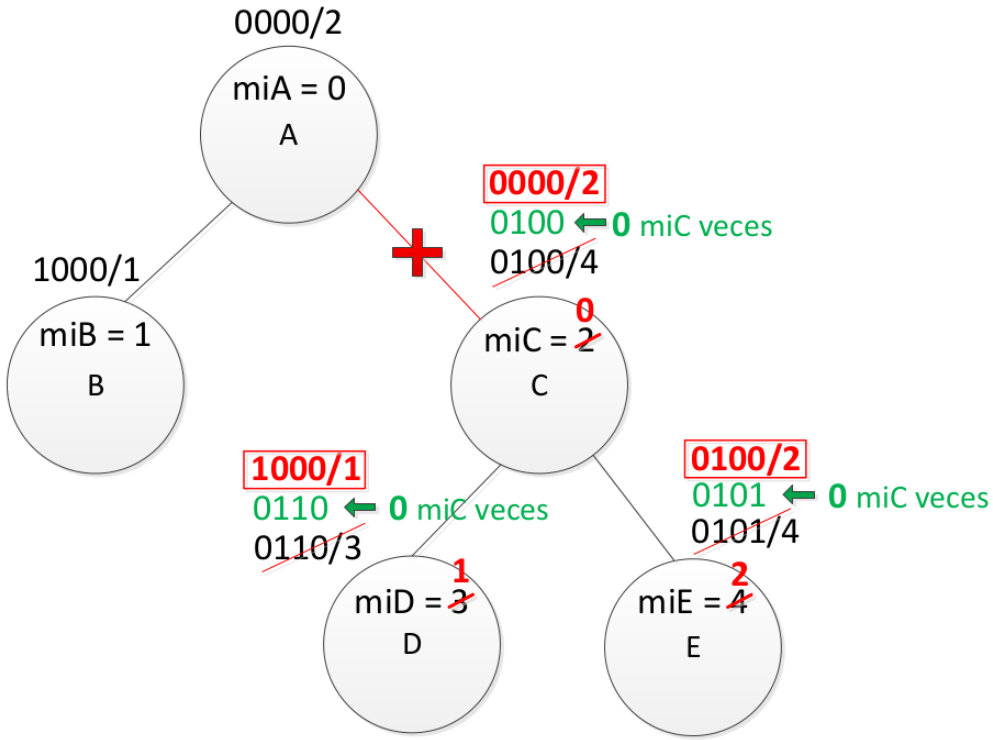
- Dirección **R** resultante $\rightarrow$ *shift* de cero a izquierda la cantidad de veces mi del nodo detector.
- $m_{frag} = m - mi_{\ nodo\ detector}$
- $mi_{frag} = mi - mi_{\ nodo\ detector}$

Principales acciones a tomar en la red fragmentada

- Asignar nuevo *número de red*.
- Asignar nueva dirección **R**.
- Asignar máscara y máscara inicial (mi).

Tratamiento

- Dirección **R** resultante → *shift* de cero a izquierda la cantidad de veces mi del nodo detector.
- $m_{frag} = m - mi_{\ nodo\ detector}$
- $mi_{frag} = mi - mi_{\ nodo\ detector}$

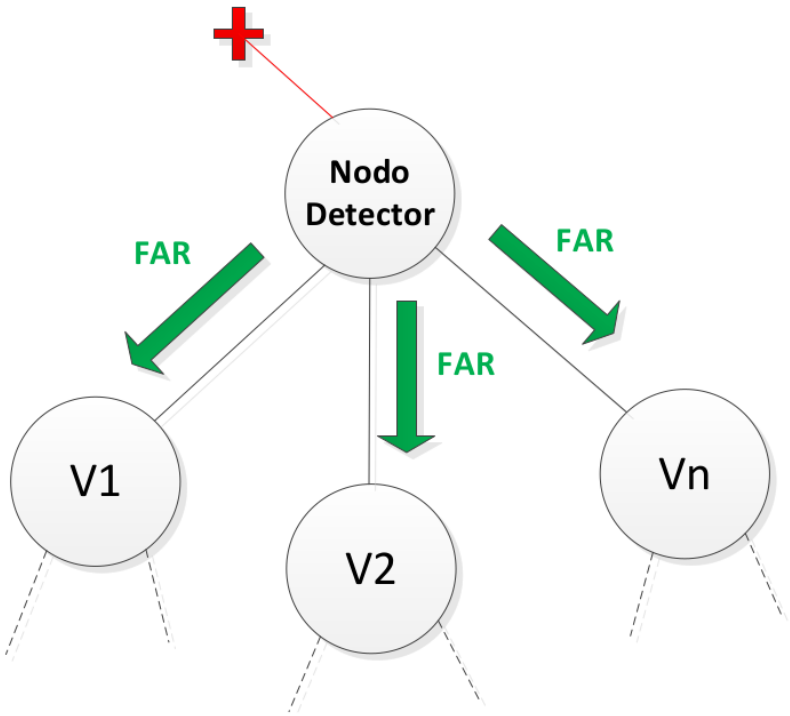


Paquetes de control

- FAR *(Fragment Address Request)* → Transporta información del *nodo detector*.
- FAN *(Fragment Address Notification)* → Confirmación del mensaje FAR.

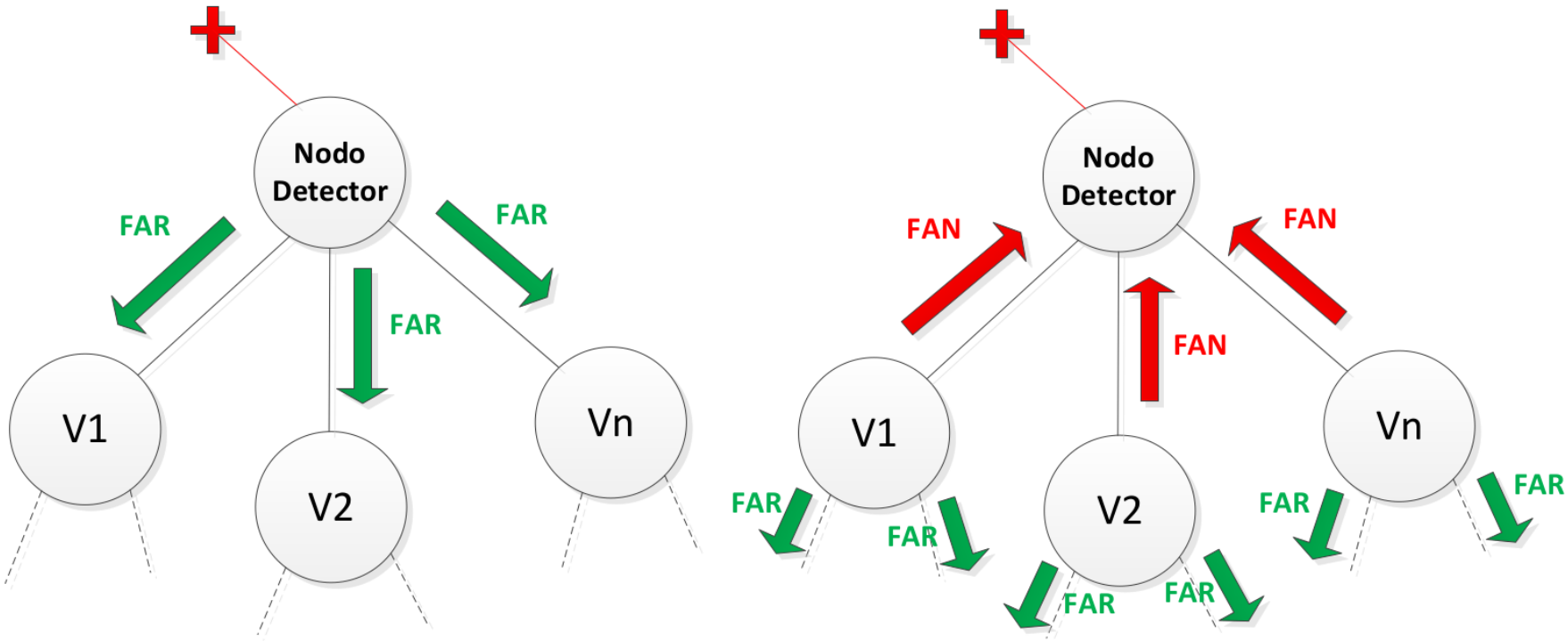
Paquetes de control

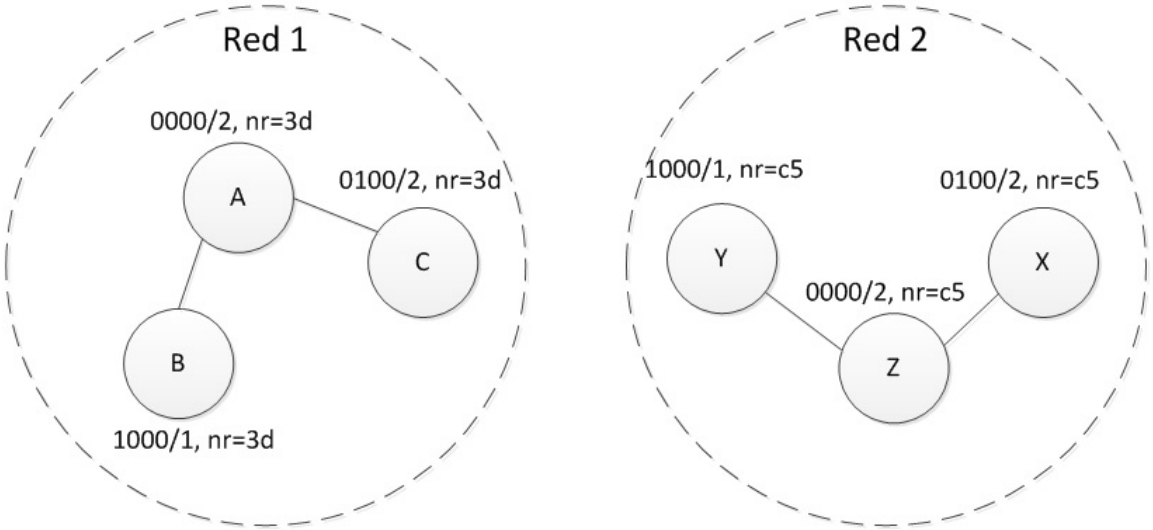
- FAR *(Fragment Address Request)* → Transporta información del *nodo detector*.
- FAN *(Fragment Address Notification)* → Confirmación del mensaje FAR.

