MATERIA: BASES DE DATOS DISTRIBUIDAS
M. en C. Euler Hernández Contreras
ESCOM-IPN
UNIDAD I SISTEMAS DISTRIBUIDOS

Contenido
1. 1Introducción a los Sistemas Distribuidos
   a) Definición
1.2 Objetivo de un Sistema Distribuido
   a) Accesibilidad a los recursos
   b) Transparencia
   c) Grado de Apertura
   d) Escalabilidad
1.3 Tipos de Sistemas Distribuidos
   a) Sistemas Distribuidos de Cómputo
   b) Sistemas Distribuidos de Información
   c) Sistemas Distribuidos Masivos

Bibliografía:

Andrew S. Tanenbaum, Maarten Van Steen. Sistemas Distribuidos Principios y Paradigmas, Segunda Edición. Pearson Education –Prentice Hall, México 2007, 686 págs.

Introducción a los Sistemas Distribuidos

Evolución de los sistemas computacionales:

1945:

Comenzó la era moderna de las computadoras
1985:

a) Las computadoras eran grandes y caras.
b) Las minicomputadoras costaban al menos decenas de miles de dólares. Las empresas tenían solamente unas cuantas, y debido a la falta de un medio de comunicación entre ellas, operaban de manera independiente.

Década de los 80's:

Surgieron dos avances relevantes.

a) El primero fue el desarrollo de los microprocesadores.
   Los microprocesadores eran máquinas de 8 bits, pero se hicieron comunes prontamente las CPU de 16,32 y 64 bits.

Una máquina que costaba 10 millones de dólares ejecutaba 1 instrucción por segundo, saltaron a máquinas que cuestan 1000 dólares y son capaces de ejecutar un millón de millones de instrucciones por segundo.

b) El segundo fue la creación de las redes de computadoras de alta velocidad.

Redes de área local o LAN (Local Area Networks), permiten la interconexión de cientos de máquinas dentro de un mismo edificio, de tal manera que es posible transferir pequeños volúmenes de información entre máquinas en unos cuantos microsegundos, más o menos.

Redes de área amplia o WAN (Wide Area Netwoks), permiten la interconexión de millones de máquinas ubicadas alrededor del mundo a velocidades que van desde los 64kbps (kilobits por segundo) hasta gigabits por segundo.

El resultado de dichas tecnologías es posible trabajar sistemas de cómputo compuestos por grandes cantidades de computadoras interconectadas mediante una red de alta velocidad. Estos sistemas se conocen como redes de computadoras o sistemas distribuidos, al contrario de los sistemas centralizados (o sistemas de un solo procesador) que por lo general constan de una sola computadora, sus periféricos, y quizás algunas terminales remotas.

Definición:

Un sistema distribuido es una colección de computadoras independientes dan al usuario la impresión de construir un único sistema coherente.

Características de la definición

1. Un sistema distribuido consta de componentes (es decir, computadoras) autónomos.
2. Los usuarios (personas o programas) creen que realmente interactúan con un sistema único.
3. Los componentes autónomos necesitan colaborar entre sí.
4. Las diferencias entre las distintas computadoras y la manera en que se comunican entre sí quedan ocultas para el usuario.
5. Los usuarios y las aplicaciones pueden interactuar de manera consistente y uniforme, sin importar dónde y cuándo tenga lugar.
6. Un sistema distribuido estará disponible de manera continua, aunque tal vez algunas partes pudieran encontrarse fuera de operación.
7. Los usuarios y aplicaciones no deben notar que las partes son separadas, o que se agregan nuevas secciones para servir a más usuarios o aplicaciones.
8. Un sistema distribuido tiene una capa de software llamada middleware, el cual da soporte a computadoras y redes heterogéneas; esta capa consta de usuarios y aplicaciones (ver Figura 1)

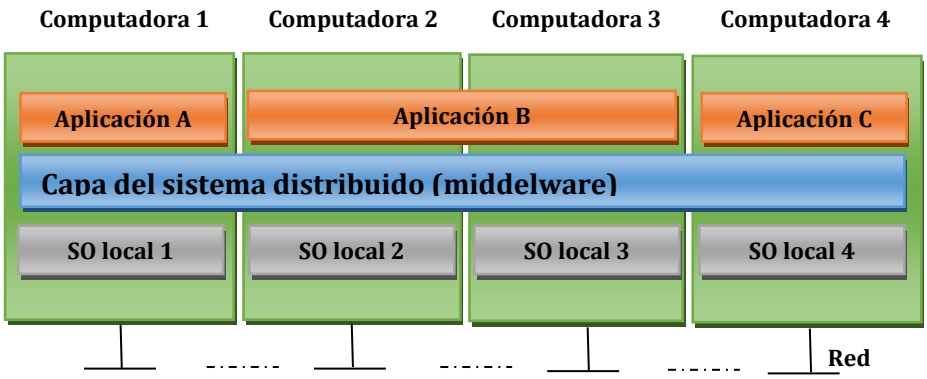


Figura 1. Un sistema distribuido organizado como middleware. La capa de middleware se extiende sobre diversas máquinas, y ofrece a cada aplicación la misma interfaz.

En la Figura 1, podemos ver cuatro computadoras conectadas en red y tres aplicaciones, de las cuales la aplicación B está distribuida entre las computadoras 2 y 3. A cada aplicación se le ofrece la misma interfaz.

Otra definición:

"*We define a distributed system as one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages*."

Our definition of Distributed Systems has the following significant consequences:

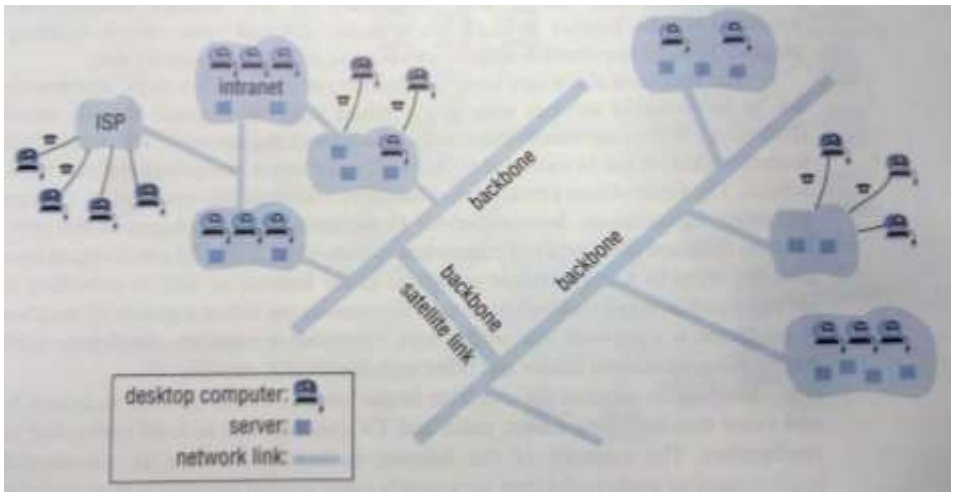
a) *Concurrency*: In a network of computers, concurrent program execution is the norm.
b) *No global clock*: When programs need to cooperate they coordinate their actions by exchanging messages.
c) *Independent failures*: All computer systems can fail and it is the responsibility of system designers to plan for the consequences of possible failures.

