Agente

* Inicio automático
* Recopilar información de SO
* Recopilar información de Red
* Recopilar información de App activa
* Enviar información mediante la librería

Servicio de recepción (Servidor Central)

* Contiene la librería para recibir los mensajes.
* Recibe mensajes enviados por los agentes y por las aplicaciones de sistemas distribuidos.
* Almacenar la información en base de datos
* Reenviarla al módulo de monitoreo dividiendo la información (agente y aplicaciones distribuidas)
* Ejecuta los scripts de SO con sus debidos parámetros

Aplicaciones de Sistemas Distribuidos

1. Características de los Sistemas Distribuidos

Una app servidor que posea un conjunto de archivos locales y puedan ser abiertos en modo lectura por el usuario (recursos), con la apertura se abre el servicio de servidor web para acceder a una página web y obtener el contenido de un index.html (apertura). al acceder otro nodo al mismo archivo no hay problema (concurrencia). Se agrega un segundo nodo servidor idéntico al primero (Escalabilidad). Si se cae un nodo, los usuarios de todas formas pueden acceder a los archivos y al servidor web (tolerancia). El nodo cliente percibe el sistema como un solo nodo (Transparencia).

1. Desafíos de los Sistemas Distribuidos

Una aplicación servidor que permita la conexión utilizando sockets seguros y SSL (Seguridad). El cliente busca un archivo y se hace checksum de el para validar que se pasó correctamente a través de la red (Tolerancia a fallos). Se agrega otro servidor idéntico al primero y los clientes ocurre nada (Escalabilidad) Si otro cliente quiere descargar el archivo, se utiliza un semáforo que controle los accesos al archivo (Concurrencia).

1. Arquitectura Cliente / Servidor

* Una app servidor que se encuentre esperando por peticiones de cliente
* Una app cliente que se conecte a un servidor indicado. y envíe un mensaje.

1. Arquitectura Punto a Punto

* Una app que permita enviar mensajes a cualquier host.

1. Sockets

* Una app que permita abrir un socket hacia otra máquina y enviarle un mensaje, y además que pueda escuchar peticiones por un puerto.

1. RMI

* Una app que permita invocar métodos de otro nodo para enviar un mensaje preestablecido.

1. Comunicación en grupo

* Una app que permita unirse a un grupo de nodos, si se encuentra unido al grupo le envía mensajes a todos.

1. Relojes lógicos: Algoritmo de Lamport

* Una app que mande marcas de reloj a otro nodos sin el algoritmo de lamport, se detiene cuando el tiempo de la actual es menor que la recibida y lo que se hace es acomodar el tiempo y continua!

1. Relojes físicos: Algoritmo de Cristian

* Una app que sirve como servidor que recibe un mensaje cada cierto tiempo de la app cliente que envía su hora actual y mediante <formula> se saca el tiempo promedio y se le envía a cada cliente.

1. Relojes físicos: Algoritmo de Berkeley

* Un app servidor que va midiendo cada cierto tiempo el tiempo de todos los clientes y realiza cálculo del promedio para enviárselo a todos los clientes.
* Un app cliente que tenga un reloj con una hora determinada en cada nodo.

1. Relojes físicos: Algoritmo con promedio (distribuido)

* Una app cliente / servidor que envía su hora cada cierto tiempo, luego se inicia un cronómetro local para recibir las demás transmisiones. Al finalizar se obtiene el tiempo local de acuerdo al promedio de todos las respuestas.

1. Exclusión mutua: Algoritmo centralizado

* Una app (servidor) que sirva como coordinador.
* Una app cliente que envíe un mensaje al coordinador cuando quiera acceder a la región crítica de otro nodo, si esta se encuentra libre, entonces accede, caso contrario se envía un mensaje de negación.

1. Exclusión mutua: Algoritmo distribuido (Ricart y Agrawala)

* Una app que envíe a los otros nodos un mensaje con la hora actual, nombre de la región y id del proceso, los otros nodos al recibir el mensaje envía un Ok cuando no es una región crítica de el. Si uno de los nodos está en una región crítica entonces no responde y guarda el mensaje en una fila\* , si uno de los nodos no está en la región, pero quiere entrar, entonces compara la marca de tiempo con la de el , si la marca recibida es menor, entonces envía Ok, caso contrario forma la solicitud en una fila y no envía nada. Cuando el emisor obtiene tantos Ok como procesos entonces puede acceder a la región crítica, cuando termina el trabajo en la región crítica le envía el mensaje al siguiente proceso en cola.