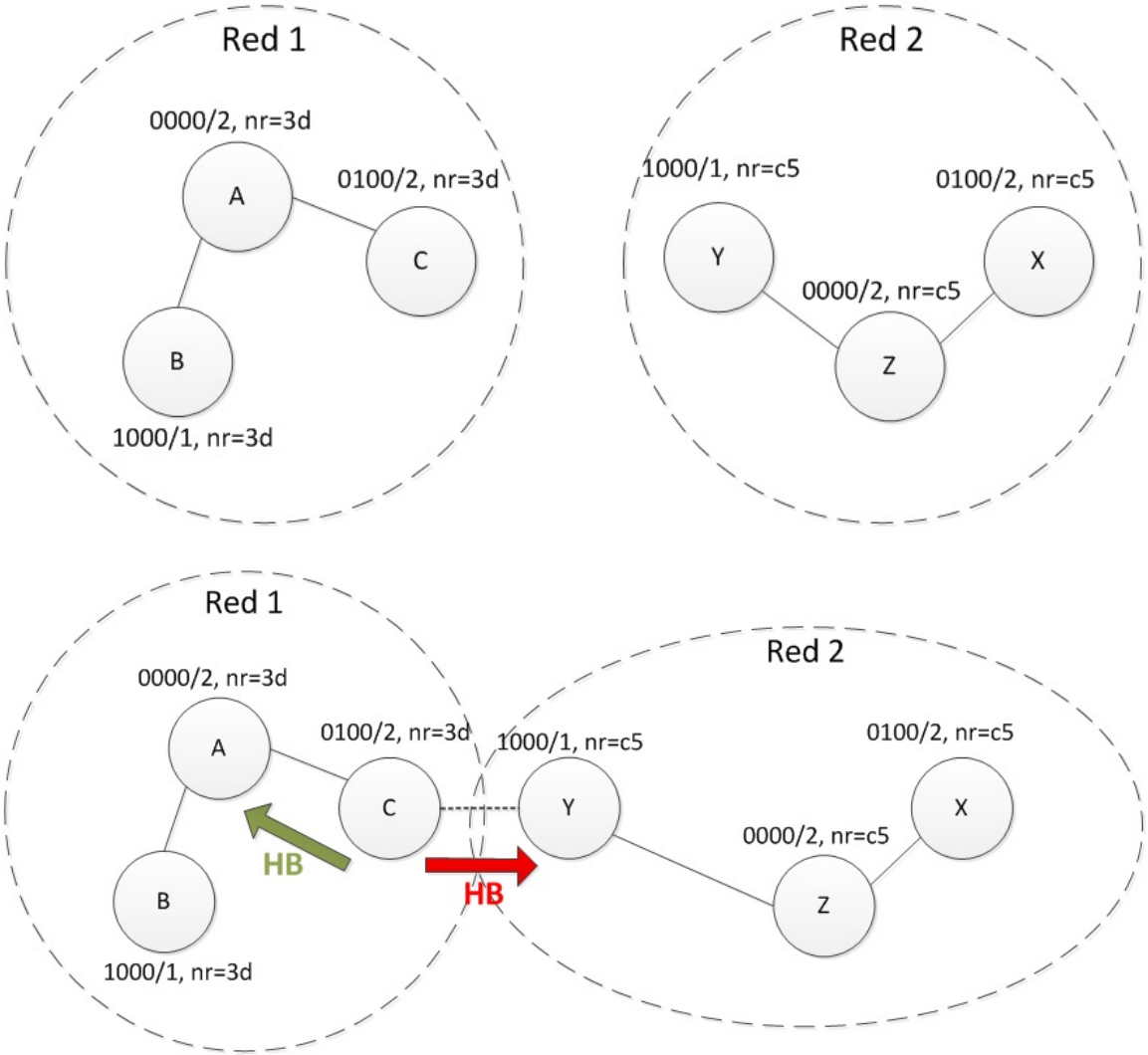


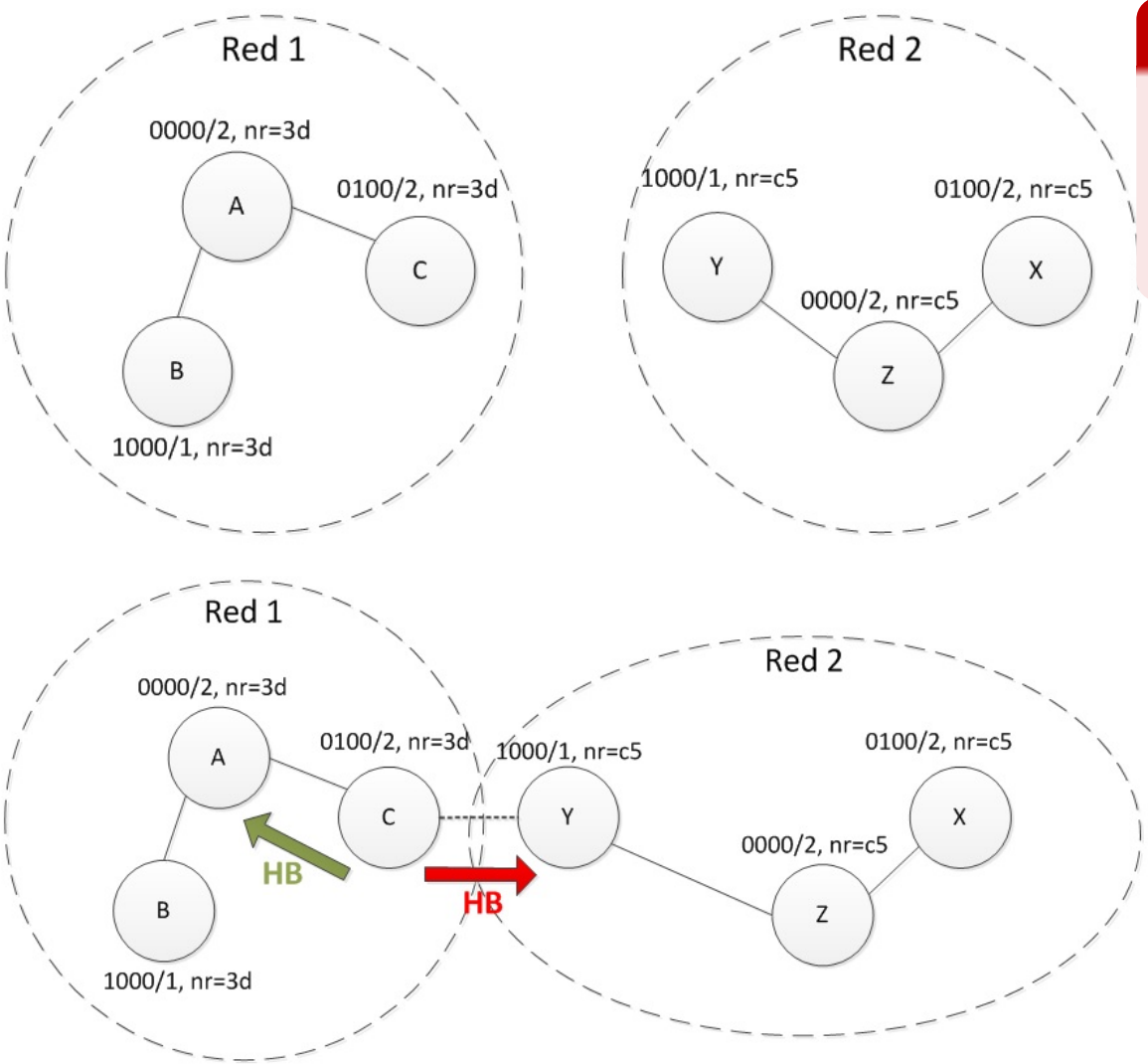
Paquetes de control

- FAR *(Fragment Address Request)* → Transporta información del *nodo detector*.
- FAN *(Fragment Address Notification)* → Confirmación del mensaje FAR.



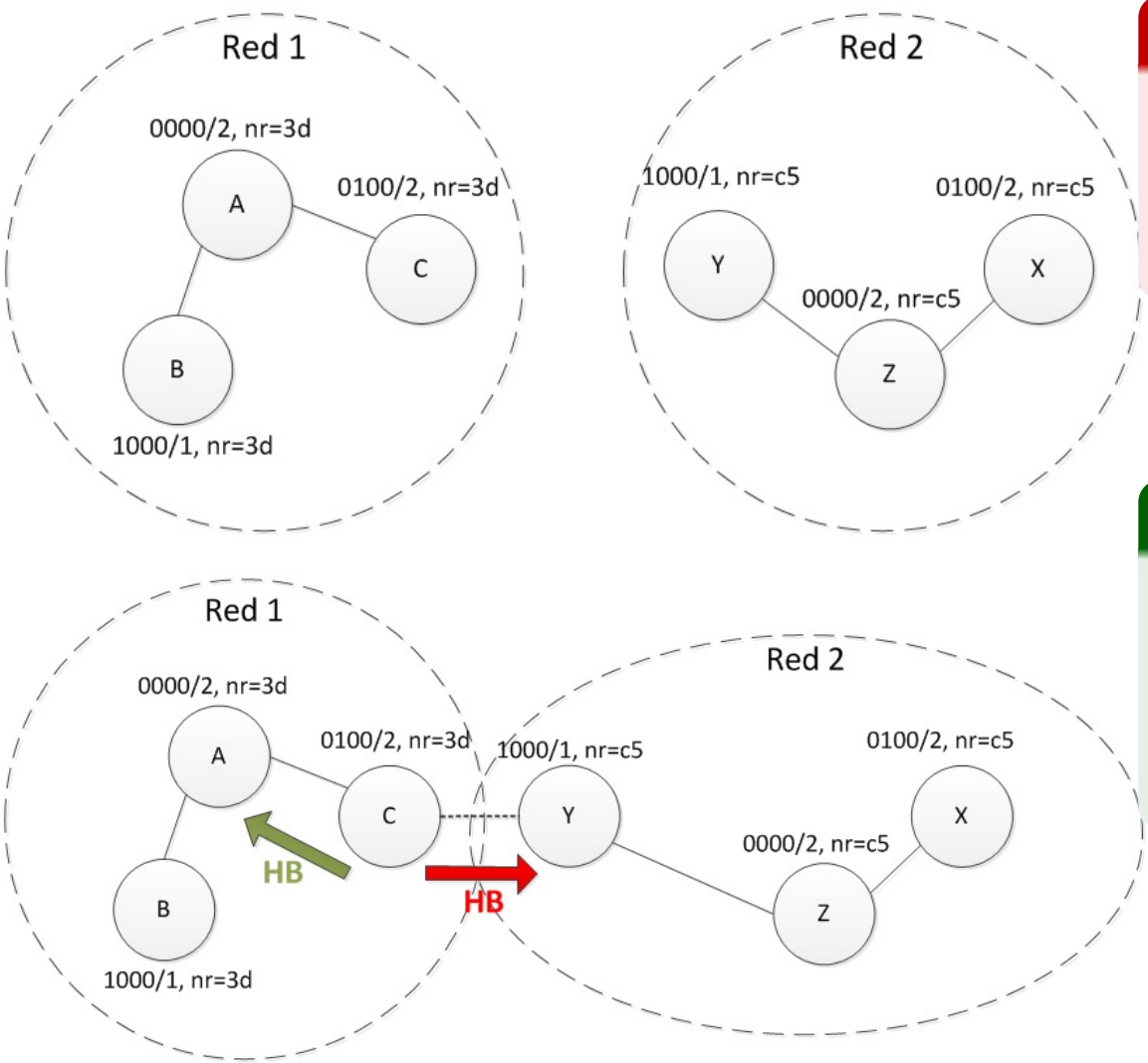

Red 1
0000/2, nr=3d
A
0100/2, nr=3d
C
B
1000/1, nr=3d
Red 2
1000/1, nr=c5
0100/2, nr=c5
Y
X
0000/2, nr=c5
Z


Red 1
0000/2, nr=3d
A
0100/2, nr=3d
C
B
1000/1, nr=3d
Red 2
1000/1, nr=c5
0100/2, nr=c5
Y
X
0000/2, nr=c5
Z
Red 1
0000/2, nr=3d
A
0100/2, nr=3d
C
HB
B
1000/1, nr=3d
Red 2
1000/1, nr=c5
0100/2, nr=c5
Y
X
HB
0000/2, nr=c5
Z



Definición

Una red se mezcla con otra cuando un nodo recibe paquetes con diferente *número de red*.



