



**MOTOR DE RED**  
**SHIFT** videogame

GAMELAB  
UNIVERSIDAD DE OVIEDO

**Alberto Carlos del Blanco Maraña**  
Madrid 28 de junio de 2007



## Tabla de Contenidos

<b>1. Introducción</b> .....	<b>4</b>
<b>2. Parámetros de la clase <i>Network</i></b> .....	<b>6</b>
2.1. Introducción .....	6
2.2. Convenios iniciales .....	8
2.3. Tipos de tramas de Datos y Control.....	9
2.3.1. Trama “ <i>Vector de Estaciones</i> ” .....	9
2.3.2. Trama “ <i>Actualización de Datos Iniciales</i> ” .....	9
2.3.3. Trama “ <i>Desconexión del Anillo</i> ” .....	10
2.3.4. Trama “ <i>Petición de Conexión</i> ” .....	10
2.3.5. Trama “ <i>Petición de Desconexión</i> ” .....	11
2.3.6. Trama “ <i>Petición de Vector de Estaciones</i> ” .....	11
2.3.7. Trama “ <i>Asentimiento</i> ” .....	12
2.4. Tipos de Operaciones.....	13
2.4.1. Conexión de una estación (ya conectada o no) .....	13
2.4.2. Desconexión de una estación .....	14
2.4.3. Pérdida del vector del anillo.....	15
2.4.4. Retransmisión del Vector de Estaciones por el Anillo .....	16
2.4.5. Desconexión del anillo por el Administrador .....	17



## Lista de Figuras

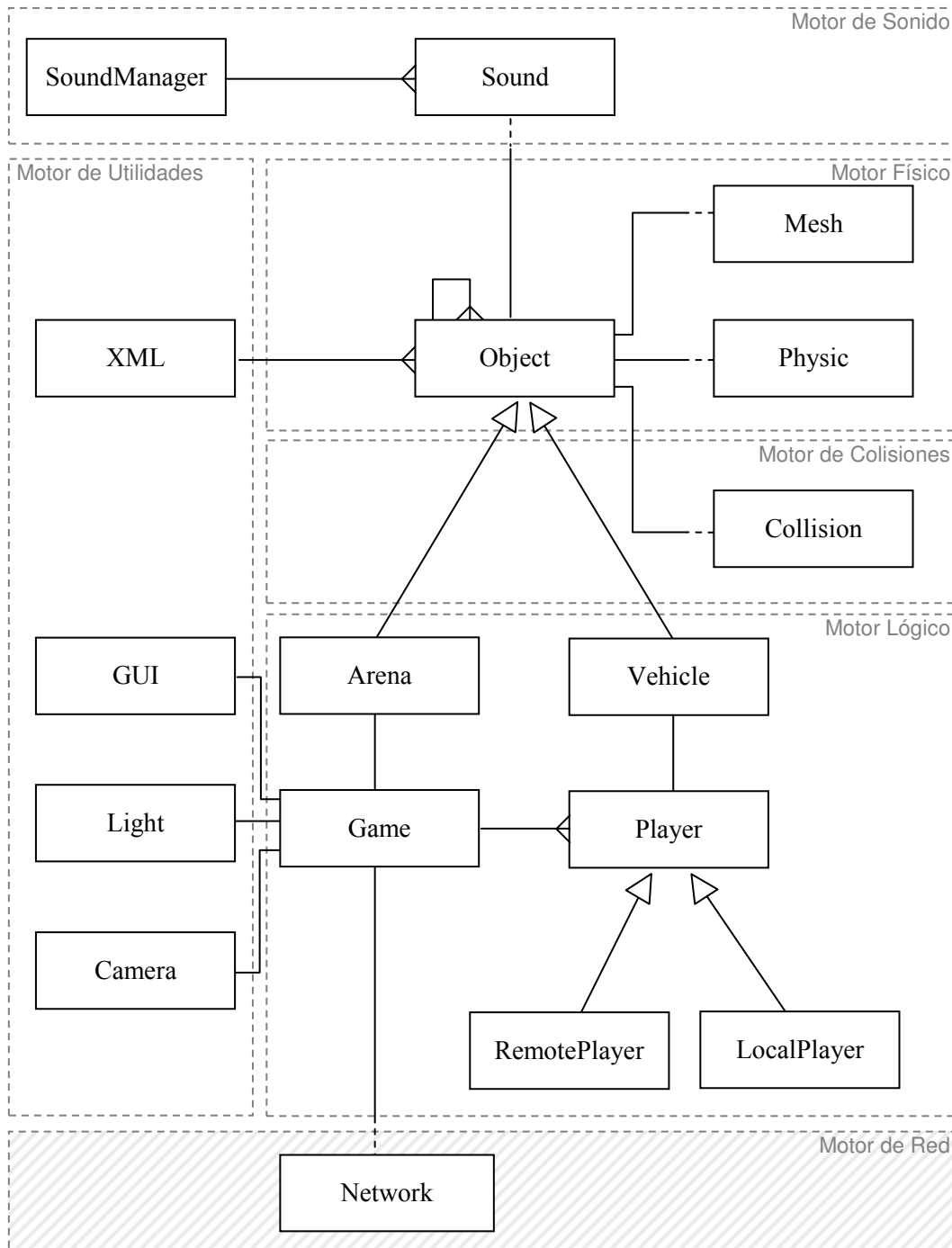
Imagen 1 : Límites del motor de red.....	4
Imagen 2 : Arquitectura de la red .....	7
Imagen 3 : Estructura de la trama .....	8
Imagen 4 : Trama “ <i>Vector de Estaciones</i> ” .....	9
Imagen 5 : Trama “ <i>Actualización de Datos Iniciales</i> ” .....	9
Imagen 6 : Trama “ <i>Desconexión del Anillo</i> ” .....	10
Imagen 7 : Trama “ <i>Petición de Conexión</i> ” .....	10
Imagen 8 : Trama “ <i>Petición de Desconexión</i> ” .....	11
Imagen 9 : Trama “ <i>Petición de Vector de Estaciones</i> ” .....	11
Imagen 10 : Trama “ <i>Asentimiento</i> ” .....	12
Imagen 11 : Conexión de una estación.....	13
Imagen 12 : Desconexión de una estación.....	14
Imagen 13 : Pérdida del vector del anillo (1) .....	15
Imagen 14 : Pérdida del vector del anillo (2) .....	15
Imagen 15 : Pérdida del vector del anillo (3) .....	16
Imagen 16 : Retransmisión del Vector de Estaciones por el Anillo.....	16
Imagen 17 : Desconexión del anillo por el Administrador.....	17

