DOCUMENTO DE DISEÑO DE SISTEMA (SDD)

MODELO DE AUTOASIGNACIÓN DE ROLES EN SISTEMAS DISTRIBUIDOS PARA NODOS EQUIVALENTES

AUTOR:

Ing. Juan Carlos Zárate Trejo – Programador/Diseñador

STAKEHOLDERS:

Dr. Félix Francisco Ramos Corchado

28 DE ABRIL DE 2023

CINVESTAV- Unidad GDL

Prefacio

Se espera que este documento proporcione los recursos necesarios para la implementación y estudio del modelo para sistemas distribuidos orientados a disponibilidad de servicios. Dichos recursos abarcarán los supuestos considerados para el diseño de la arquitectura, la arquitectura misma, librerías, lenguajes de programación, interfaces y demás tecnologías que hacen posible el funcionamiento.

Se espera que los principales usuarios de este documento sean:

- Clientes del sistema.
- Arquitecto de software.
- Ingeniero administrador de proyecto.
- Ingeniero de implementación del sistema.
- Ingeniero de pruebas del sistema.
- Ingeniero de mantenimiento del sistema.

Contenido

Introducción

En este documento se presentarán los aspectos principales en cuanto a diseño se refiere del modelo y su implementación para solventar el problema de la caída de nodos en una red distribuida mediante la pseudo - descentralización y algoritmos de coordinación entre nodos independientes. En el área de los sistemas distribuidos, continuamente se aborda el tema del fallo de dispositivos y la posibilidad de respaldar sus funciones, sin embargo, en ocasiones se trata de una tarea que requiere inversiones adicionales o bien, redundancia en el sistema, lo cual podría ser complicado de gestionar a nivel de software y hardware.

La propuesta presentada en este trabajo aborda dichos aspectos desde la dinámica de la disponibilidad de recursos, es decir, qué tareas hay por hacer y quién las puede hacer.

Resumen del documento

Los objetivos de diseño específicos están contenidos en este documento, así como la estructura arquitectónica que espera ser implementada; además de identificar cualidades de un patrón de diseño con las características de diseño del sistema propuesto. El patrón de diseño arquitectónico propuesto permite modelar la arquitectura apropiada específicamente a este requerimiento mediante un modelo para separar la comunicación de los datos dentro del sistemas.

1. Propósito

Este documento describe los requerimientos de software del sistema para gestión de redes de nodos con persistencia en disponibilidad de servicios.

1.1 Alcance del proyecto

El modelo se desarrollará enfocándose inicialmente en el supuesto de redes locales distribuidas y donde los nodos cuentan con recursos computacionales equivalentes.

Alcance:	Este documento de requerimientos de software es la base del desarrollo de software del proyecto. Describe los siguientes tópicos: - Requerimientos del usuario. - Requerimientos del sistema. - Requerimientos funcionales. - Requerimientos no funcionales. - Requerimientos de dominio.

2. Referencias:

IEEE Estándar 1016-1998: Práctica recomendada de IEEE para el diseño de sistema de software. IEEE, 1998.

Antes de leer este documento de diseño del sistema (SDD), se recomienda leer el documento de análisis de requerimiento (SRS), que proporciona todos los requerimientos funcionales, requerimientos no funcionales y de dominio descritos sobre este sistema.

3. Definiciones

Sistema distribuido: Un sistema distribuido es un conjunto de programas informáticos que utilizan recursos computacionales en varios nodos de cálculo distintos para lograr un objetivo compartido común.

API: Application Programming Interface (Interfaz de programación de aplicación).

Socket: Los sockets son canales de comunicación que permiten que procesos no relacionados intercambien datos localmente y entre redes. Un único socket es un punto final de un canal de comunicación bidireccional.

Seekers: Término utilizado en el proyecto para referirse a uno de los modos de nodo caracterizado por buscar en la red para confirmar la presencia o ausencia de un nodo maestro.

Master: Término utilizado ene el proyecto para referirse al modo maestro de los nodos, caracterizado por administrar las conexiones y gestionar los nodos que forman parte de la red, así como aquellos que entran y salen.

Threads: Los hilos (threads) son accesos al sistema de administración de memoria de un dispositivo para la gestión del consumo y la prioridad de ejecución de una operación que se ejecuta durante el procesamiento de información de la aplicación.

