PROGRAMACIÓN DE SISTEMAS DISTRIBUIDOS

Práctica 1 – Servicio de Mensajería

Descripción

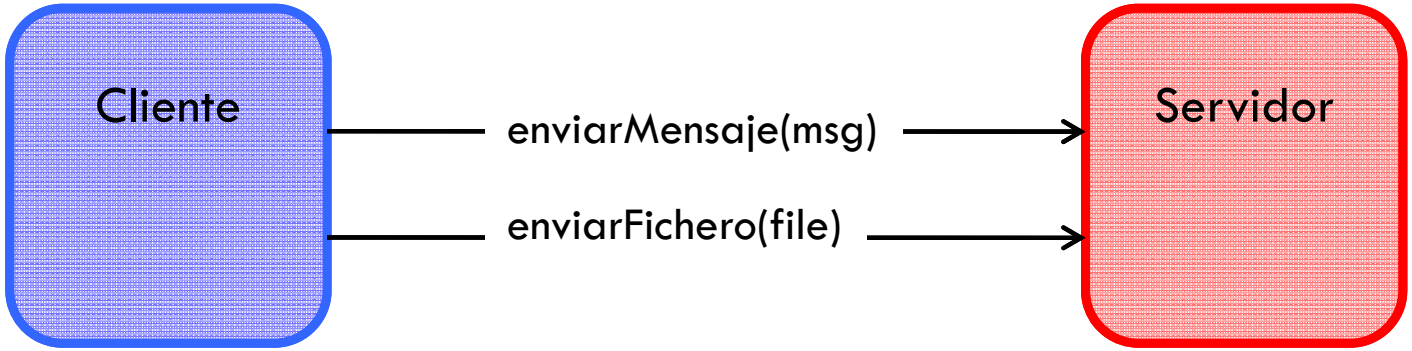
- Desarrollo de un servicio de mensajería
  - Similar al funcionamiento de la popular aplicación para móviles: **whatsApp**
- Implementación:
  - Arquitectura cliente/servidor
  - Servicios Web (gSOAP)
  - Lenguaje de programación C
- Servicios a implementar
  - Envío de mensajes *cortos* de texto
  - Envío de ficheros binarios (limitados por tamaño)

Descripción

- La idea parece sencilla
  - Enviar mensajes (strings) entre usuarios
- Parece fácil…
  - Los clientes envían los mensajes al servidor
  - El servidor calcula a quién va dirigido el mensaje
  - Se envía el mensaje al destino
- ¿Y para enviar ficheros?
- ¿Y para enviar un mensaje a un grupo de usuarios?

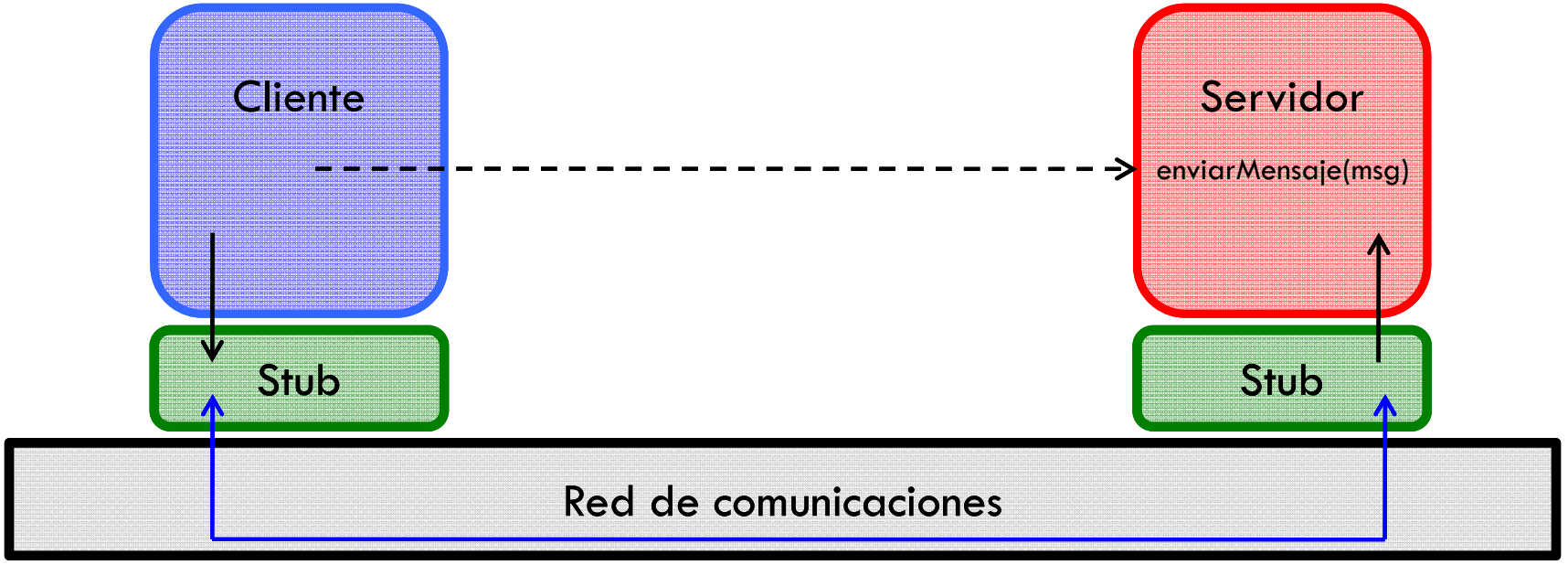
Arquitectura

- Comunicación cliente-servidor
  - **No** hay comunicación directa cliente-cliente
- 2 servicios básicos:
  - Envío de mensajes
  - Envío de ficheros



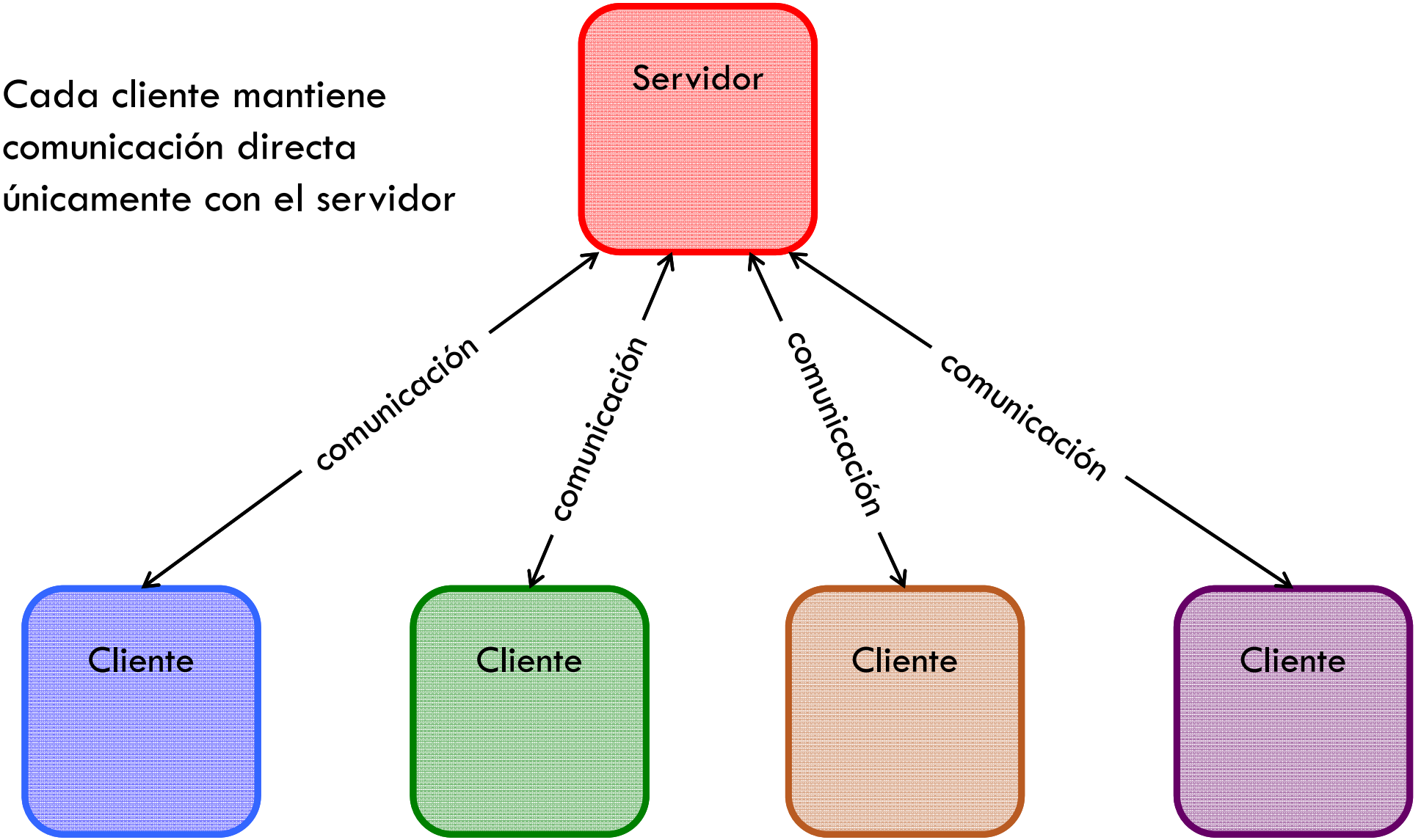
Arquitectura

- Servicios Web
  - Nos dan transparencia de comunicación
  - Definimos los servicios
    - Las comunicaciones se generan de forma automática