# 1. Introducción

El presente documento detalla el sistema de red del videojuego SHIFT, como parte del proyecto fin de carrera realizado por Alberto Carlos del Blanco para la E.P.S.I.G. de la Universidad de Oviedo.

El módulo de red permite compartir los parámetros necesarios que habiliten el modo multijugador.

Dentro del esquema general del motor gráfico, el módulo de red queda limitado por el ámbito rayado:



**Imagen 1 : Límites del motor de red**



Cómo refleja en el diagrama, el módulo de red queda implementado por la clase:

- **Network**  
Implementa la lógica de red necesaria para incorporar un protocolo sencillo de comunicación entre diferentes instancias de la aplicación, de forma que se pueda mantener una situación de escena global y situaciones locales que serán actualizadas continuamente.

Al igual que en otros módulos del videojuego, existen infinidad de librerías que permiten implementar protocolos para el mantenimiento de datos globales entre diferentes instancias de red. Sin embargo no hay que olvidar el carácter educativo de este proyecto, por lo que se prefiere implementar un protocolo ad-hoc a bajo nivel (utilizando únicamente librerías de sockets) conforme las necesidades del videojuego:

- Independiente del interfaz de red, para ello se abstrae la comunicación mediante sockets.
- Ad-hoc a las necesidades del videojuego, y especialmente integrado con la finalidad a la que va a estar dedicado.
- Protocolo sencillo y sin mucha sobrecarga en la red, lo que permitirá mayor capacidad de refresco de las escenas locales, mayor cantidad de usuarios simultáneos, mayor realismo, jugabilidad, etc.

De hecho, el protocolo aquí implementado será esencial para la capacidad multijugador del videojuego. En los siguientes apartados se darán más detalles sobre este protocolo.



## 2. Parámetros de la clase *Network*

### 2.1. Introducción

El objetivo consiste en el diseño de un protocolo sencillo para compartir datos globales entre varias instancias de red. Para ello se tomará una **estructura lógica de anillo**, dónde se transmitirá, junto con el testigo del anillo, la información global de la escena que sea necesaria. En cada momento, cada estación que reciba el anillo procederá como sigue:

- Actualizará sus datos locales del resto de jugadores con la información global que recibe
- Actualizará los datos globales de él mismo con los datos locales que tiene
- Pasará el testigo a la siguiente estación del anillo

En la red se distinguirán dos tipos de procesos, un **proceso administrador** que se encarga de la administración del anillo y varios **procesos estaciones** que lo forman. Lógicamente, desde el punto de vista del videojuego:

- El que crea la partida multijugador arranca un proceso administrador y un proceso estación, es decir actuará como servidor y cliente al mismo tiempo
- El resto de jugadores que se unan a la partida, arrancarán sólo un proceso estación que se unirá al anillo por medio del proceso administrador del equipo que actúa como servidor.