The term "resource" is a rather abstract one, but it best characterizes the range of things that can usefully be shared in a networked computer system. It extends from hardware components such as disks and printers to software-defined entities such as files, databases and data objects of all kinds. It includes the stream of video frames that emerges from a digital video camera and the audio connection that a mobile phone call represents.

Examples of distributed systems.

The Internet, intranets and the emerging technology of networks based on mobile devices. They are designed to exemplify the wide range of services and applications that are supported by computer networks and to begin the discussion of the technical issues that underline their implantation.

The Internet



A typical portion of the internet

The Internet is a vast interconnected collection of computer network of many different types. There are programs running on the computers connected to it interact by passing messages, employing a common means of communication. The design and constructions of the Internet communication mechanisms (the Internet protocols) is a major technical achievement, enabling a program running anywhere to address messages to programs anywhere else.

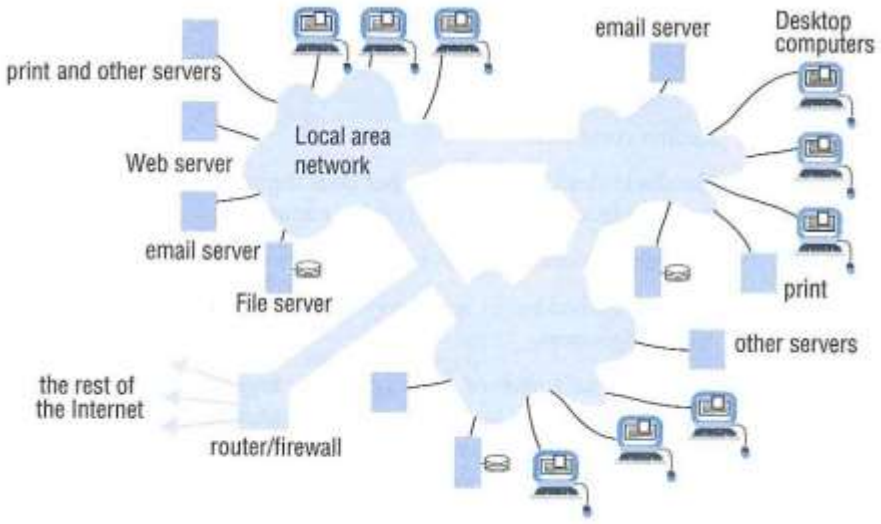
The Internet is also a very large distributed system. It enables users, wherever they are, to make use of services such as the World Wide Web, email and file transfer. Internet Service Providers (ISPs) are companies that provide modems links and other types of connection to individual users and small organizations, enabling them to access services anywhere in the Internet as well as providing local services such as email and web hosting. The intranets are linked together by backbones. A backbone is a network link with a high transmission capacity, employing satellite connections, fibre optic cables and other high-bandwidth circuits.

Multimedia services are available in the Internet, enabling users to access audio and video data including music, radio and TV channels and to hold phone and video conferences. The capacity of the Internet to handle the special communication requirements of multimedia data is currently quite limited because it does not provide the necessary facilities to reserve network capacity for individual streams of data.

Intranets

An intranet is a portion of the Internet that is separately administered and has a boundary that can be configured to enforce local security policies. It is composed of several local area networks (LANs) linked by backbone connections. The network configuration of a particular intranet is the responsibility of the organization that administers it and may vary widely –ranging from a LAN on a

single site to a connected set of LANs belonging to branches of a company or other organization in different countries.



A Typical intranet

An intranet is connected to the Internet via router, which allows the users inside the intranet to make use of service elsewhere such as the Web or email. It also allows the users in other intranets to access the services it provides. Many organizations need to protect their own services from unauthorized use by possible malicious users elsewhere. Companies also want to protect themselves from harmful programs such as viruses entering and attacking the computers in the intranet and possibly destroying valuable data.

The role of a firewall is to protect an intranet by preventing unauthorized messages leaving or entering. A firewall is implemented by filtering incoming and outgoing messages, for example according to their source or destination.

Mobile and ubiquitous computing

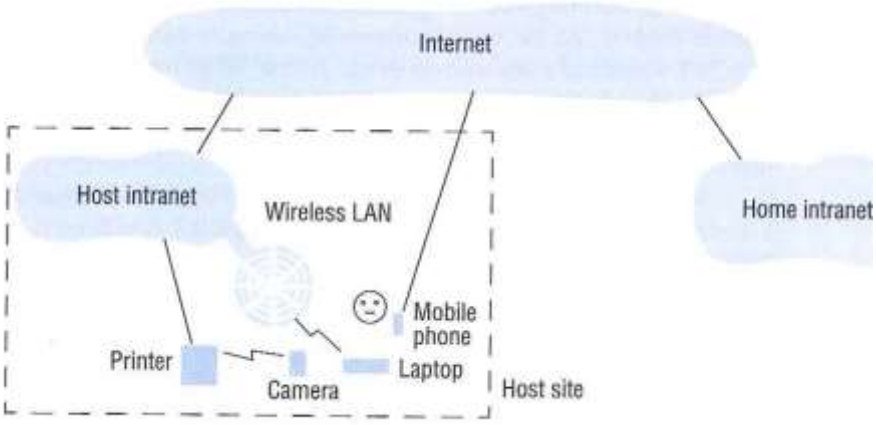
Technological advances in device miniaturization and wireless networking have led increasingly to the integration of small and portable computing devices into distributed systems. These devices include:

- Laptop computers
- Handle devices, including personal digital assistants (PDA's), mobile phones, pagers, video cameras and digital cameras.
- Wearable devices, such as smart watches with functionality similar a PDA.
- Devices embedded in appliances such a washing machines, hi-fi systems, cars and refrigerators.

