|  |  |  |
| --- | --- | --- |
|  | **Instituto Politécnico Nacional**  Escuela Superior de Cómputo |  |
|  | Desarrollo de Sistemas Distribuidos |  |
|  | Prof. **Benjamín Cruz Torres** |  |

**Práctica No. 4**  
**Sincronización en Sistemas Distribuidos**

Grupo: 4CV{}

|  |  |
| --- | --- |
|  | Equipo: {Num Equipo} |
| Integrantes: |  |
|  |  |
|  |  |
|  |  |

*Fecha:{Fecha de realización}*

Práctica 4: Sincronización en sistemas distribuidos

Objetivo de la Práctica Que comprenda la importancia de la sincronización de relojes en un sistema distribuido.

Tecnologías a aplicar: Sockets, RMI, POO, Protocolos de comunicación, Bases de Datos, algoritmo de Christian, algoritmo de Berkeley.

## Competencias objetivo

Trabajo en equipo, comunicación en sistemas distribuidos, sincronización.

## Actividades

Desarrollar una aplicación que reciba una cadena o un carácter a través de dos servidores en la nube. Ambos servidores deberán tener sus relojes sincronizados. De acuerdo con los siguientes requerimientos:

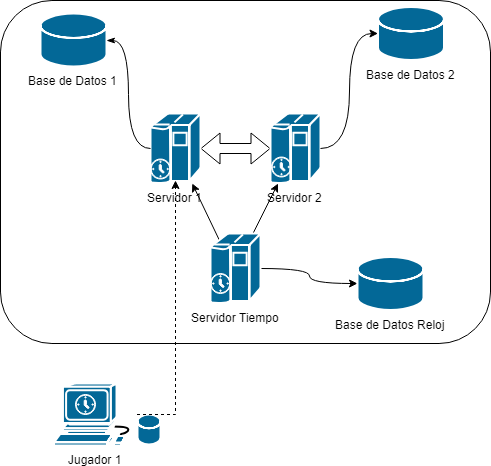


Figura 1. Sincronización usando un algoritmo de Sincronización

* Servidores en la nube NO

## Requerimientos funcionales

* Los servidores son los únicos conectados y con acceso a sus respectivas bases de datos.
* En la interfaz de usuario del Jugador cliente habrá un reloj digital. YA
* En la interfaz de usuario del Jugador hay un botón “Enviar”.  YA
* El botón “Enviar” mandará una cadena o un carácter al servidor correspondiente.  YA
* El servidor que reciba el dato lo mostrará en un mensaje en la línea de comando.  YA
* Cada servidor almacenará en su base de datos el dato enviado, junto con la hora en que se recibió y el identificador del Jugador (dirección IP), también si fue una cadena o un carácter. YA
* La hora se determinará por el reloj de la interface del servidor, no la del sistema.   YA
* Al inicio los relojes de cada Servidor y Usuario tienen horas diferentes elegidas al azar. YA
* El Servidor de Tiempo tendrá que sincronizar el sistema cada δ segundos. GUZ

## Requerimientos no funcionales

* El jugador hará peticiones solamente a uno de los servidores. OK
* Las dos bases de datos mantienen la información replicada. Esto es, hay una comunicación constante con entre los dos servidores.
* El usuario puede modificar el reloj del Jugador. OMAR
* El formato de hora de los relojes es de 24 hrs. OK
* Tomar en cuenta la latencia al enviar los ajustes por parte del servidor de tiempo. GUZ
* Para sincronizar se utilizará el algoritmo de Berkeley, el de Cristian o el de Berkeley Distribuido. GUZ
* Una vez que se calcula la nueva hora de referencia (o UTC en el caso de Cristian), el coordinador envía los ajustes correspondientes. GUZ
* Considerar “1 hora razonable” al hacer los cálculos. OK

# Introducción

{Agrega una breve introducción sobre el tema a tratar}

# Análisis

{Indica aquí la fase de análisis de tu práctica}

# Diseño

{Indica aquí la fase de diseño de tu práctica}

{Incluye los diagramas correspondientes}

# Desarrollo de la práctica

{Indica los pasos que usaste para resolver la práctica}

{Incluye capturas de pantalla de la aplicación}

# Pruebas de funcionamiento

{Indica aquí las pruebas realizadas}

{Incluye capturas de pantalla con las pruebas realizadas}

# Conclusiones

{Indique una breve conclusión sobre la práctica realizada}

# Bibliografía

{Incluye las referencias bibliográficas utilizadas}

Consideraciones finales

Descarga el documento antes de llenarlo.

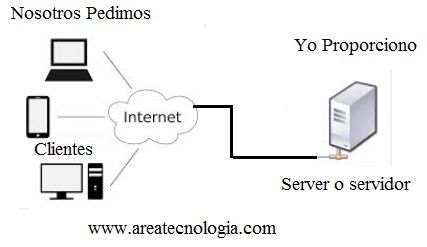
Este documento se debe llenar en equipo, aunque la práctica la deben hacer TODOS los integrantes del mismo.

Después de llenar el documento, guárdalo como PDF y envíalo a través del tema correspondiente en la plataforma *de aprendizaje virtual correspondiente*.

Queda estrictamente prohibido cualquier tipo de plagio a otros equipos o grupos de este semestre o anteriores. En caso de incurrir en esta falta, se anulará la asignación correspondiente y se bajarán 2 puntos al (los) equipo (s) involucrados.