SDD: Documento de Diseño del Sistema.

SRS: Documento de requerimientos del sistema.

Python: Es un lenguaje de propósito general que se puede utilizar para desarrollo web, análisis de datos, inteligencia artificial, automatización de tareas, entre otros.

RPC: "Remote Procedure Call" o llamada a procedimiento remoto en español. Es una técnica que permite que un programa acceda a otro en un sistema remoto como si fuera local.

RMI: Es una tecnología desarrollada por Sun para permitir la colaboración de objetos que están localizados remotamente. Esta tecnología se enmarca en la idea de permitir colaboración entre Objetos Remotos.

4. Propósito del sistema

El propósito de este sistema es proporcionar una referencia al diseño de sistemas distribuidos donde se busque la gestión de los nodos miembros de dicho sistema, a su vez que se gestiona los estados de desconexión de los mismos y el reemplazo de roles.

4.1 Objetivo de diseño

El diseño del modelo de comunicación tiene como objetivo principal garantizar la disponibilidad del sistema distribuido mediante la implementación de mecanismos de tolerancia a fallos que permitan la detección y recuperación de errores en tiempo real. Para lograr esto, se deben definir y diseñar los diferentes componentes del sistema, incluyendo el nodo maestro y los nodos, estableciendo los protocolos y mecanismos de comunicación necesarios para su correcto funcionamiento. Además, el diseño también debe asegurar la escalabilidad y flexibilidad del sistema, permitiendo que se adapte a los cambios en el entorno y en los requisitos del usuario sin comprometer su rendimiento. Esto implica la implementación de un diseño modular y extensible, que permita la incorporación de nuevos nodos y funcionalidades de manera sencilla.

Asimismo, el diseño debe satisfacer los requisitos funcionales y no funcionales definidos para el sistema. En cuanto a los requisitos funcionales, se deben definir las funciones que el sistema debe realizar, como la gestión de los nodos, la detección de fallos y la elección de un nuevo maestro. Por otro lado, los requisitos no funcionales deben asegurar el rendimiento y la eficiencia del sistema, como la tolerancia a fallos, el tiempo de respuesta y el uso adecuado de los recursos del sistema.

4.2 Rendimiento

El objetivo del diseño del modelo de comunicación para el sistema distribuido es asegurar un alto rendimiento en términos de velocidad, capacidad, escalabilidad y eficiencia en el uso de los recursos del sistema. Para lograr esto, se deben realizar ciertas suposiciones sobre el sistema, como que se espera un alto volumen de tráfico de red y que los nodos pueden estar geográficamente dispersos. Una de las principales consideraciones en el diseño es el uso eficiente de los recursos del sistema, como el ancho de banda de red, el espacio de almacenamiento y la capacidad de procesamiento. El diseño debe minimizar la sobrecarga de la red, por lo

que se deben establecer protocolos de comunicación optimizados y eficientes para minimizar la cantidad de datos que se transmiten entre los nodos y el nodo maestro.

Además, el diseño debe ser escalable para acomodar el crecimiento futuro del sistema, por lo que se deben implementar soluciones que permitan añadir nuevos nodos sin afectar el rendimiento general del sistema. Esto puede lograrse mediante el diseño de un sistema modular y distribuido, que permita la adición y eliminación de nodos en tiempo real. Otra consideración importante es la tolerancia a fallos, que es esencial para garantizar el rendimiento del sistema en caso de que uno o varios nodos fallen. Se deben establecer protocolos y mecanismos de recuperación de fallos que minimicen el impacto en el rendimiento del sistema en caso de fallos de nodos individuales.

4.3 Usabilidad

Se espera garantizar la usabilidad del modelo, a través de la implementación de un diseño intuitivo y fácil de usar que permita a los usuarios interactuar con el modelo de manera efectiva y eficiente. Para lograr esto, se deben tener en cuenta las necesidades y características del modelo de comunicación, como la gestión del nodo maestro y los nodos secundarios, la verificación de la conectividad de los nodos y la tolerancia a las fallas.

El diseño del software debe proporcionar una interfaz clara y fácil de entender que permita a los usuarios interactuar con el modelo de manera efectiva, y que sea fácil de usar para usuarios con diferentes niveles de conocimiento técnico. Además, se debe garantizar la facilidad de instalación, configuración y uso del modelo, a través de la implementación de una guía de usuario clara y detallada, así como la disponibilidad de soporte técnico para ayudar a los usuarios en caso de cualquier problema.

