**TRABAJO PRACTICO**



**NOMBRE:** SIRIA BARRIGA VILLARROEL

**Nº REGISTRO:**219215944

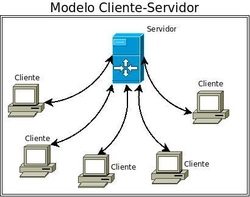
**CARRERA**: ING. INFORMATICA

**MATERIA:** PROG. DE APLICACIÓN DE TIEMPO REAL

**AÑO:** 2023

**Liste al menos 3 medios (a nivel de software) por los que se puede establecer comunicaciones para las Aplicaciones en Tiempo Real**

**Socket**



Cuando dos procesos que están en hosts diferentes necesitan intercambiar información a través de la red, ya sea la red local o Internet, es necesario que abran un socket para establecer la comunicación y también para intercambiar cualquier flujo de datos. Generalmente el flujo de datos se hace de forma fiable y ordenada, no obstante, dependemos de si los procesos están utilizando el protocolo TCP o el protocolo UDP, a continuación, os explicamos brevemente las principales características de ambos protocolos:

* El protocolo TCP es un protocolo de la capa de transporte que es orientado a conexión, esto significa que antes de intercambiar los datos reales hay un paso previo para establecer una comunicación. Este protocolo también garantiza que toda la transmisión de los datos se hace sin errores, el propio TCP se encarga de reenviar los datos nuevamente en caso de que el receptor no los reciba a tiempo o los reciba dañados, además, garantiza también el orden, por lo que nos aseguramos de que los procesos van a recibir todos los datos en orden desde su origen.
* En el caso del protocolo UDP, no es orientado a conexión, no hay un paso previo en la comunicación, sino que se envían los datos directamente. Este protocolo no garantiza que la transmisión se realice sin errores, aunque hará todo lo posible para que sí lo haga, además, tampoco garantiza el orden de los datagramas que el origen envíe al destino. La parte positiva de UDP es que tiene una cabecera muy pequeña y es muy rápido, ya que no hay una fase de establecimiento de la conexión.

Para que dos procesos puedan comunicarse entre sí, es necesario que un proceso sea capaz de localizar al otro, y que ambos procesos sean capaces de intercambiar información a través de la red. Por supuesto, estamos usando una arquitectura cliente-servidor, por lo que uno de los dos procesos es el que debe iniciar la comunicación. Para que dos procesos se comuniquen necesitamos tener un socket.

Un socket, independientemente de si usamos el protocolo TCP o el protocolo UDP, viene definido por los siguientes parámetros:

* Protocolo de la capa de transporte utilizado: TCP o UDP
* Dirección IP de origen: puede ser una IP pública o privada.
* Dirección IP de destino: puede ser una IP pública o privada.
* Puerto de origen o local: este puerto suele estar entre los puertos 49152 hasta 65535, son los denominados como puertos dinámicos o privados. Pero no tiene por qué ser así, puede utilizar cualquier puerto de origen, por ejemplo, un servidor web que usa el puerto 443 para las conexiones HTTPS usará el puerto de origen 443 para las conexiones.
* Puerto de destino o remoto: este puerto puede ser cualquier puerto, es necesario que el otro proceso o host esté escuchando este puerto.

Gracias a todos estos parámetros, podemos hacer que cada una de las conexiones que se realicen sean únicas, de esta forma, tanto el origen como el destino podrán identificar perfectamente la conexión y empezar a intercambiar datos.

**CORBA**

CORBA (Common Object Request Broker Architecture) es un estándar definido por el grupo de gestión de objetos (OMG) que permite el funcionamiento conjunto de componentes de software escritos en distintos lenguajes informáticos y que se ejecutan en distintos sistemas.

CORBA es un estándar para distribuir objetos a través de redes de modo que las operaciones en dichos objetos puedan llamarse de forma remota. CORBA no está asociado con ningún lenguaje de programación en particular y cualquier lenguaje que tenga un enlace CORBA puede utilizarse para llamar e implementar objetos CORBA. Los objetos se describen en una sintaxis denominada IDL (Interface Definition Language).