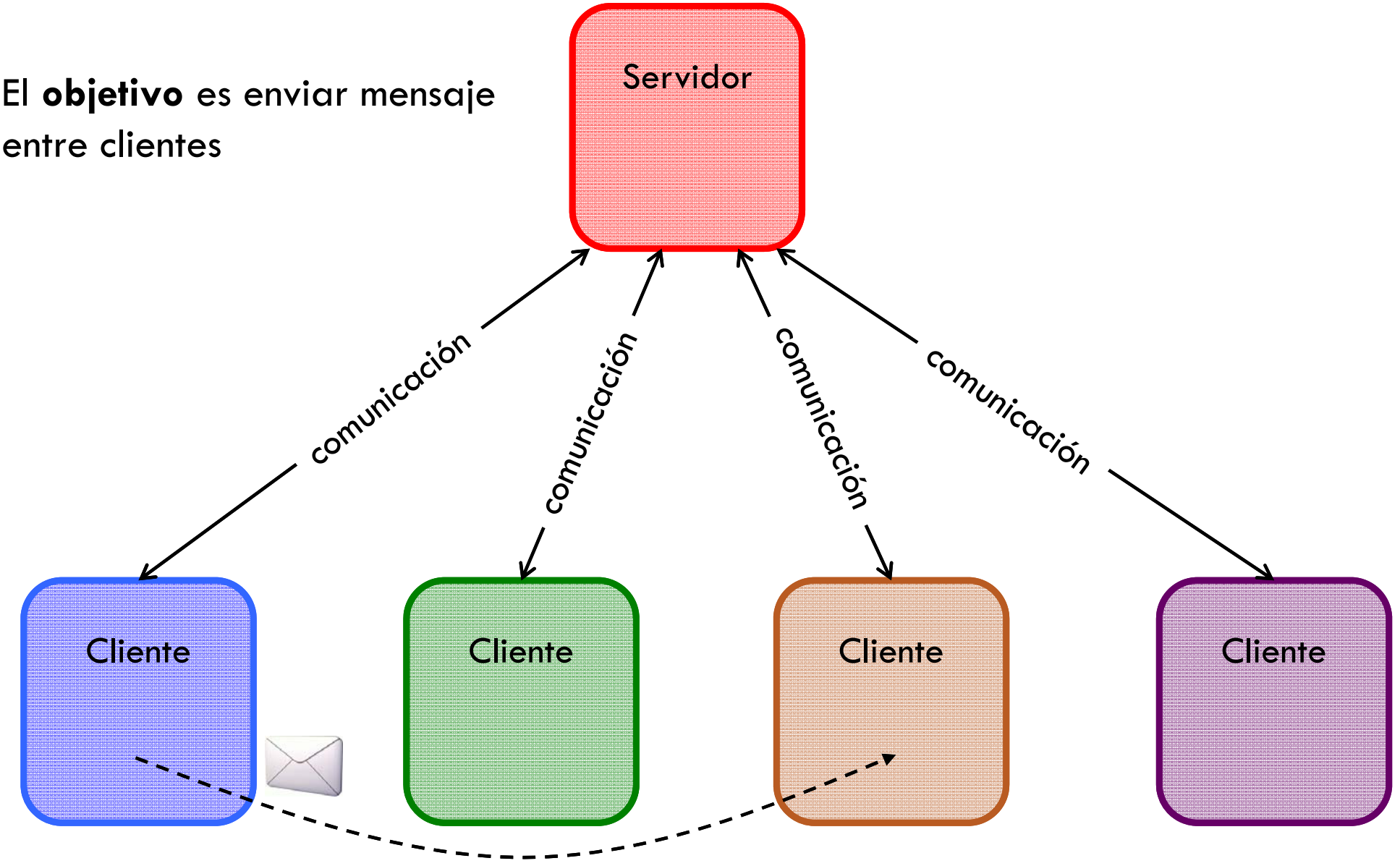
Arquitectura

Cada cliente mantiene comunicación directa únicamente con el servidor

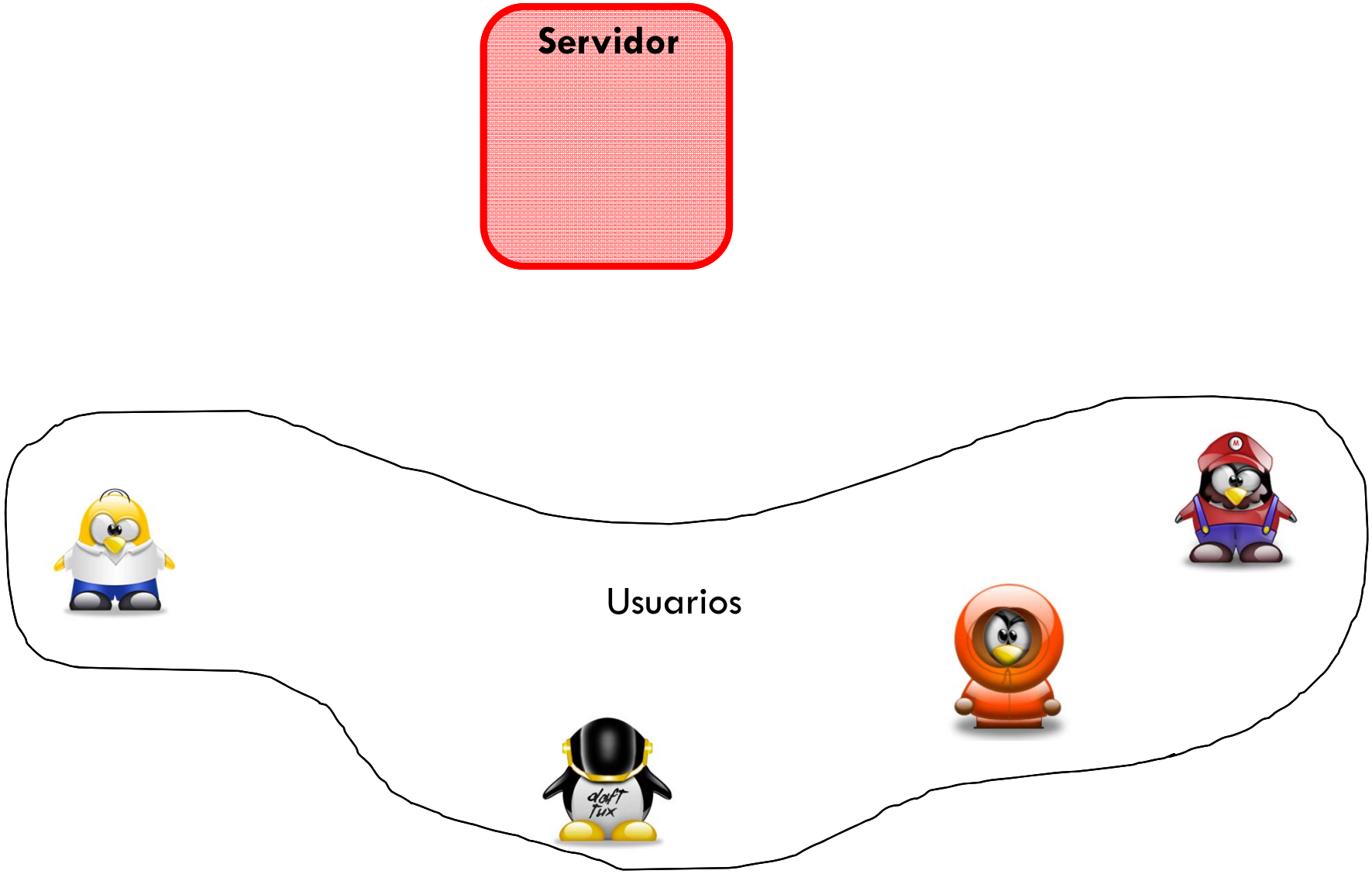


Arquitectura

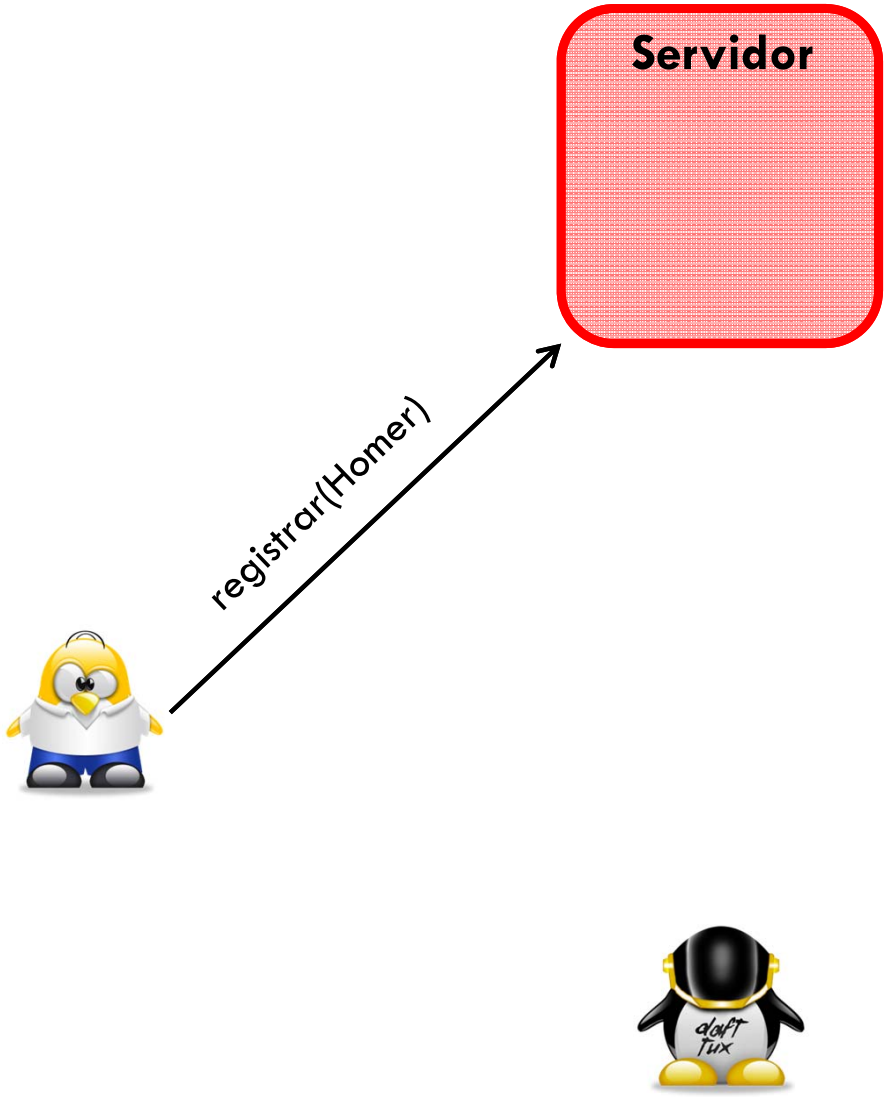
El **objetivo** es enviar mensaje entre clientes



Gestión de usuarios



Gestión de usuarios



Antes de utilizar el servicio, cada usuario debe registrarse en el servidor

Gestión de usuarios

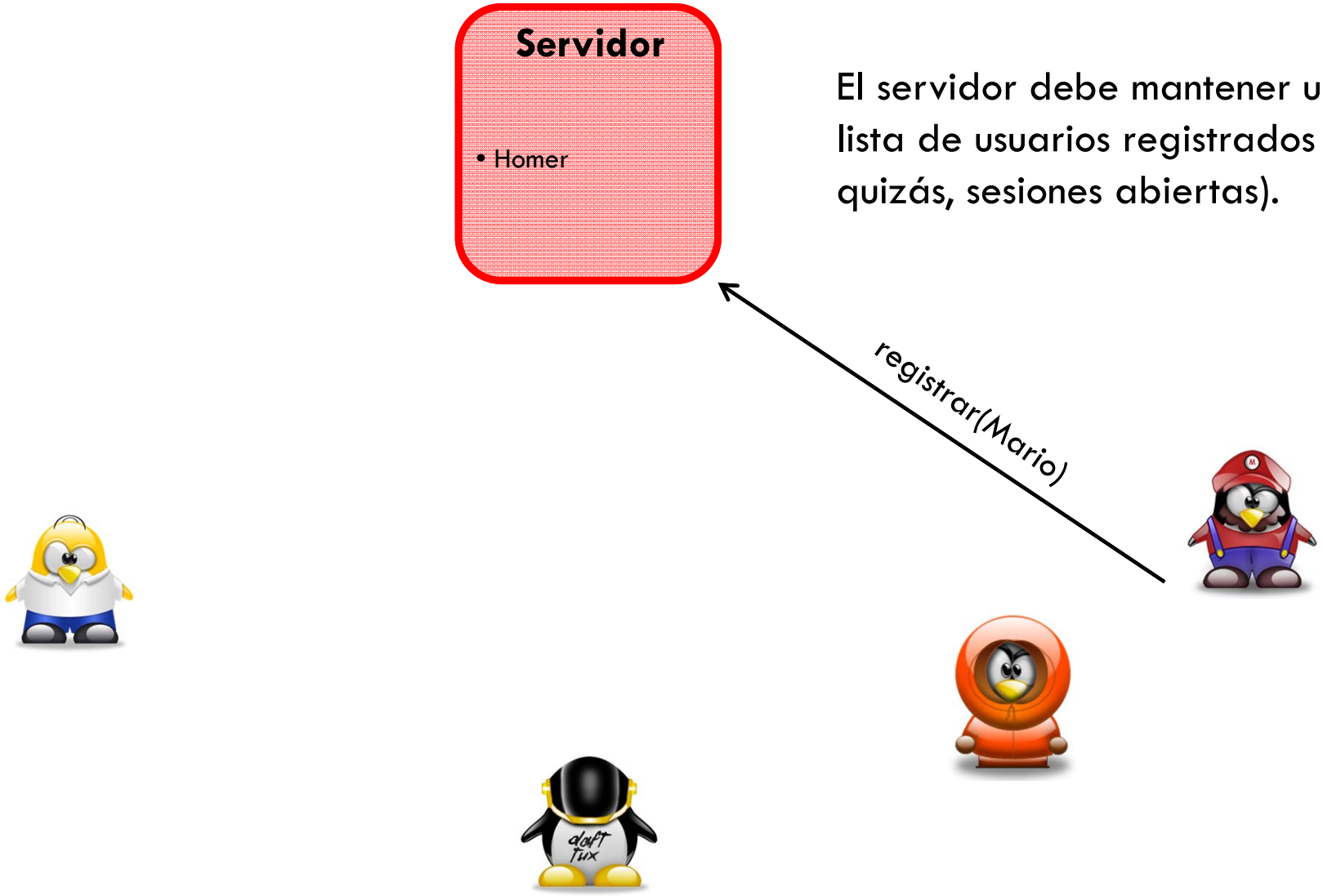
Servidor

- Homer

El servidor debe mantener una lista de usuarios registrados.

Si el usuario no va a conectarse siempre desde el mismo dispositivo, el servidor también podría mantener una lista de sesiones abiertas (en particular, con el fin de evitar que un mismo usuario abra múltiples sesiones simultáneos)

Gestión de usuarios



El servidor debe mantener una lista de usuarios registrados (y quizás, sesiones abiertas).