4.4 Confiabilidad

La recuperabilidad y tolerancia a fallos que se espera como respuesta del sistema, permitiendo un modelo que permita que el sistema donde se implemente la integración de nodos nuevos, permitir la salida de otros y el respaldo de servicios.

Estrategia:

La tolerancia a fallos en el modelo de comunicación distribuida consiste en un equilibrio entre los módulos de gestión, que permita controlar las conexiones y el consumo de servicios de manera eficiente para evitar la saturación del sistema. Además, se deben detectar y reportar los errores para su corrección durante la fase de pruebas del sistema, sin que se vea afectada la operatividad del nodo maestro y de los demás nodos.

4.5 Mantenibilidad

La mantenibilidad de este sistema se enfocará principalmente en la adición de servicios de control y seguridad, como su integración con métodos de cifrado para la distribución de tokens de comunicación. El mantenimiento del sistema se ha enfocado en el desarrollo de herramientas para el monitoreo constante del estado del sistema y la identificación de posibles fallas. Se ha implementado un sistema de logs detallado que registra todas las acciones realizadas por el sistema, lo que facilita la identificación de problemas y la realización de mantenimiento preventivo.

El sistema también cuenta con una estructura de pruebas y validación que permite detectar errores de forma temprana y asegurar la calidad del código. Para facilitar la integración de nuevas funcionalidades y mejoras, se han implementado prácticas de desarrollo ágil que permiten la iteración constante y la incorporación de cambios de forma rápida y segura.

5. Funcionalidad general del sistema

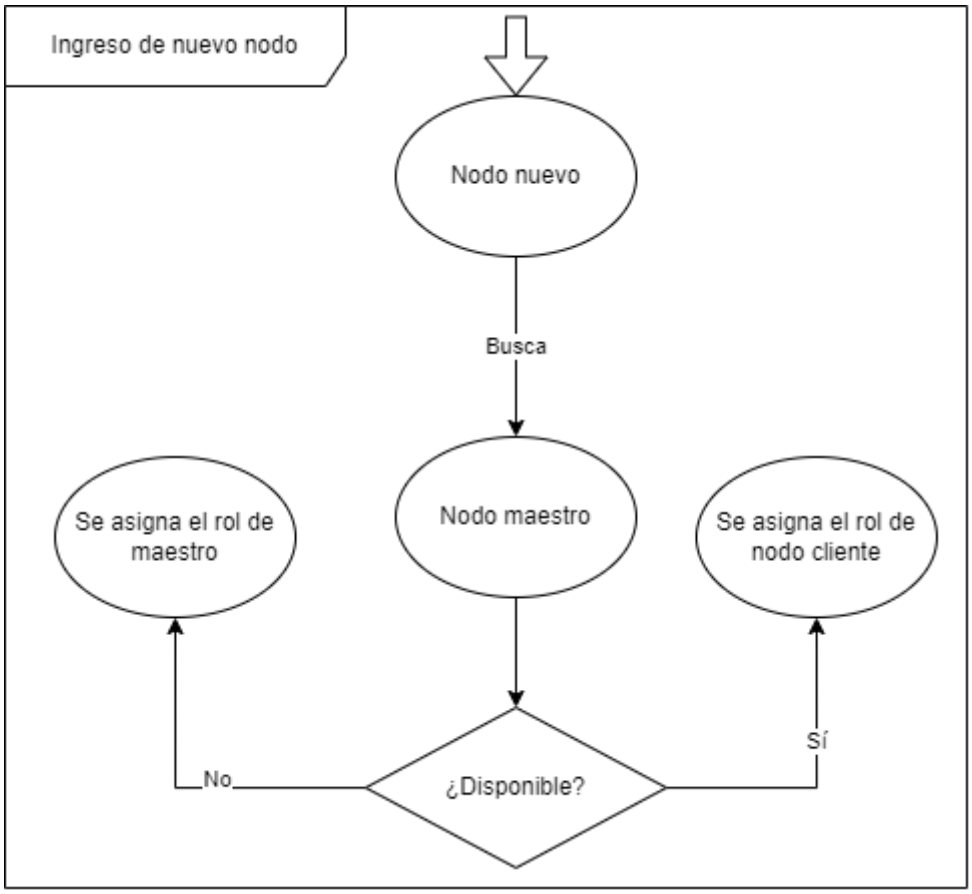
El proyecto consiste en un modelo de comunicación para un sistema distribuido donde el nodo maestro se encarga de gestionar al resto de nodos (Verificar que se mantengan en línea), a su vez que, si un nodo se desconecta, este tiene un tiempo de tolerancia para recuperarse antes de que el maestro deje de verificar su conectividad y lo ignore por completo. Por otro lado, los nodos pueden darse cuenta de que el maestro ha caído, y llevan a cabo una subrutina para decidir un nuevo maestro, comenzando nuevamente el proceso.