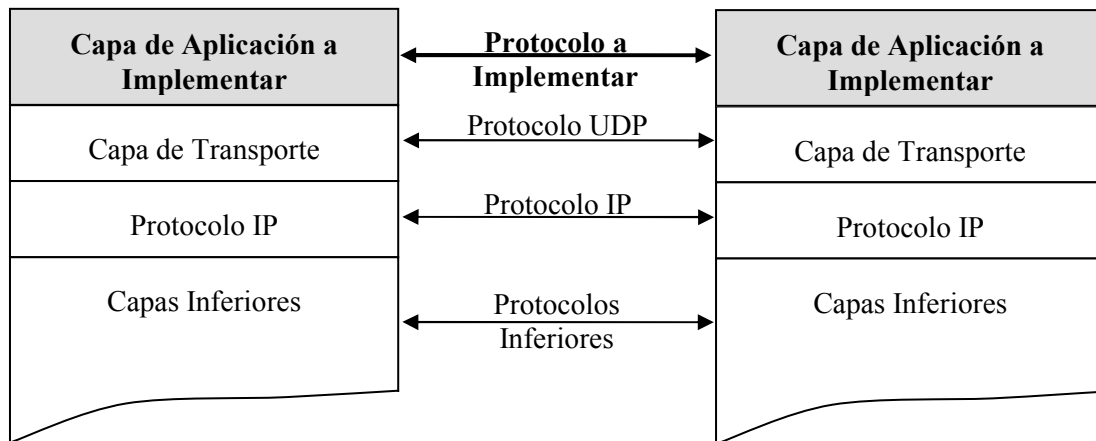
Este planteamiento implica por tanto el diseño de un protocolo de comunicaciones que ofrezca los servicios básicos para permitir una comunicación transparente y fiable en el anillo.

Existen varias posibilidades para hacerlo, pero en este caso se han seguido los siguientes criterios:

- Se descarta usar una librería de comunicaciones para videojuegos (como por ejemplo las que incluye DirectX) por el carácter educativo del proyecto.
- Se descarta implementar las comunicaciones a nivel de aplicación TCP, ya que de cara a los videojuegos es más común usar el nivel de transporte UDP. Esto complicará el protocolo, ya que obliga a implementar mecanismos que aseguren la fiabilidad en la transmisión.
- Se descarta multiplexar las transferencias en varios puertos, uno de control y otro de datos, por no complicar más el protocolo y por no usar más puertos de cara a la aplicación.



Por lo tanto la aplicación se va a construir haciendo uso de la arquitectura TCP/IP, partiendo de los servicios del protocolo de transporte UDP. Este protocolo dispone una comunicación extremo a extremo multiplexada mediante puertos, pero no proporciona mecanismos de acuse de recibo, secuenciación de mensajes, control de flujo, detección de errores, etc. Por tanto, el protocolo que se diseñe (que comunicará las capas de aplicación) debe disponer de ciertos servicios para ofrecer un nivel razonable de fiabilidad en la conexión.



**Imagen 2 : Arquitectura de la red**

En los siguientes apartados, se van a describir brevemente los aspectos más destacados del diseño del **protocolo** y de la **aplicación** que han sido implementados.



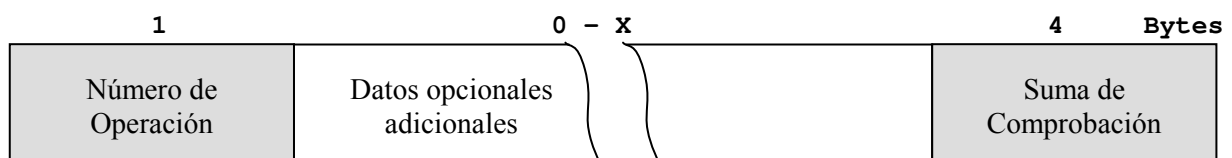
## 2.2. Convenios iniciales

De manera resumida, el protocolo diseñado parte de las siguientes características (se irán detallando en cada uno de los apartados posteriores):

- La **estructuración** de la comunicación se hará mediante tramas, tanto de datos como de control.
- Para la **transparencia de la comunicación** el protocolo estará orientado a carácter, utilizando si es preciso campo de longitud para las tramas de tamaño variable.
- Se supone una **comunicación bidireccional entre cada estación y el administrador**.
- Se supone una **comunicación unidireccional entre las estaciones**, ya que entre estos tipos de proceso se va a simular una red con estructura en anillo.
- Se consideran mecanismos de **control de transmisión**: las comunicaciones bidireccionales contarán con asentimientos, temporizadores y reenvíos, mientras que las comunicaciones unidireccionales considerarán únicamente temporizadores.
- Se consideran mecanismos de **detección de errores**: se incluye un campo (de un byte) con la suma de comprobación de cada trama, facilitando así el descarte de tramas defectuosas. El cálculo de este dato se realiza como la suma acumulativa de cada uno de los bytes de la trama.
- Se consideran mecanismos de **seguridad**: cada proceso sólo tratará los datos que reciba de aquellas direcciones (IP y puerto) que le puedan enviar ese tipo de datos.
- Se consideran mecanismos de **control de flujo**, buffereando la recepción de datos.
- El **número de estaciones** del anillo puede estar limitado por el tamaño de la trama UDP y por los temporizadores utilizados, en principio el anillo se limita a un máximo de 16 estaciones.
- Los **datos de conexión necesarios para cada estación** del anillo: dirección de su antecesor, dirección de su sucesor y dirección del proceso administrador. (el vector de equipos que circula por la red no es necesario para la conexión del anillo).
- Los **datos de conexión necesarios para el administrador**: las direcciones de todas las estaciones que forman el anillo.