Gestión de usuarios

Servidor

- Homer
- Mario

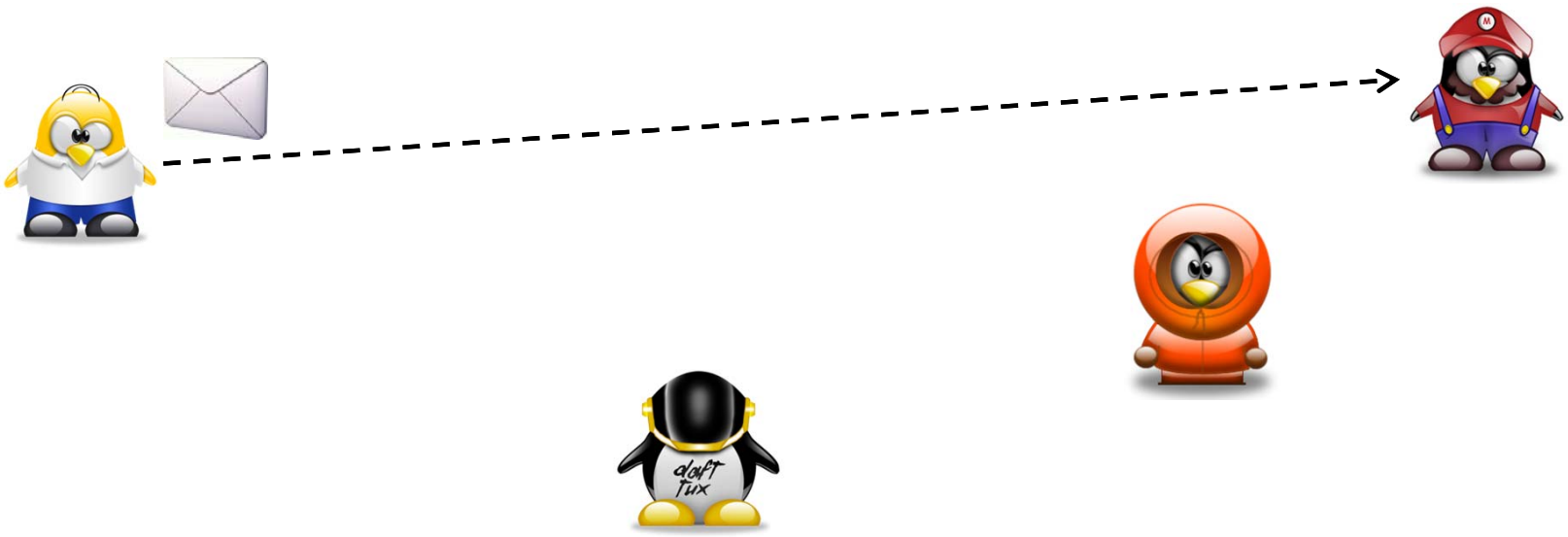
En este punto, ¿podría Homer enviarle un mensaje a Mario?

Gestión de usuarios

Servidor

- Homer
- Mario

En este punto, ¿podría Homer enviarle un mensaje a Mario?



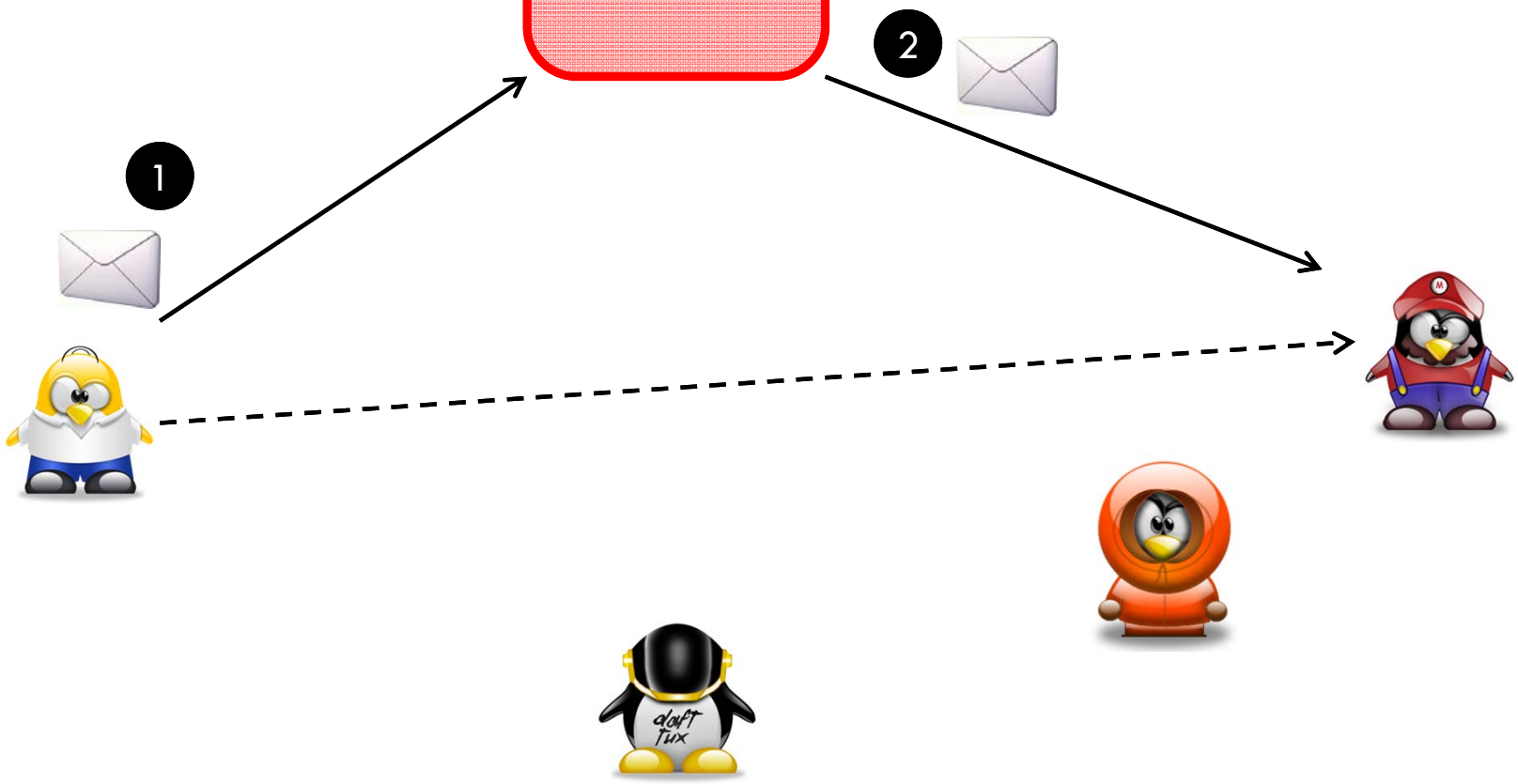
Gestión de usuarios

Hay que pasar por el servidor…

Servidor

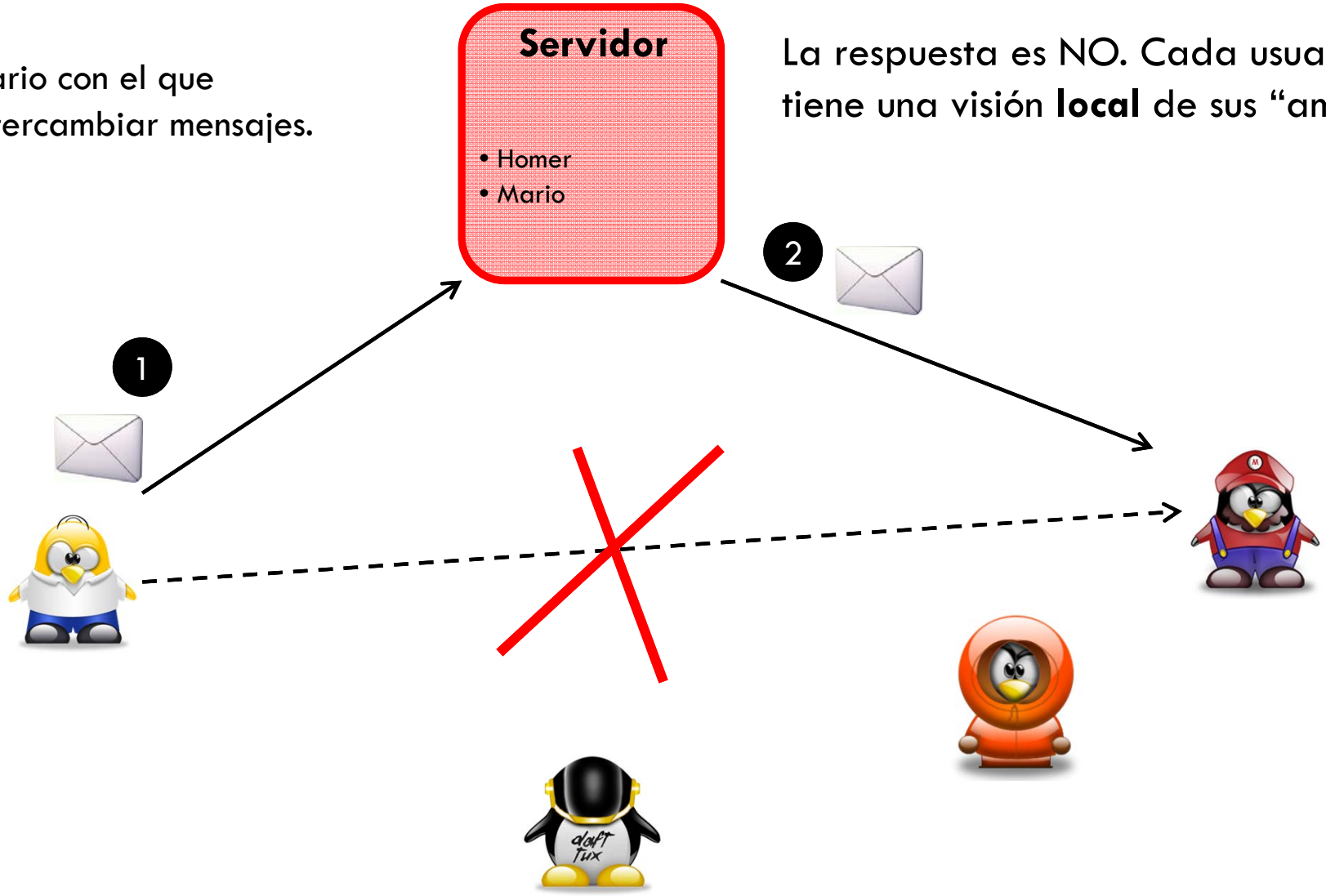
- Homer
- Mario

En este punto, ¿podría Homer enviarle un mensaje a Mario?



Gestión de usuarios

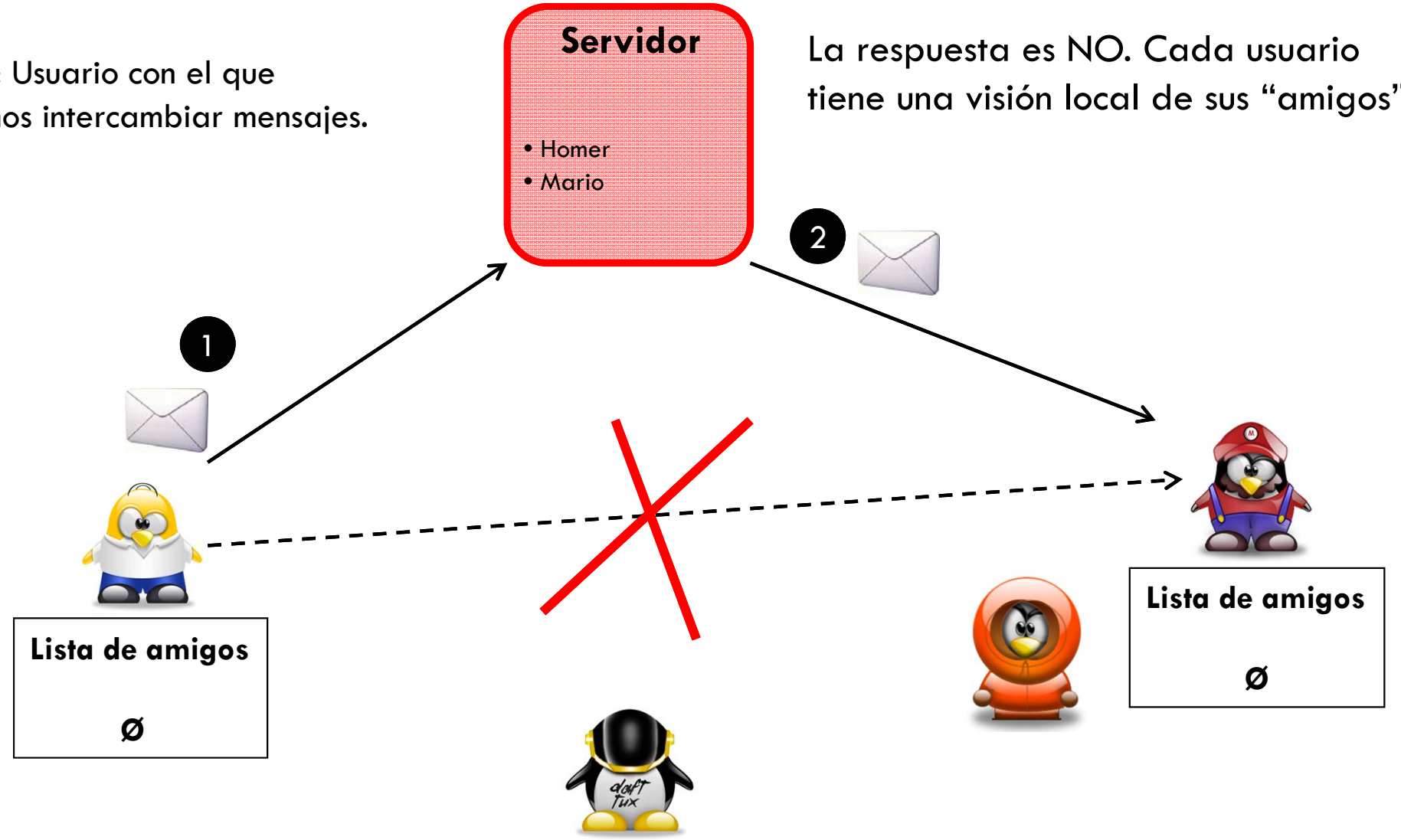
Amigo: Usuario con el que podemos intercambiar mensajes.