6. Modelo de software

El modelo arquitectónico describe a alto nivel el diseño propuesto vital para el funcionamiento del sistema.

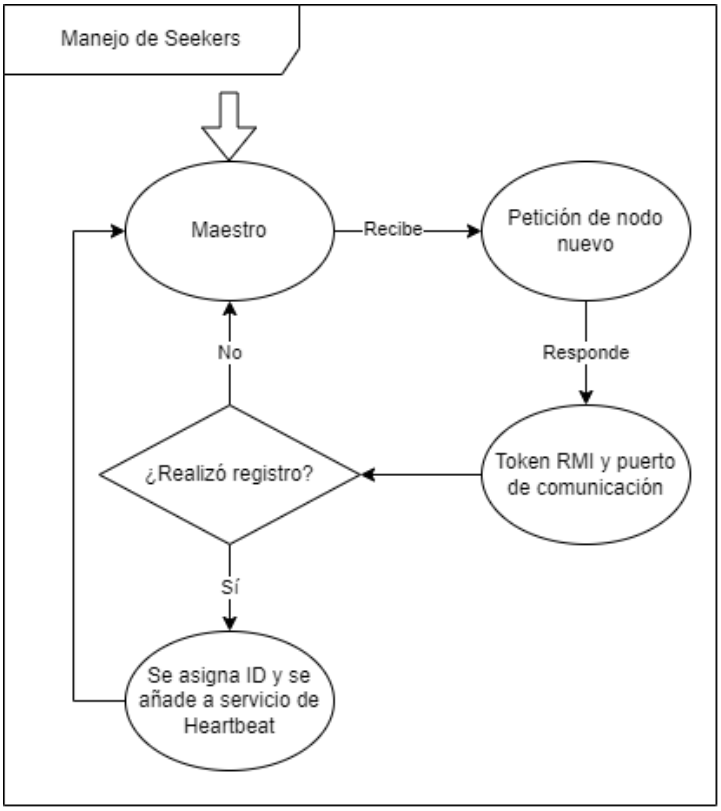
6.1 Ingreso de nodos nuevos

Describe el procedimiento que se realiza cuando un nodo inicia su ejecución por primera vez. Dado que es posible que no haya ya un maestro este puede asumir dicho rol, en caso contrario, se vuelve parte de la red consultando al maestro en turno.