Definición

Una red se mezcla con otra cuando un nodo recibe paquetes con diferente *número de red*.

Detección

1. Procesar un paquete HB con distinto *número de red*.
2. *número de red* local $<$ *número de red* del HB.

Principales acciones a tomar en los nodos de la red que se mezcla

- Asignar *número de red*.
- Asignar dirección **R** acorde a la red resultante.
- Asignar máscara y máscara inicial (*mi*).

Principales acciones a tomar en los nodos de la red que se mezcla

- Asignar *número de red*.
- Asignar dirección **R** acorde a la red resultante.
- Asignar máscara y máscara inicial (mi).

Tipo de tratamiento

- Mecanismo 1 → Aplicable a los nodos hijos del *par receptor*.
- Mecanismo 2 → Aplicable a los nodos que no son *hijos* del *par receptor*.

Mecanismo 1

Dos casos:

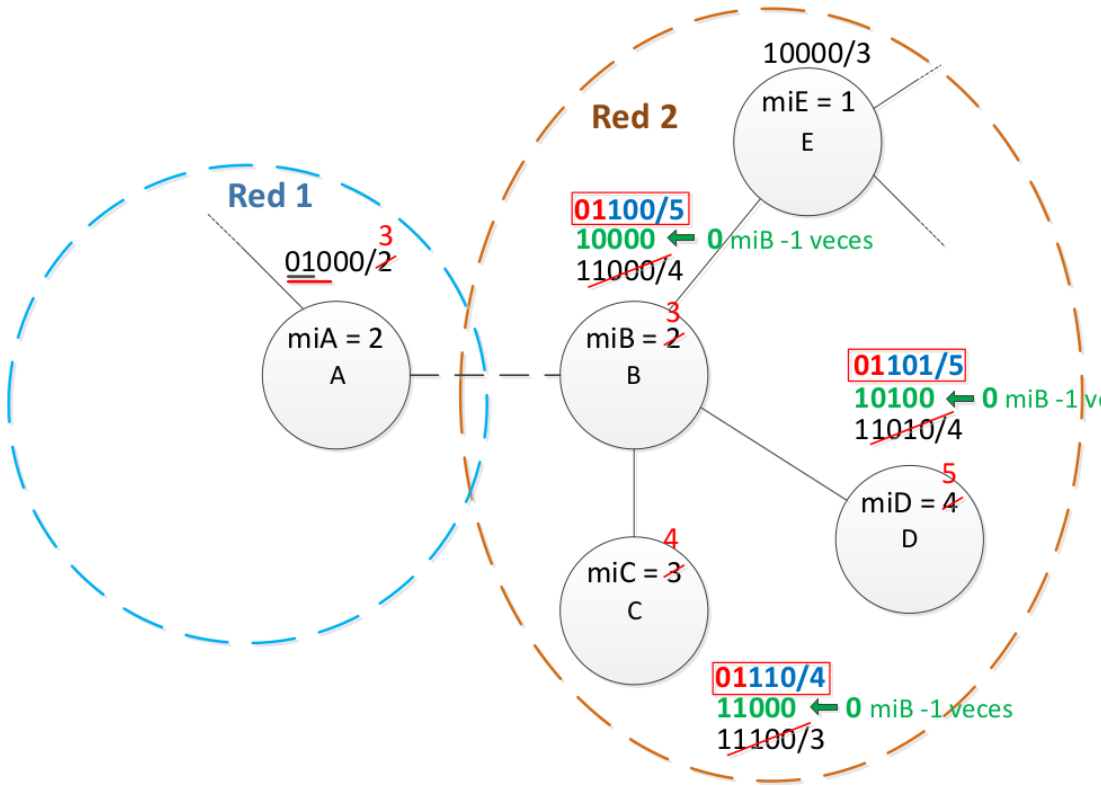
1. **Caso 1** $\rightarrow$ Dirección **R** *par receptor* $\neq$ todos los bits en cero.
2. **Caso 2** $\rightarrow$ Dirección **R** *par receptor* $=$ todos los bits en cero.

Tratamiento Caso 1

1. De la dirección **R** $\rightarrow$ *shift* de cero a izquierda la cantidad de veces de $(mi_{\ par\ recpetor} - 1)$.
2. Del punto anterior $\rightarrow$ *shift* de cero a derecha la cantidad de veces la $m_{\ par\ recpetor}$.
3. Dirección **R** resultante $\rightarrow$ operación OR entre el punto 2 y dirección **R** del *par emisor*.
4. $m_{mezcla} = m_{\ par\ emisor} + m - (mi_{\ par\ receptor} - 1)$
5. $mi_{mezcla} = m_{\ par\ emisor} + 1 + (mi - mi_{\ par\ receptor})$

Tratamiento Caso 1

1. De la dirección **R** → *shift* de cero a izquierda la cantidad de veces de $(mi\ _{par\ recpetor} - 1)$.
2. Del punto anterior → *shift* de cero a derecha la cantidad de veces la $m\ _{par\ recpetor}$.
3. Dirección **R** resultante → operación OR entre el punto 2 y dirección **R** del *par emisor*.
4. $m_{mezcla} = m\ _{par\ emisor} + m - (mi\ _{par\ receptor} - 1)$
5. $mi_{mezcla} = m\ _{par\ emisor} + 1 + (mi - mi\ _{par\ receptor})$

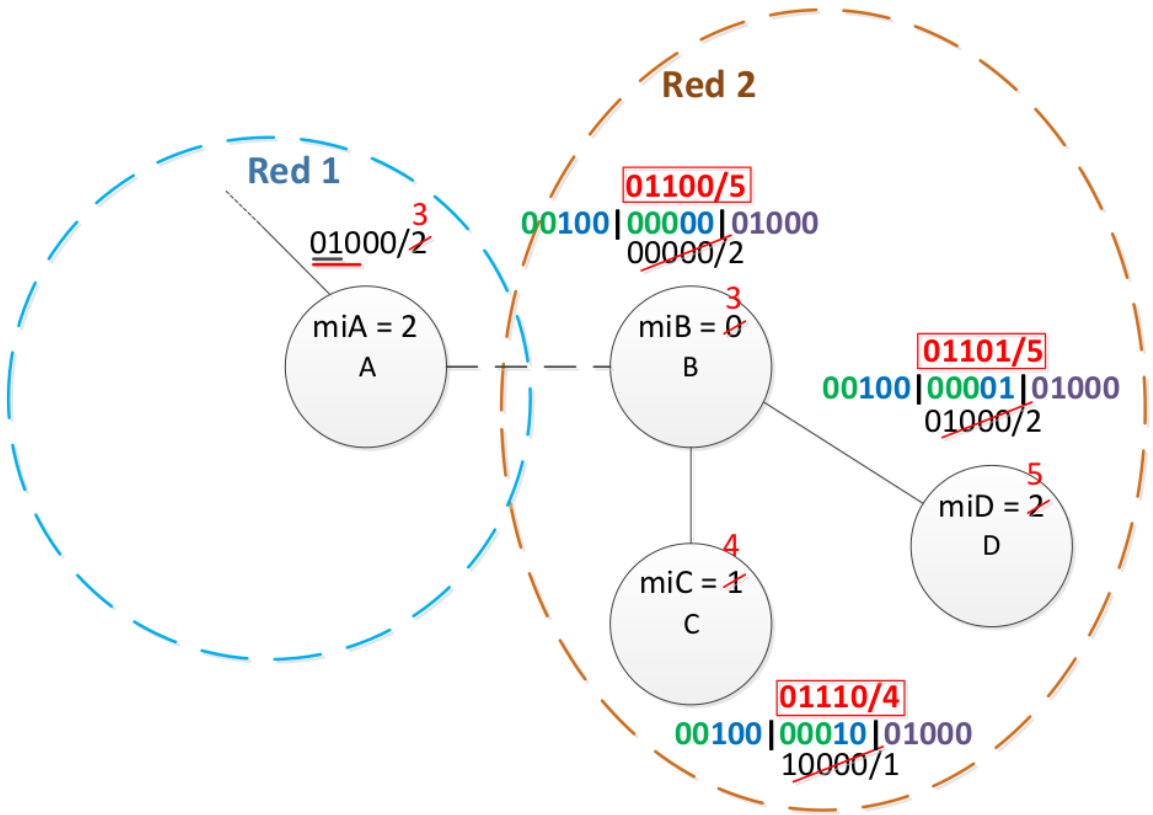


Tratamiento Caso 2

1. De la dirección de todos los bits en cero → set en 1 el bit mas significativo.
2. Del punto 1 → *shift* de cero a derecha la cantidad de $m_{\ par\ emisor}$.
3. De la dirección **R** → *shift* de cero a derecha la cantidad de $m_{\ par\ receptor}$.
4. Dirección **R** resultante → operación OR entre el punto 2, 3 y la dirección **R** del *par emisor*.
5. La m se obtiene ídem caso 1.
6. La mi se obtiene ídem caso 1.

Tratamiento Caso 2

1. De la dirección de todos los bits en cero → set en 1 el bit mas significativo.
2. Del punto 1 → *shift* de cero a derecha la cantidad de $m_{\ par\ emisor}$.
3. De la dirección **R** → *shift* de cero a derecha la cantidad de $m_{\ par\ receptor}$.
4. Dirección **R** resultante → operación OR entre el punto 2, 3 y la dirección **R** del *par emisor*.
5. La m se obtiene ídem caso 1.
6. La mi se obtiene ídem caso 1.



