Un ***sistema distribuido*** funciona mediante la interconexión de varios equipos. Las tres ***características fundamentales*** son:

1. Está ***formado por más de un elemento*** independiente.
2. Los ***elementos*** del sistema distribuido ***no están sincronizados entre sí,*** cada uno tiene su reloj.
3. Todos ***los elementos están conectados a una red*** que permite su comunicación.

El ***Cloud Computing / La Nube*** es el sistema distribuido que más se usa en la actualidad. Este ofrece un servicio al usuario a través de internet, en el que el proveedor abstrae los mecanismos con los que se proporcionan los servicios. Este proceso de abstracción es la virtualización de servicios.

Las aplicaciones que forman el sistema distribuido se comunican entre sí con una finalidad. Este proceso implica los siguientes ***conceptos fundamentales:***

1. ***Mensaje:*** Información que se intercambia entre las aplicaciones.
2. ***Emisor:*** Aplicación que envía el mensaje
3. ***Receptor:*** Aplicación que recibe el mensaje. Un mismo mensaje puede tener varios.
4. ***Paquete:*** Es la unidad básica de información que se usa para transmitir un mensaje a través de una red. Este debe dividirse en uno o varios paquetes.
5. ***Canal de comunicación:*** Medio por el que se transmiten los paquetes, conectando al emisor y receptor.
6. ***Protocolo de comunicaciones:*** Conjunto de reglas que indican cómo intercambiar un paquete entre los elementos interconectados. Define la secuencia de paquetes que se deben transmitir y el formato del mensaje.

……………………………………………………………………………………………………………

Para que las aplicaciones puedan comunicarse deben existir ***mecanismos*** que lo permitan. Estos están formados por elementos hardware y software, que se organizan en una ***jerarquía / pila de protocolos.*** Esta establece cómo deben trabajar entre ellos.

El ***protocolo IP*** es el estándar de comunicaciones de Internet y en la mayoría de sistemas distribuidos. Este ***define una pila de protocolos completa,*** que consta de los siguientes niveles:

1. ***Nivel de Red:*** Es el más bajo, está compuesto por el hardware de comunicaciones y sus controladores. Se encarga de transmitir los paquetes. Funciona con redes LAN, WAN o ambas.
2. ***Nivel de Internet / IP:*** Se compone del software encargado de dirigir los paquetes por la red, asegurando que lleguen a su destino.
3. ***Nivel de Transporte:*** Compuesto por software con la función de crear el canal de comunicación, descomponer el mensaje en paquetes y gestionar su transmisión (UDP y TCP).
4. ***Nivel de Aplicación:*** Lo componen las aplicaciones que forman el sistema distribuido. Estas hacen uso del resto de niveles para transferir mensajes entre ellas.

Las aplicaciones de un sistema distribuido se sitúan en un nivel superior al de aplicación, cuando un emisor decide enviar un mensaje se producen las siguientes operaciones:

1. La aplicación del emisor entrega el mensaje al nivel de transporte.
2. El protocolo del nivel de transporte descompone el mensaje en paquetes y los pasa al nivel de Internet.
3. El nivel de Internet localiza al receptor y calcula la ruta completa para los paquetes y entrega los paquetes al nivel de red.
4. El nivel de red transmite los paquetes hasta el receptor.
5. Una vez que el receptor los recibe, pasa los paquetes de nuevo al nivel superior de red y a su vez al nivel de Internet.
6. El nivel de Internet comprueba que los paquetes han llegado correctamente. Si es así los envía al nivel de transporte.
7. El nivel de transporte los agrupa para formar el mensaje y lo envía al nivel de aplicación.
8. La aplicación receptora recibe el mensaje.

El ***protocolo de transporte TCP*** es el más usado en la ***pila IP.*** Este ***divide los mensajes*** del nivel aplicación ***en paquetes*** que se envían al nivel Internet y los combina para formar mensajes que se pasan al nivel superior aplicación. Características:

1. ***Garantiza que los datos no se pierden,*** siempre que la comunicación sea posible.
2. Garantiza que ***los mensajes lleguen en orden*** en el que se envían.
3. Es un ***protocolo orientado a conexión.***

Un ***protocolo orientado a conexión*** usa un ***canal de comunicación*** entre dos aplicaciones que permanece abierto durante un tiempo determinado. Cuando se transmite información usando uno de estos protocolos se llevan a cabo las siguientes operaciones:

1. ***Establece la conexión:*** Se realiza al inicio de las comunicaciones y sirve para crear el canal de comunicación, que estará abierto hasta que uno de los dos lo cierre.
2. ***Envío de mensajes:*** Se puede realizar tantas veces como sea necesario mientras la conexión sea estable. Usa el canal de comunicación para enviar los mensajes en paquetes.
3. ***Cierre de la conexión:*** Se realiza cuando se quiere interrumpir la comunicación.

