CORBA: Servicios de objetos

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas

Ingeniería Informática

Universidad Carlos III de Madrid

Contenidos

1. Servicios de CORBA
2. El servicio de eventos de OMG
3. Recomendaciones para la implementación de consumidores y proveedores

Contenidos

1. **Servicios de CORBA**
2. El servicio de eventos de OMG
3. Recomendaciones para la implementación de consumidores y proveedores

Servicios de Objetos CORBA

- Similares a librerías de alto nivel.
- Gestión de recursos (objetos) distribuidos.
- Estandarización de servicios comunes a varias arquitecturas.
- Servicios no dependen del cliente ni del tipo de datos.
- Tratamiento de excepciones / situaciones anómalas.

Servicios de Objetos CORBA

- Objetivos:
  - Separación del *interface* y la implementación.
  - Las referencias a objetos se especifican mediante *interfaces*.
  - Clientes dependen de los *interface*.
  - *Interfaces* soportan mecanismos de herencia.
  - Uso de excepciones y códigos de notificación.

Servicios de Objetos CORBA

- Principios de diseño:
  - Ofertar una solución eficiente.
  - Posibilidad de combinación.
  - Ofertar servicios genéricos (independientes de la plataforma).
  - Permitir implementaciones tanto locales como remotas.
  - La calidad del servicio depende de la implementación.
  - Uno de *interface* con soporte de *callback*.
  - No utiliza espacios con identificadores globales.

Servicios de Objetos CORBA

- *Naming Service*
- *Concurrency Service*
- *Event Service*
- *Logging Service*
- *Security Service*
- *Time Service*
- *Notification Service*
- *Transactions Service*

Contenidos

1. Servicios de CORBA
2. **El servicio de eventos de OMG**
3. Recomendaciones para la implementación de consumidores y proveedores

Servicio de eventos OMG

- Paradigma *peer-to-peer* en CORBA: uso *callback cliente*.
- Necesidades de:
  - Modelo de comunicaciones desacopladas.
  - Soporte a multidifusión.
- Términos cliente-servidor son significativos respecto a una única petición.

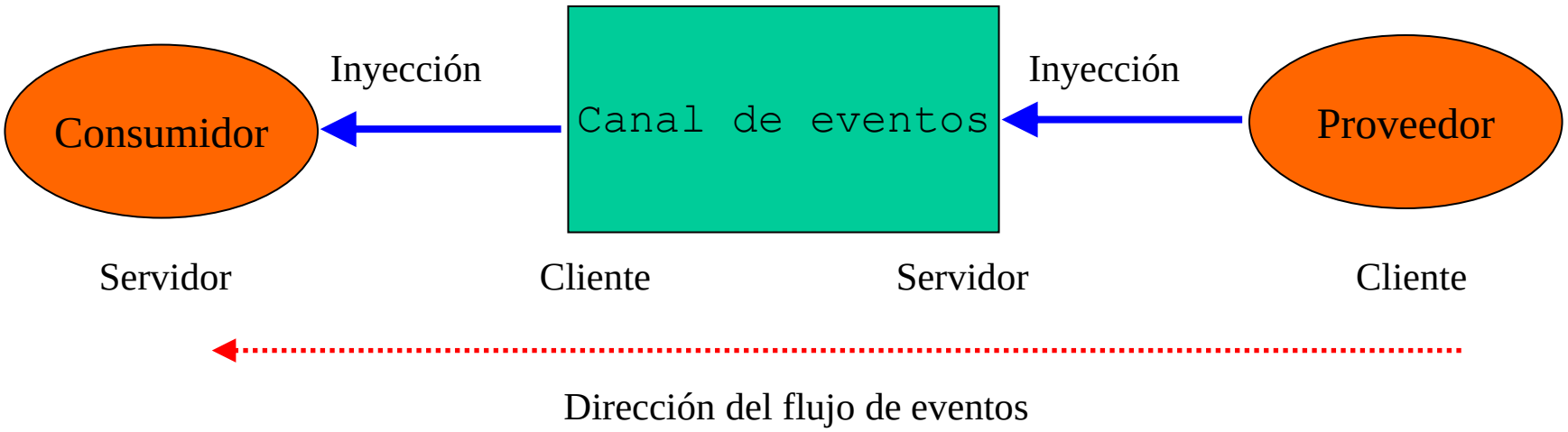
Servicio de eventos OMG

- Limitaciones en el uso de *callbacks*:
  - Igualdad de las referencias a objeto.
    - Is_equivalent()
  - Persistencia de los *callbacks*.
  - Fallo de *callbacks*.
  - Escalabilidad.
  - Acoplamiento.