Servidor

* **ordenador y sus programas**, que están **al servicio de otros ordenadores**
* atiende y responde a las peticiones que le hacen los otros ordenadores (clientes)
* arquitectura que siguen los servidores es el de **cliente-servidor**, es decir **el cliente/s pide y el servidor proporciona** los recursos o servicios
* Utilizados para gestionar los recursos de una red. Red más conocida y más grande es **Internet**
* **“Servidor deberá estar siempre encendido”**



* En la siguiente imagen vemos el apilamiento de los servidores de una empresa que se dedica a proporcionar almacenamiento de información, también llamado [Cloud Computing](https://www.areatecnologia.com/informatica/cloud-computing.html).  
  

<https://www.areatecnologia.com/informatica/servidor-y-tipos.html>



Plataformas Cloud Gratuitas

<http://xhrist14n.blogspot.com/2012/03/plataformas-cloud-gratuitas.html>

# **Ejemplo socket java Cliente/Servidor**

El servidor es el encargado de responder las peticiones de los clientes previo proceso de las mismas. Un servidor puede atender peticiones de varios clientes. En este ejemplo solo atenderá a un cliente, es recomendable el uso de Thread para sistemas multi-clientes, pero eso lo veremos en otro tutorial.

<http://www.jc-mouse.net/proyectos/ejemplo-socket-java-clienteservidor>

<http://www.jc-mouse.net/proyectos/ejemplo-socket-java-clienteservidor>

ventajas

* reducción de costes

[herramientas](http://bbvaopen4u.com/es/actualidad/cinco-herramientas-para-que-los-emprendedores-saquen-provecho-del-cloud) para programadores que permiten picar código en la nube: **[Codio](https://codio.com/" \t "_blank)**, [**Cloud9**](https://c9.io/), **[Codenvy](https://codenvy.com/" \t "_blank)** o **[Nitrous](https://www.nitrous.io/" \t "_blank)** son algunos ejemplos. Son **Entornos de Desarrollo Integrado (IDEs) con las ventajas de una solución alojada en la nube.**

* **Son algo más que un editor de código**: igual que los IDEs se podrían definir como editores de código con esteroides, las plataformas de desarrollo en la nube podrían pasar por ser IDEs con esteroides. Este tipo de herramientas son más bien Plataformas de Desarrollo como Servicio (Development Platform as a Service- dPaaS), que ofrecen una serie de funcionalidades sobre costes y productividad más elevadas.
* **Menos tiempo de implementación**: este tipo de plataformas reducen mucho los tiempos de instalación. Menos tiempo, menos costes.
* **Programa desde cualquier sitio**: no es necesario estar delante de tu máquina para continuar programando tu producto. Al estar alojada en la nube, **es posible hacerlo desde cualquier sitio y dispositivo** (ordenador o tableta). Único requisito: estar conectado a la Red
* **Trabajo colaborativo en tiempo real**: una de sus grandes ventajas es que varios desarrolladores pueden estar trabajando en el mismo proyecto a la vez y utilizar servicios de chat online para comunicarse.
* **Personalización del entorno de desarrollo**: este tipo de herramientas permiten instalar dependencias para los proyectos de forma independiente. Cuando se programa en local, en muchas ocasiones se pueden tener complicaciones porque las dependencias para unos proyectos afectan a otros o perjudican a otras aplicaciones web.

<https://bbvaopen4u.com/es/actualidad/desarrollo-en-la-nube-ventajas-y-plataformas-para-programadores3>

En Sitepoint, la página de contenido especializado para desarrolladores, [hacen un listado de las virtudes de utilizar la nube para programar](http://www.sitepoint.com/developing-in-the-cloud-an-introduction/):

<https://www.sitepoint.com/developing-in-the-cloud-an-introduction/>

computo en la nube:

<https://es.wikipedia.org/wiki/Computación_en_la_nube>

Otro detalle que merece la pena conocer es el uso de servidores de aplicaciones basados en Java EE. Un 50% de dichos servidores son Tomcat, un 37% IBM WebSphere, un 22% utiliza Oracle WebLogic y, finalmente, el 18% se decanta por Red Hat JBoss.

Mientras tanto, a nivel de herramientas de desarrollo de aplicaciones en Java, Eclipse parece ser la más popular para estos profesionales, ya que es utilizada por el 66% de los encuestados. Muy por debajo se sitúan otras ofertas como NetBeans (17%) y Oracle JDeveloper (12%).

<https://www.silicon.es/los-desarrolladores-en-java-se-mueven-a-la-nube-2238>

# Tutorial: Compilación de una aplicación web de Java mediante Azure Cosmos DB y la API de SQL

<https://docs.microsoft.com/es-es/azure/cosmos-db/sql-api-java-application>

red virtual

Una **red virtual** PowerVM permite conectividad entre las particiones de un **servidor** , si existe un puente, con otros servidores. Puede crear varias redes virtuales en un sistema gestionado y, a continuación, conectar las particiones a dichas redes.

Una VLAN (Virtual Local Area Network) permite segmentar de forma lógica la red física.

Una VLAN es un método que segmenta de forma lógica una red física para que la conectividad de la capa 2 esté restringida a miembros que pertenezcan a la misma VLAN.

Un adaptador Ethernet virtual permite que las particiones envíen y reciban tráfico de red sin un adaptador Ethernet físico dedicado. Se crea un adaptador Ethernet virtual cuando conecta una partición a una red virtual.

Dentro del sistema, los adaptadores Ethernet virtuales se conectan a un conmutador Ethernet virtual IEEE 802.1Q.

<https://www.ibm.com/support/knowledgecenter/es/POWER8/p8efd/p8efd_powervm_virtual_network_concept.htm>

mysqld --defaults-file="C:\\Program Files\\MySQL\\MySQL Server 8.0\\bin\\my.ini" –-init-file=C:\\mysql-init.txt