La respuesta es NO. Cada usuario tiene una visión **local** de sus "amigos"

Gestión de usuarios

Amigo: Usuario con el que podemos intercambiar mensajes.



La respuesta es NO. Cada usuario tiene una visión local de sus "amigos"

Gestión de usuarios

Homer quiere ser amigo de **Mario**

Servidor

- Homer
- Mario

Se envía una solicitud de amistad al servidor.

solicitarAmistad(Mario)

1

Lista de amigos

Ø

Lista de amigos

Ø

Gestión de usuarios

En el servidor se marca la solicitud de amistad como pendiente (pending)

Servidor

- Homer
- Mario

2

- Pending (Homer)

Lista de amigos

Ø

Lista de amigos

Ø

Gestión de usuarios

Homer está esperando…

Servidor

- Homer
- Mario
  - Pending (Homer)

El servidor pregunta a Mario si acepta la solicitud de Homer

3

aceptarSolicitud(Homer)

Lista de amigos

Ø

Lista de amigos

Ø

Gestión de usuarios

Homer está esperando…

Servidor

- Homer
- Mario
  - Pending (Homer)

El servidor pregunta a Mario si acepta la solicitud de Homer

4

confirmarSolicitud(Homer)

Lista de amigos

Ø

Lista de amigos

Ø

Gestión de usuarios

Homer está esperando…

Servidor

- Homer (Mario)
- Mario (Homer)

5

El servidor actualiza la lista de amigos de los involucrados

Lista de amigos

Ø

Lista de amigos

Ø

Gestión de usuarios

Homer está esperando…

Servidor

- Homer (Mario)
- Mario (Homer)

El servidor actualiza la lista de amigos de los involucrados

6 haConfirmado(Mario)

Lista de amigos

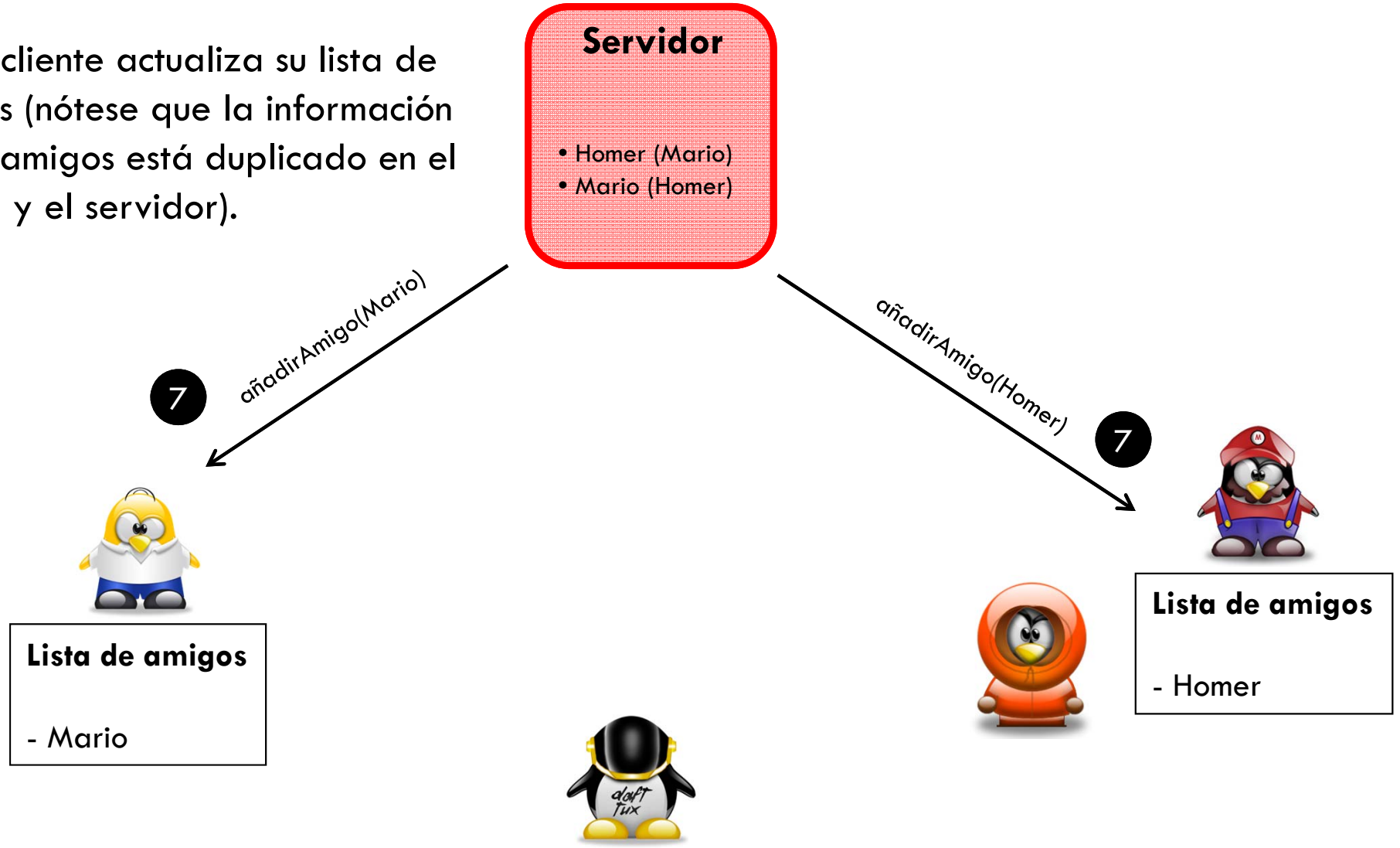
Ø

Lista de amigos

Ø

Gestión de usuarios

Cada cliente actualiza su lista de amigos (nótese que la información sobre amigos está duplicado en el cliente y el servidor).



Gestión de usuarios

Ahora sí podrían intercambiar mensajes!

Servidor

- Homer (Mario)
- Mario (Homer)

Lista de amigos

- Mario

Lista de amigos

- Homer

Gestión de usuarios

- Datos que nos hacen falta en el servidor:
  - Lista de usuarios
    - Cada usuario tendrá una lista de amigos
- Datos que nos hacen falta en el cliente:
  - Lista de amigos

Envío de mensajes

- Cada usuario puede enviar mensajes a otros usuarios
- Los mensajes los reciben amigos del usuario
- ¿Un usuario puede enviar un mensaje a alguien no amigo?

Envío de mensajes

- Cada usuario puede enviar mensajes a otros usuarios
- Los mensajes los reciben amigos del usuario
- ¿Un usuario puede enviar un mensaje a alguien no amigo?
  - Sí!
  - Pero, ¿qué pasa con el mensaje?
    - El servidor lo descarta

Envío de mensajes

1) Se dan de alta Kenny y Daft Punk.
2) Actualizamos la lista de amigos

Nos hace falta un servicio para enviar mensajes
El servicio lo proporciona el servidor

¿Lo llamamos **enviarMensaje(msg, usuario)**?

Servidor
- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- Cada vez que se conecta un usuario, tiene una IP
  - Distintos sitios = IPs distintas
- ¿Guardamos las IPs de los clientes?
- ¿Y si un usuario no está conectado?
  - ¿Le llegan los mensajes?
- ¿Cómo envía el servidor un mensaje al cliente?
  - ¿Y si está desconectado?

Servidor
- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- Por partes:
  - Las IPs no se guardan
  - El servidor NO es un cliente
  - No hacen falta las IP de cada usuario
  - La IP del servidor es siempre la misma
    - La de los clientes NO

Servidor

- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- Vamos a enviar un mensaje!
  - Homer envía uno a Mario
  - Comentamos los pasos…

Servidor

- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

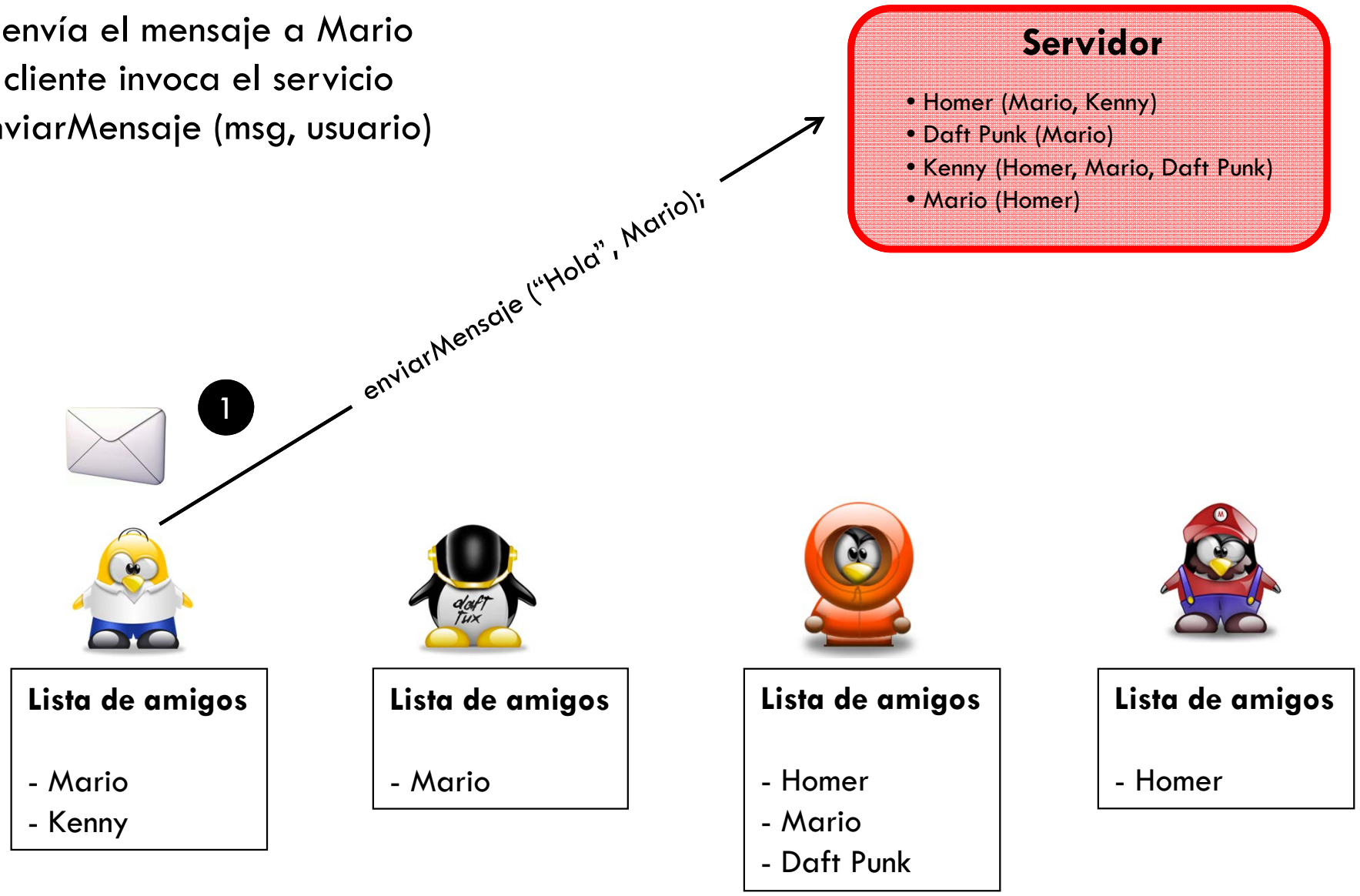
- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- Homer envía el mensaje a Mario
  - El cliente invoca el servicio
  - enviarMensaje (msg, usuario)



Envío de mensajes

- El servidor comprueba:
  - a) Que Mario existe
  - b) Que Mario es amigo de Homer
- Hasta aquí todo OK!