Tratamiento

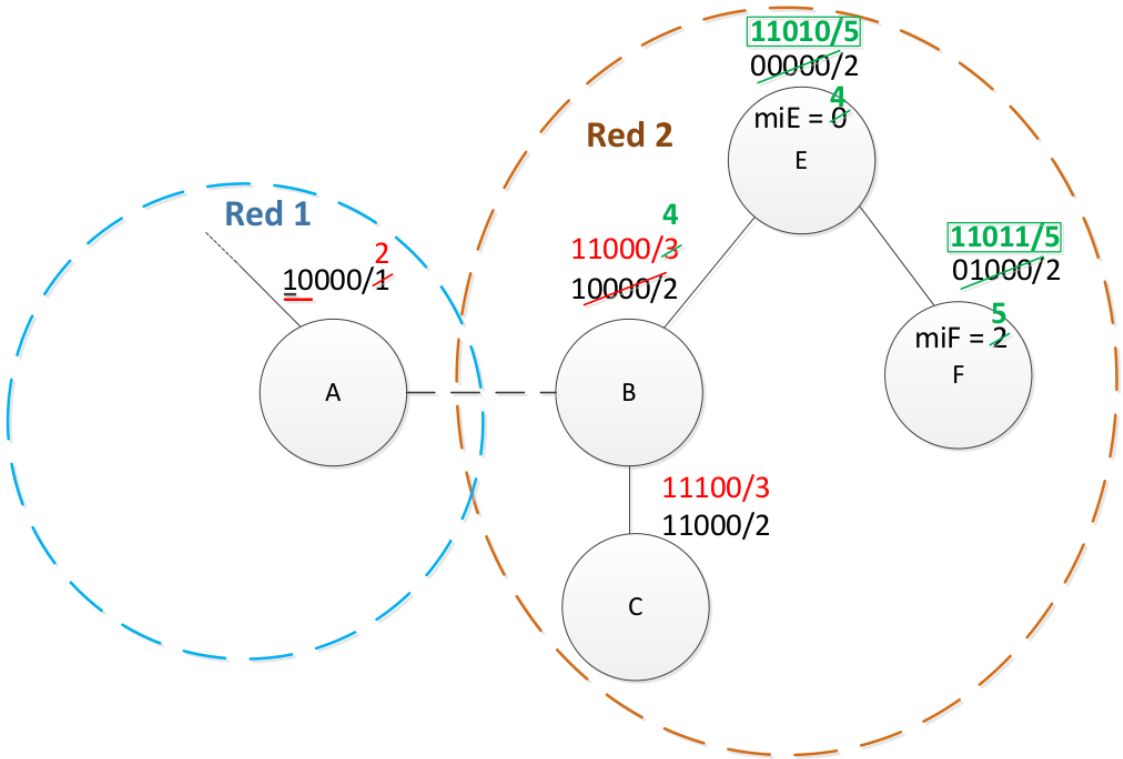
1. De la dirección **R** y m resultante del mecanismo del nodo predecesor de la rama, se asigna la siguiente dirección **R** y m del espacio de direcciones.

2. La mi se calcula sumando la m del nodo predecesor de la rama, mas uno.

Tratamiento

1. De la dirección **R** y m resultante del mecanismo del nodo predecesor de la rama, se asigna la siguiente dirección **R** y m del espacio de direcciones.

2. La mi se calcula sumando la m del nodo predecesor de la rama, mas uno.

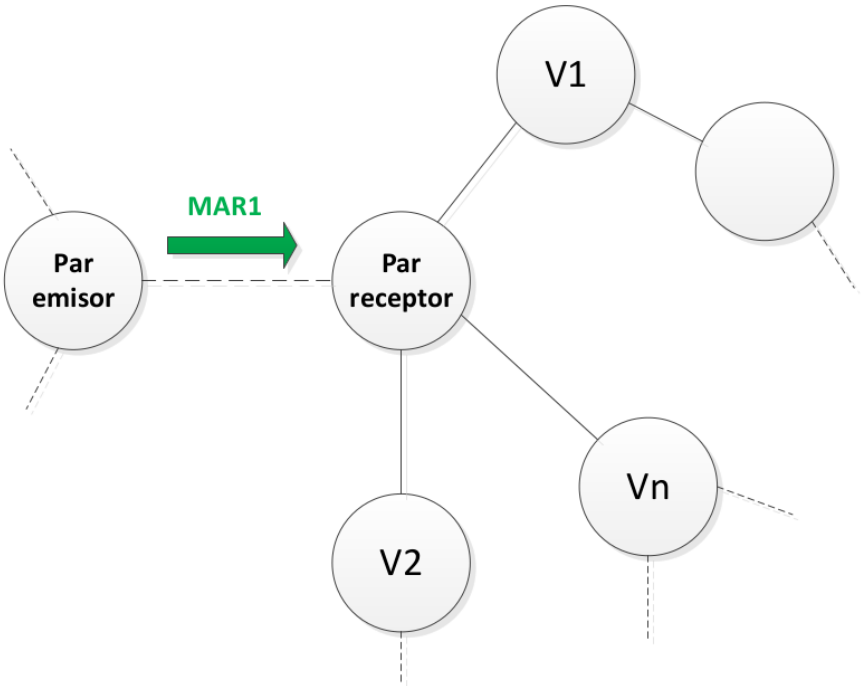


Paquetes de control

- MAR1 *(Mix Address Request 1)* → Transporta información del mecanismo 1.
- MAR2 *(Mix Address Request 2)* → Transporta información del mecanismo 2.
- MAN1 *(Mix Address Notification 1)* → Confirmación del mensaje MAR1.
- MAN2 *(Mix Address Notification 2)* → Confirmación del mensaje MAR2.

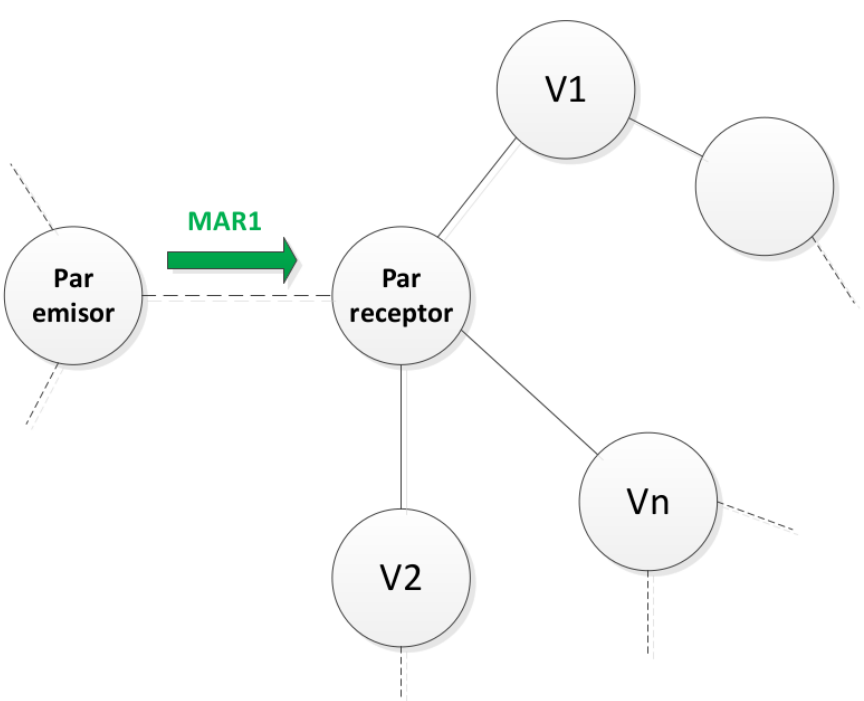
Paquetes de control

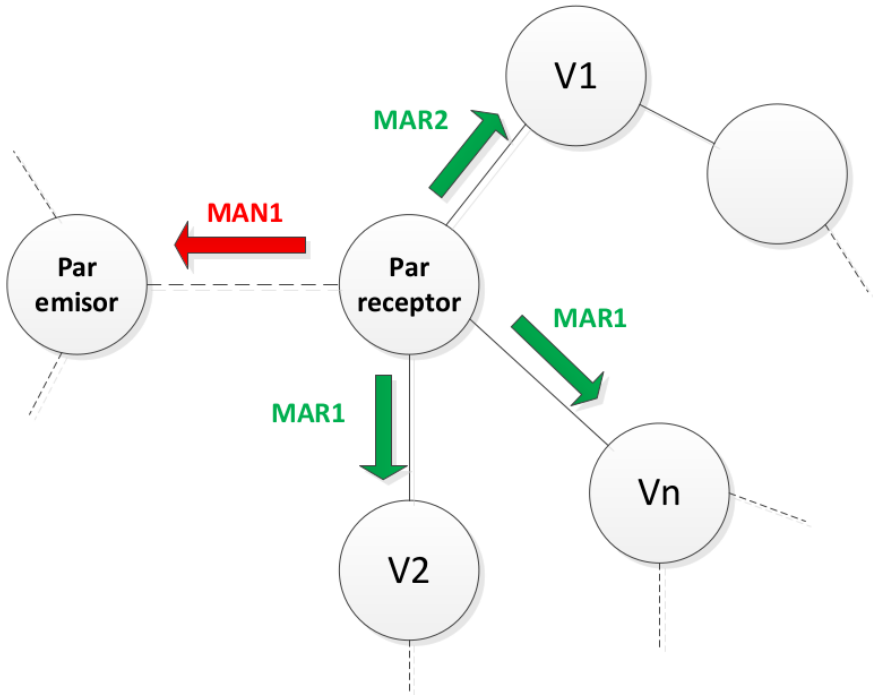
- MAR1 *(Mix Address Request 1)* → Transporta información del mecanismo 1.
- MAR2 *(Mix Address Request 2)* → Transporta información del mecanismo 2.
- MAN1 *(Mix Address Notification 1)* → Confirmación del mensaje MAR1.
- MAN2 *(Mix Address Notification 2)* → Confirmación del mensaje MAR2.



Paquetes de control

- MAR1 *(Mix Address Request 1)* → Transporta información del mecanismo 1.
- MAR2 *(Mix Address Request 2)* → Transporta información del mecanismo 2.
- MAN1 *(Mix Address Notification 1)* → Confirmación del mensaje MAR1.
- MAN2 *(Mix Address Notification 2)* → Confirmación del mensaje MAR2.





Trabajos anteriores

1. Implementación ANTop ruteo reactivo → Desarrollo en núcleo Linux 2.6.

Trabajos anteriores

1. Implementación ANTop ruteo reactivo → Desarrollo en núcleo Linux 2.6.

Aportes

1. Adaptación en un SO núcleo Linux 4.10.0.
2. Correcciones en el campo de:
   - Direccionamiento IPv6.
   - Gestión de direcciones secundarias.
   - Gestión de direcciones recuperadas.
   - Ruteo reactivo.
   - Servicio de resolución de nombres *Rendez Vous*.
3. Programación en C los algoritmos de fragmentación y mezcla.