Teniendo en cuenta estas consideraciones, la estructura genérica de una trama del protocolo constará de tres partes:

- Una cabecera de un byte que indica el tipo de trama
- Una zona opcional de datos con tamaño variable
- Cuatro bytes finales con la suma de comprobación del contenido de la trama.



**Imagen 3 : Estructura de la trama**

A continuación se comentará la estructura particular de las diferentes tramas de datos que se pueden recibir (en función del primer byte), tanto por el proceso administrador como por el proceso estación.



## 2.3. Tipos de tramas de Datos y Control

### 2.3.1. Trama “*Vector de Estaciones*”

Se trata de la trama de datos que circula alrededor del anillo, siendo además la única que circulará por el mismo. Contiene la información global de la escena y a continuación la dirección (IP y Puerto) y los datos propios de cada estación del anillo.

- **Proceso emisor:** estación o el administrador (cada vez que se pierde el vector)
- **Proceso receptor:** estación (solo será procesada si la envía la estación antecesora o el administrador)

Aspecto y tamaño de los datos de la trama:

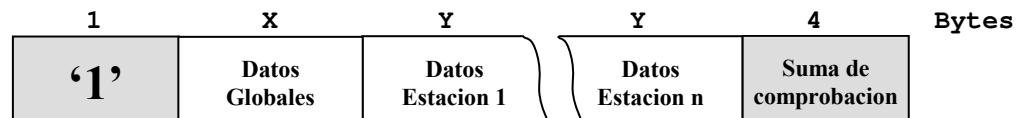


Imagen 4 : Trama “*Vector de Estaciones*”

### 2.3.2. Trama “*Actualización de Datos Iniciales*”

Se trata de una trama de datos que puede emitir el administrador para *actualizar la información de conexión que tiene cada estación en el anillo*: dirección del equipo antecesor y dirección del equipo posterior. La longitud es por tanto fija, ya que solo se envían estas dos direcciones (IP y puerto, cada bloque será de 16 bytes).

Estas tramas permiten por tanto la construcción dinámica del anillo por parte del administrador.

- **Proceso emisor:** administrador
- **Proceso receptor:** estación (solo será procesada si la envía el administrador)

Aspecto y tamaño de los datos de la trama:

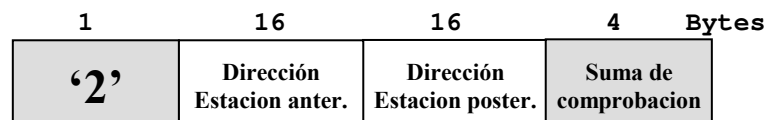


Imagen 5 : Trama “*Actualización de Datos Iniciales*”



### 2.3.3. Trama “Desconexión del Anillo”

Se trata de una trama de control que puede emitir el administrador para *obligar a una estación que abandone de manera forzosa el anillo*. Este hecho se utiliza para desconectar a todas las estaciones cuando el proceso administrador finaliza. Como no requiere datos adicionales, la trama tan sólo contiene el tipo y la suma de comprobación.

- **Proceso emisor:** administrador
- **Proceso receptor:** estación (solo será procesada si la envía el administrador)

Aspecto y tamaño de los datos de la trama:

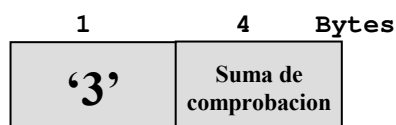


Imagen 6 : Trama “Desconexión del Anillo”

### 2.3.4. Trama “Petición de Conexión”

Se trata de una trama de control que puede emitir una *estación aún no conectada al anillo y que desea conectarse al mismo*. Al no requerir datos adicionales, la trama tan sólo contiene el tipo y el *checksum*.

- **Proceso emisor:** estación
- **Proceso receptor:** administrador (se procesará la trama desde cualquier dirección, aunque sea de una estación ya conectada al anillo)

Aspecto y tamaño de los datos de la trama:

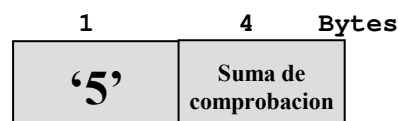


Imagen 7 : Trama “Petición de Conexión”



### 2.3.5. Trama “*Petición de Desconexión*”

Se trata de una trama de control que puede emitir una *estación conectada al anillo y que desea desconectarse del mismo*. En realidad un proceso estación puede desconectarse del anillo sin enviar esta trama (no afectará a la funcionalidad ya que el anillo se recompondrá dinámicamente). Al no requerir datos adicionales, la trama tan sólo contiene el tipo y el *checksum*.