The portability of many of these devices, together with their ability to connect conveniently to networks in different places, makes mobile computing possible.

Mobile computing is the performance of computing tasks while the user is on the move, or visiting places others their unusual environment. In mobile computing, users who are away from their “home” intranet are still provided with access to resources via the devices they carry with them. They can continue to access the Internet; they can continue to access resources in their home intranet; and there is increasing provision for users to utilize resources such as printers that are conveniently nearby as they move around.

Ubiquitous computing, is the harnessing of many small, cheap computational devices that are present in users’ physical environments, including the home, office and even natural settings. The term “ubiquitous” is intended to suggest that small computing devices will eventually become so pervasive in everyday objects that are scarcely noticed. That is, their computational behavior will be transparently and intimately tied up with their physical function.



Portable and handheld devices in a distributed system

The figure shows the user’s home intranet and the host intranet at the site that the user is visiting. Both intranets are connected to the rest of the Internet.

The users has access to three forms of wireless connection. Their laptop has a means of connecting to the host’s wireless LAN. This network provides coverage of a few hundred of metres. It connects to the rest of the host intranet via a gateway. The user also has a mobile (cellular) telephone, which is connected to the Internet. The phone give access to pages of simple information, which is presents on its small display. Finally, the user carries a digital camera, which can communicate over a personal area wireless network (with range up to about 10m) with a device such as a printer.

Distributed System are everywhere. The construction of distributed systems produces many challenges:

Heterogeneity: They must be constructed from a variety of different networks, operating systems, computer hardware and programming languages.

Openness: Distributed systems should be extensible- the first step is to publish the interfaces of the components, but the integration of components written by different programmers is a real challenge.

Security: Encryption can be used to provide adequate protection of shared resources and to keep sensitive information secret when is transmitted in messages over a network.

Scalability: A distributed system is scalable if the cost of adding a user is a constant amount terms of the resources that must be added.

Failure handling: Any process, computer or network may fail independently of the others. Therefore each component needs to be aware of the possible ways in which the components it depends on may fail and be designed to deal with each of those failures appropriately.

Concurrency: The presence of multiple users in a distributed system is a source of concurrent requests to its resource. Each resource must be safe in a concurrent environment.

Transparency: The aim is to make certain aspects of distribution invisible to the application programmer so that they need only be concerned with the design of their particular application. For example, they need not be concerned with its location or the details of how its operations are accessed by other components, or whether it will be replicated or migrated.

Objetivo de un sistema distribuido

Un sistema distribuido debe hacer que los recursos sean fácilmente accesibles; debe ocultar de manera razonable el hecho de que los recursos están distribuidos por toda la red; debe ser abierto; y debe ser escalable.

Accesibilidad a los recursos

Un sistema distribuido debe facilitar a los usuarios (y a las aplicaciones) el acceso a los recursos remotos, y compartirlos de manera controlada y eficiente.

Recursos: éstos pueden ser impresoras, computadoras, dispositivos de almacenamiento, datos, archivos, páginas web, y redes, entre otros.

Conectar usuarios y recursos facilita la colaboración y el intercambio de información.

La conectividad de internet está provocando:

a) La creación de numerosas organizaciones virtuales que trabajan juntas por medio de un *groupware*. Groupware: Es software para la edición colaborativa, la teleconferencia y más.
b) Ha propiciado el comercio electrónico, lo cual nos permite comprar y vender todo tipo de bienes sin tener que ir físicamente a una tienda o incluso salir de casa.
c) La seguridad debido a la conectividad y el intercambio se vuelve cada vez más importante ya que los sistemas proporcionan poca protección en contra del espionaje y de la intrusión en las comunicaciones.
d) Un problema relacionado es que el aumento de la conectividad puede provocar una comunicación no desea, tal como el correo basura, con frecuencia llamado *spam*. En tales casos, se debe protegerse mediante el uso de filtros de información especiales que seleccionan los mensajes entrantes basándose en su contenido.

Transparencia

Un sistema distribuido es transparente si es capaz de presentarse ante los usuarios y las aplicaciones como si se tratara de una sola computadora.

Tipos de transparencia

Transparencia	Descripción
Acceso	Oculta diferencias en la representación de los datos y la forma en que un recurso accede a los datos
Ubicación	Oculta la localización de un recurso
Migración	Oculta el que un recurso pudiera moverse a otra ubicación
Reubicación	Oculta el que un recurso pudiera moverse a otra ubicación mientras está en uso
Replicación	Oculta en número de copias de un recurso
Concurrencia	Oculta que un recurso puede ser compartido por varios usuarios que compiten por él
Falla	Oculta la falla y recuperación de un recurso

Transparencia de acceso:

a) Ocultar las diferencias en la representación de los datos y la manera en que el usuario accede a dichos recursos.
b) Oculta las diferencias en la arquitectura de las máquinas.
c) Lo más importante es llegar a un acuerdo con respecto a la manera en que representamos los datos en las diferentes máquinas y sistemas operativos.

Transparencia de Ubicación:

a) Los usuarios no pueden determinar en qué ubicación física se localiza el sistema.
b) La transparencia de ubicación se puede lograr si asignamos solamente nombres lógicos a los recursos, esto es, nombres en los cuales la

ubicación de un recurso no quede secretamente codificada. Un ejemplo de dicho nombres es una URL.

Transparencia de migración:

Se refiere al hecho de reubicar los recursos de un sistema distribuido sin afectar la manera en que podemos acceder a dichos recursos.

Transparencia de reubicación:

Hace referencia a reubicar los recursos mientras accedemos a ellos sin que el usuario o la aplicación lo noten. Por ejemplo, cuando los usuarios móviles pueden continuar usando sus computadoras portátiles inalámbricas mientras se mueves de un lugar a otro sin desconectarse (temporalmente).

Transparencia de replicación:

La replicación en un sistema distribuido permite incrementar la disponibilidad de los recursos o mejorar el rendimiento desde donde se accede a él y tiene que ver con el hecho de ocultar que existen distintas copias del recurso.

Transparencia de concurrencia:

Consiste en hacer que el acceso concurrente a un recurso compartido deje a ese recurso en un estado consistente. La consistencia se puede alcanzar a través de mecanismos de bloqueo mediante los cuales se concede al usuario, por turno, el acceso exclusivo al recurso deseado.