Un ***protocolo no orientado a conexión*** usa un ***canal de comunicación propio para cada mensaje,*** por lo que ***no hay que establecer la conexión ni cerrarla.***

El ***protocolo de transporte UDP*** es otro de los fundamentales de la ***pila IP.*** Este funciona de manera similar a TCP pero tiene unas características diferentes:

1. Es un protocolo no orientado a conexión, por lo que es más rápido.
2. No garantiza que lleguen todos los mensajes siempre.
3. No garantiza que los mensajes lleguen en orden de envío.
4. Permite enviar mensajes de 64 KB como máximo.
5. Los mensajes se denominan ***“Datagramas”.***

Los ***sockets*** son el ***mecanismo de comunicación básico*** fundamental ***para realizar transferencias de información entre aplicaciones*** a través de redes internas LAN o Internet. Proporcionan una abstracción de la pila de protocolos, ofreciendo una interfaz sencilla para intercambiar mensajes.

***Un socket / enchufe representa el extremo de un canal de comunicación.*** Para establecer la conexión entre dos aplicaciones ambas deben crear sus sockets y conectarlos entre sí. ***Una vez se han conectado se crea una tubería privada a través de la red,*** que permite a las aplicaciones enviar y recibir mensajes. Para ello, las aplicaciones escriben y leen en sus sockets.

Una máquina se identifica por su dirección IP, en ella pueden haber varias aplicaciones con sus sockets funcionando. ***Para identificar las aplicaciones se usan los puertos,*** estos son un ***número que identifica al socket de una aplicación*** en la máquina. Un puerto solamente puede estar asignado a un socket en concreto. Cuando una ***aplicación crea un socket y lo asigna a un puerto*** se dice que ***el socket está escuchando por ese puerto.***

Hay dos t***ipos de sockets*** dependiendo de su funcionalidad y del protocolo de nivel de transporte que usan:

1. ***Sockets stream:*** Son ***orientados a conexión*** y hacen uso del ***protocolo TCP.*** Son fiables y aseguran el orden de entrega correcto. Estos se usan para comunicarse siempre con el mismo receptor, manteniendo el canal de comunicación abierto en ambas partes.

Cuando se establece una conexión entre sockets stream se deben seguir una serie de pasos. En esta secuencia uno de los elementos de la comunicación debe actuar como ***proceso servidor*** y otro como ***proceso cliente.*** ***El servidor crea el socket primero y espera a que el cliente cree el suyo y se conecte, creando un canal.***

***Proceso del cliente:***

1. ***Creación del socket:*** Crea el socket cliente y le asigna un puerto.
2. ***Conexión del socket:*** Se localiza el socket servidor y se crea el canal de comunicación que los une. Para ello se debe conocer la IP y el puerto.
3. ***Envío y recepción de mensajes:*** Ya se pueden enviar y recibir mensajes mediante operaciones de lectura y escritura sobre el socket cliente.
4. ***Cierre de conexión:*** Para terminar se debe cerrar el socket cliente.

***Proceso del servidor:***

1. ***Creación del socket:*** Crea el socket servidor.
2. ***Asignación de dirección y puerto:*** Mediante la operación bind se le asigna una dirección IP y un número de puerto concreto al socket.
3. ***Escucha:*** Se debe configurar para que escuche por el puerto asignado. Mediante la ***operación listen,*** se prepara al socket para ***aceptar conexiones.***
4. ***Aceptación de conexiones:*** Mediante la ***operación connect*** ***conecta un cliente al socket.*** La ***operación accept bloquea el proceso servidor ya que espera una conexión del cliente.*** Cuando llega una petición de conexión, el servidor crea un ***nuevo socket*** que se usará solamente en la conexión con ***dicho cliente.*** Así el socket principal del servidor está libre para mantener la escucha de más peticiones.
5. ***Envío y recepción de mensajes:*** Ya se pueden enviar y recibir mensajes a través de operaciones de escritura y lectura sobre su nuevo socket.
6. ***Cierre de la conexión:*** Se cierra el socket de la conexión con el cliente pero no el principal.
7. ***Sockets datagram: No son orientados a conexión*** y hacen ***uso del protocolo UDP.*** Pueden usarse para ***enviar datagramas*** (mensajes) ***a muchos receptores,*** ya que ***usan un canal temporal para cada envío.*** No son fiables ni aseguran el orden de entrega correcto. En estos ***no se diferencian los sockets clientes de los servidores.***

Todas las aplicaciones que usan este tipo de socket ***siguen los siguientes pasos*** para enviar mensajes:

1. ***Creación del socket:*** Crea un socket.
2. ***Asignación de dirección y puerto:*** Asigna una IP y un puerto al socket mediante la ***operación bind.***
3. ***Envío y recepción de mensajes:*** Con esto ya se pueden enviar y recibir datagramas mediante la ***operación send.***
4. ***Cierre de la conexión:*** Se cierra usando la operación close.