Servidor

- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

2

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

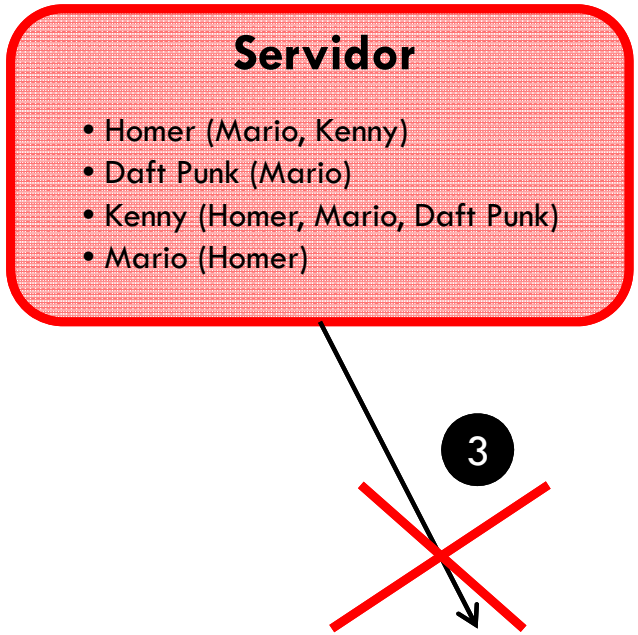
- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- Hay que enviar el mensaje a Mario…
  - ¿Cómo?



Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

- Homer
- Mario
- Daft Punk

Lista de amigos

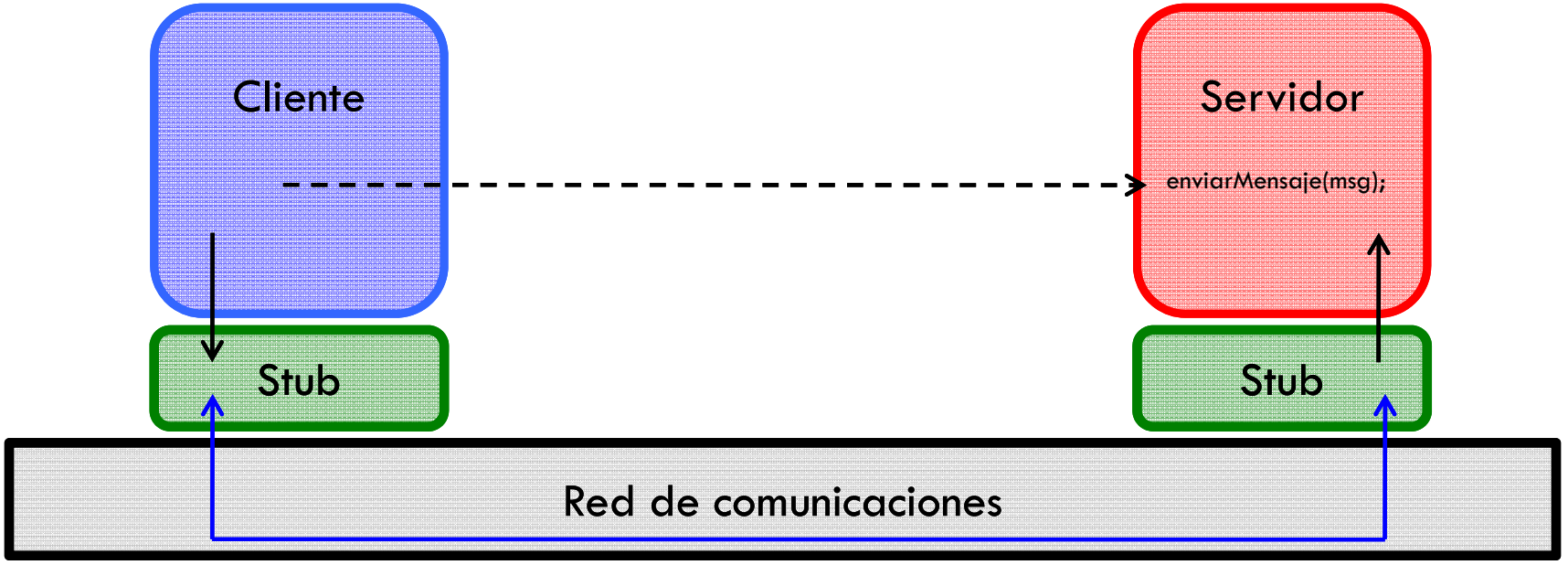
- Homer

Arquitectura

El cliente invoca *enviarMensaje()* en el servidor

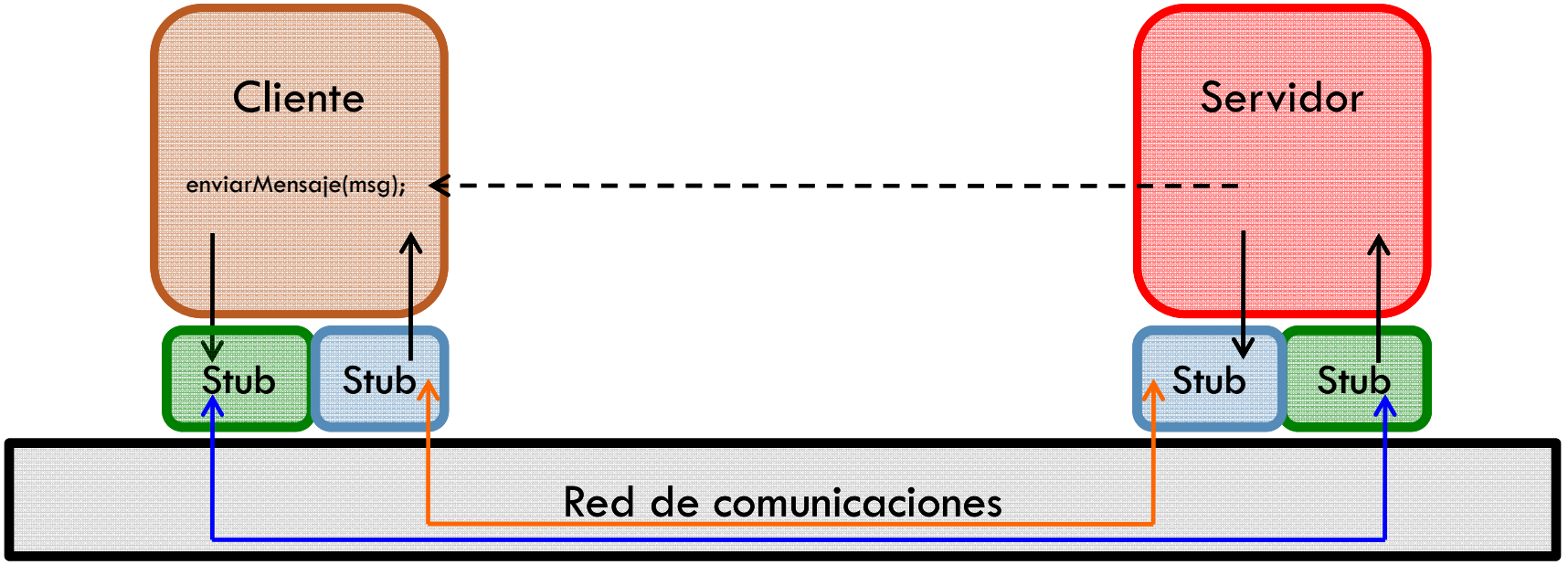
Toda la parte de comunicación la hacen los stubs

¿Qué hace el servidor con el mensaje?



Arquitectura

Un callback…



Arquitectura

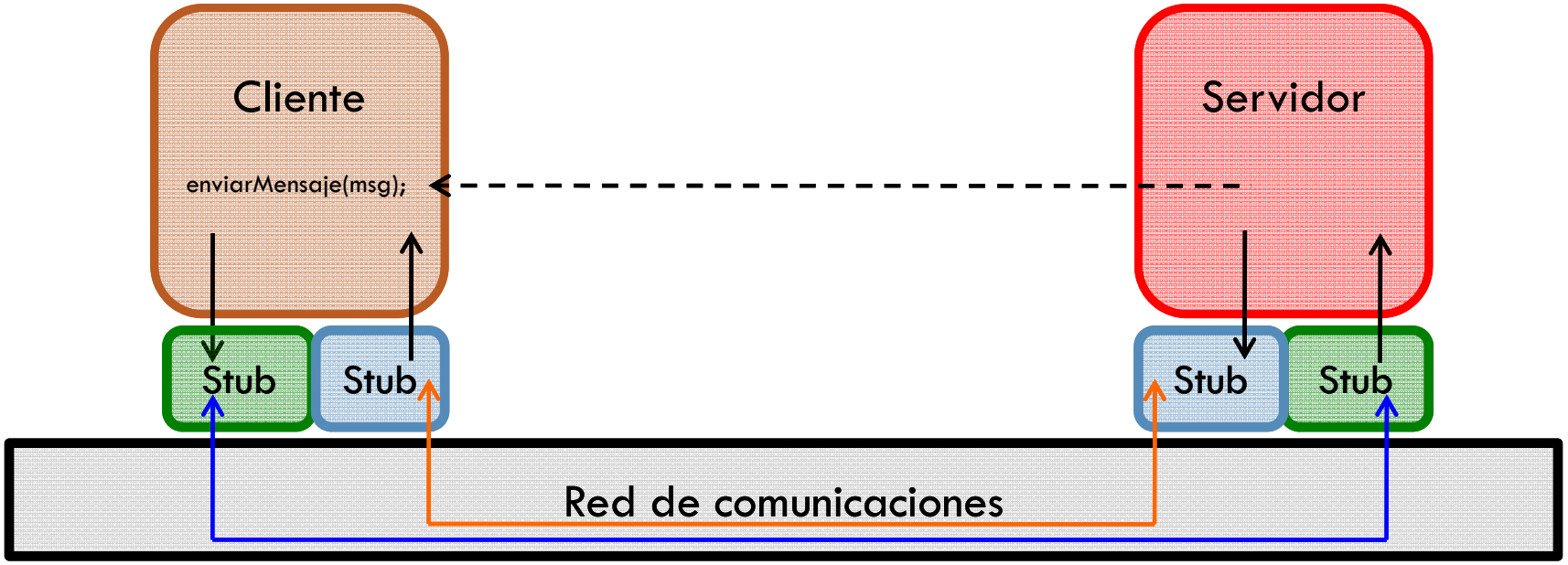
Un callback…

El servidor actúa como cliente y el cliente como servidor

Generamos más stubs para llevar a cabo esta comunicación

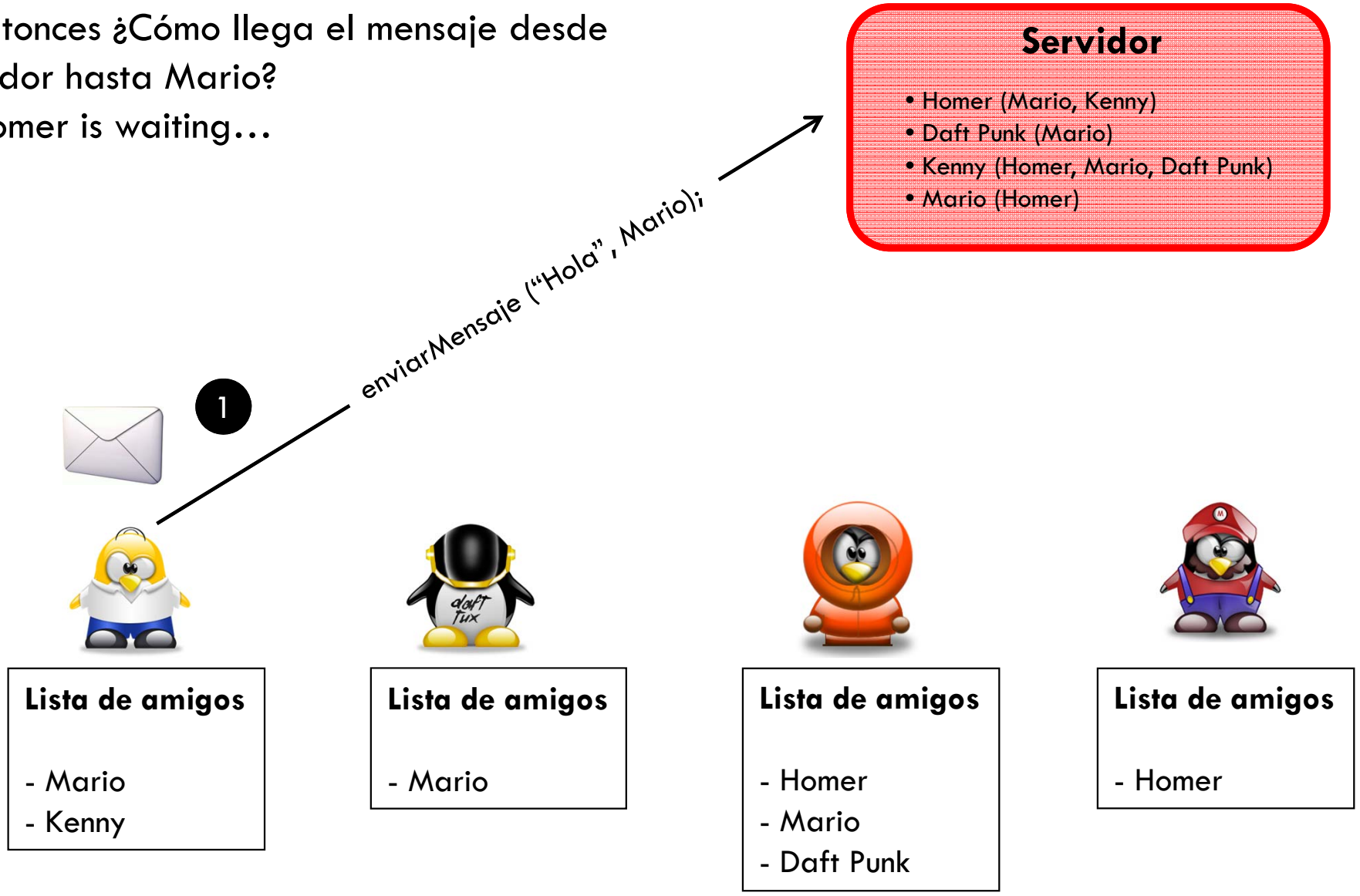
¿No sería duplicar el mismo servicio pero en sentido contrario?