Transparencia a fallas:

Es lograr que el usuario no se percate de que un recurso (del cual quizá nunca oyó hablar) deja de funcionar correctamente, y que después el sistema repare la falla. Enmascarar las fallas es uno de los problemas más difíciles de solucionar en los sistemas distribuidos, e incluso resulta imposible lograrlo cuando hacemos ciertas suposiciones aparentemente realistas. La principal dificultad radica en la falta de habilidad para distinguir entre un recurso muerto y un recurso penosamente lento.

Grado de Apertura

a) *Abierto.*
   Un sistema distribuido abierto, es un sistema que ofrece servicios de acuerdo con las reglas estándar que describen la sintaxis y la semántica de dichos servicios. Ejemplo en las redes de computadoras, las reglas

estándar gobiernan formato, contenido significado de los mensajes enviados y recibidos.

b) Interoperabilidad.
Define la extensión mediante la cual dos implementaciones de sistemas o componentes de fabricantes distintos puedan coexistir y trabajar juntos si únicamente se apoyan en sus servicios mutuos tales como se especifica mediante un estándar común.

c) Portabilidad
Define la extensión mediante la cual una aplicación desarrollada para un sistema distribuido A se pueda ejecutar, sin modificación, en un sistema distribuido B que comparte la misma interfaz que A.

d) Extensible
En un sistema extensible, debiera ser relativamente fácil agregar partes que se ejecutan en sistemas operativos diferentes, o incluso reemplazar todo un sistema de archivos.

Escalabilidad

Se puede medir en las siguientes dimensiones:

a) Respecto a su tamaño.
Significa que podemos agregar fácilmente usuarios y recursos.
b) Geográficamente.
Es aquel en el que los usuarios y recursos pueden radicar muy lejos unos de los otros.
c) Administrativamente.
Puede ser fácil manejar incluso si involucra muchas organizaciones administrativas diferentes.

Tipos de Sistemas Distribuidos

Tenemos los siguientes: sistemas distribuidos de cómputo, sistemas distribuidos de información, y sistemas distribuidos embebidos.

1. Sistemas Distribuidos de Cómputo.

Entran los sistemas de cómputo de alto rendimiento y tenemos dos tipos:
a) Cómputo en Cluster
Consta de una colección de estaciones de trabajo similares, o computadoras personales, conectadas cercanamente por medio de una red local de alta velocidad. Además, cada nodo ejecuta el mismo sistema operativo.
b) Cómputo en malla (Grid)
Consta de sistemas distribuidos construidos generalmente como un conjunto de sistemas de cómputo, en donde cada sistema podría caer dentro de un dominio administrativo diferente, y podría ser muy

diferente cuando nos referimos al hardware, software, y la tecnología de red instalada.

Sistemas de cómputo en cluster:

La computación en cluster se utiliza para la programación en paralelo donde un solo programa (de cálculo intensivo) corre paralelamente en múltiples máquinas.

Características (Ver Figura 2):

a) Cada cluster consta de una colección de nodos controlados y se accede a ellos mediante un solo nodo maestro.
b) El nodo maestro manipula la ubicación de los nodos para un programa paralelo particular.
c) El nodo maestro mantiene una cola de procesamiento por lotes de trabajos enviados y proporciona una interfase para los usuarios del sistema.
d) El nodo maestro ejecuta el middleware necesario para la ejecución de programas y la administración del cluster.
e) El middleware está formado por bibliotecas necesarias para ejecutar programas paralelos.
f) Una computadora en cluster ofrece mayor transparencia en la distribución al aparentar ser una sola computadora.
g) Son sistemas homogéneos: mismo sistema operativo, están conectadas en la misma red, en esencia son las mismas computadoras.

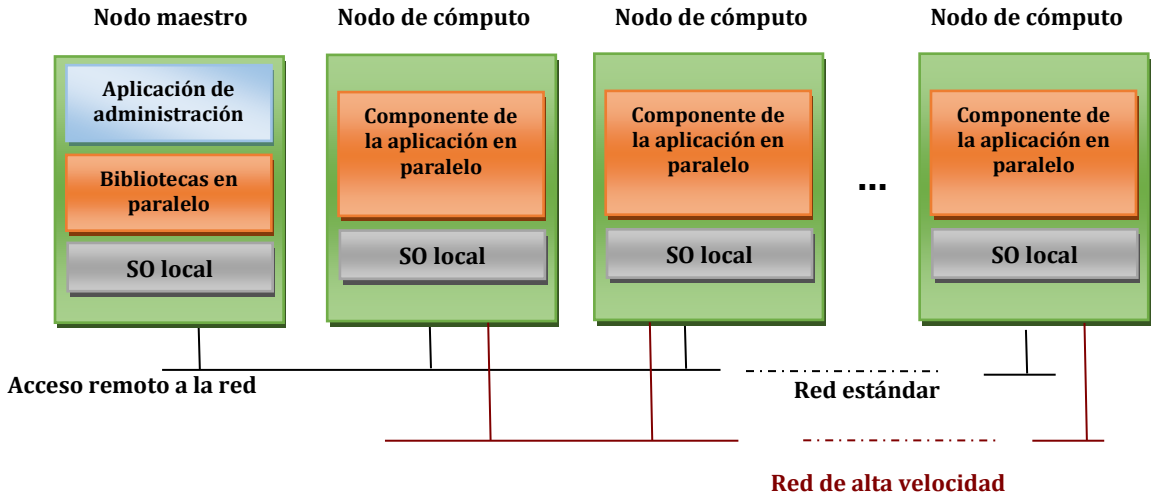


Figura 2. Ejemplo de un sistema de cómputo en cluster.

2. Sistemas de cómputo en grid.

Tienen un alto grado de heterogeneidad: no se hacen suposiciones de ninguna índole con respecto al hardware, sistemas operativos, redes, dominios administrativos, políticas de seguridad, etc.

El propósito de los sistemas de cómputo en grid es reunir recursos diferentes de diferentes organizaciones para permitir la colaboración de un grupo de personas o instituciones. (Esta colaboración se realiza en forma de una organización virtual).

La gente de esta organización virtual tiene derechos de acceso a los recursos que proporciona la organización. Los recursos constan de servidores, facilidades de almacenamiento, y bases de datos.

Arquitectura en capas para sistemas de cómputo en grid.