- **Proceso emisor:** estación
- **Proceso receptor:** administrador (se procesará la trama sólo si la envía un quipo del anillo)

Aspecto y tamaño de los datos de la trama:

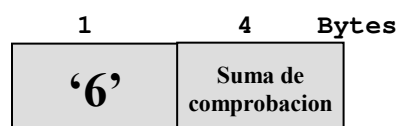


Imagen 8 : Trama “*Petición de Desconexión*”

### 2.3.6. Trama “*Petición de Vector de Estaciones*”

Se trata de una trama de control que puede emitir una *estación conectada al anillo y que reclama el vector de estaciones tras el vencimiento de su temporizador*. Al no requerir datos adicionales, la trama tan sólo contiene el tipo y el *checksum*.

- **Proceso emisor:** estación
- **Proceso receptor:** administrador (se procesará la trama sólo si la envía un quipo del anillo)

Aspecto y tamaño de los datos de la trama:

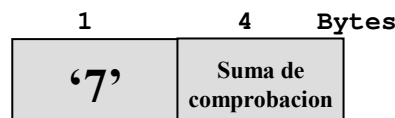


Imagen 9 : Trama “*Petición de Vector de Estaciones*”



### 2.3.7. Trama “Asentimiento”

Se trata de una *trama de control de transmisión para las comunicaciones bidireccionales* entre una estación y el administrador. No requiere datos adicionales, con lo que sólo tiene el tipo y el *checksum*.

- **Proceso emisor:** estación o administrador
- **Proceso receptor:** estación o administrador (sólo recibe por la dirección a la que hemos enviado las tramas 3, 6 o 7)

Aspecto y tamaño de los datos de la trama:

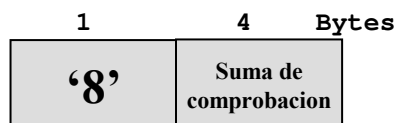


Imagen 10 : Trama “Asentimiento”



## 2.4. Tipos de Operaciones

A continuación se especificarán de forma simplificada las operaciones básicas del protocolo.

En general, en todas estas operaciones se realizan las comprobaciones de checksum, se disponen temporizadores para evitar bloqueos de procesos a la espera de respuestas y se comprueban direcciones de origen para evitar accesos no válidos. Además, todas las operaciones de control se repetirán un máximo de `MAX_INTENTOS` veces en caso que no se logren correctamente.

### 2.4.1. Conexión de una estación (ya conectada o no)

Suponiendo que la estación Y y la estación Z pertenecen al anillo y son consecutivas. Si una estación X se quiere conectar al anillo y además resulta que:  $Dirección Y < Dirección X < Dirección Z$

Entonces la conexión de la estación X será del siguiente modo:

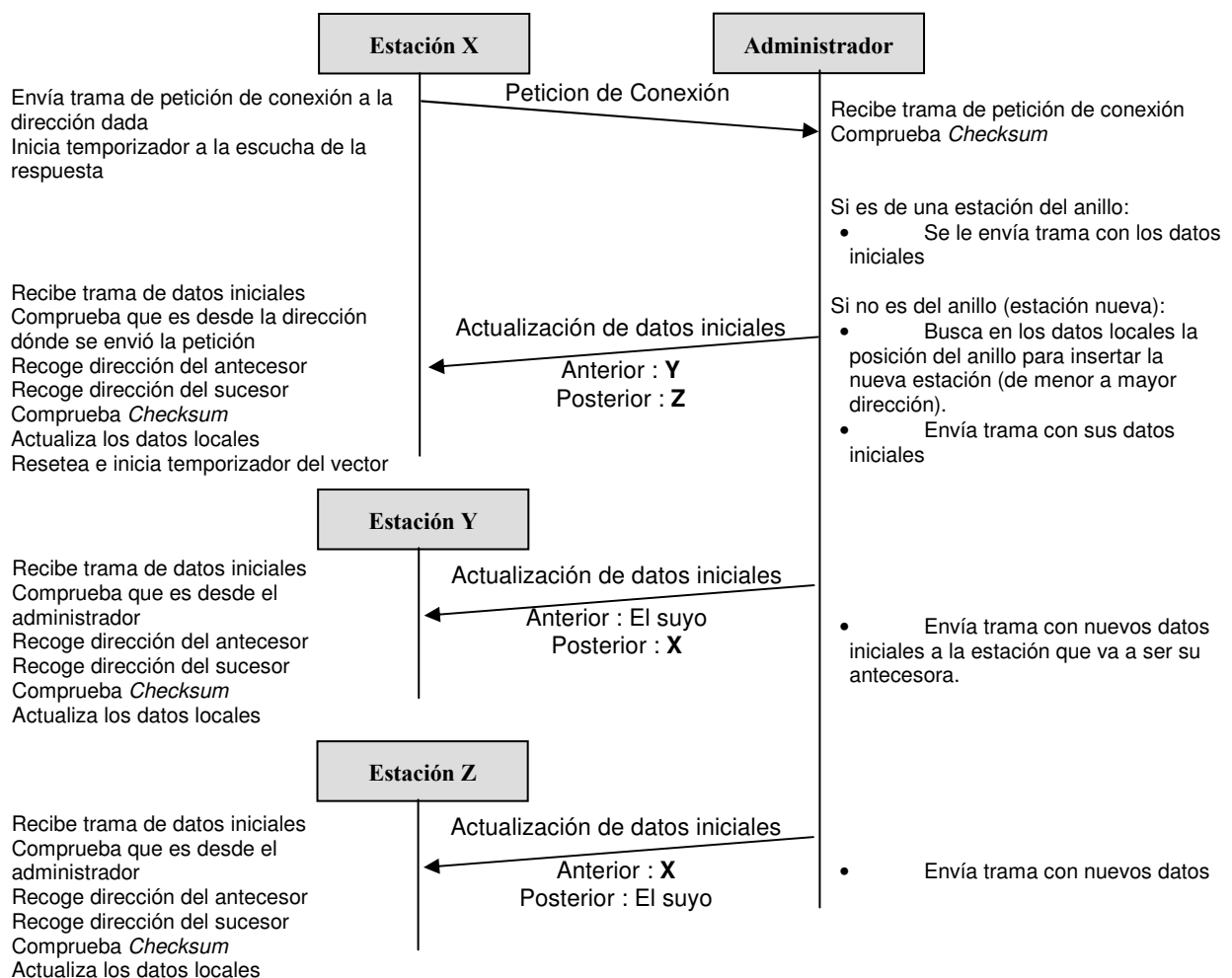


Imagen 11 : Conexión de una estación

De esta forma hemos insertado a X entre Y y Z y el anillo vuelve a estar consistente. Este proceso puede derivar en la pérdida del vector (si antes de actualizar Z el vector llega a Y que lo pasa a X que a su vez lo pasará a Z, quién lo rechazará al no provenir de Y), lo cual se solucionará mediante otra operación que será descrita más adelante.

Por otra parte, la información del vector que circula por el anillo será actualizada por las propias estaciones como también se verá más adelante.



### 2.4.2. Desconexión de una estación

Suponiendo que las estaciones X, Y y Z pertenecen al anillo y son consecutivas (*Dirección Y < Dirección X < Dirección Z*). Si la estación X desea desconectarse del anillo, lo hará del siguiente modo:

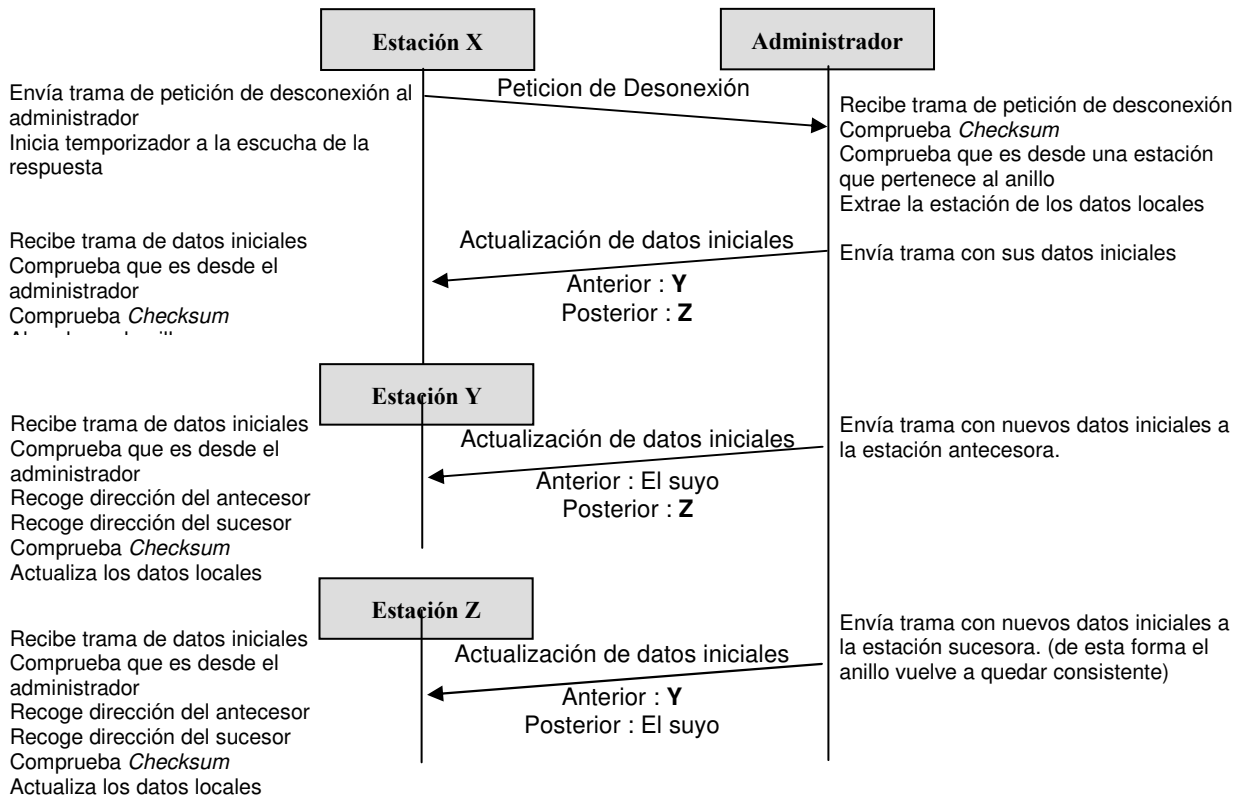


Imagen 12 : Desconexión de una estación

De esta forma hemos extraído a X y el anillo vuelve a estar consistente. Del mismo modo que en el caso anterior, este proceso puede derivar en la pérdida del vector.