Los sockets datagram ***usan un mismo socket*** para enviar a distintos receptores, por lo que solo habrá que cerrar ese al finalizar.

En Java existen tres ***clases principales*** que permiten la comunicación por sockets:

1. ***java.net.Socket:*** Para la creación de sockets stream cliente.
2. ***java.net.ServerSocket:*** Para la creación de sockets stream servidor.
3. ***java.net.DatagramSocket:*** Para la creación de sockets datagram.

Los ***datagramas*** se representan en Java con objetos de la ***clase DatagramPacket,*** uno de estos tiene un datagrama de tamaño fijo asociado a un destinatario o emisor, es decir a una IP y un puerto concretos.

El ***método send()*** de la ***clase DatagramSocket*** permite ***enviar datagramas*** de la anterior clase siempre que se hayan inicializado bien. El ***método receive()*** permite ***recibir los datagramas y almacenarlos en objetos*** de la clase ***DatagramPacket.*** Esta clase ***usa*** un objeto de la ***clase InetAddress para representar la*** ***dirección IP.***

***Las direcciones IP y los sockets*** son las ***herramientas fundamentales*** que usan las ***aplicaciones distribuidas*** que usan protocolos de transporte.

***Un*** ***modelo de comunicaciones*** ***es una arquitectura*** ***que especifica cómo se comunican entre sí los diferentes elementos de una aplicación distribuida,*** además define el número de elementos y la función de cada uno. Los modelos ***más usados son*** los ***cliente/servidor*** y los de ***comunicación en grupo,*** además de los ***P2P.***

* El ***modelo cliente/servidor*** es el más sencillo, en el que ***un servidor ofrece servicios a uno o más clientes.*** El servidor debe ser fácilmente accesible y conocido por los clientes por su IP y nº de puerto. Este modelo se basa en un mecanismo de comunicación mediante ***petición y respuesta.*** Un modelo cliente/servidor normalmente ***se implementa usando sockets stream pero también se pueden implementar con otros tipos.***
* El ***modelo de comunicación en grupo*** es la alternativa más común al anterior. A diferencia del último, ***no existen roles diferenciados,*** en este existe un ***conjunto de dos o más elementos que cooperan entre sí.*** ***A este conjunto se le llama grupo y los elementos que lo forman son todos iguales.*** En este modelo ***los mensajes se transmiten mediante radiado, que los envía de manera simultánea a todos los miembros del grupo,*** ***sin establecer comunicaciones punto a punto.*** Este modelo puede implementarse usando sockets stream o datagrams:

1. Si se usan ***sockets stream cada elemento debe establecer una conexión con el resto,*** por lo que ***cada elemento debe tener un socket.***
2. Si se usan ***sockets datagram cada elemento necesitará un único socket.*** Requiere menos sockets pero es menos fiable.

***La comunicación en grupo*** ***hace uso de*** ***bibliotecas de comunicación.*** Estas son ***software diseñado para implementar modelos*** de comunicación como este último y ofrecen muchas ventajas y facilitan su implementación, configuración y depuración.

***Los modelos cliente/servidor y de comunicación en grupo presentan dos limitaciones:***

1. ***Reparto de roles fijo:*** La función de un elemento no varía nunca.
2. ***Mecanismo de comunicación único:*** Los elementos del sistema distribuido se comunican siempre de la misma forma.

Además de estos dos procedimientos básicos, el ***protocolo IP incorpora el multicast,*** que ***a diferencia de los métodos*** ***unicast*** consiste en ***enviar el mismo mensaje a varios destinatarios de forma simultánea usando una IP multicast.*** ***Para poder recibirlo, todos los sockets deben estar configurados con la misma IP multicast.*** Esto supone un ***riesgo para la seguridad*** de la red.

***Tipos de envío en el protocolo IP***

* **Unicast:*****Envío*** de mensajes ***entre*** un ***único emisor y un receptor*** determinado. Lo emplean los sockets ***TCP y UDP.***
* **Multicast:**Envío de mensajes a un ***grupo de destinatarios,*** distinguidos por una dirección ***IP multicast.***
* **Broadcast:*****Envío simultáneo*** de un mensaje ***a todos los miembros de una red local.***
* **Anycast:**Similar a multicast o broadcast, pero el mensaje ***es entregado solamente al destinatario más cercano en la red.***

Las ***redes P2P / peer to peer*** son uno de los modelos de ***comunicación híbrida más potentes,*** usados para ***intercambio de archivos*** y aplicaciones comerciales. Una red P2P está ***formada por un grupo de elementos distribuidos que colaboran por un objetivo común.*** En estas, los elementos ***pueden*** ***desempeñar los roles de cliente/servidor,*** es decir, ***cada elemento puede variar según la función que necesite*** en determinado momento, lo que le da mucha ***flexibilidad*** al modelo P2P. Además estas eliminan los problemas más importantes del cliente/servidor tradicional, ya que puede seguir funcionando con otro servidor si uno cae.