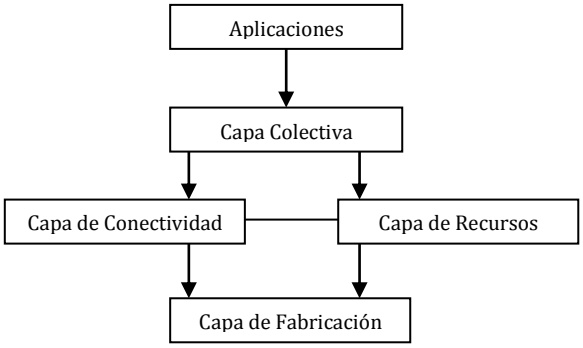


Figura 3. Arquitectura en capas para sistemas de cómputo en grid.

Esta arquitectura está compuesta por cuatro capas:

a) *Fabricación*
   Proporciona las interfaces para recursos ubicados en un sitio específico. Permiten el intercambio de recursos dentro de una organización virtual.
   Proporcionan funciones para consultar el estado y las capacidades de un recurso, junto con funciones para la administración real de un recurso (por ejemplo, bloqueos de recursos).

b) *Conectividad*
   Consiste en protocolos de comunicación para dar soporte a las transacciones del grid que abarcan el uso de múltiples recursos.
   Contiene Protocolos para transferir datos entre los recursos o simplemente para acceder a un recurso desde una ubicación remota.
   Contiene protocolos de seguridad para autenticar usuarios y recursos.

c) *Recursos*
   Es responsable de la administración de un solo recurso.

Ofrece funciones útiles para obtener la información de configuración sobre un recurso específico, o, en general para realizar operaciones especificas tales como la creación de un proceso o la lectura de datos.

d) Colectiva
Se encarga de manipular el acceso a múltiples recursos, y, por lo general, consta de servicios para descubrir recursos, ubicación y calendarización de tareas dentro de múltiples recursos, replicación de datos, y así sucesivamente.

e) Aplicación
Consta de aplicaciones que operan dentro de una organización virtual y hacen uso del ambiente de cómputo en grid.

Sistemas Distribuidos de Información

Sistemas de Procesamiento de transacciones

Estos hacen referencia a los sistemas de bases de datos, donde las operaciones se llevan a cabo generalmente en forma de transacciones.

Programar usando transacciones requiere de primitivas de transacción especiales que deben ser proporcionadas ya sea por el sistema distribuido subyacente o por el lenguaje del sistema en tiempo de ejecución.

Primitiva	Descripción
BEGIN_TRANSACTION	Marca el inicio de una transacción
END_TRANSACTION	Termina la transacción e intenta continuar
ABORT_TRANSACTION	Finaliza la transacción y reestablece los viejos valores
READ	Lee los datos desde un archivo, una tabla u otra fuente
WRITE	Escribe los datos en un archivo, una tabla, o en otra fuente

Tabla 1. Ejemplo de primitivas de transacción

Características de las transacciones (ACID)

1. Atómicas: Para el mundo exterior, la transacción es indivisible.
2. Consistentes: La transacción no viola sistemas invariantes.
3. Aisladas: Las transacciones concurrentes no interfieren entre si.
4. Durables: Una vez que se confirma una transacción, los cambios son permanentes.

Atómicas

Esta propiedad garantiza que cada transacción ocurra completamente, o se omite, y si ocurre, sucede en una sola acción instantánea e indivisible.
Mientras que una transacción se encuentra en progreso, otros procesos (estén o no involucrados en la transacción) no pueden ver ningún estado intermedio.

Consistentes

Esta propiedad establece que las transacciones son consistentes. Lo cual significa que si el sistema tiene ciertas invariantes que deben permanecer siempre, si se mantuvieron antes de la transacción, también permanecerán después.

Por ejemplo, en un sistema bancario, una invariante clave es la ley de conservación de dinero. Después de cada transferencia interna, la cantidad de dinero presente en el banco debe ser la misma que existía antes de la transferencia, pero por un breve momento durante la transacción, esta invariante puede violarse. Sin embargo, la violación no es visible fuera de la transacción.

Aisladas o en Serie

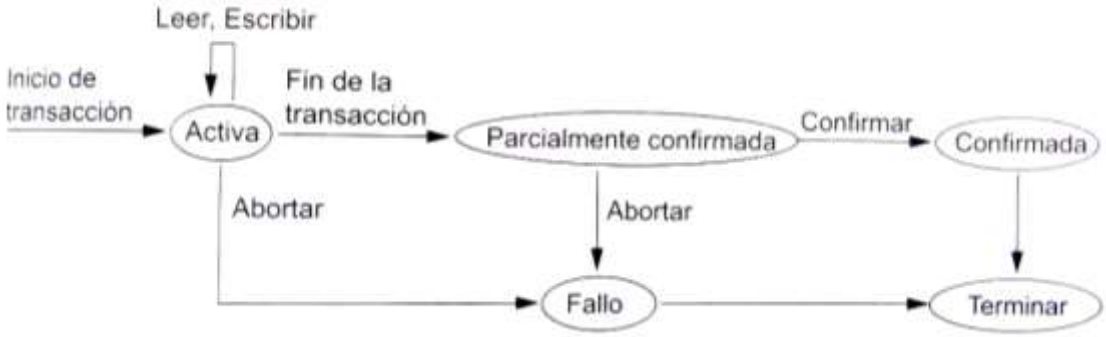
Esto significa que si dos o mas transacciones se están ejecutando al mismo tiempo, para cada una de ellas y para otros procesos, el resultado final luce como todas las transacciones se ejecutarán en secuencia, en cierto orden (según el sistema).

Durables

Esto se refiere al hecho de que una vez confirmada una transacción, no importa lo que suceda, la transacción continúa y los resultados se vuelven permanentes. Ninguna falla ocurrida después de la confirmación puede deshacer los resultados u ocasionar que se pierdan.