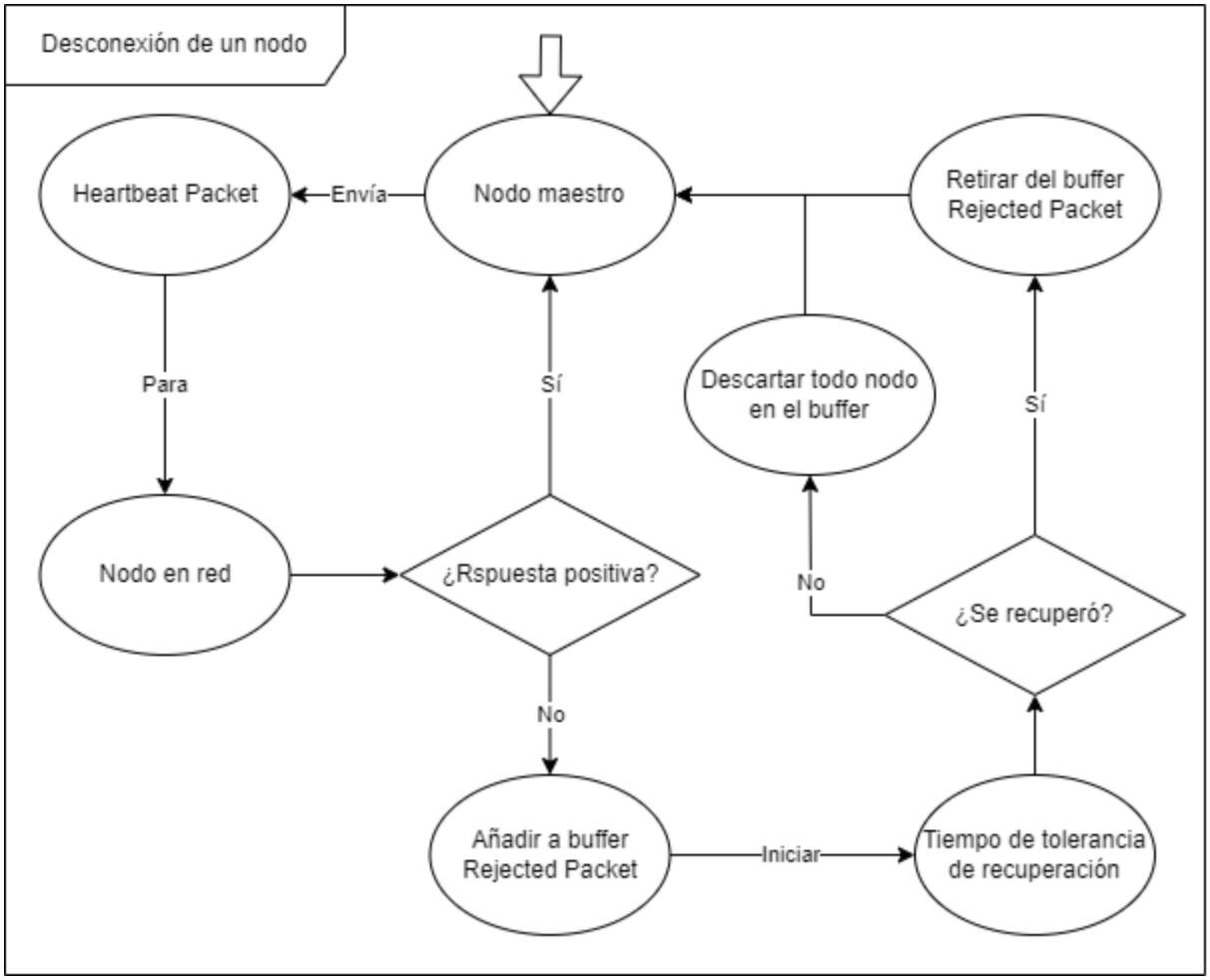
6.2 Manejo de Seekers

Cuando los nodos inician su etapa de descubrimiento, entran en modo Seeker, de manera que su tarea solo sea buscar un maestro durante un tiempo determinado. En cuanto el maestro de turno recibe una petición de entrada de un Seeker, este responde con el Token para llamar mediante RMI al método correspondiente para registrarse y que el maestro tenga conocimiento del puerto por el cual pueda enviar los paquetes Heartbeat. Este registro permite también que el nodo obtenga un identificador y un evento para iniciar sus propios servicios.