### 2.4.3. Pérdida del vector del anillo

Esta situación se puede dar al inicio del anillo (cuando se conecta la primera estación) o cada vez que se pierde el vector por la caída de estaciones, y en algunos casos por la entrada y salida ordenada de las mismas.

Como es lógico, tras el reclamo del vector por una estación del anillo, el resto de las estaciones no tardarán en reclamarlo también, por lo que el administrador podrá recopilar información suficiente para recomponer el anillo dinámicamente y entregar un nuevo vector a la primera estación.

De esta forma, en una **primera fase de recogida de peticiones**, por cada estación que continúe en el anillo se realizarán las siguientes operaciones :

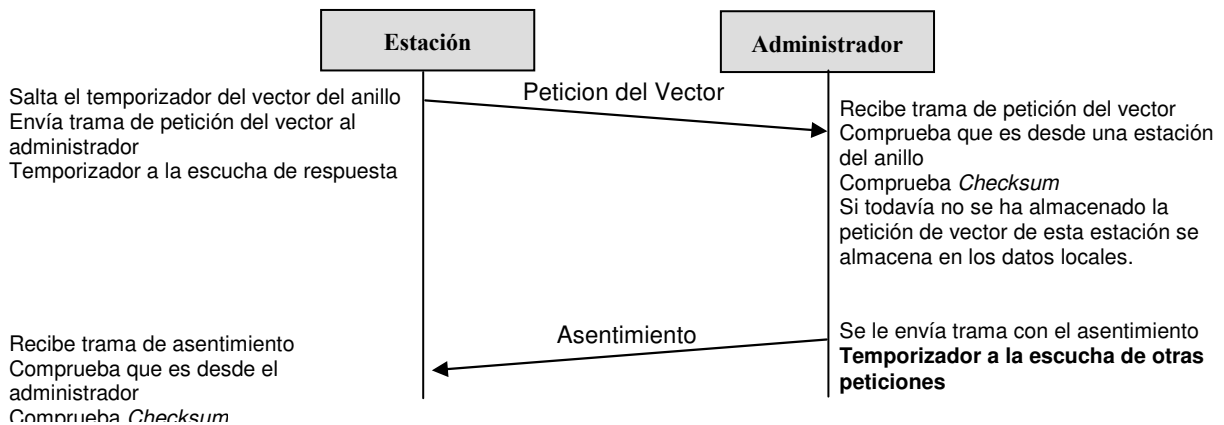


Imagen 13 : Pérdida del vector del anillo (1)

Tras vencer el temporizador del administrador se supone que todas las estaciones que aún continúan en el anillo han reclamado ya el vector, por lo que el administrador se encuentra en condiciones de enviar los datos iniciales a cada una de ellas y de pasar el vector a la primera de la lista.

Por tanto, en una **segunda fase de reconstrucción del anillo**, por cada estación que realizó petición del vector, se realizarán las siguientes operaciones:

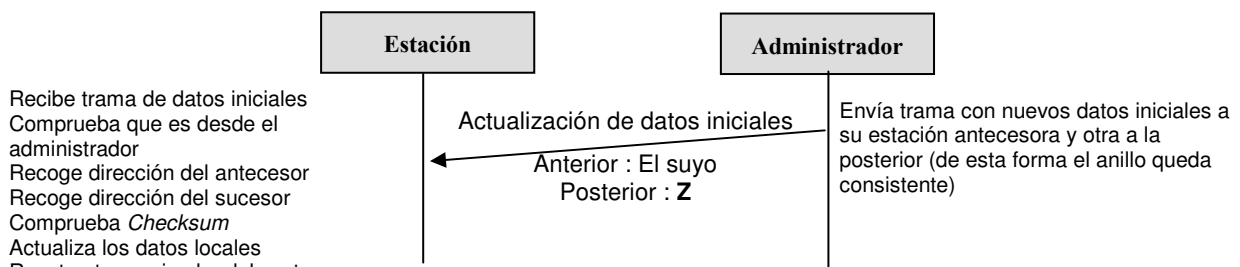
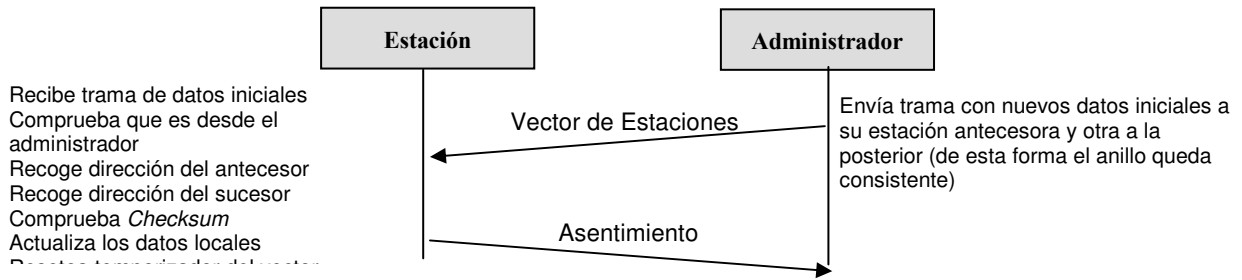