CORBA incluye cuatro componentes:

**ORB (Object Request Broker)**

ORB (Object Request Broker) maneja la comunicación, ordenación y desordenación de parámetros, de modo que el manejo de parámetros es transparente para aplicaciones de cliente y servidor CORBA.

**Servidor CORBA**

El servidor CORBA crea objetos CORBA y los inicializa con un ORB. El servidor coloca las referencias a los objetos CORBA dentro de un servicio de denominación de modo que los clientes puedan acceder a los mismos.

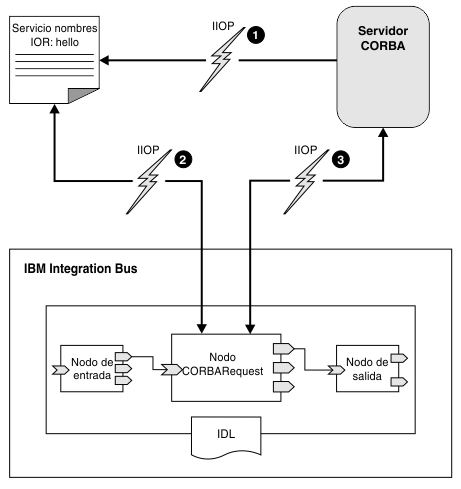
**Servicio de nombres**

El servicio de denominación mantiene referencias a objetos CORBA.

**Nodo CORBARequest**

El nodo CORBARequest actúa como un cliente CORBA.

El diagrama siguiente muestra las capas de comunicación entre IBM® Integration Bus y CORBA.



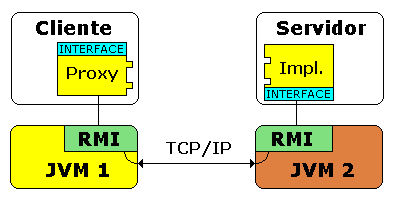
El diagrama ilustra los pasos siguientes.

1. Las aplicaciones de servidor CORBA crean objetos CORBA y colocan las referencias a objetos en un servicio de denominación de modo que los clientes puedan llamar a los objetos.
2. En el momento de despliegue, el nodo contacta con un servicio de denominación para obtener una referencia de objeto.
3. Cuando llega un mensaje, el nodo utiliza la referencia de objeto para llamar a una operación en un objeto del servidor CORBA.

**RMI (Remote Method Invocation)**

RMI (Remote Method Invocation) es un mecanismo que permite realizar llamadas a métodos de objetos remotos situados en distintas (o la misma) máquinas virtuales Java, compartiendo así recursos y carga de procesamiento a través de varios sistemas.

RMI permite exportar objetos como objetos remotos para que otro proceso remoto pueda acceder directamente como un objeto Java. Todos los objetos de una aplicación distribuida basada en RMI deben ser implementados en Java. Esta es una de las principales ventajas de RMI, ya que RMI forma parte del API de Java, con lo que la integración de objetos remotos en aplicaciones distribuidas se realiza sin necesidad de usar recursos adicionales (como por ejemplo un lenguaje de descripción de interfaces o IDL). De hecho, se utiliza la misma sintaxis para una llamada a un objeto remoto o un objeto local.



El cliente invoca a los objetos remotos mediante la interfaz remota. Un servicio de nombres (registro RMI) reside en el host proporcionando el mecanismo que el cliente usa para encontrar uno más servidores iniciales RMI.

La interacción con el objeto remoto se lleva a cabo a través de la interfaz remota. Esencialmente, ésta describe los métodos que pueden ser invocados de forma remota, y que el objeto remoto implementa. Cuando se obtiene una referencia a un objeto remoto, el objeto no se envía a través de la red al cliente que lo solicita. En su lugar se genera un objeto proxy o stub que constituye el proxy de la parte del cliente del objeto remoto. Todas las interacciones del cliente se realizarán con esta clase stub, la cual es responsable de gestionar los datos entre el sistema local y el remoto. Muchos clientes pueden tener referencias a un único objeto remoto. Cada cliente tiene su propio objeto stub que representa al objeto remoto, pero dicho objeto remoto NO se replica.