¿Esto es una buena solución? Quizás para un servicio de chat pero aquí NO



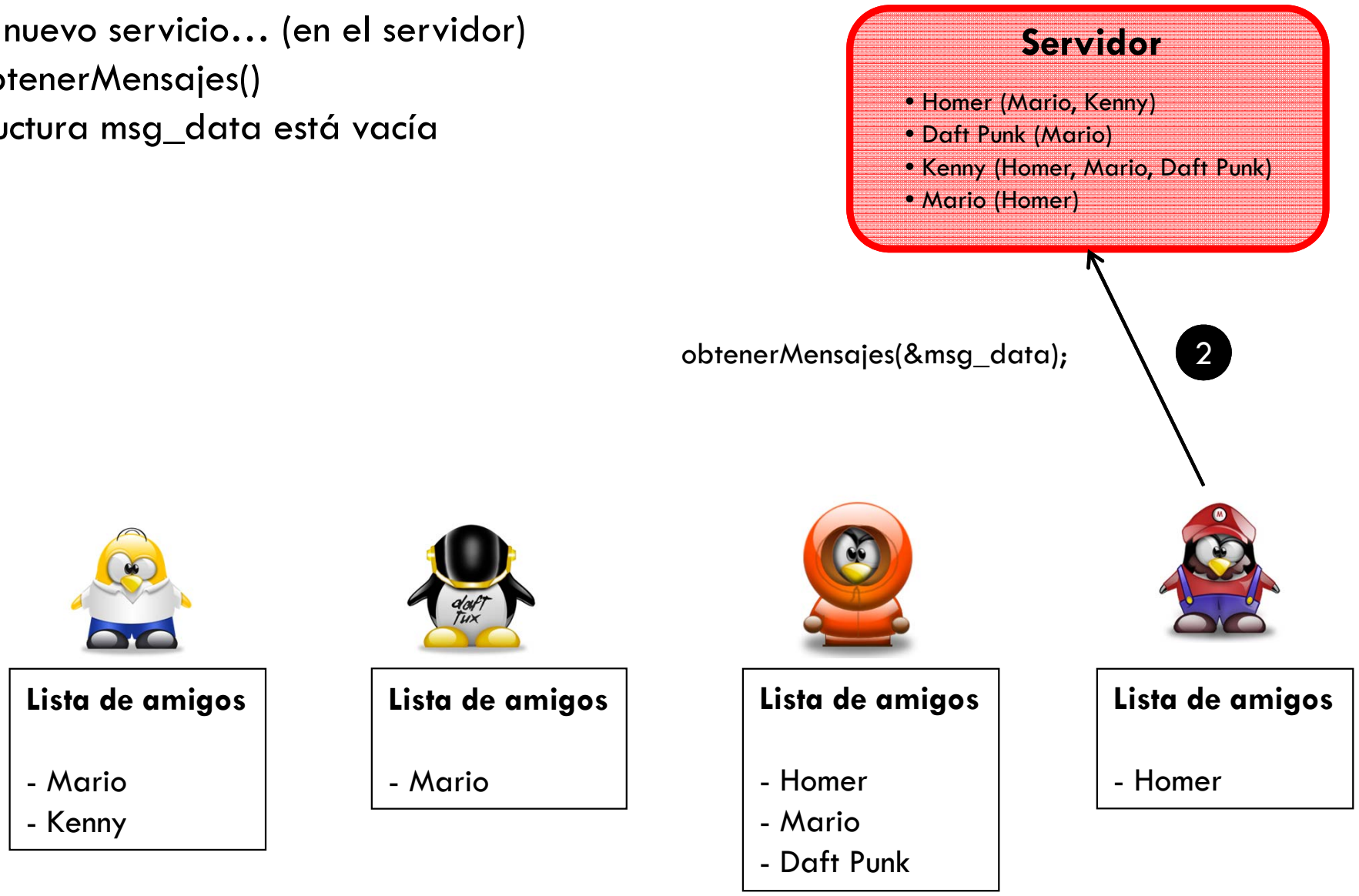
Envío de mensajes

- Pero entonces ¿Cómo llega el mensaje desde el servidor hasta Mario?
  - Homer is waiting…



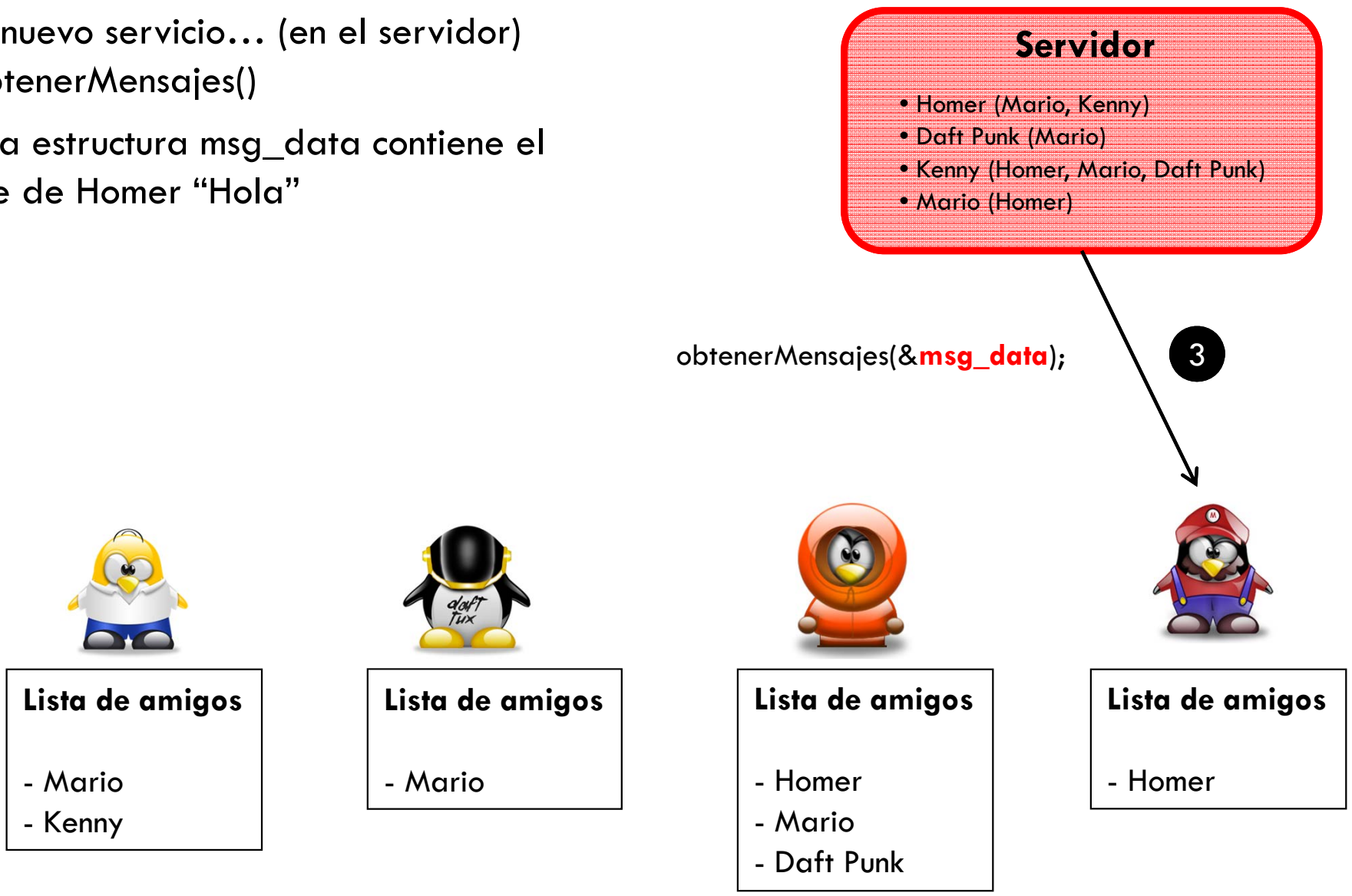
Envío de mensajes

- Con un nuevo servicio… (en el servidor)
  - obtenerMensajes()
- La estructura msg_data está vacía



Envío de mensajes

- Con un nuevo servicio… (en el servidor)
  - obtenerMensajes()
- Ahora la estructura msg_data contiene el mensaje de Homer “Hola”



Envío de mensajes

- ¿Y ya está?
  - ¿Y si me envía 2 mensajes?
  - ¿Cómo sincronizamos envío/obtener?
- Es complicado…
  - Deberíamos para a Homer para que no envíe más mensajes
- ¿Es esto una buena solución?

Servidor

- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- ¿Y ya está?
  - ¿Y si me envía 2 mensajes?
  - ¿Cómo sincronizamos envío/obtener?
- Es complicado…
  - Deberíamos para a Homer para que no envíe más mensajes
- ¿Es esto una buena solución?
  - **NO**

Servidor

- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- Mucha complejidad
  - Es incómodo esperar para enviar mensajes
- ¿Solución?

Servidor

- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

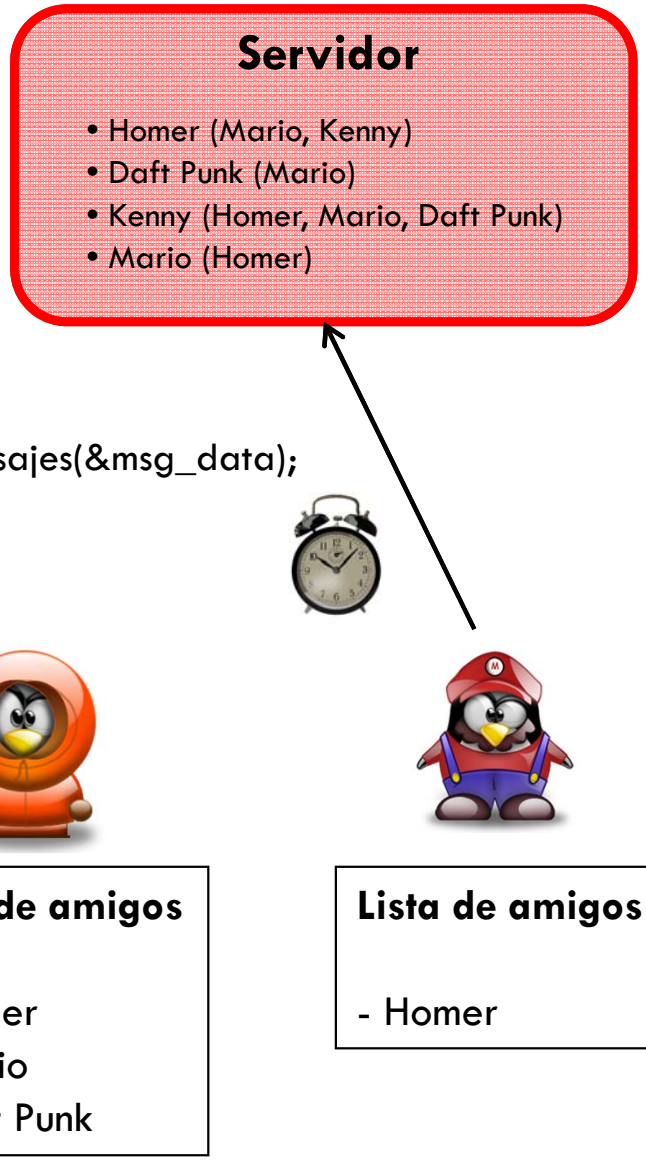
- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- Mucha complejidad
  - Es incómodo esperar para enviar mensajes
- ¿Solución?
  - Utilizar alarmas
  - Solicitar mensajes cada cierto tiempo



Envío de mensajes

- Ya tenemos
  - Gestión de usuarios
  - Listas de amigos
  - Alarmas para obtener mensajes
  - ¿Qué falta?
    - ¿Hemos terminado?