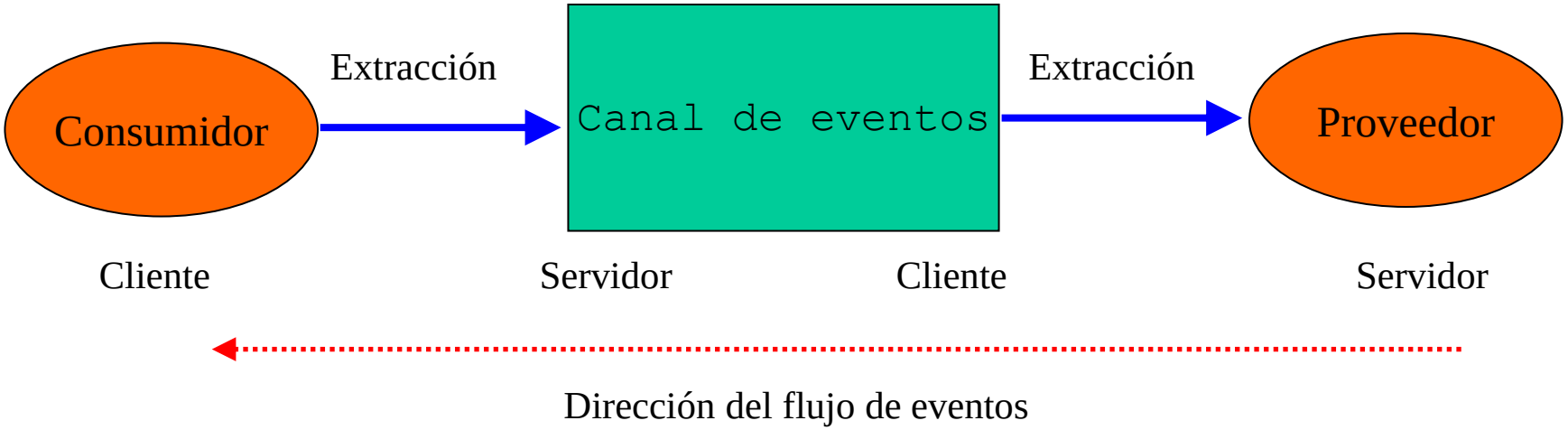
Servicio de eventos OMG

- Componentes:
  - Proveedores.
  - Consumidores.
  - Canal de eventos.
- Modelos asociados al servicio de eventos OMG:
  - Modelo de inyección (*push*)
  - Modelo de extracción (*pull*)

Modelo de inyección (*push*)



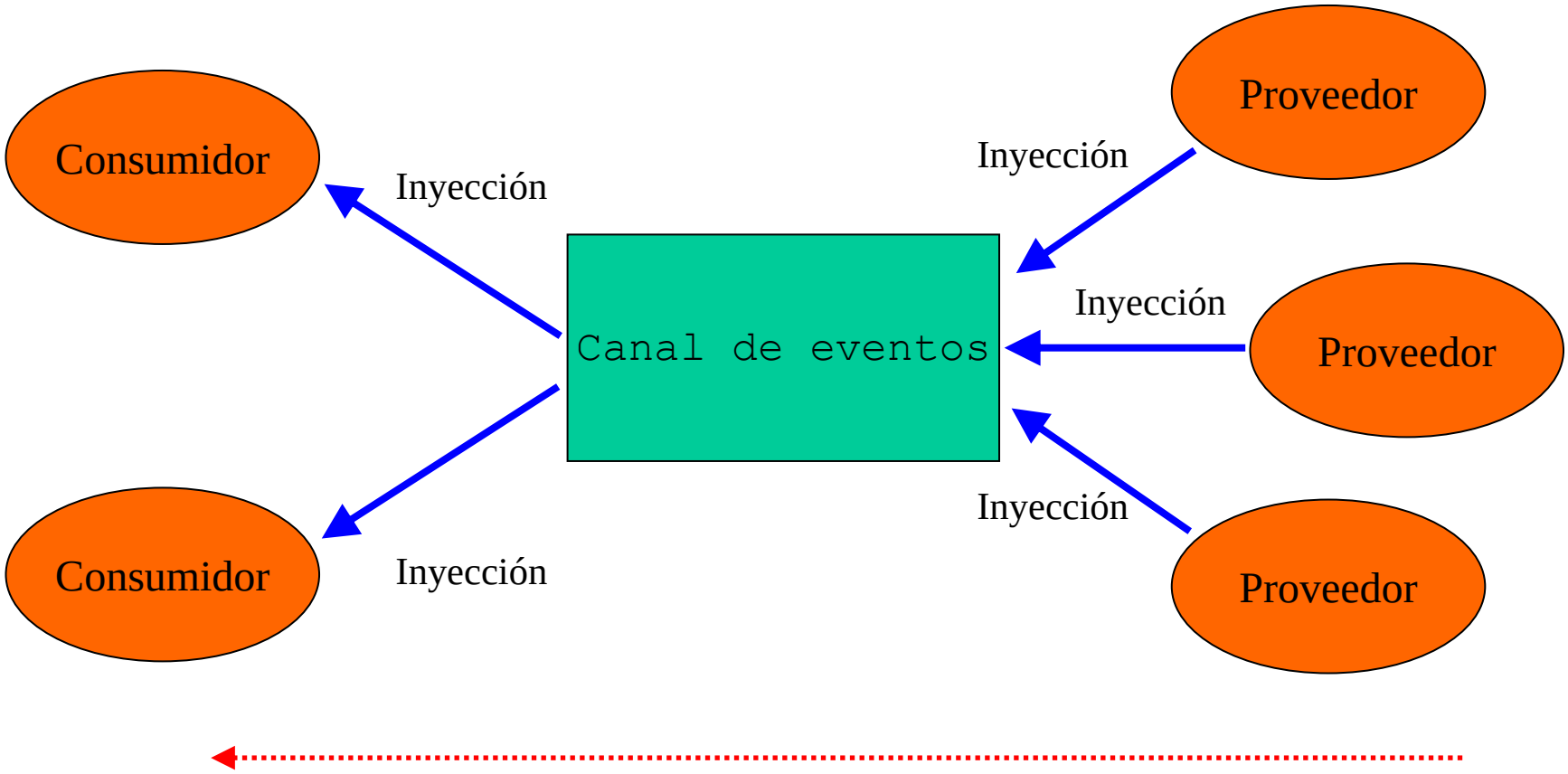
Modelo de extracción (*pop*)