Estados de la transacción

Para llevar a cabo la recuperación de la base de datos, el sistema tiene que saber cuando la transacción inicia, termina y se confirma o aborta. El Gestor de Recuperación es el encargado de llevar a cabo las siguientes operaciones:



1. *Inicio de transacción*: Principio de la ejecución de la transacción.
2. *Leer o escribir*: Implica una operación de lectura o escritura de elementos de la BD que se efectúan como parte de una transacción.
3. *Fin de la transacción*: Especifica que las operaciones de leer o escribir han terminado y marca el límite de la ejecución de una transacción.
Se verifican si los cambios introducidos se pueden aplicar permanentemente a la BD (confirmar) o si deben abortarse porque viola el control de concurrencia u otra razón.
4. *Confirmar transacción*: Señala que la transacción terminó con éxito y que cualquier cambio (actualización) ejecutadas por ella se pueden confirmar sin peligro en la BD y que no se cancelarán.
5. *Revertir (o abortar):* Implica que la transacción terminó sin éxito y que cualquier cambio o efecto que pueda haberse aplicado a la BD se debe cancelar.

Bitácora del sistema (Archivo LOG, diario o journal)

Se almacena la información necesaria para deshacer -en caso de fracasar (*rollback*)- o rehacer -en caso de recuperar (*rollforward*)- las transacciones.

Rollback: Este se emplea cuando se verifica la consistencia de la BD y se encuentra en esta inconsistente, se deben de deshacer las transacciones anteriores hasta tener nuevamente consistencia.
Rollforward: Se considera cuando en un momento determinado la transacción inició y terminó exitosamente pero su resultado no está reflejado en el repositorio de datos.

La bitácora almacena las siguientes entradas:

1. [**start_Transaction**, *T*]
2. [**write_item**, *T*,*X*, *valorAnterior*, *nuevoValor*]
3. [**read_item**, *T*,*X*]
4. [**commit**, T]
5. [**abort**, T]

En otras palabras considera:
- Identificador de la transacción
- Hora de la modificación
- Identificador del registro afectado
- Tipos de acción
- Valor anterior del registro
- Nuevo valor del registro

Transacción anidada

Se construye a partir de cierta cantidad de substransacciones. La transacción de más alto nivel puede dividirse en subprocesos hijos que se ejecutan en paralelo entre sí, en diferentes máquinas, para mejorar el rendimiento de simplificar la programación.

Cada uno de estos subprocesos también pueden ejecutar una o más substransacciones, o dividirse en sus propios subprocesos hijos.

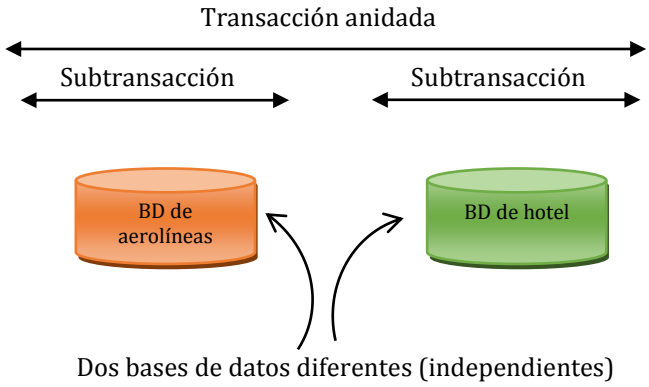


Figura 4. Transacción Anidada

Las subtransacciones dan lugar a un sutil pero importante, problema. Imagine que una transacción inicia diversas substracciones en paralelo, y que una de éstas se confirma volviendo visibles los resultados para la transacción padre. Después de más cálculos, la transacción padre aborta y reestablece todo el sistema en el estado que tenía antes de iniciada la transacción de más alto nivel. Como consecuencia, los resultados de la substransacción que se confirmó, a pesar de todo, deben deshacerse. Por tanto, la permanencia a la que nos referimos antes sólo es aplicable a transacciones del más alto nivel.

En los sistemas distribuidos, las transacciones anidadas son importantes para que proporcionen una forma natural de distribuir una transacción a través de varias máquinas.

En los primeros días de los sistemas middleware empresariales, el componente que manejaba transacciones distribuidas (anidadas) conformaba la parte central para integrar aplicaciones al nivel servidor o de base de datos. A este componente se le conoce como *Monitor de procesamiento de transacciones* o *monitor TP*.

Su principal tarea era permitirle a una aplicación el acceso a múltiples servidores\bases de datos ofreciendo un modelo de programación transaccional.

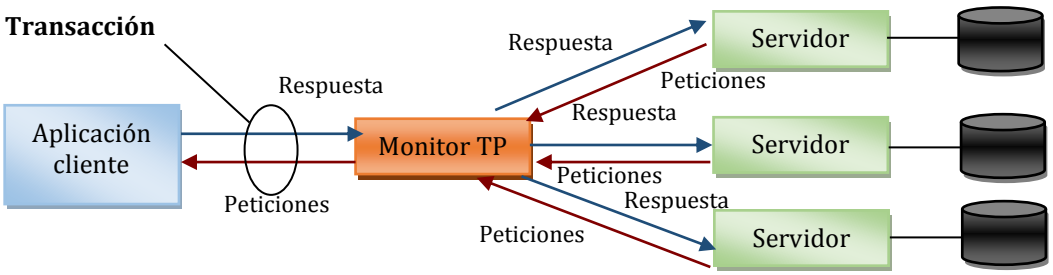


Figura 5. Rol de un monitor TP en sistemas distribuidos

Integración de aplicaciones empresariales

Toma en cuenta el hecho de que los componentes de las aplicaciones debían de ser capaces de comunicarse entre sí de manera directa y no sólo mediante un comportamiento de petición respuesta soportado por sistemas de procesamiento de transacciones (como lo hacen en la Figura 5).

Esta necesidad de comunicación dio pie a muchos modelos de comunicación diferentes. La idea principal fue que las aplicaciones existentes pudieran intercambiar información de manera como se muestra en la Figura 6.

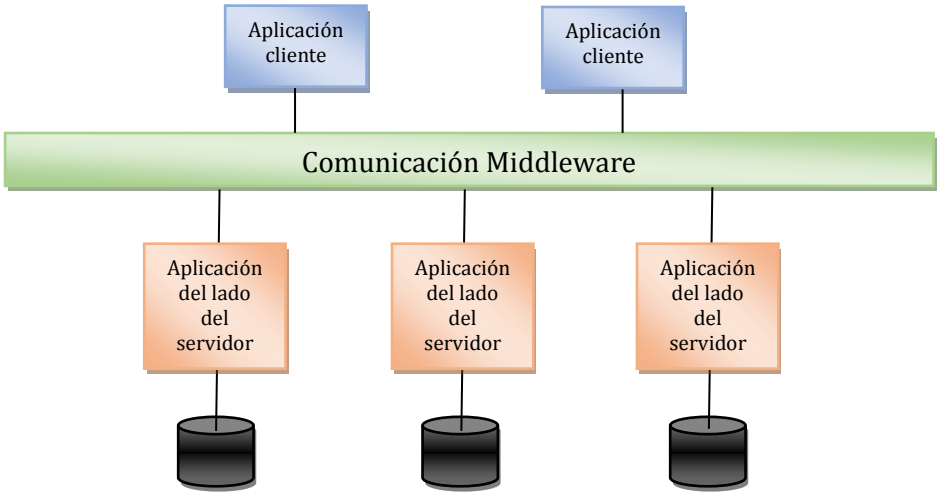


Figura 6. Middleware como facilitador de comunicación al integrar aplicaciones empresariales.