Servidor

- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

Envío de mensajes

- Un ejemplo para ver si está todo…
  - Homer envía un mensaje a Kenny
  - Mario envía un mensaje a Kenny
  - Daft está durmiendo
  - Kenny… no sabemos donde está!

Servidor

- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

Lista de amigos

- Mario
- Kenny

Lista de amigos

- Mario

Lista de amigos

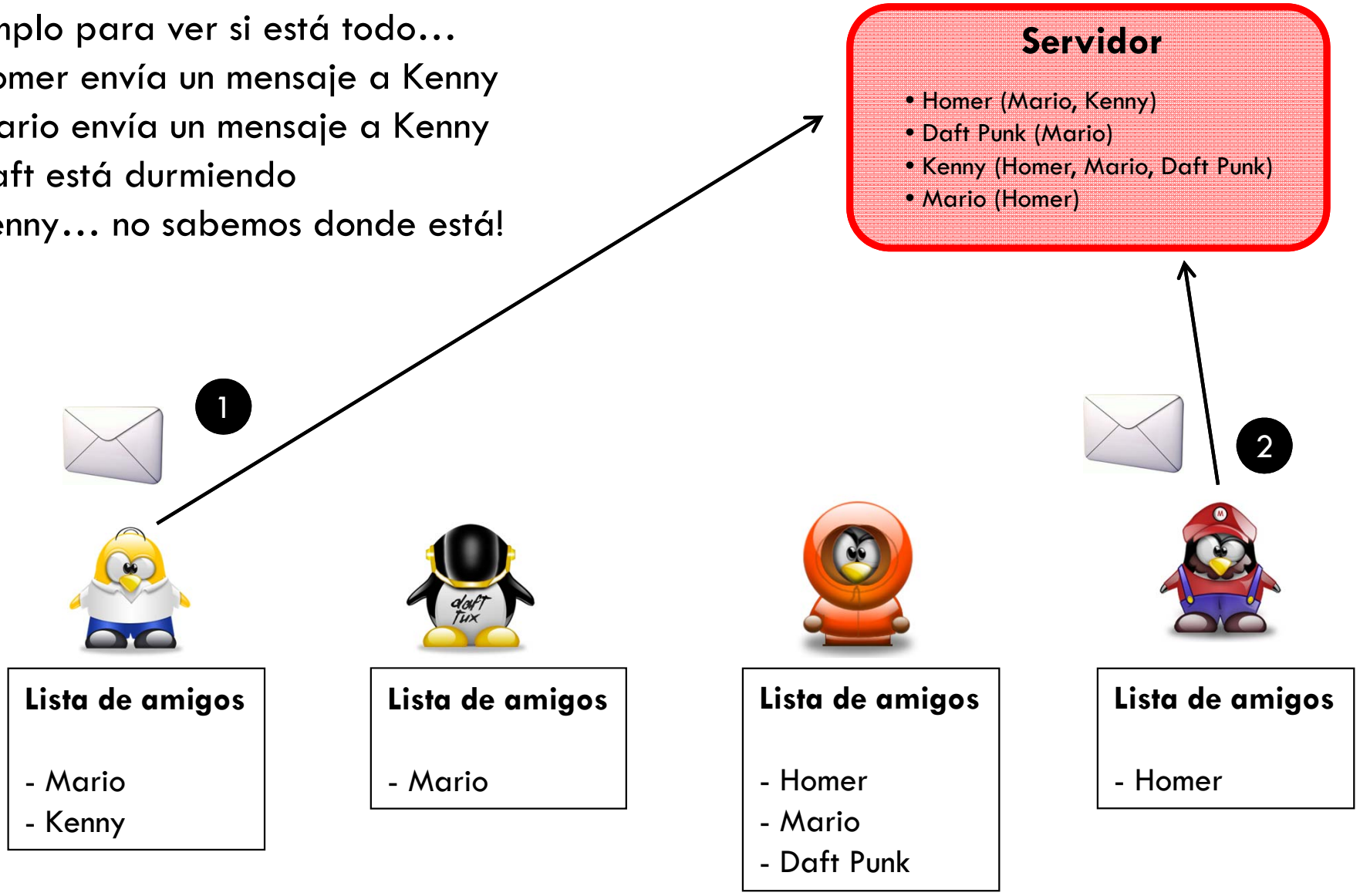
- Homer
- Mario
- Daft Punk

Lista de amigos

- Homer

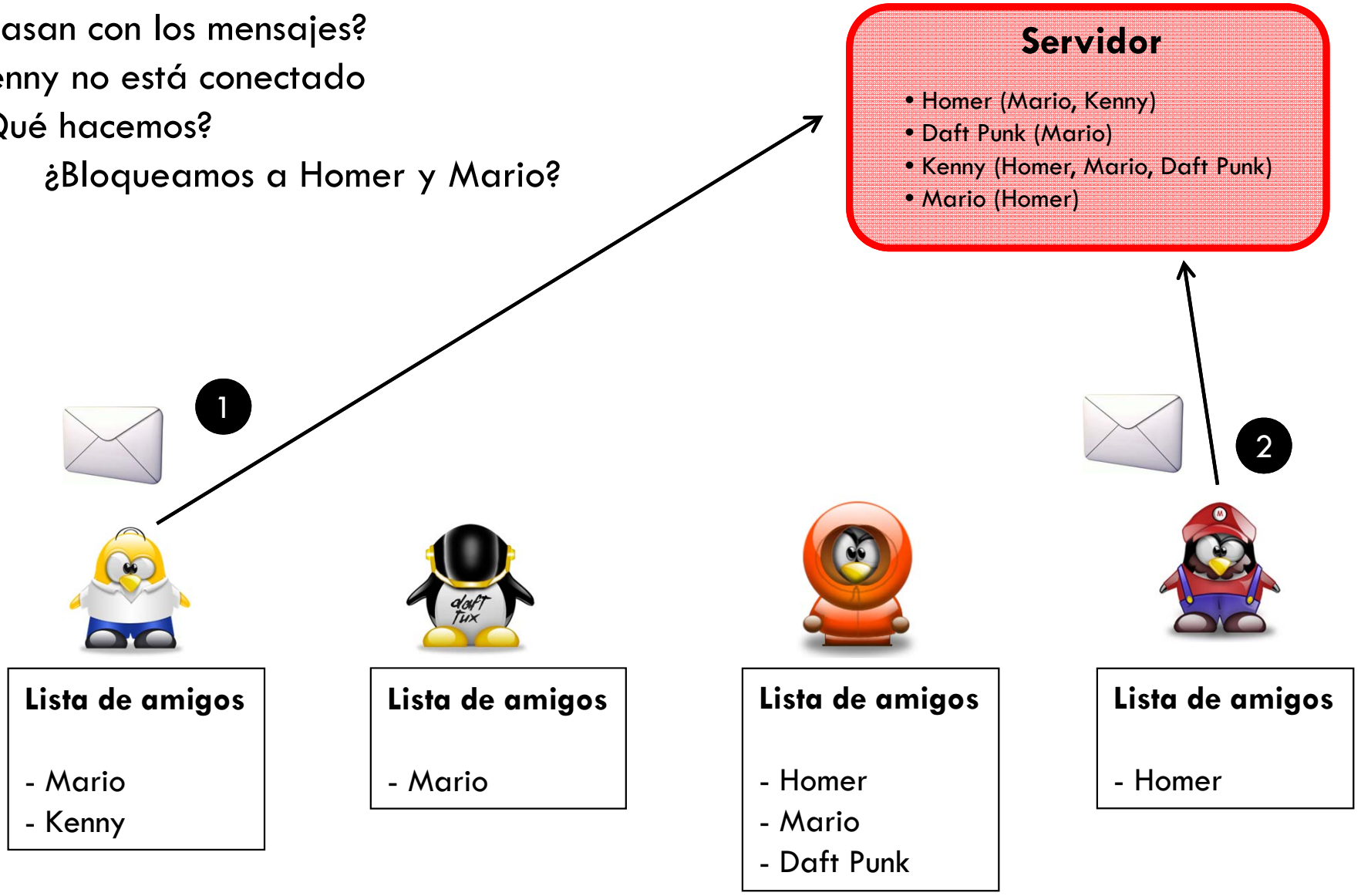
Envío de mensajes

- Un ejemplo para ver si está todo…
  - Homer envía un mensaje a Kenny
  - Mario envía un mensaje a Kenny
  - Daft está durmiendo
  - Kenny… no sabemos donde está!



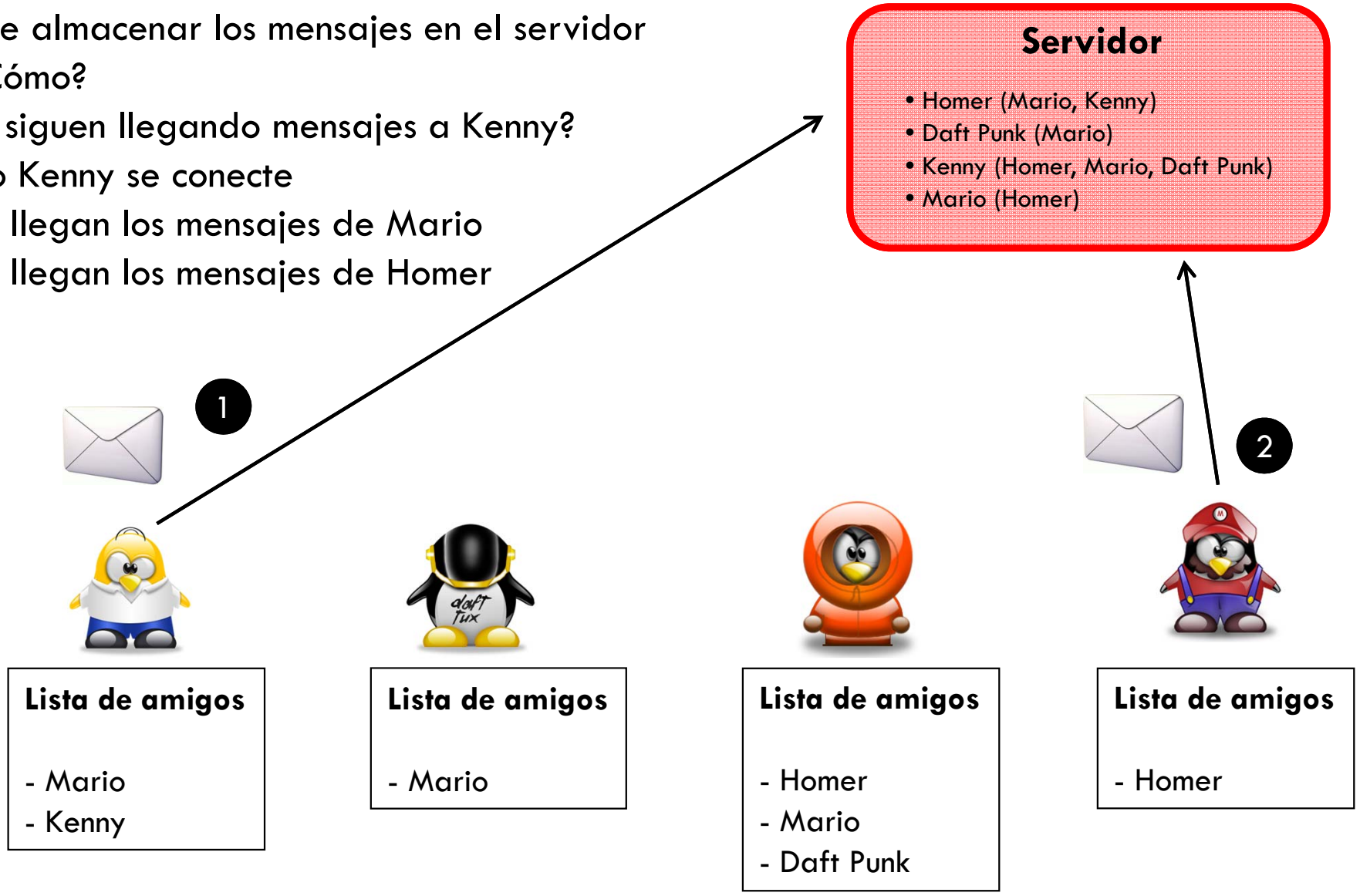
Envío de mensajes

- ¿Qué pasan con los mensajes?
  - Kenny no está conectado
  - ¿Qué hacemos?
    - ¿Bloqueamos a Homer y Mario?