1. Algoritmo de selección: Grandulón

* Una app que envíe un mensaje P a los demás mayores que el, si nadie responde, gana P y manda mensajes a los demás indicando que es coordinador, si uno de los procesos mayores a P responde, P deja de ser coordinador. (Se podría tener un nodo coordinador desde un principio, se cae y luego se elige el coordinador).

1. Algoritmo de selección: Anillo

* Cada nodo sabe cual es el siguiente, cuando el coordinador no funciona, cada nodo coloca su id de proceso y envía el mensaje al siguiente, cuando el primero recibe el mensaje, elige el coordinador de acuerdo a un criterio!

1. Tipos de fallas

-Falla de congelación: Se detiene el servidor luego de estar funcionando normalmente.

-Falla de omisión: El servidor no responde a las peticiones entrantes (del cliente).

-Falla de tiempo: La respuesta del servidor queda fuera del intervalo de tiempo específicado.

-Falla de respuesta: La respuesta del servidor es incorrecta.

1. Fallas bizantinas.

* Una app que envíe a todos los demás nodos su id y cada uno recolecta los resultados en un vector, se transfiere el vector a todos y se compara cada casilla de los vectores, si no hay mayoría se coloca como UNKNOWN

1. Arquitectura cliente servidor: Modelo de acceso remoto

* Un app servidor que contenga archivos ubicados en diferentes lugares y sean accedido a través de una interfaz
* Un app cliente que contenga la interfaz de los archivos disponibles para descargar.

1. Arquitectura cliente servidor: Modelo de carga y descarga

* Una app servidor que contenga archivos.
* Una app cliente que acceda al servidor, descargue un archivo, lo utilice y vuelva a cargar el archivo!

1. Sistemas de archivos basados en Clúster.

* Un app servidor que permita dividir un archivo muy grande en partes y almacenarlo en cada servidor \*
* Una app cliente que acceda al archivo.

1. Domain name system (DNS)

* Un app servidor que contenga un conjunto de dominios asociados a ips.
* Un app cliente que dado un dominio lo envíe al servidor y este devuelva el ip.

1. Lightweight Directory Access Protocol (LDAP)

* Una app servidor que contenga un conjunto de usuarios guardados en una estructura
* Un app cliente que envie el nombre de un usuario (uid) y devuelva el árbol de jerarquía.

1. Remote Method Invocation (RMI)

* Una app que muestre rmi a fondo…

1. Enterprise Java Beans (EJB)

* EJB Con estado y sin estado.

1. SOAP

* Una app servidor que contenga el servicio mediante el WSDL?
* Una app cliente que consuma ese servicio

1. REST

* Una app servidor que contenga el servicio
* Un app cliente que consuma el servicio a través de la URL.

Librería:

* Recibir mensajes a través de un puerto
* Enviar mensajes a uno o más destinatarios
* Atributos configurables (ip destino, ip remitente, mensaje) etc
* Simple: m.mensaje(“hola mundo”);

Módulo de Ciclo de Vida

* Controlar los nodos por parte del usuario
* Botones para ver los eventos, un evento es: iniciar la aplicación, detener un nodo, todos los nodos o enviar un mensaje a un nodo específico.
* Realiza el deploy de los ejecutables en cada raspberry
* Contiene información de la cantidad de nodos disponibles en el sistema distribuido.
* Se comunica con el servidor central a través de la librería de mensajes quien localiza el script específico para su ejecución

Módulo de Monitoreo

* Permite visualizar los mensajes enviados por los agentes y las aplicaciones a través del servidor central.
* Posee dos tipos de mensaje a mostrar: De configuración y de información entre los nodos.
* Muestra la información en tiempo real

Módulo de Gestión

* Permite crear, consultar, modificar o eliminar un tópico o pregunta
* Requiere de un registro previo por parte del usuario
* Consultar toda las tablas de la base de datos.