Existen diversos tipos de comunicación middleware:

a) Llamadas a procedimientos remotos (RPC).
   Un componente de aplicación puede enviar de manera efectiva una petición a otro componente de aplicación si se realiza una llamada a un procedimiento local, lo cual resulta en una petición que se empaca como un mensaje y se envía al componente invocado (remoto).

b) Invocaciones a métodos remotos (RMI).
   Una RMI es básicamente lo mismo que un RPC, excepto que la RMI opera sobre objetos en lugar de aplicaciones.

   RPC y RMI tienen la desventaja de que el que llama y el llamado necesitan estar en ejecución y estar activos al momento de la comunicación.

c) Middleware orientado a mensajes (MOM).
   Las aplicaciones simplemente envían mensajes a puntos de contacto lógicos, en general descritos mediante un sujeto.
   Las aplicaciones pueden indicar su interés por un tipo específico de mensaje, después de lo cual la comunicación middleware se encargará de que esos mensajes se entreguen a aquellas aplicaciones.

Middleware Layer

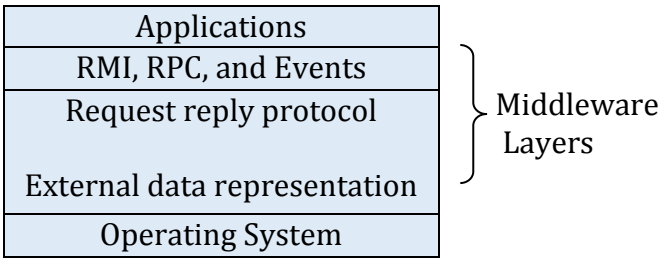


Figure 7. Middleware Layers

Middleware: Software that provides a programming model above the basic buinding blocks procesess and message passing. The middleware layer uses protocols based on messages between processes to provide its higher-level abstractions such a remote invocations and event, as illustrated in Figure 7. For example, the remote method invocation abstraction is based on the request-reply protocol.

The request-reply protocol

The following protocol is based on a trio of communication primitives: *doOperation*, *getRequest* and *sendReply*, as show in Figurate 8.

The *doOperation* method is used by clients to invoke remote operations. Its arguments specify the remote object and which method to invoke, together with additional information (arguments) required by the method. Its result is an RMI reply.

GetRequest is used by a server process to acquire service requests as show im Figure 8. When the server has invoked the method in the specified object it then uses *sendReply* to send the reply message to the client. When the reply message is received by the client the original *doOperations* is unblocked and execution of the client program continues.

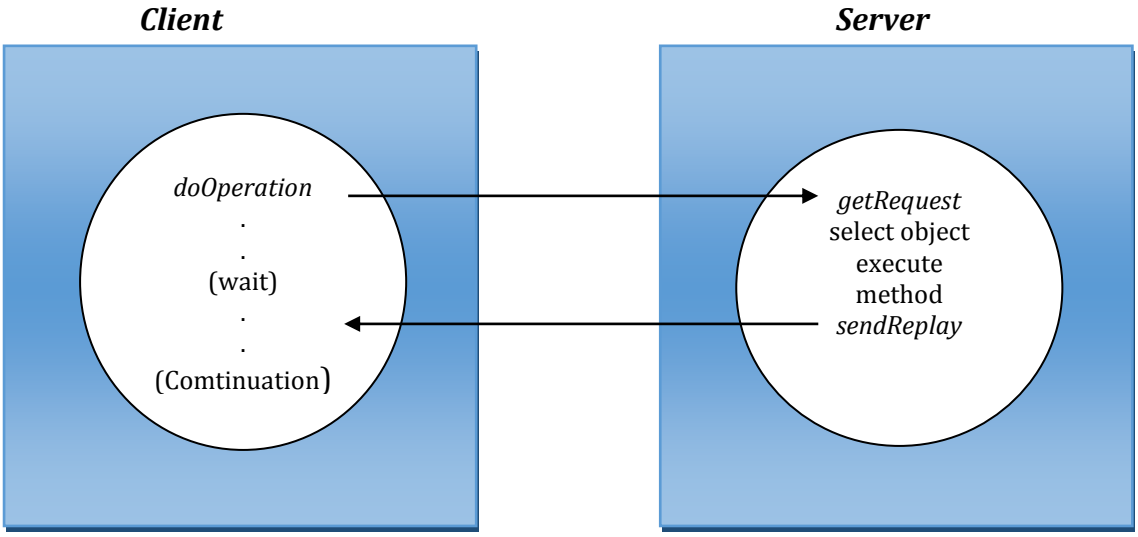
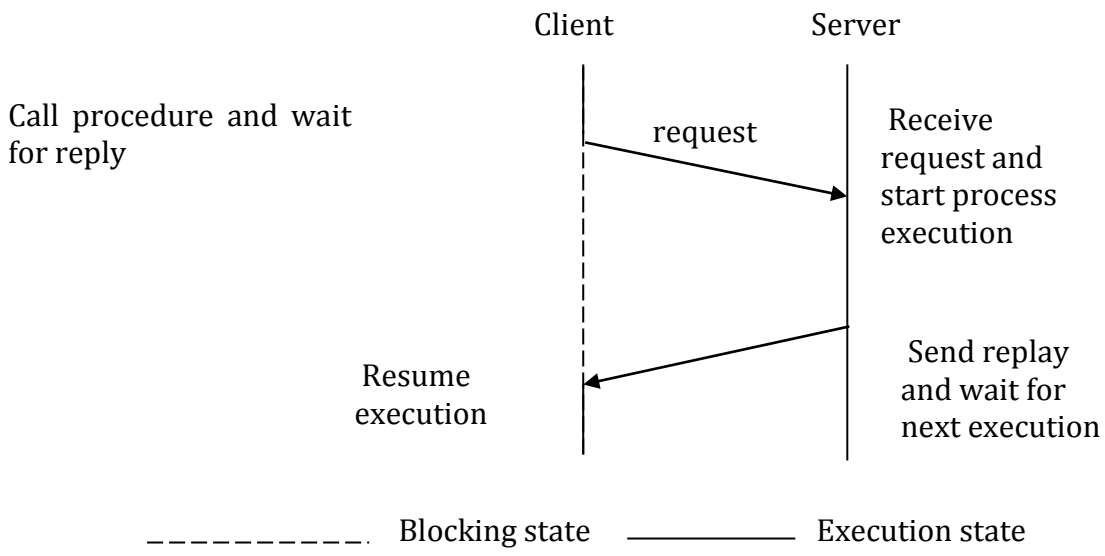