Temario

Simulaciones

- ¿Por qué necesitamos simular?

Simulaciones

- ¿Por qué necesitamos simular?
- Mininet (http://mininet.org/).

Simulaciones

- ¿Por qué necesitamos simular?
- Mininet (http://mininet.org/).
- Simulación de redes aleatorias.

Simulaciones

- ¿Por qué necesitamos simular?
- Mininet (http://mininet.org/).
- Simulación de redes aleatorias.
- Simulación de la fragmentación de una red.

Simulaciones

- ¿Por qué necesitamos simular?
- Mininet (http://mininet.org/).
- Simulación de redes aleatorias.
- Simulación de la fragmentación de una red.
- Simulación de la mezcla de dos redes.

1. Herramienta de simulación de redes cableadas.
2. Nodo $\rightarrow$ maquina virtual.
3. Sistema operativo: Linux.
4. Programación: Python.

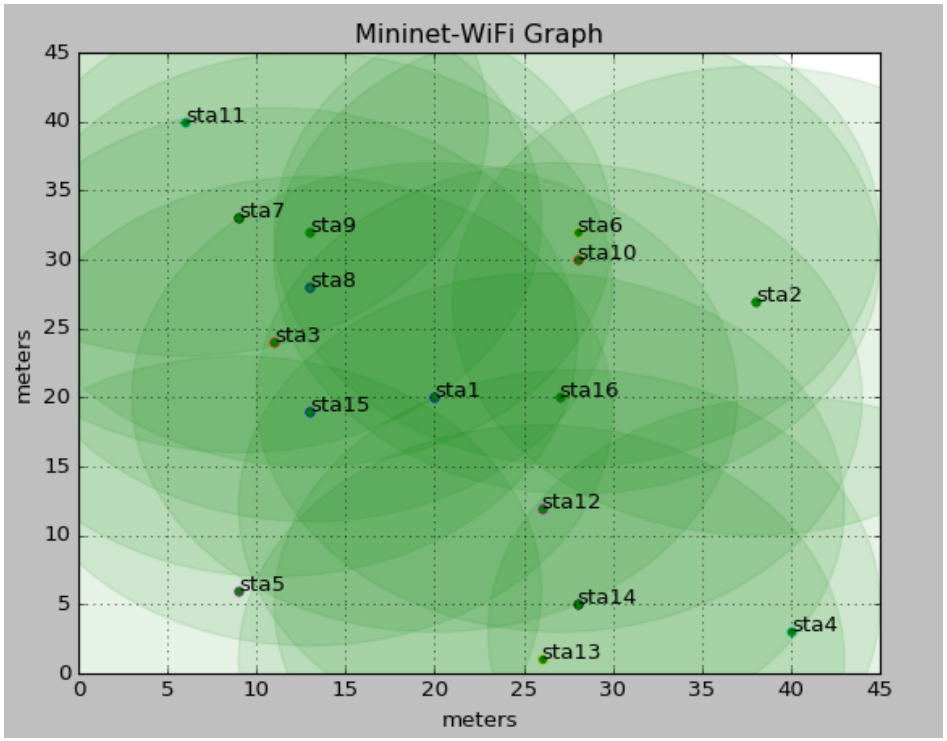
1. Herramienta de simulación de redes cableadas.
2. Nodo → maquina virtual.
3. Sistema operativo: Linux.
4. Programación: Python.

Mininet Wifi

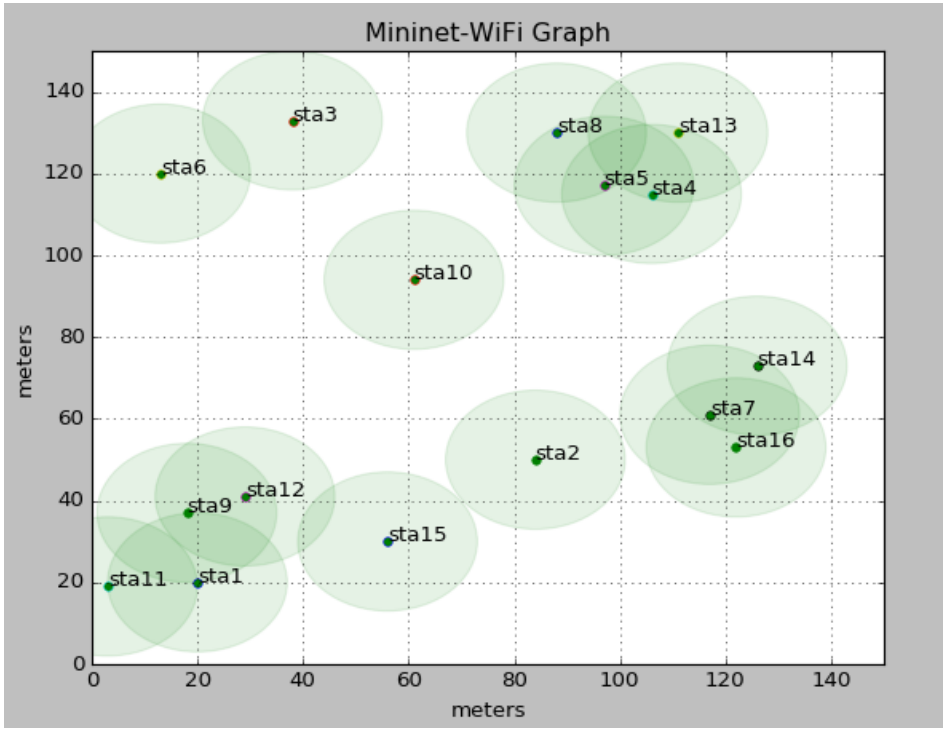
1. *fork* del emulador de red Mininet.
2. Funcionalidades ampliadas a entornos de redes Wireless.
3. Se agregan:
   - Estaciones Wifi virtualizadas.
   - Puntos de accesos → Usa controladores inalámbricos estándar de Linux.
4. Cada nodo tiene posición y movilidad.

- Dos modos de simulación: Nodos cercanos y Nodos lejanos.
- 16 nodos máximos en cada simulación.
- 10 simulaciones en cada modo.