Envío de mensajes

- Hay que almacenar los mensajes en el servidor
  - ¿Cómo?
- ¿Y si le siguen llegando mensajes a Kenny?
- Cuando Kenny se conecte
  - Le llegan los mensajes de Mario
  - Le llegan los mensajes de Homer



Envío de mensajes

- Almacenamos los mensajes en el servidor, ¿Cómo?
  - ¿En ficheros?
  - ¿Un fichero por destinatario?
  - ¿Un fichero por pareja (emisor, destinatario)?

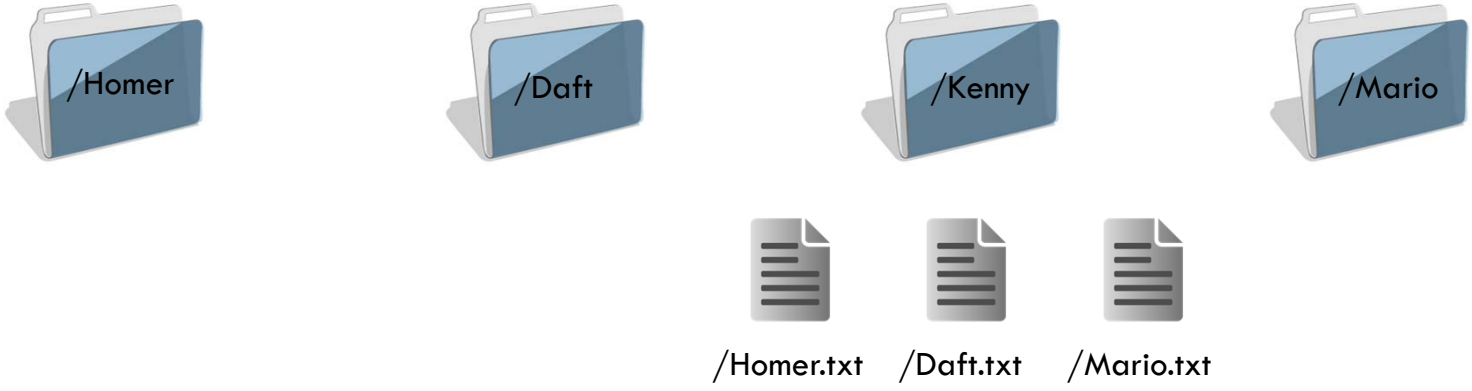
Servidor
- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)

- Posibles soluciones
  1. Un fichero por destinatario
     - Más fácil mantener el orden de llegada entre mensajes con distintos emisores
  2. Un directorio por destinatario
     - que contiene un fichero por emisor
     - Menos accesos concurrentes
  3. Una base de datos
     - MySQL, por ejemplo

Envío de mensajes

- Ilustración de la primera solución:
  - Un directorio por usuario (destinatario)
  - que contiene un fichero por emisor

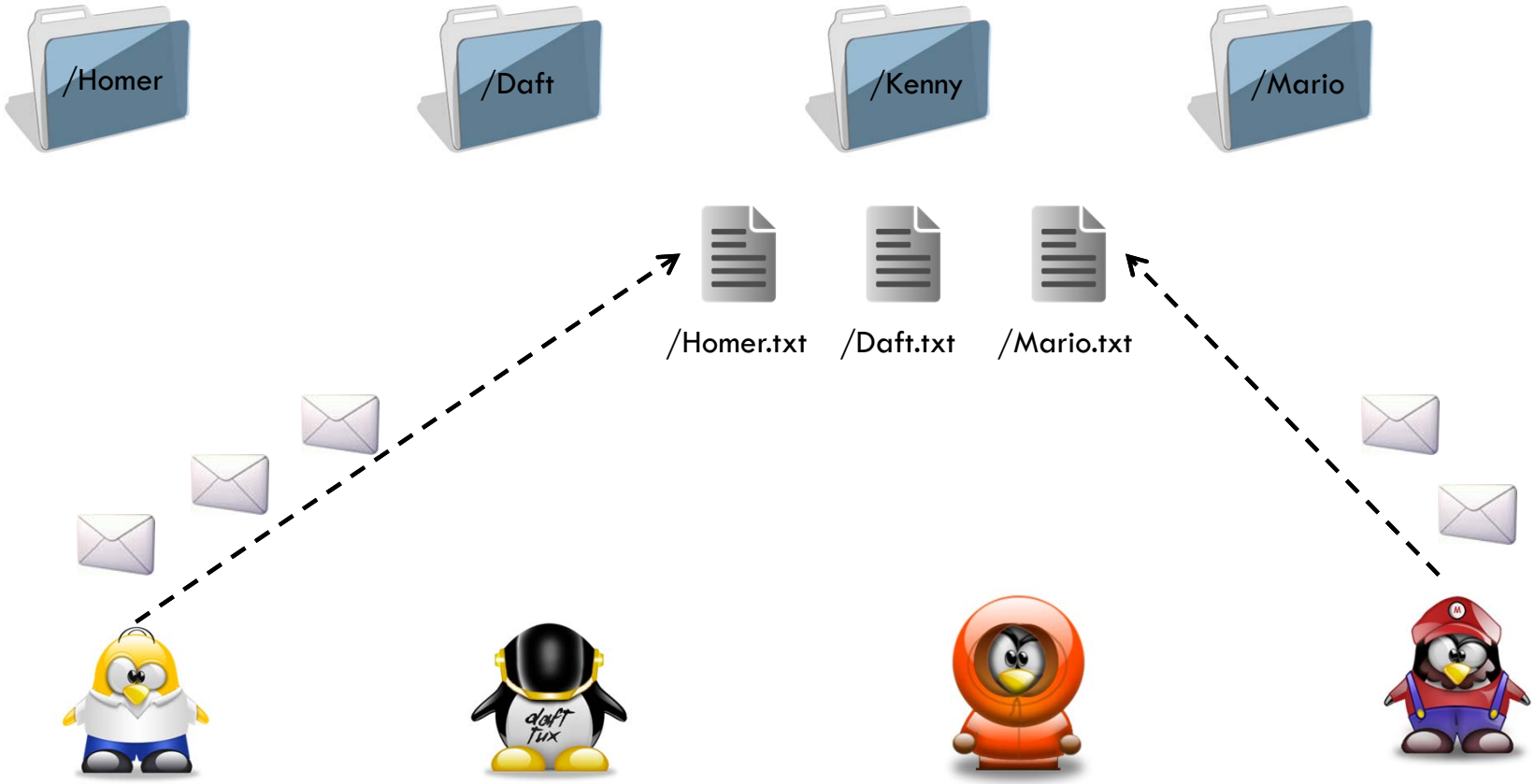
Servidor
- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)



Envío de mensajes

- Ilustración de la primera solución:
  - Todos los mensajes del mismo emisor:
    - se graban en el mismo fichero
    - en la carpeta del destinatario

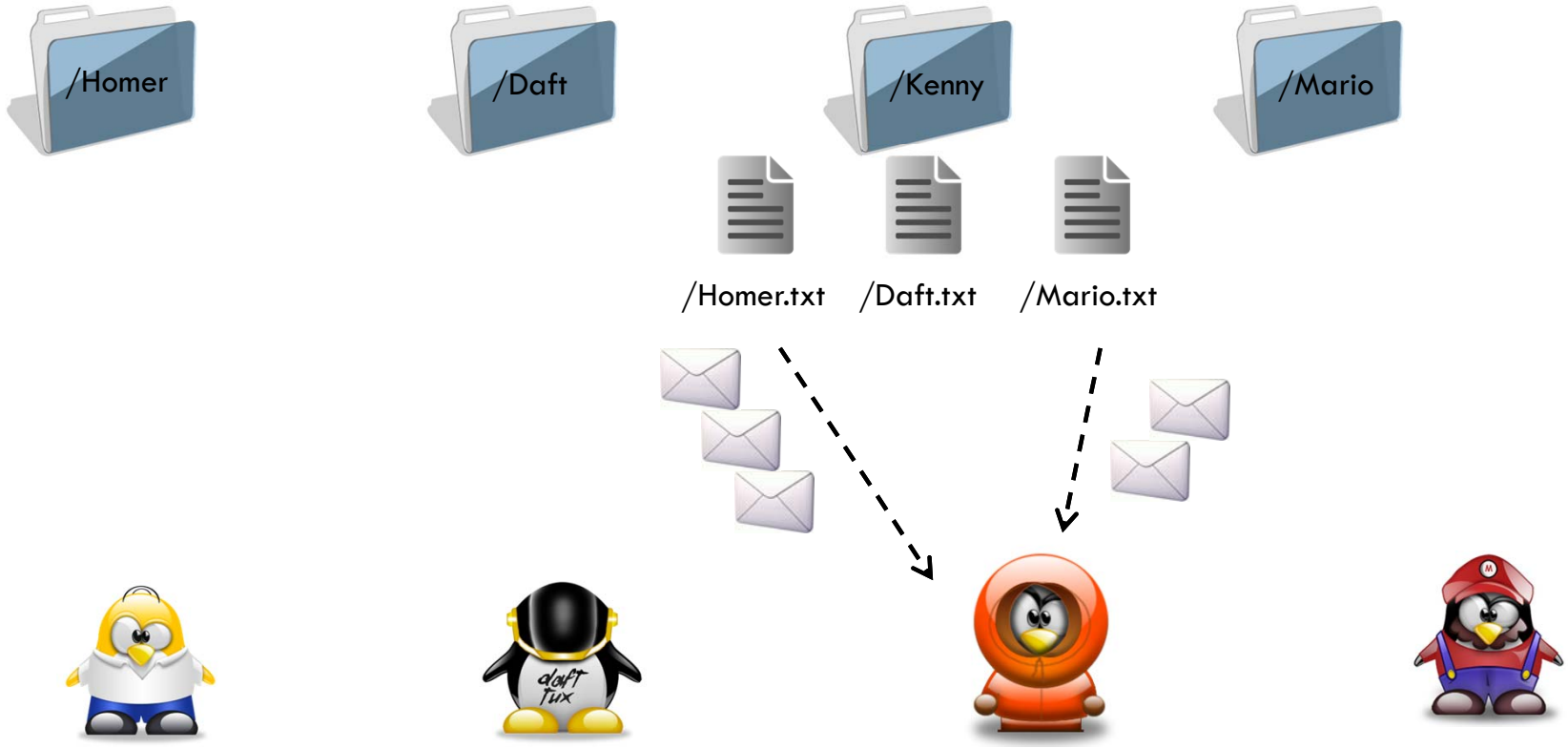
Servidor
- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)



Envío de mensajes

- Ilustración de la primera solución:
  - Cuando Kenny se conecte:
    - Recibe **los 5 mensajes pendientes**
      - 3 de Homer y 2 de Mario
    - También puede recibir otras informaciones

Servidor
- Homer (Mario, Kenny)
- Daft Punk (Mario)
- Kenny (Homer, Mario, Daft Punk)
- Mario (Homer)



En resumen

- Definir los servicios que va a ofrecer el servidor y los estructuras de datos usados
  - Darse de alta y de baja, quizás abrir sesión (¿cómo se cierran las sesiones?)
  - Enviar mensaje y enviar petición de amistad
  - Obtener datos pendientes (nótese: un mensaje por dato sería muy ineficiente)
    - Mensajes, peticiones de amistad, confirmaciones de entrega (*double check*)
- Definir las estructuras de datos en el servidor
  - Para la gestión de usuarios y sus amistades
  - Para la gestión de los mensajes y sus confirmaciones, y de las peticiones de amistad
- Definir las estructuras de datos en el cliente
  - Para los mensajes recibidos, las entregas confirmadas, los amistades actuales,…
- Definir en el cliente cómo recibir los mensajes, p.ej. con una alarma
- Definir la política de almacenamiento de datos en el servidor
  - Persistencia: cuando y cómo escribir a disco, cómo recuperar de una caída o un apago
  - Guardar los mensajes para siempre (¡listos para cuando los pida el NSA!)

Otros aspectos…

- Otros aspectos
  - Diseñar la interfaz de usuario (GUI): en modo texto
  - Interceptar el control-C, tanto en el cliente como en el servidor
  - Implementar un servidor multi-hilo y gestionar la concurrencia
- Implementar el envío de ficheros, posibilidades:
  1. Igual que mensajes pero con `void*`
     - Se envía dentro del mensaje SOAP codificado con base64 (genera mensajes grandes)
  2. Con MIME multipart, posibilidades:
     a) Mecanismo estándar MTOM (`http://www.w3.org/TR/soap12-mtom`)
     b) Mecanismo no-estándar SwA (`http://www.w3.org/TR/SOAP-attachments`)
- Implementar el uso de grupos
  - Grupos con administrador, solo el administrador puede modificar el grupo
  - ¡Cuidado! No necesariamente todos son amigos de todos
- Otras Mejoras
  - Utilizar SSL / Comprimir ficheros antes de enviarlos