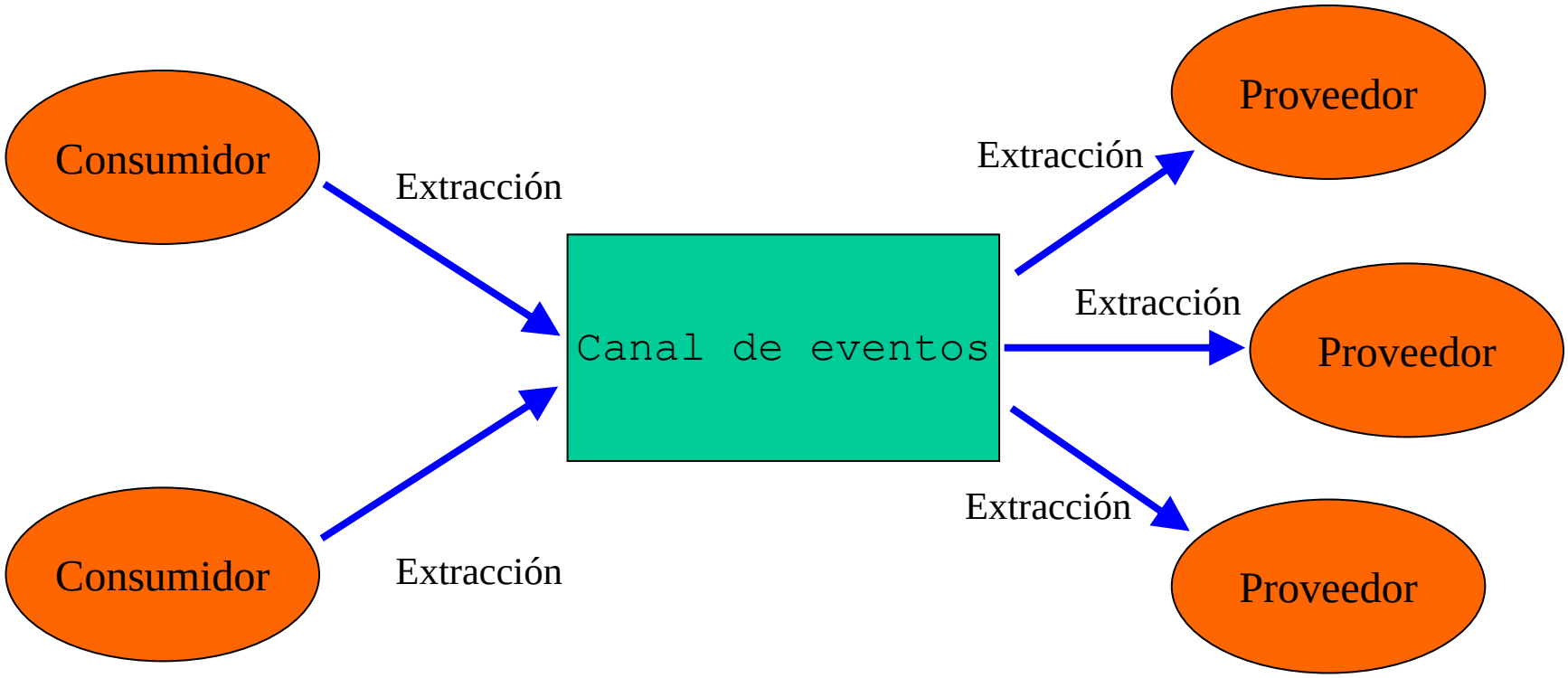
Servicio de eventos OMG

- Modelos proporcionados por el canal de eventos:
  - Modelo canónico de inyección.
  - Modelo canónico de extracción.
  - Modelo híbrido de inyección/extracción.
  - Modelo híbrido de extracción/inyección.