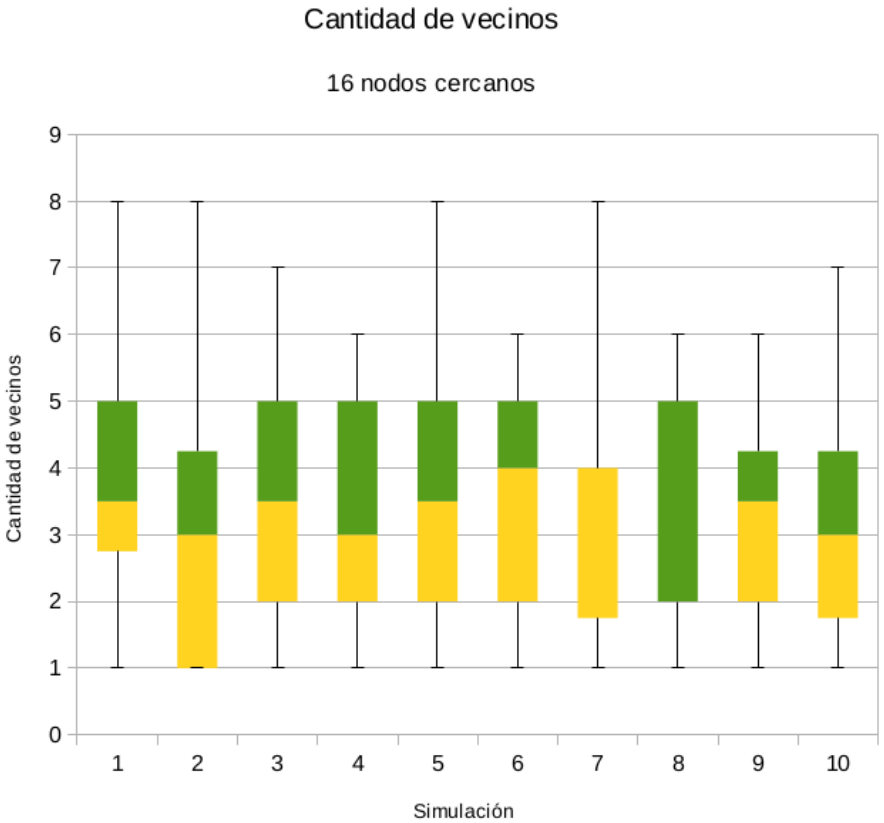
Nodos cercanos

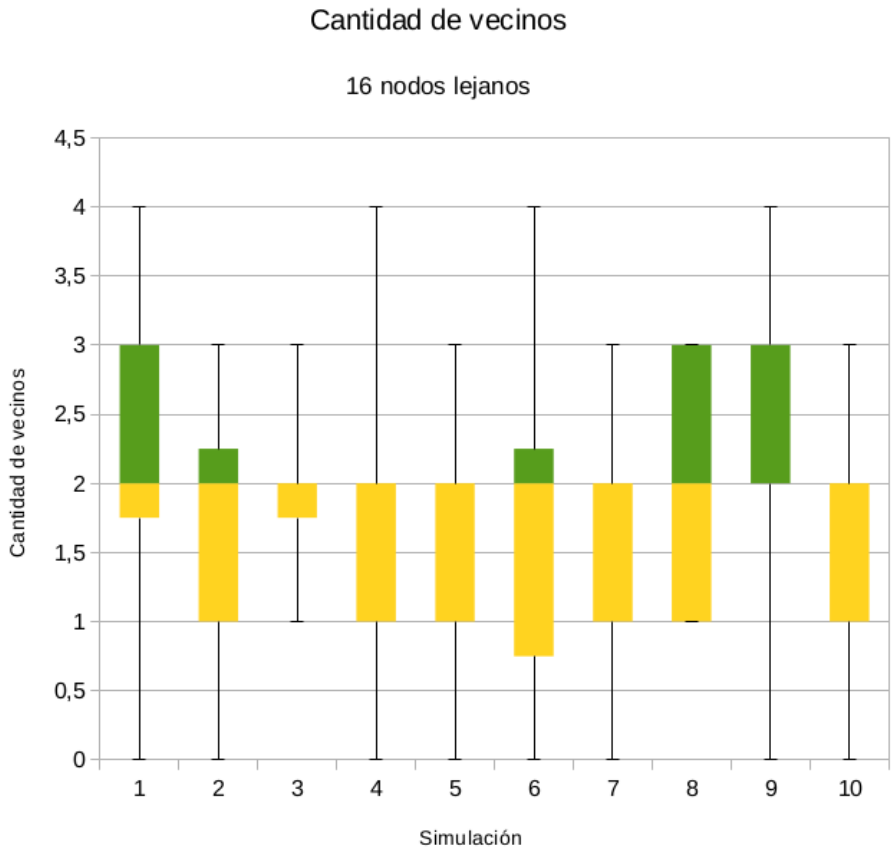


Nodos lejanos

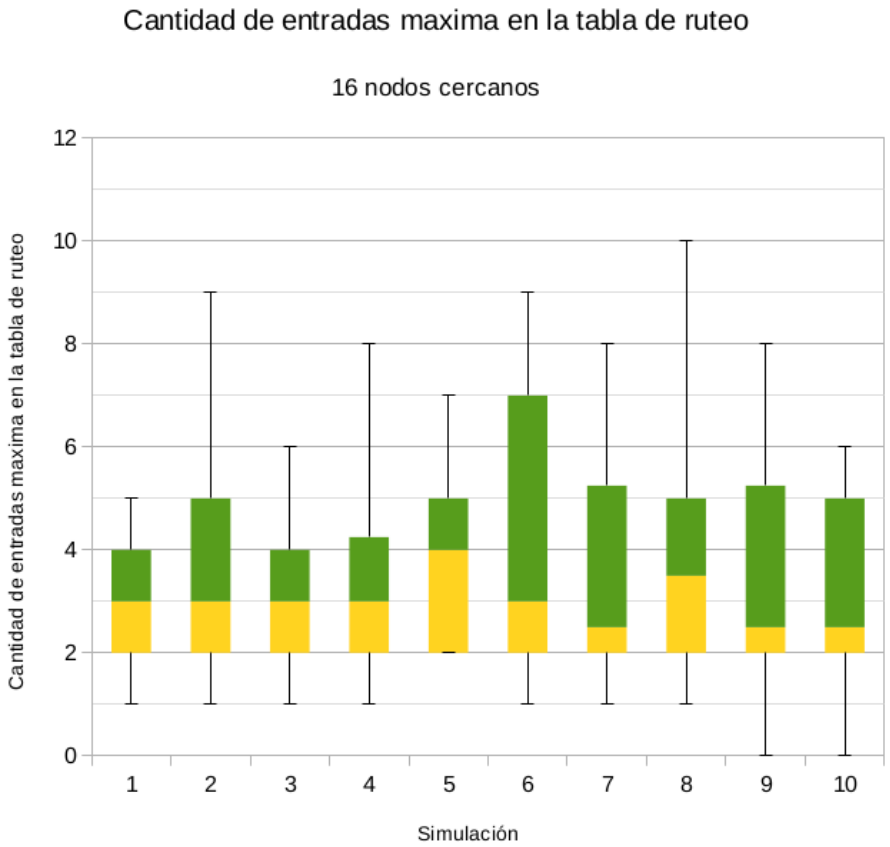


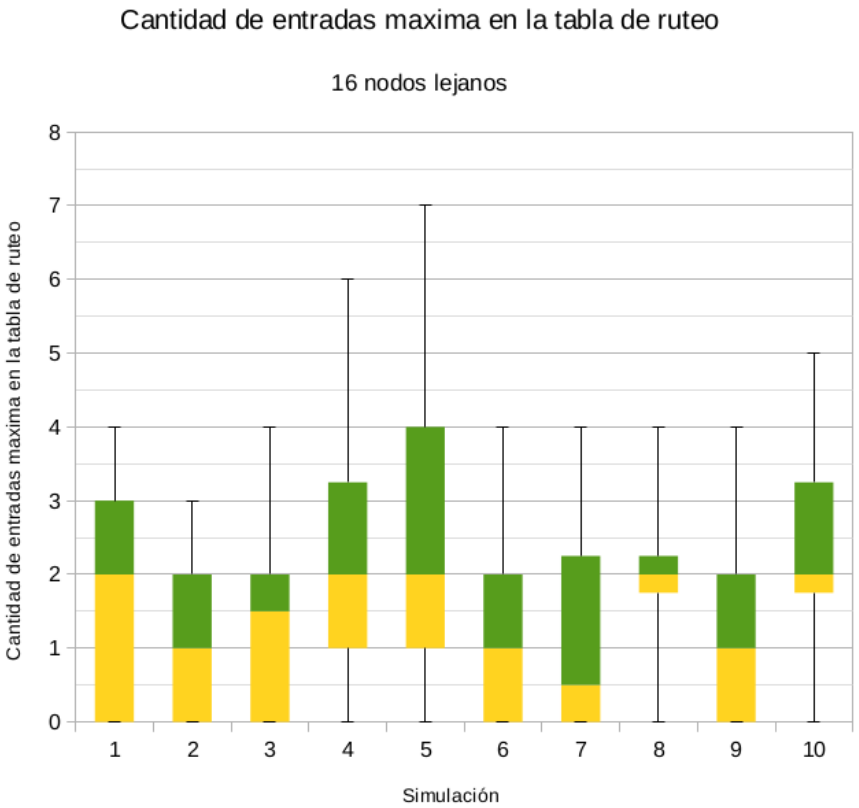
Cantidad de vecinos



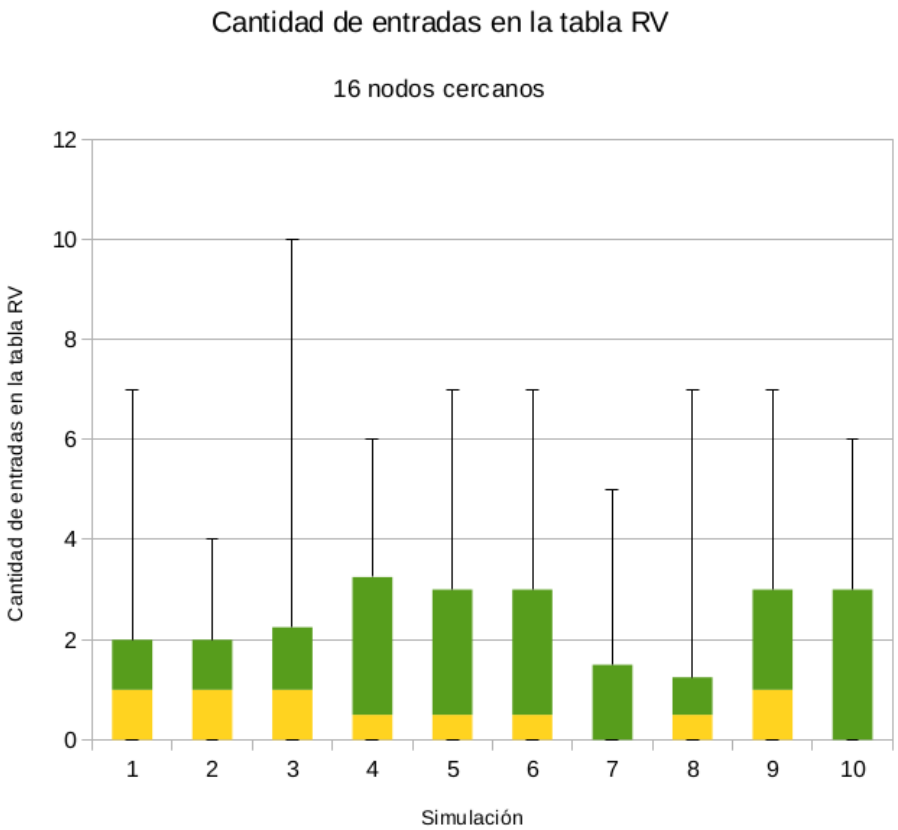


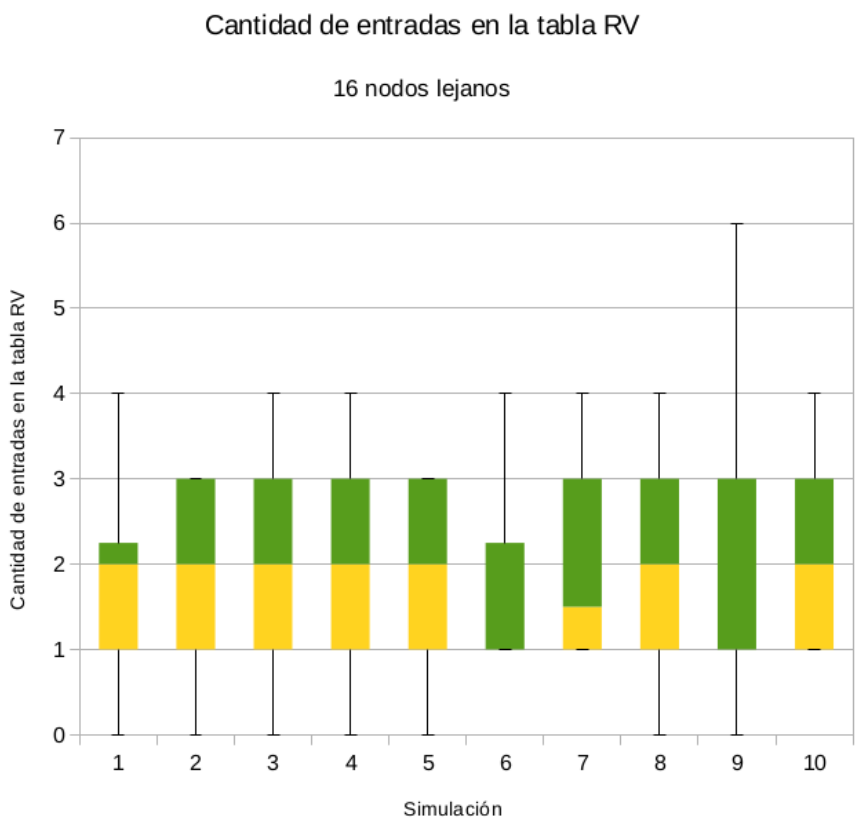
Cantidad de entradas máximas en la tabla de ruteo





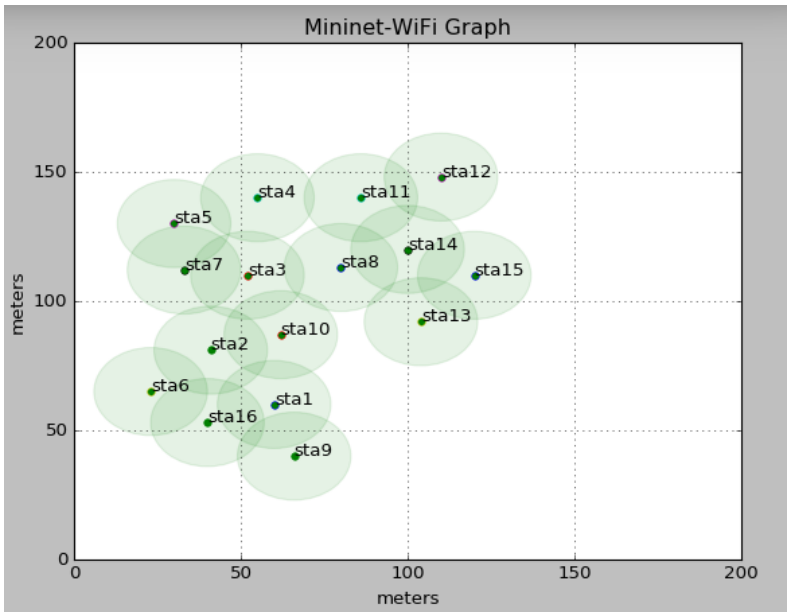
Cantidad de entradas en la tabla de *Rendez Vous*