Imagen 14 : Pérdida del vector del anillo (2)

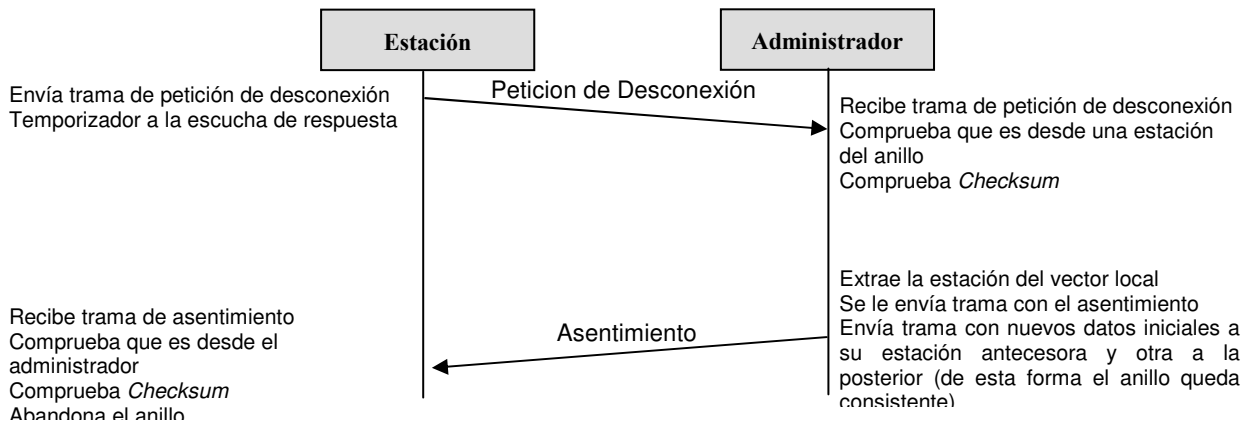


Por último, en una **tercera fase de puesta en marcha**, se envía el vector de estaciones a la primera de la lista para iniciar el anillo.



**Imagen 15 : Pérdida del vector del anillo (3)**

#### 2.4.4. Retransmisión del Vector de Estaciones por el Anillo



**Imagen 16 : Retransmisión del Vector de Estaciones por el Anillo**



### 2.4.5. Desconexión del anillo por el Administrador

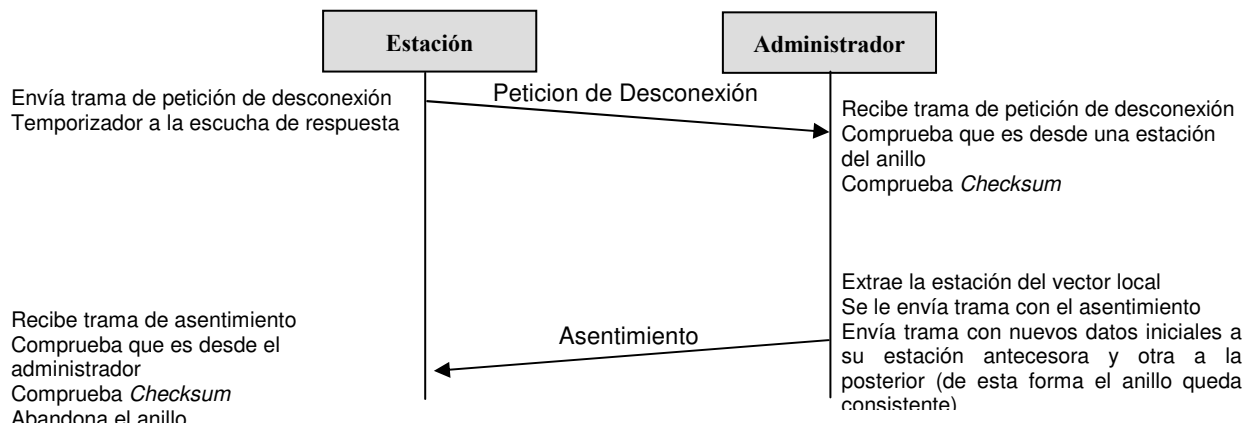


Imagen 17 : Desconexión del anillo por el Administrador

Alberto Carlos del Blanco Maraña  
Madrid 28 de junio de 2007