En la figura 3, se muestra la arquitectura DCOM. El COM Run-Time proporciona los métodos de objetos remotos a clientes y componentes. Se utiliza el DCE RPC y el Security Provider para generar un paquete de red estándar, conforme a las especificaciones de DCOM.

4. REMOTE METHOD INVOCATION (RMI)

El método de invocación remota (RMI) define y soporta un modelo objeto distribuido para el lenguaje Java, ocultando al programador el ORB subyacente y proporcionando una API que facilita la construcción de aplicaciones distribuidas.

EL API para desarrollar aplicaciones que utilicen RMI fue desarrollado por SunSoft, una filial de Sun Microsystems, en 1997. El RMI es suministrado gratuitamente junto con el Java Development Kit (JDK).

RMI permite crear objetos remotos utilizando las interfaces a Java. Un objeto remoto, en Java, es un objeto cuyos métodos son invocados desde otra máquina virtual Java (JVM). Esta diseñado para construir aplicaciones Java/Java, es decir, el lenguaje utilizado para la implementación de los objetos en el cliente y en el servidor es Java.

La arquitectura del RMI está definida por tres niveles de abstracción (Figura 4): la capa stub/skeleton, la capa de referencia remota y la capa de transporte.

La capa stub/skeleton está compuesta por dos componentes con funcionalidades similares: el stub (del lado del cliente) y el skeleton (del lado del servidor). Tanto el Stub como el Skeleton son generados automáticamente, utilizando el compilador rmic, a partir de la clase servidor. El Stub se encarga de organizar y serializar los parámetros del objeto, proporcionando al cliente una interfaz y un comportamiento consistente con el objeto remoto. El skeleton se comunica directamente con la implementación del objeto y la capa de referencia remota del servidor.

La capa de referencia remota es la encargada de determinar el protocolo de invocación, el cual es independiente de los stubs y skeletons. Cada objeto remoto utiliza un determinado protocolo de invocación, el cual puede ser: punto a punto, grupo de objetos duplicados, soporte para una estrategia específica de duplicación y estrategias de reconexión. Por ejemplo, la capa de referencia remota se encarga de determinar cuando el servidor es un objeto simple o un objeto duplicado, requiriendo en este último caso, comunicación con las diferentes copias.

La capa de transporte es la encargada de enviar los datos entre el cliente y el servidor. Cuando se hace la invocación de un objeto, determina la dirección correcta donde éste se encuentra. También administra la conexión entre el cliente y el servidor, por lo que monitoria...
constante: la misma. Si se interrumpe una conexión o si ocurre un error de transferencia entre los datos, se encarga de notificar a la capa de referencia remota de esta situación, por medio de excepciones.

5. DEFINICIÓN DE CRITERIOS

El propósito de esta sección es definir un conjunto general de criterios que permitan establecer un marco de comparación común a diferentes tecnologías para objetos distribuidos. Estos criterios se definen a partir del estudio de las especificaciones técnicas de las principales tecnologías para objetos distribuidos: CORBA, DCOM y RMI [OOTM95][Chau97][Albert98].

(a) **Interoperabilidad a nivel de lenguajes**: Especifica si la tecnología provee los mecanismos necesarios para comunicar objetos implementados en diferentes lenguajes de programación.

(b) **Interoperabilidad a nivel de plataformas**: Especifica si la tecnología provee los mecanismos necesarios para comunicar objetos que se encuentran en sistemas operativos y plataformas de hardware heterogéneas.

(c) **Lenguaje de definición de interfaces**: Este criterio describe las características del lenguaje de definición de interfaces (IDL) proporcionado por la tecnología.

(c.1) **Mecanismo de construcción de interfaces**: Específica los mecanismos de construcción de interfaces que el IDL utiliza para definir una interfaz en función de otras interfaces.

(c.2) **Relación entre la interfaz y la implementación de una clase**: Este criterio establece la relación entre la definición de la interfaz y la implementación de la clase. Esta relación puede ser de uno a uno ó de muchos a uno. Una relación de muchos a uno, significa que una clase es implementada a partir de varias definiciones de interfaces.

(d) **Protocolos**: El protocolo es el encargado de establecer la forma de comunicación en una red. A través del protocolo las diferentes aplicaciones ubicadas en maquinas distintas y sistemas operativos heterogéneos pueden comunicarse entre sí. Este criterio especifica los protocolos de comunicación subyacentes utilizados por la tecnología.

(e) **Recolector de basura**: En un sistema distribuido, un recolector de basura, en el servidor, debe ser capaz de eliminar automáticamente los objetos remotos que no están referenciados por un cliente. Este criterio identifica si la tecnología utiliza un mecanismo de recolección de basura y las características que este posee.

(f) **Transmisión de parámetros**: Se describen los diferentes mecanismos de pase de parámetros utilizados en la tecnología.

(f.1) **Transmisión de objetos por valor**: Determina si un objeto puede ser pasado como parámetro por valor. En este caso se obtiene una copia del objeto transmitido por el cliente en el servidor.