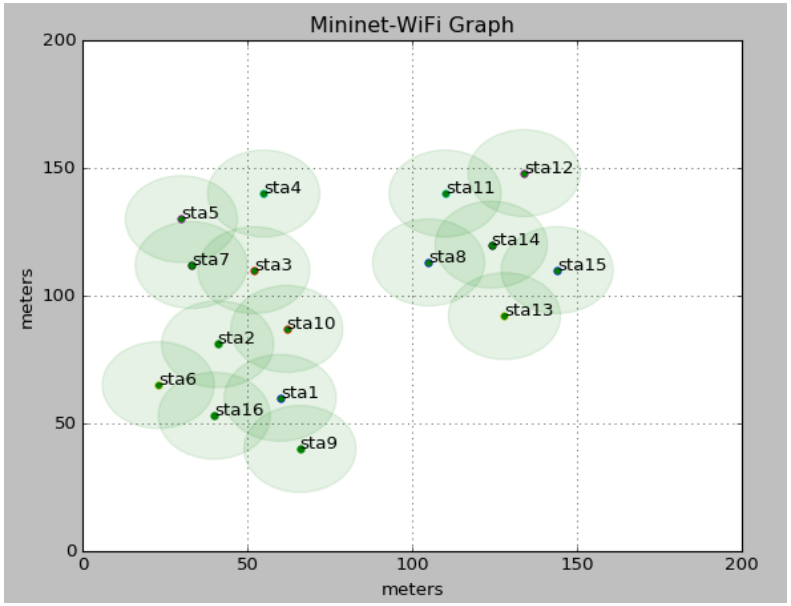
- 16 nodos en total.
- Configuración de posicionamiento de nodos formando una red.
- Movilidad en el tiempo de una porción de la red → Fragmentación.
- Capturas de paquetes de control FAR y FAN → `Wireshark`.
- Registro de direcciones **R** de los nodos antes y después de la fragmentación.
- Corroborar conectividad entre distintos nodos de la red fragmentada. → Consistencia del protocolo.

Tiempo de simulación= 0



Nodo	Dirección primaria	Dirección secundaria	Nº red
sta1	2001:db8:**0:0:0:0:0:1**	-	0
sta2	2001:db8:0:0:**8000:0:0:1**	-	0
sta3	2001:db8:0:0:**c000:0:0:1**	-	0
sta4	2001:db8:0:0:**e000:0:0:1**	2001:db8:0:0:**f000:0:0:1**	0
sta5	2001:db8:0:0:**d000:0:0:1**	-	0
sta6	2001:db8:0:0:**a000:0:0:1**	-	0
sta7	2001:db8:0:0:**9000:0:0:1**	-	0
sta8	2001:db8:0:0:**c800:0:0:1**	-	0
sta9	2001:db8:0:0:**4000:0:0:1**	-	0
sta10	2001:db8:0:0:**2000:0:0:1**	-	0
sta11	2001:db8:0:0:**cc00:0:0:1**	2001:db8:0:0:**cd00:0:0:1**	0
sta12	2001:db8:0:0:**ce00:0:0:1**	-	0
sta13	2001:db8:0:0:**ca00:0:0:1**	2001:db8:0:0:**cb00:0:0:1**	0
sta14	2001:db8:0:0:**c900:0:0:1**	-	0
sta15	2001:db8:0:0:**c980:0:0:1**	-	0
sta16	2001:db8:0:0:**6000:0:0:1**	-	0

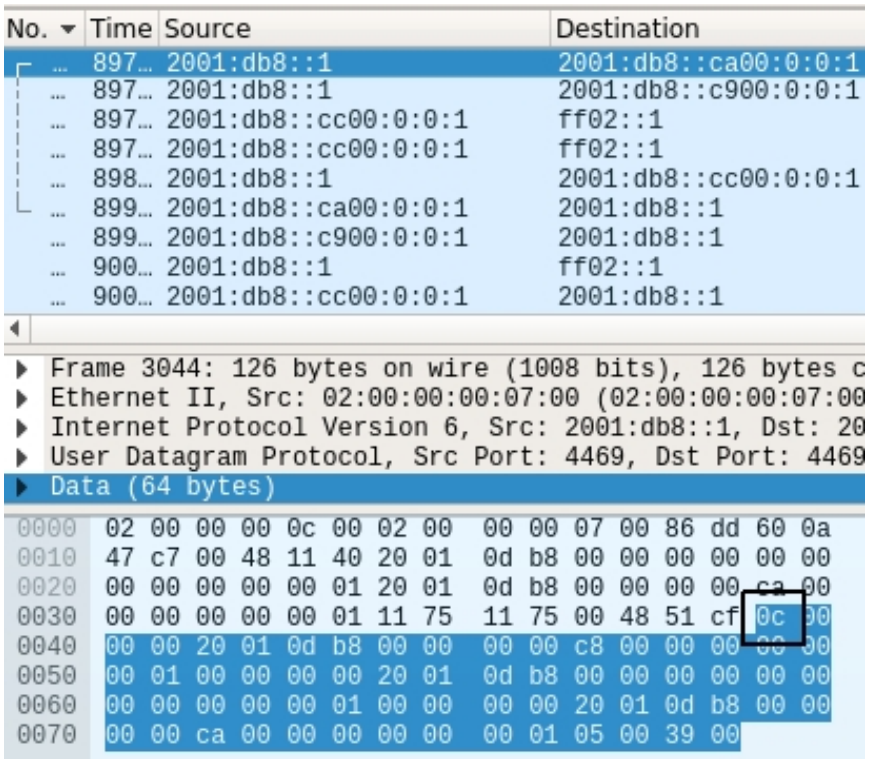
Tiempo de simulación=300s

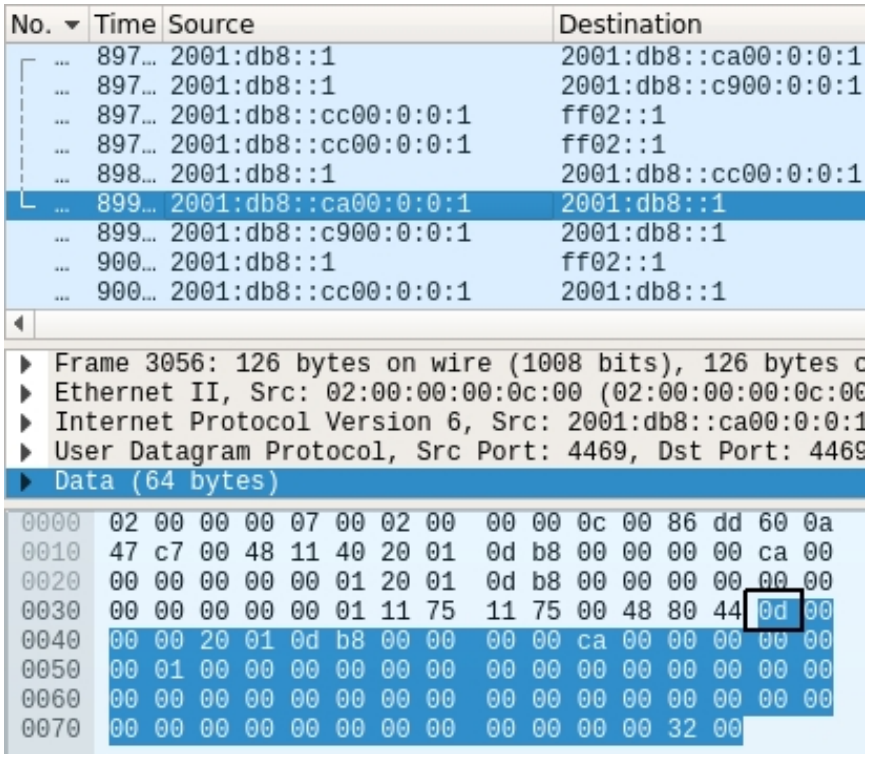


Red 1			
Nodo	Dirección primaria	Dirección secundaria	Nº red
sta1	2001:db8:0:0:**0:0:0:1**	-	0
sta2	2001:db8:0:0:**8000:0:0:1**	-	0
sta3	2001:db8:0:0:**c000:0:0:1**	-	0
sta4	2001:db8:0:0:**e000:0:0:1**	2001:db8:0:0:**f000:0:0:1**	0
sta5	2001:db8:0:0:**d000:0:0:1**	-	0
sta6	2001:db8:0:0:**a000:0:0:1**	-	0
sta7	2001:db8:0:0:**9000:0:0:1**	-	0
sta9	2001:db8:0:0:**4000:0:0:1**	-	0
sta10	2001:db8:0:0:**2000:0:0:1**	-	0
sta16	2001:db8:0:0:**6000:0:0:1**	-	0

Red 2			
Nodo	Dirección primaria	Dirección secundaria	Nº red
sta8	2001:db8:0:0:**0:0:0:1**	-	7
sta11	2001:db8:0:0:**8000:0:0:1**	2001:db8:0:0:**a000:0:0:1**	7
sta12	2001:db8:0:0:**c000:0:0:1**	-	7
sta13	2001:db8:0:0:**4000:0:0:1**	2001:db8:0:0:**6000:0:0:1**	7
sta14	2001:db8:0:0:**2000:0:0:1**	-	7
sta15	2001:db8:0:0:**3000:0:0:1**	-	7

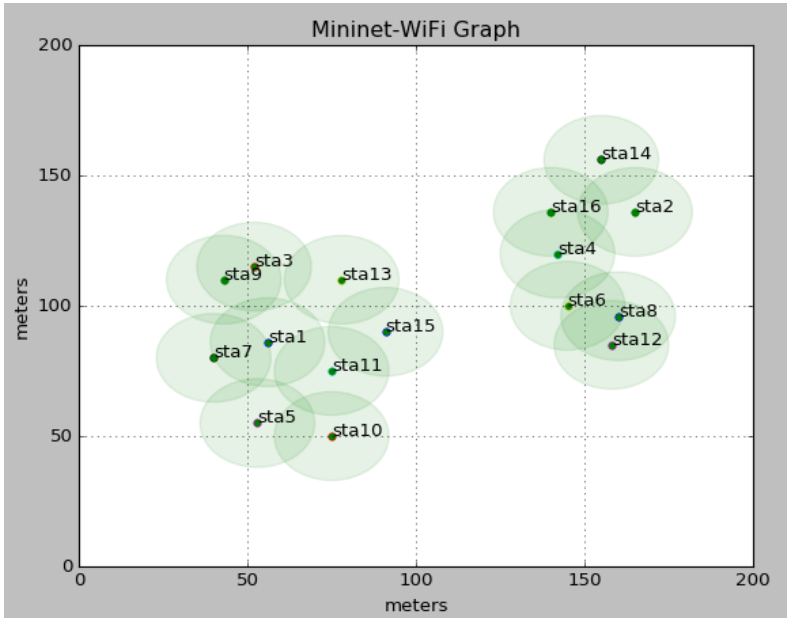
Captura de paquetes de control





- 16 nodos en total.
- Configuración de posicionamiento de nodos formando dos redes.
- Movilidad en el tiempo de una de las redes → Mezcla.
- Capturas de paquetes de control MAR1, MAR2, MAN1 y MAN2 → `Wireshark`.
- Registro de direcciones **R** de los nodos antes y después de la mezcla.
- Corroborar conectividad entre distintos nodos de la red mezclada. → Consistencia del protocolo.