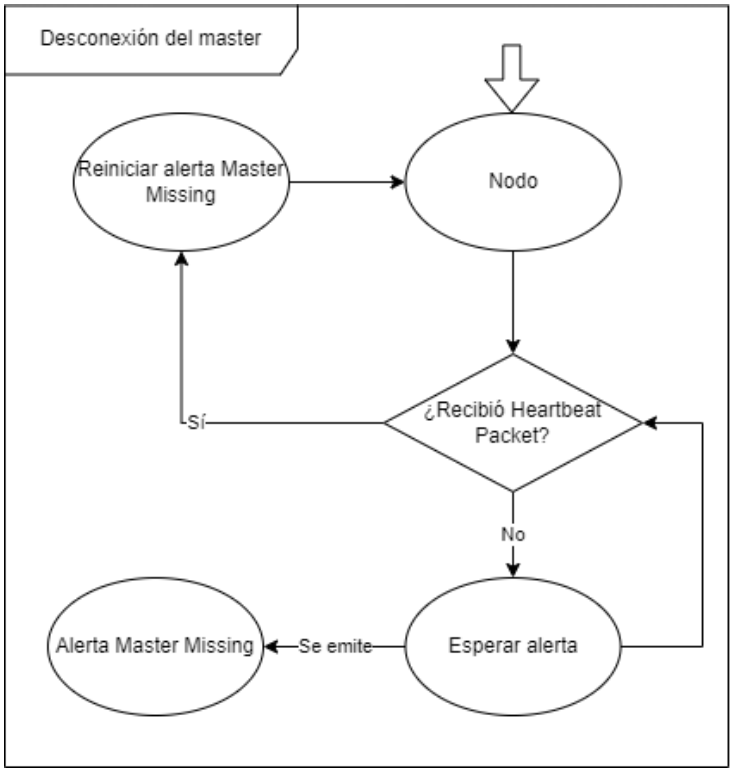
6.3 Desconexión de un nodo

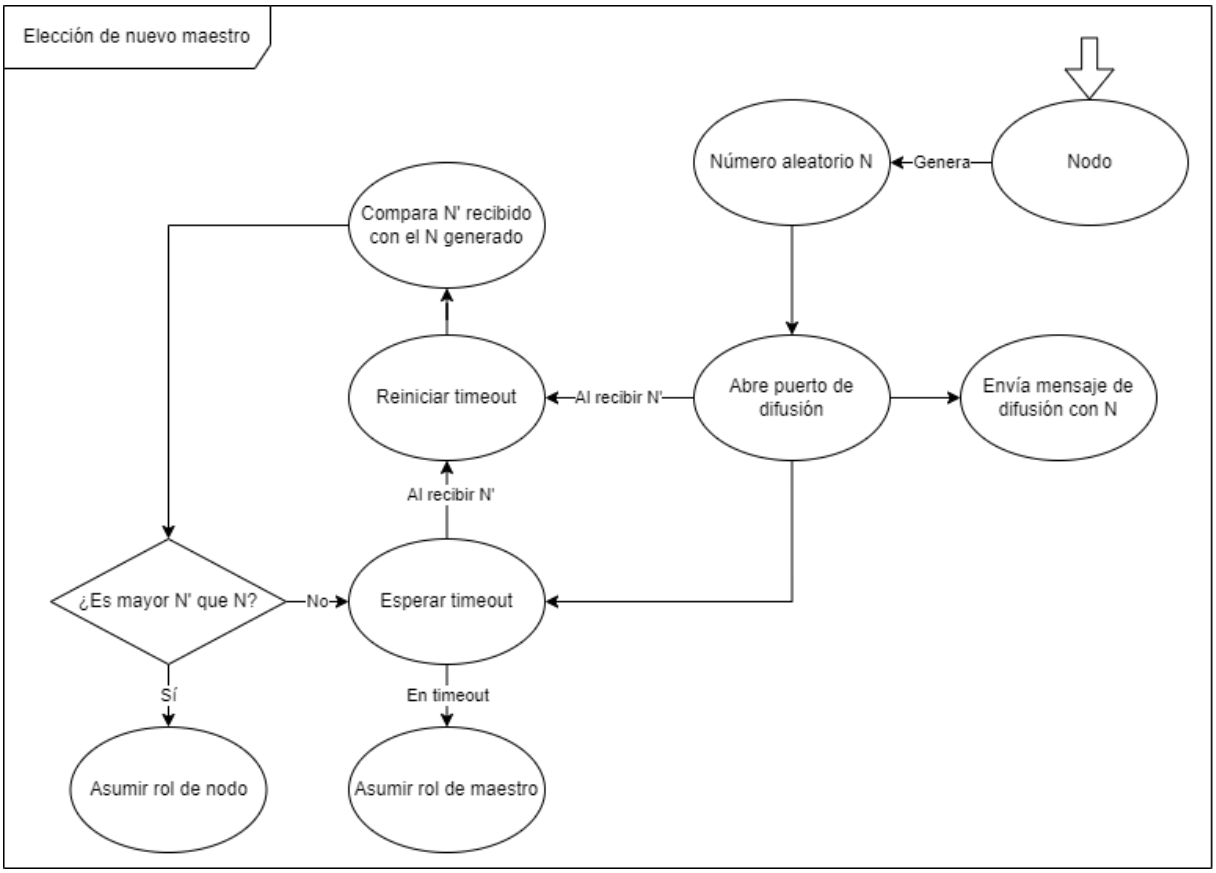
Si un nodo falla, el maestro asume que solo se ha perdido el paquete de Heartbeat, por lo cual ingresa al nodo a un buffer de verificación de paquetes rechazados. Para cualquier nodo en dicho buffer se cuantifica el margen disponible desde que rechazó el primer paquete. Este margen permite que el nodo pueda reintegrarse conservando los datos que el maestro haya recolectado de él. Por otro lado, una vez que algún nodo sobrepasa dicho margen, este es eliminado del registro del maestro, de manera que el maestro ya no dedica recursos a intentar comunicarse con dicho nodo.