Modelo canónico de inyección:
el canal de eventos representa un notificador

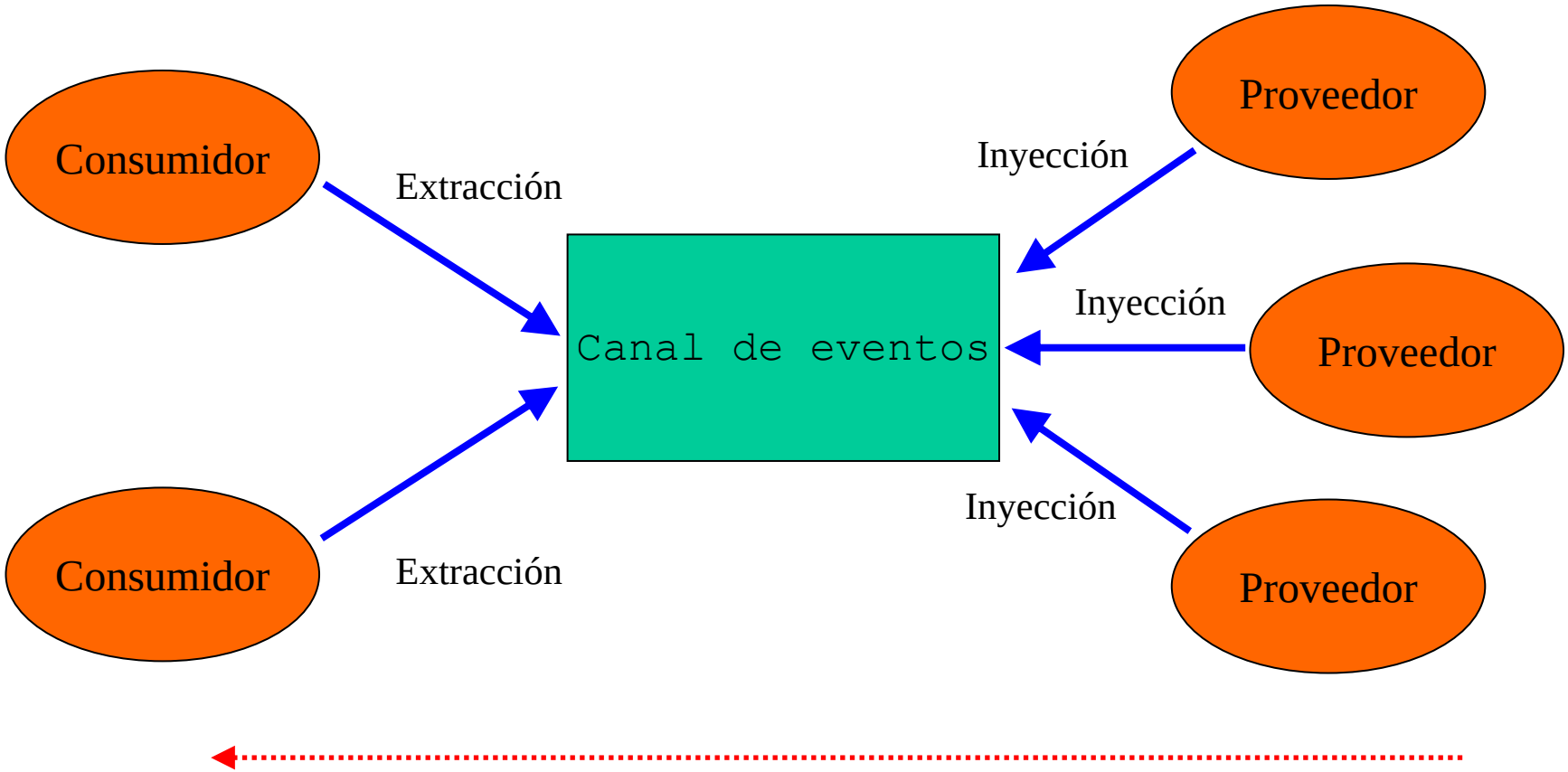


Modelo canónico de extracción:
el canal de eventos representa un procurador

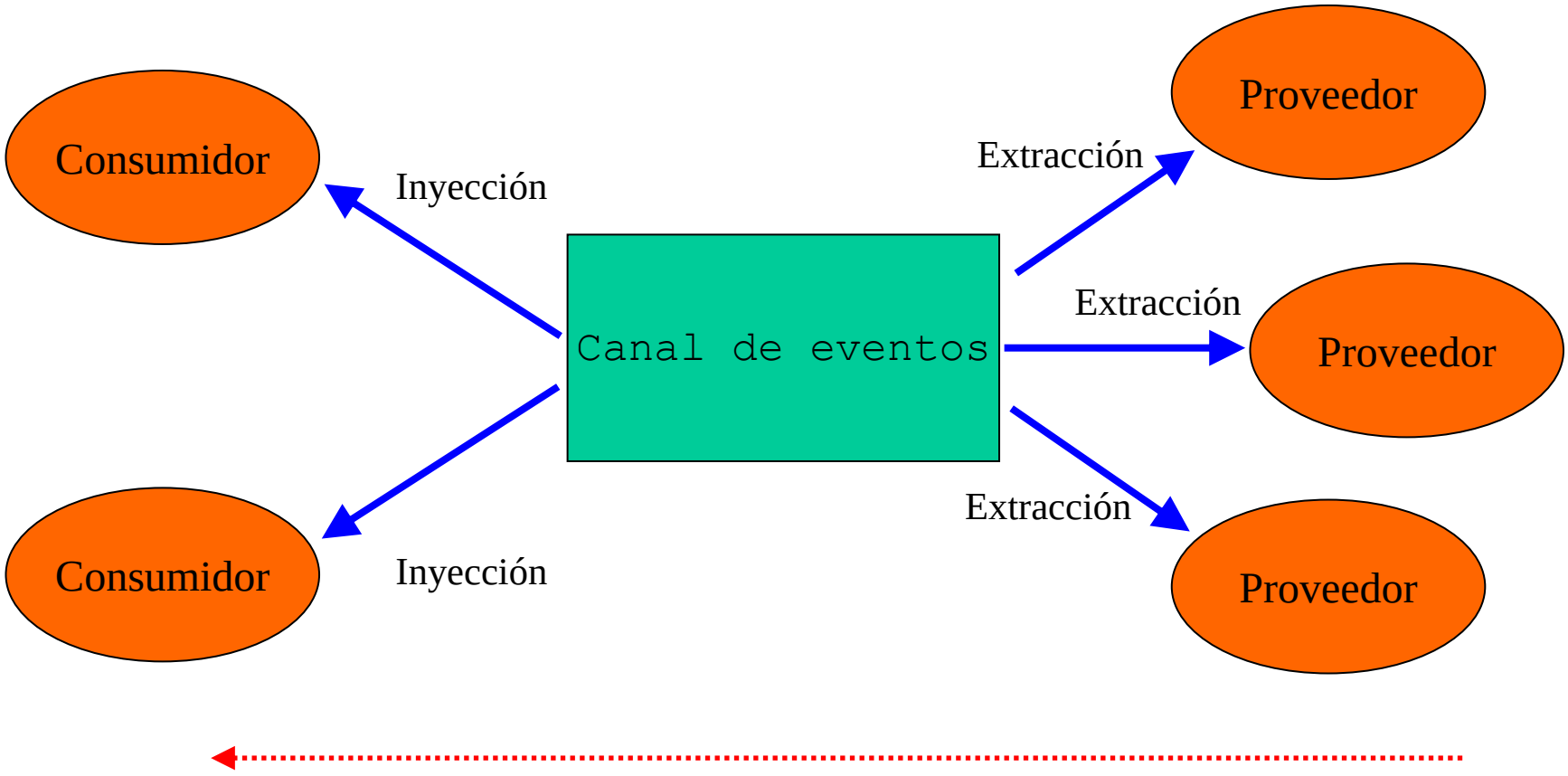


Dirección del flujo de eventos

Modelo hibrido de inyección/extracción:
el canal de eventos representa una cola

