Telecomunicaciones y sistemas distribuidos
Redes y telecomunicaciones
Proyectos de aprobación final
2022/2024

1. Una empresa tiene puntos de venta distribuidos de ciertos productos. Cada punto de venta tiene su propio depósito de artículos. Un nodo especial designado (que puede cambiar) es el único que puede actualizar precios de productos. Cualquier punto de venta podrá vender productos sin disponibilidad en su depósito local, tomándolos de otros depósitos, si existieran. Los clientes pueden conectarse con cualquier punto de venta para realizar consultas de precios o compras de productos.
   Se pide:
   a. Diseñar e implementar un sistema distribuido para lograr el comportamiento descripto.
   b. Debe garantizar la consistencia de los datos: Un producto debe tener el mismo precio en todos los nodos y las existencias deben quedar en estados consistentes luego de una venta.
   c. Debe tolerar fallas en el nodo especial o de los puntos de venta.
   d. Definir y documentar la arquitectura del sistema, su diseño y funcionamiento, el formato de los mensajes y las herramientas utilizadas.

2. Un sistema de venta de pasajes distribuido permite al usuario comprar tickets para llegar de un destino A a B posiblemente con combinaciones (conexiones) entre servicios de diferentes compañías. Cada compañía ofrece un conjunto de servicios entre diferentes puntos, cada uno con una cierta capacidad (número de asientos).
   Además, cada compañía tiene su propio nodo (servidor) que procesa requerimientos de la forma
   a. getFreeSeats(serviceId)
   b. reserve(serviceId, seat_number)
   c. confirm_reserve(serviceId, seat_number)
   d. cancel_reserve(serviceId, seat_number)
   Se debe diseñar implementar un sistema que no permita la sobreventa de pasajes.
   Los nodos (servidores) de las companías pueden presentar fallas de comunicación.

3. Una aplicación a escala global requiere mantener el estado de los usuarios con su estado (conectado/desconectado). Los clientes pueden consultar a cualquier nodo del sistema por el estado de un cliente. Los datos deben estar uniformemente distribuidos entre los nodos de la red para poder responder a consultas con costo a lo sumo logarítmico (en base a la cantidad de mensajes).
   Se pide:
   a. Diseñar e implementar un sistema distribuido para lograr el comportamiento descripto.
   b. Debe ser tolerante a fallas. Ante la falla de un nodo, el sistema global no debería perder información sobre los usuarios.
   c. Documentar la solución elegida y su fundamentación, los formatos de mensajes y las herramientas utilizadas.

4. Diseñar e implementar una aplicación distribuida que permita editar en forma compartida un documento, Cada cliente se conectará a un nodo (peer) de la red el cual contiene una réplica del documento. Los peers deberán colaborar en la edición colaborativa con el objetivo de mantener las réplicas consistentes. Los clientes pueden emitir comandos de edición de la forma *insert(c,p)* y *delete(p)*, en cualquier momento. La operación *insert(c,p)* representa la inserción del caracter *c* en la posición *p* del documento. La operación *delete(p)* debe eliminar el caracter en la posición *p*.
   Ayuda: Usar algún algoritmo de transformación de operaciones.

5. Dados N procesos, implementar una primitiva `broadcast(msg)` *confiable atómico*. Esto significa que todos los procesos correctos (sin fallas) coinciden en el conjunto de mensajes recibidos y son entregados (delivered) a la aplicación en el mismo orden (con respecto a otros mensajes de otros broadcasts). La operación debería ser exitosa aún cuando el emisor falla luego del broadcast. Se debe notar que este problema es equivalente a que todos los procesos correctos deban coincidir en la secuencia de mensajes a entregar a la aplicación.

6. Diseñar e implementar un sistema de gestión de datos distribuidos con replicación. Cada dato se accede por su nombre y debe estar replicado en al menos dos *data servers*. Un cliente podrá acceder a cualquier servidor. En caso que ese servidor no contenga ese archivo deberá redirigir al cliente la lista de servidores que lo tienen para que luego éste contacte a alguno de ellos. Las operaciones sobre un dato pueden ser de *read* o *write*. Cada servidor mantiene una lista de los archivos que mantiene (por simplicidad, fijado en tiempo de inicialización del sistema). Cuando un servidor queda fuera de línea, al volver debe actualizar sus datos desde el otro servidor que mantiene las copias de sus archivos.

7. ***Proyecto para alumnos/as del curso Redes y Telecomunicaciones***

   Desarrollar una aplicación web que permita interactuar a estudiantes y profesores en una *pizarra virtual*. El sistema deberá permitir varias pizarras en uso simultáneamente por diferentes cursos (profesores y alumnos). Inicialmente el único que puede dibujar en la pizarra es el profesor pero puede darle la tiza a un alumno determinado. Cada usuario (profesores y alumnos) se deben *unir* a una pizarra. La interface web debe mostrar los participantes usando una pizarra con su rol (profesor o alumno).
   Enlaces útiles:
   1. https://developer.mozilla.org/es/docs/Web/API/Canvas_API/Tutorial
   2. Websockets clients:
   https://developer.mozilla.org/es/docs/Web/API/WebSockets_API/Writing_WebSocket_client_applications
   3. WebSockets con node.js.
   https://dev.to/codesphere/getting-started-with-web-sockets-in-nodejs-49n0

Nota: Para el desarrollo de un proyecto se evaluará la selección de herramientas, middleware y la solución de los algoritmos seleccionados. Cada selección debe estar fundamentada apropiadamente. Incluir las referencias bibliográficas correspondientes.