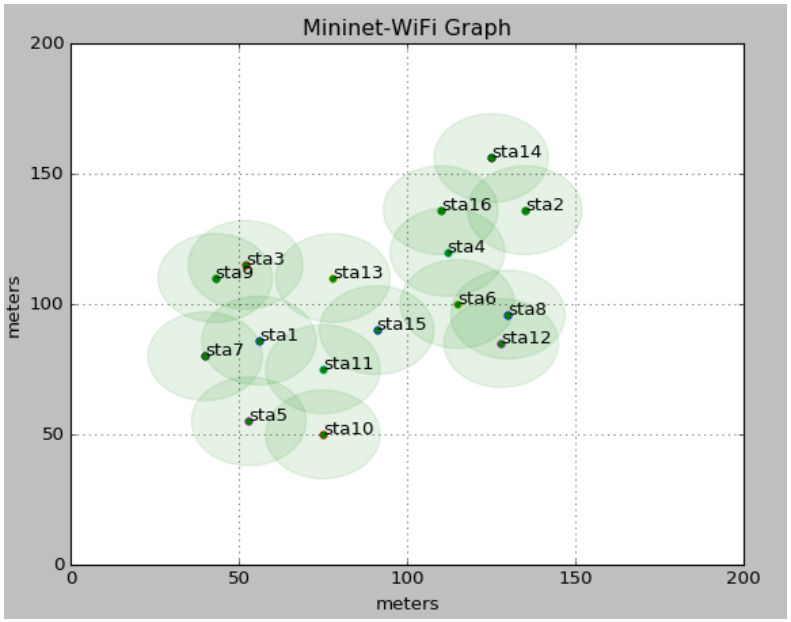
Tiempo de simulación=0



Red 1			
Nodo	Dirección primaria	Dirección secundaria	Nº red
sta1	2001:db8:0:0:**0:0:0:1**	2001:db8:0:0:**2000:0:0:1**	0
sta3	2001:db8:0:0:**8000:0:0:1**	-	0
sta5	2001:db8:0:0:**4000:0:0:1**	2001:db8:0:0:**4800:0:0:1**	0
sta7	2001:db8:0:0:**6000:0:0:1**	-	0
sta9	2001:db8:0:0:**c000:0:0:1**	2001:db8:0:0:**e000:0:0:1**	0
sta10	2001:db8:0:0:**5000:0:0:1**	-	0
sta11	2001:db8:0:0:**800:0:0:1**	-	0
sta13	2001:db8:0:0:**a000:0:0:1**	-	0
sta15	2001:db8:0:0:**b000:0:0:1**	-	0

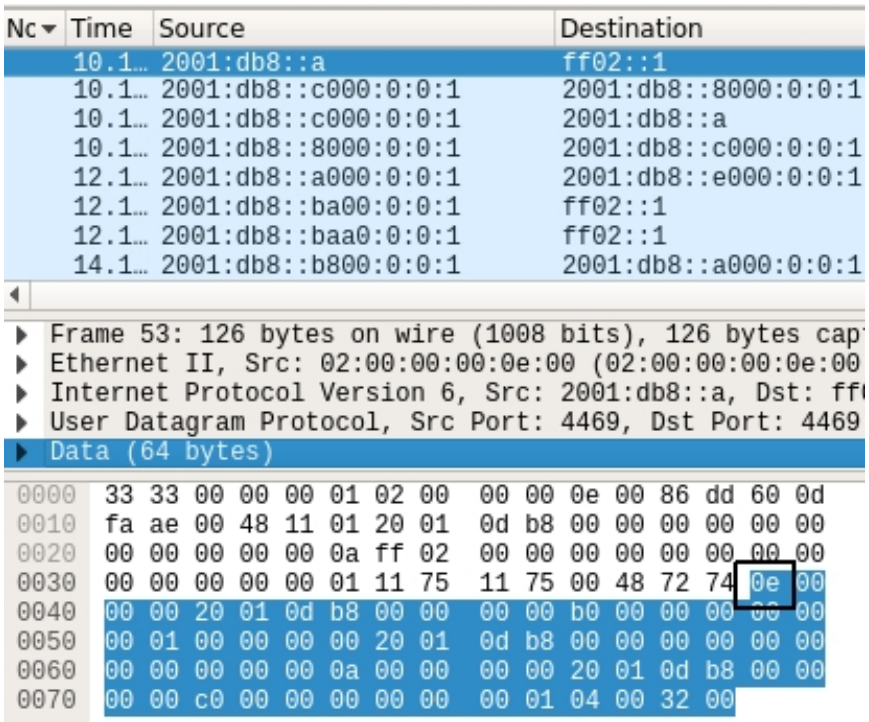
Red 2			
Nodo	Dirección primaria	Dirección secundaria	Nº red
sta2	2001:db8:0:0:**0:0:0:1**	-	1
sta4	2001:db8:0:0:**8000:0:0:1**	-	1
sta6	2001:db8:0:0:**c000:0:0:1**	2001:db8:0:0:**e000:0:0:1**	1
sta8	2001:db8:0:0:**a000:0:0:1**	-	1
sta12	2001:db8:0:0:**b000:0:0:1**	-	1
sta14	2001:db8:0:0:**4000:0:0:1**	2001:db8:0:0:**6000:0:0:1**	1
sta16	2001:db8:0:0:**2000:0:0:1**	-	1

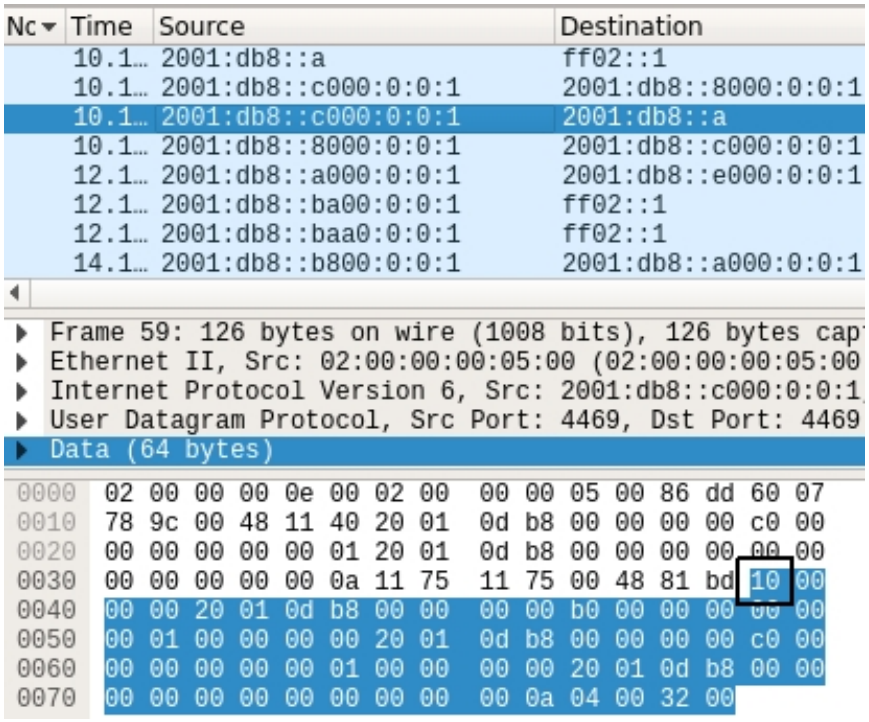
Tiempo de simulación= 200s



Red 1			
Nodo	**Dirección primaria**	**Dirección secundaria**	**Nº red**
sta1	2001:db8:0:0:**0:0:0:1**	2001:db8:0:0:**2000:0:0:1**	0
sta2	2001:db8:0:0:**b980:0:0:1**	-	0
sta3	2001:db8:0:0:**8000:0:0:1**	-	0
sta4	2001:db8:0:0:**b900:0:0:1**	2001:db8:0:0:**b920:0:0:1**	0
sta5	2001:db8:0:0:**4000:0:0:1**	2001:db8:0:0:**4800:0:0:1**	0
sta6	2001:db8:0:0:**b800:0:0:1**	-	0
sta7	2001:db8:0:0:**6000:0:0:1**	-	0
sta8	2001:db8:0:0:**b940:0:0:1**	-	0
sta9	2001:db8:0:0:**c000:0:0:1**	2001:db8:0:0:**e000:0:0:1**	0
sta10	2001:db8:0:0:**5000:0:0:1**	-	0
sta11	2001:db8:0:0:**800:0:0:1**	-	0
sta12	2001:db8:0:0:**b950:0:0:1**	-	0
sta13	2001:db8:0:0:**a000:0:0:1**	-	0
sta14	2001:db8:0:0:**b9c0:0:0:1**	-	0
sta15	2001:db8:0:0:**b000:0:0:1**	-	0
sta16	2001:db8:0:0:**b9a0:0:0:1**	-	0

Captura de paquetes de control





Temario

Contribuciones de esta tesis

- Desarrollo de algoritmos de fragmentación y mezcla de redes ANTop.
- Mejoras en la implementación ANTop → Soporta fragmentación y mezcla.
- Verificación de funcionamiento → `Mininet Wifi` *(open source)*.
- Analisis de datos de las simulaciones.

Contribuciones de esta tesis

- Desarrollo de algoritmos de fragmentación y mezcla de redes ANTop.
- Mejoras en la implementación ANTop → Soporta fragmentación y mezcla.
- Verificación de funcionamiento → `Mininet Wifi` *(open source)*.
- Analisis de datos de las simulaciones.

Trabajos futuros

- Función *Hash* → Tamaño de tablas *Rendez Vous* pareja entre los servidores.
- Algoritmo de decisión de mezcla → Red que será renombrada es la que tenga menor cantidad de nodos.
- Múltiples fragmentaciones, mezclas.

GRACIAS