Un **sistema distribuido** está formado por un conjunto de programas (sistemas), los programas se ejecutan en procesos diferentes, se comunican entre sí y contribuyen a la ejecución de una tarea común - **168158**
Un sistema distribuido también puede ser visto como un conjunto de computadoras autónomas que dan la ilusión de ser un sistema único y homogéneo que explota recursos locales y remotos para una tarea. **-166831**
Los procesos pueden ser ejecutados en la misma o en diferentes maquinas. **163221**
Las computadoras estan ligadas por una infraestructura ya sea de comunicación y de un software de sistemas y de aplicación. - **163221**
Las máquinas de un sistema distribuido pueden:

- Estar en el **mismo sitio**
- Estar **interconectadas en sitios diferentes** y comunicarse a través de una rd. **167319**

Los **DOS (Distributed operating Systems)** son sistemas operativos fuertemente acoplados para multiprocesadores o sistemas de cómputo homogéneo, cuyo objetivo principal es esconder y administrar recursos del hardware. **-166831**
Los **NOS (Network Operating Systems)** son sistema operativos débilmente acoplados para sistemas de multicómputo heterogéneos. Su principal objetivo es ofrecer servicios locales a clientes remotos. **-166912**
Sistema Operativo basado en Middleware: Capa adicional sobre los Network Operating Systems que implementa servicios de propósito general, cuyo objetivo y ventaja es ofrecer transparencia en la distribución. **167319**
SOCKETS
Socket: Un socket es la interfaz entre un proceso de aplicación y una capa de transporte. **158434**
Tipos de sockets:

- Sockets de internet **158434**
- Sockets de Unix**158434**
- Sockets X.25 **158434**

Tipos de Internet Sockets;

- **Stream Sockets(SOCK_STREAM): 167319**
  - Connection-oriented
  - Rely on TCP to provide reliable two-way connected communication
- **Diagram Sockets (SOCK_DIAGRAM): 167319**
  - Connection is unreliable
  - Rely on UDP

TCP : for Transmission Control Protocol, is a connection-oriented protocol used to establish and maintain a connection between 2 or more entities. It performs several actions such as segmenting data into packets that can be delivered by the network. Those packets are send and received from the Network Layer. TCP manages the flow control of the exchange. It can request non delivered packets from the sender. Until every packets hasn't been received during a transaction, TCP is still working. This particular aspect of management makes this protocol used for the transfer of sensible data. **403783**
UDP : for User Datagram Protocol, is a non connection-oriented protocol which means that once packets are send, there is no more connection between the 2 entities. Unlike TCP, there is no management of lost packets, which makes it a less reliable but faster protocol. It is used for non required reliable data such as streaming. **403783**
Connection Protocols: Connection-oriented and Connectionless **167319**
Main Operations: socket(), bind(), listen(), connect(), accept(), send(), recv(), sendto(), recvfrom(), close(). **163221**

- **Connect():** connects to a remote host. **163221**
- **Bind():** Es atar el servicio a un puerto **168663**

Miscellaneous routine: getpeername(), gethostname() **163221**

Procesos Sincronos: El cliente hace una peticion al servidor y queda a la espera de la respuesta del servidor para continuar con su ejecución **(168663)**

Procesos Aincronos: El cliente hace una peticion al servidor pero no se queda esperando la respuesta, sino que continúa su ejecución y durante la ejecución es que el servidor me retorna una respuesta **(168663)**

Background:

Two types of "Byte ordering":

¨ **Network Byte Order:** High-order byte of the number is stored in memory at the lowest address

¨ **Host Byte Order:** Low-order byte of the number is stored in memory at the lowest address Ø

Network stack (TCP/IP) expects Network Byte Order **170716**

Conversions: htons() - HosttoNetworkShort ¨

htonl() - HosttoNetworkLong ¨

ntohs() - NetworktoHostShort ¨

ntohl() - NetworktoHostLong **170716**

Socket Structures

- struct sockaddr: Holds socket address information for many types of sockets.
- struct sockaddr_in: A parallel structure that makes it easy to reference elements of the socket address. **168158**