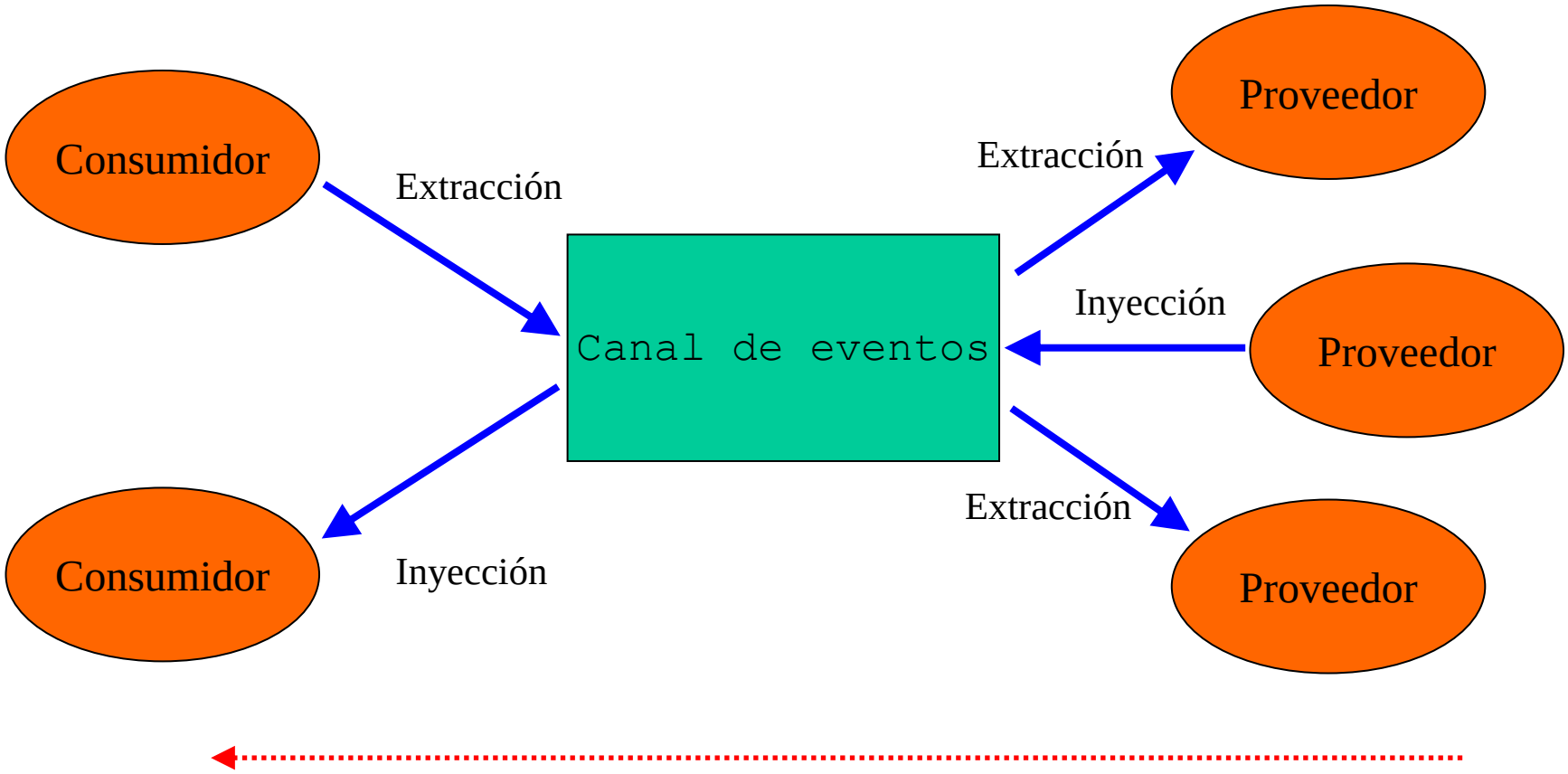


Modelo hibrido de inyección/extracción:
el canal de eventos representa un agente inteligente



Modelo mixto:

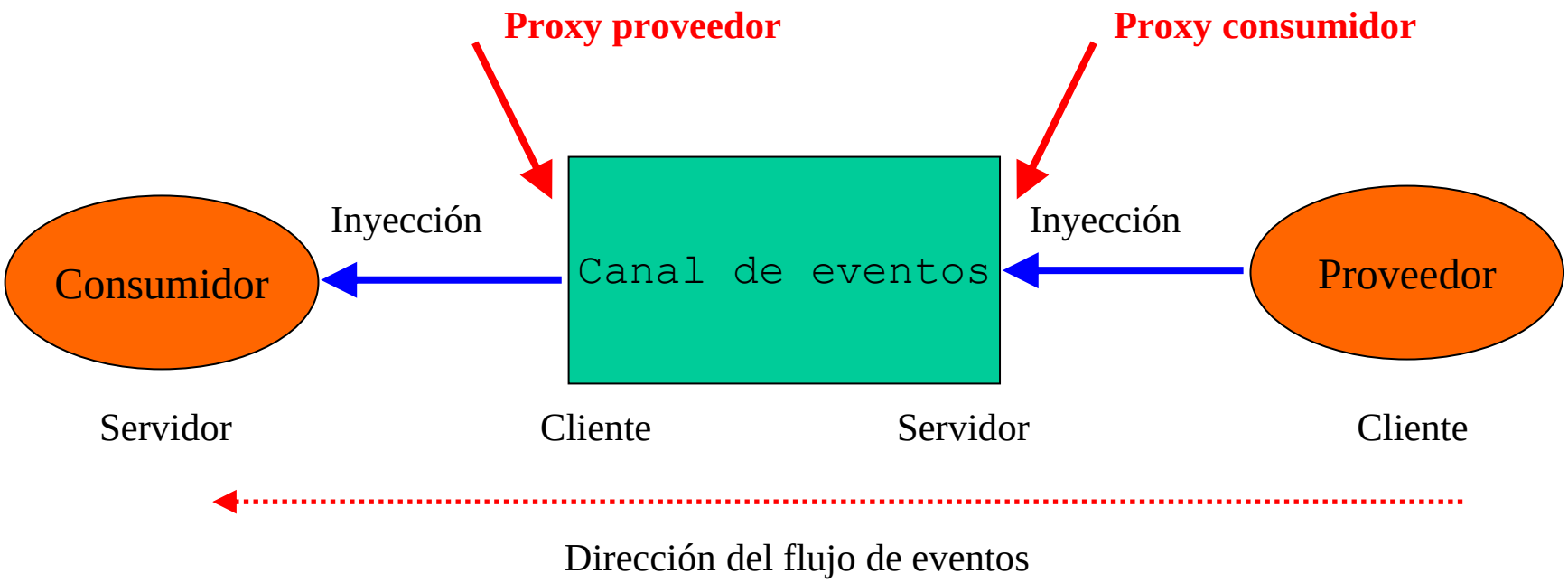
Cada consumidor recibe todos los eventos de todos los proveedores



Contenidos

1. Servicios de CORBA
2. El servicio de eventos de OMG
3. **Recomendaciones para la implementación de consumidores y proveedores**

Implementación de consumidores y proveedores



Implementación de consumidores y proveedores

1. Crear canal de eventos.
2. Implementar de servidor:
   - Consumidor de inyección.
   - Proveedor de extracción.
3. Obtener de una referencia del canal de eventos.
4. Obtener referencias:
   - `ConsumerAdmin` para registrar consumidor.
   - `SupplierAdmin` para registrar proveedor.
   - `Destroy`
5. Obtener referencia al objeto *proxy* desde las referencias anteriores.
6. Llamar a la operación de conexión.

Obtener una referencia del canal de eventos

```
import org.omg.CosNaming.*;
import org.omg.CosEventChannelAdmin.*;
import org.omg.CosEventComm.*;
import org.omg.CORBA.Any;

 org.omg.CosEventChannelAdmin.EventChannel ecs = null;
 org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(argv, null);
 try {
      NamingContext nc =NamingContextHelper.narrow
                (orb.resolve_initial_references("NameService"));
      NameComponent [] name = {new NameComponent(
                ChannelServer.CHANNELNAME, "")};
      ecs = EventChannelHelper.narrow(nc.resolve(name));
      } catch (Exception ex) {
         ex.printStackTrace();
```

Obtener referencias del canal de eventos para registrar proveedor o consumidor

```
interface EventChannel
  {
    ConsumerAdmin for_consumers();
    SupplierAdmin for_suppliers();
    void destroy();
  };
```