Figure 8. Request-replay conmmunication

RPC (Remote Procedure Call)

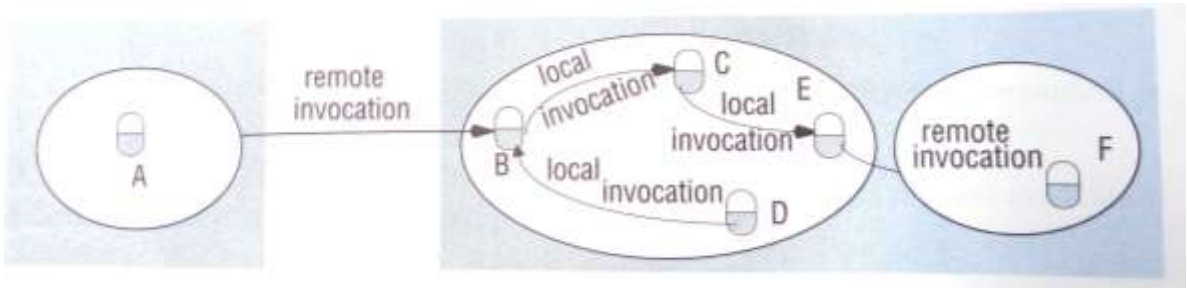
Remote procedure model, allows client programs to call procedure in server programs running in separate processes and generally in different computes from the client.



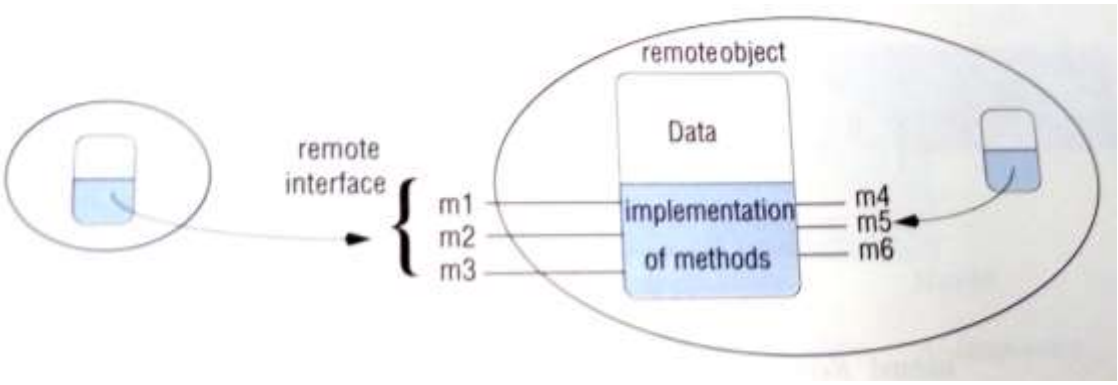
RMI (Remote Method Invocation)

Remote method invocation allows an object living in one process to invoke the methods of an object living in another process.

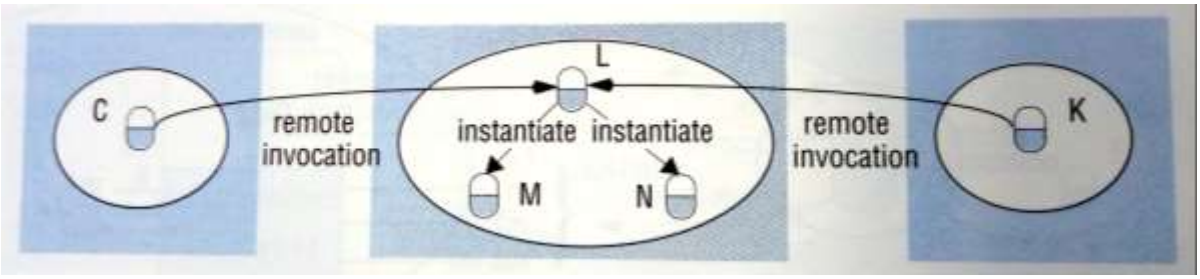
Remote and local Invocations



A remote object and its remote interface



Instantiation of remote objects



Sistemas Distribuidos Masivos

Este tipo de sistema distribuido se caracteriza por su estabilidad ya que hacen referencia a la introducción de dispositivos de cómputo móviles y embebidos.

Los dispositivos se caracterizan por ser pequeños, de baterías, portátiles, y tienen sólo una conexión inalámbrica, aunque no todas estas características son aplicables a todos los dispositivos.
Una característica importante es su carencia general de control administrativo humano. Los dispositivos son configurados por sus propietarios, ya que de otro modo necesitan descubrir automáticamente su ambiente y "adaptarse" de la mejor manera posible.

Requerimientos para las aplicaciones móviles

a) Incluir cambios contextuales
   Significa que un dispositivo debe estar continuamente consciente de que su ambiente puede cambiar en cualquier momento.

b) Fomentar composiciones a la medida
   Se refiere al hecho de que muchos dispositivos de sistemas masivos se utilizarán en formas muy diferentes por distintos usuarios. Como resultado, la suite de aplicaciones que se ejecutan en un dispositivo debe ser sencilla de configurar, ya se por el usuario o mediante la interpolación automatizada (pero controlada).

c) Reconocer el intercambio como algo común
   Un aspecto importante de los sistemas dominantes es que los dispositivos generalmente ingresan al sistema para acceder a la información. Esto sucede para facilitar la lectura, el almacenamiento e intercambiar información. Debido a la intermitente y cambiante conectividad de los dispositivos, el espacio donde reside la información accesible muy probablemente cambiará a cada momento.