Modelo cliente servidor : El protocolo cliente-servidor es un modo de transacción (a menudo a través de una red) entre varios programas o procesos: uno, llamado cliente, envía peticiones; el otro, llamado servidor, espera las peticiones del cliente y las responde. El servidor ofrece un servicio al cliente. El cliente y el servidor estan generalmente pero no necesariamente en dos máquinas distintas.**403715**

Peer to peer comunicación: Peer-to-peer (a menudo abreviado como P2P) es un modelo de intercambio en red en el que cada entidad es a la vez cliente y servidor, en contraste con el modelo cliente-servidor explicado anteriormente. En este modo de comunicación, dos o más ordenadores se conectan directamente para intercambiar datos. Cada máquina puede funcionar como cliente o como servidor.**403715**

Sockets help the application processes to communicate with each other using standard Unix file descriptors. **163221**

Designación y ligado:

- **Ligado estático:** ninguna llamada al servidor de nombres (llamada resuelta en tiempo de compilación). **- 166912**
- **Ligado a la primera llamada:** consulta del servidor de nombres sólo en la primera llamada. **- 166912**
- **Ligado a cada llamada:** consulta del servidor de nombres a cada llamada. **- 166912**

RPC (Remote Procedure Call): Es un simple sistema de middleware que sirve como herramienta de alto nivel para la implementación del modelo cliente – servidor . Además, cuenta con la ventaja de que se tiene la forma y efecto idéntico al de la llamada a un proceso local. **165974**

Escalibilidad: Adaptación del sistema distribuido para tener más usuarios y a la vez estos obtienen una respuesta rápida para ello se suele añadir procesadores más veloces. Por lo tanto, los componentes no deberían necesitar cambiar cuando la escala del sistema aumenta. **165974**

Transparencia: Los sistemas distribuidos deben de ser percibidos por los usuarios y las aplicaciones como un todo y no como una colección de componentes que

cooperan entre si. Por lo que, cuenta con varias propiedades que todo sistema distribuido debe tener. Por ejemplo, esconder objetos de información sin el conocimiento de donde se encuentran como cuando se consulta una pagina de internet. **165974**

Openess: Se refiere a extensiones y mejoras en un sistema distribuido. Las interfaces detalladas de los componentes deben ser publicadas integrando nuevos componentes con los ya existentes tomando en cuenta diferencias en la representación de los datos de los tipos de interfaz en diferentes procesadores. **165974**

Pros y Contras del Uso de RPC (Remote Procedure Call).168724

Pros:

- Forma y efecto idéntico al de la llamada a un proceso local
- Las aplicaciones no deben ser modificadas al pasar al RPC
- Validación local antes de pasar a la distribución
- Simplicidad de uso

Contras:

- **Semántica compleja en caso de falla**
- Los procesos cliente y servidor pueden fallar independientemente
- La red es fuente de incertidumbre
- **Restricciones sobre los parámetros**
- Paso de estructuras complejas posible pero difícil

Stub: Es el código (oculto) correspondiente al envío de la petición (marshall) y la recepción de la respuesta (unmarshall).Tanto la petición como la recepción ocurren por parte del cliente **168663**

Remote Procedures(Designación y ligado):168724

Objetos a designar: El proceso llamado, el sitio del servidor

- Designación independiente de la localización
- Posibilidad de reconfiguración (errores, distribución de carga)

Momento del ligado: Precoz (estático) o tardío (dinámico)

Estático: localización del servidor conocida desde la composición

Dinámico: localización del servidor en tiempo de ejecución

- Designación simbólica de servicios (no asociada a un sitio de ejecución)
- Posibilidad de implementación o de selección

Tipos de Representacion de parámetors en Modelo RPC(168724)

Solución normalizada (ASN.1):

Sintaxis abstracta (Abstract Syntax Notation) para representar la estructura de los datos
Codificación independiente de las máquinas, SO's
Herramientas de generación disponibles para máquinas específicas

Otras soluciones:

Representación externa común

- Ejemplo: SUN XDR (External Data Representation) usada por NFS (conversiones inútiles si cliente y servidor son del mismo tipo)

Conversión explícita (al servidor) a partir de una representación local al cliente

Negociación entre el cliente y el servidor