Obtener referencia al objeto *proxy*

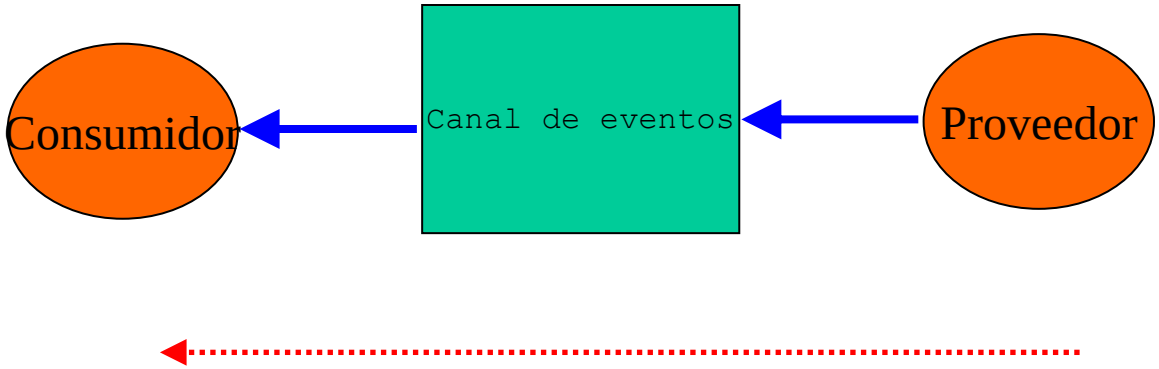
```
interface ConsumerAdmin
  {
    ProxyPushSupplier obtain_push_supplier();
    ProxyPullSupplier obtain_pull_supplier();
  };

interface SupplierAdmin
  {
    ProxyPushConsumer obtain_push_consumer();
    ProxyPullConsumer obtain_pull_consumer();
  };
```

Obtener referencia al objeto *proxy*

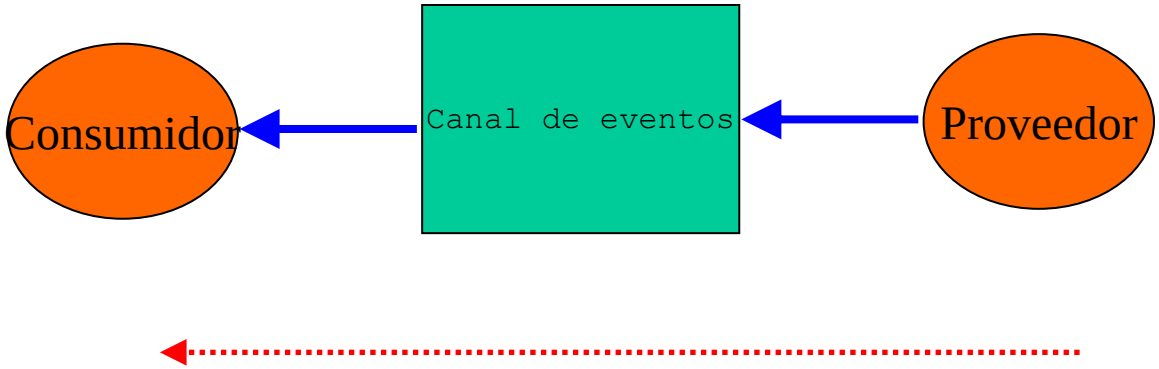
```
interface ProxyPushConsumer: CosEventComm::PushConsumer
{
  void connect_push_supplier(in CosEventComm::PushSupplier push_supplier)
                             raises (AlreadyConnected);
};
interface ProxyPullSupplier: CosEventComm::PullSupplier
{
  void connect_pull_consumer(in CosEventComm::PullConsumer pull_consumer)
                             raises (AlreadyConnected);
};
interface ProxyPullConsumer: CosEventComm::PullConsumer
{
  void connect_pull_supplier(in CosEventComm::PullSupplier pull_supplier)
                             raises (AlreadyConnected, TypeError);
};
interface ProxyPushSupplier: CosEventComm::PushSupplier
{
  void connect_push_consumer(in CosEventComm::PushConsumer push_consumer)
                             raises (AlreadyConnected, TypeError);};
```

Proveedor por inyección




```
SupplierAdmin     sa;
ProxyPushConsumer ppc;
sa  = ecs.for_suppliers();
ppc = sa.obtain_push_consumer();
…
ppc.push( event );
```

Consumidor por inyección




```
ConsumerAdmin     ca;
ProxyPushSupplier pps;
ca  = ecs.for_consumers();

pps = ca.obtain_push_supplier();
PushConsumer pt = new
   _PushConsumerImplBase();
orb.connect(pt);
pps.connect_push_consumer( pt );
```

Servicio de eventos OMG

- Consideraciones del modelo de inyección:
  - Es el más difundido.
  - Menores requisitos de almacenamiento.
  - Evita sobrecarga de muestreo.
- Consideraciones del modelo de extracción:
  - Delega en el muestreo por parte del consumidor.
  - Es necesaria una política de eliminación de eventos.
  - Puede producir alto tráfico de red.

Servicio de eventos OMG

- Limitaciones del servicio de eventos:
  - Proveedores múltiples.
  - Falta de fiabilidad.
  - Falta de filtrado.
  - Falta de especificaciones.
  - Mensajes asíncronos y llamadas independientes del tiempo.

CORBA: Servicios de objetos

Grupo ARCOS

Desarrollo de Aplicaciones Distribuidas

Ingeniería Informática

Universidad Carlos III de Madrid