6.4 Desconexión del maestro

Dado que los nodos no almacenan información del maestro en este modelo, no existe una rutina de recuperación del maestro de parte de los nodos. Para esta situación, los nodos poseen una alerta que es reiniciada cada vez que el master envía un Heartbeat Packet. Dicha alarma se configura de manera que el nodo pueda asegurar que, si el maestro se encuentra en línea, en algún momento dentro de ese margen llegará un Heartbeat Packet. Si esto no se cumple, la alarma sonará, el nodo asume que el maestro ha caído y se iniciará la rutina de consenso mediante la cual se coordinará con el resto de los nodos para determinar al nuevo maestro.

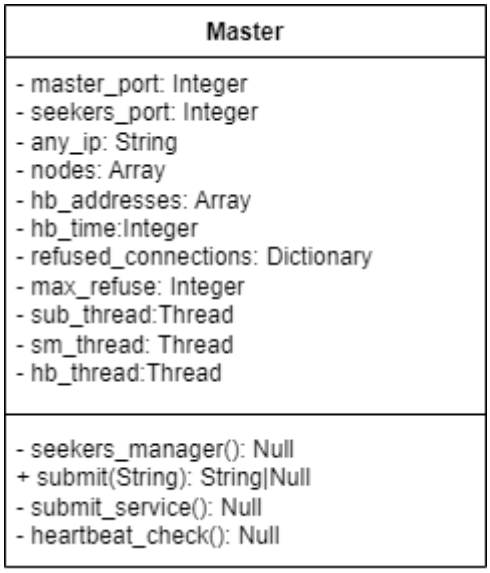

Desconexión del master
Reiniciar alerta Master Missing
Nodo
¿Recibió Heartbeat Packet?
Sí
No
Alerta Master Missing
Se emite
Esperar alerta


Elección de nuevo maestro
Número aleatorio N
Genera
Nodo
Compara N' recibido con el N generado
Reiniciar timeout
Al recibir N'
Abre puerto de difusión
Envía mensaje de difusión con N
Al recibir N'
¿Es mayor N' que N?
No
Esperar timeout
Sí
En timeout
Asumir rol de nodo
Asumir rol de maestro

7. Diagramas de clase

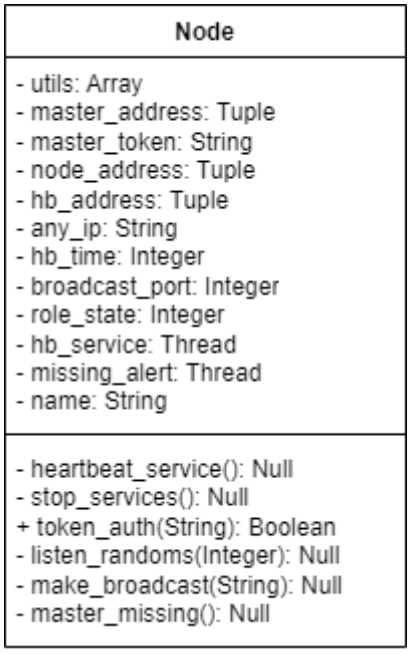
7.1 Maestro

Posee las propiedades y métodos necesarios para la gestión de nodos nuevos y el servicio de verificación de estado de conexión de los nodos miembros.



7.2 Nodo

Posee las propiedades y métodos necesarios para ejecutar las tareas de registro, respuesta al servicio Heartbeat y consenso para determinar el nuevo maestro.



7.3 Seeker

Posee la funcionalidad necesaria para determinar si existe un maestro en la red y el formato para interpretar el mensaje de respuesta con el Token y la dirección.