PRÁCTICA 4.
MEMORIA.

Gonzalo Jiménez Carretero
Luis Ignacio Pastor Cristóbal
2º Ing. Informática

APARTADO 4

- **¿Cómo añadirías nuevos métodos de minado y validación?**

Si se quisieran crear nuevos métodos de minado o validación bastaría con crear la clase correspondiente, haciendo que implementase la interfaz "*IMiningMethod*" (ubicada en el paquete "*MiningMethods*") o la interfaz "*IValidateMethod*" (en el paquete "*ValidateMethods*"), respectivamente. Estas interfaces poseen los prototipos necesarios para hacer tanto el minado como la validación de forma correcta, que serán usados a la hora de interactuar con las transacciones y los bloques.

Por ejemplo, si quisiera crear una clase de minado avanzado ("*AdvancedMining*", por ejemplo), debería de crear su clase correspondiente, implementar la interfaz mencionada anteriormente, y sobrescribir las funciones "*createHash*" y "*mineBlock*" con la implementación deseada.

- **¿Cómo añadirías nuevos tipos de mensaje que impliquen otros procesamientos por parte de los distintos tipos de nodos?**

Antes de responder a este apartado, hemos de hablar de la implementación que hemos realizado en la práctica. En nuestro caso, una vez le llega a un nodo/subnet, se hacen comprobaciones para que, en función del mensaje recibido, ejecutará una acción u otra. Obviamente, esta solución no es fuente de buenas prácticas, y es un código que deberíamos de haber evitado a toda costa; no obstante, por problemas que hemos tenido hemos priorizado que funcione correctamente antes que tener un código limpio, aunque como resultado tengamos los casos casi "*hardcodeados*". Además, hemos de mencionar que, aunque se nos dijo que no podíamos tocar las interfaces, creemos que añadiendo alguna función más se podría haber hecho más genérico la detección de los mensajes. Explicada toda la situación, para este apartado tenemos dos posibles respuestas, cómo es posible hacerlo en nuestro código y cómo se podría hacer en un código bien implementado.

En nuestro código, bastaría con añadir en la clase "*Node*" y "*Subnet*" los casos para los mensajes nuevos en la función "*broadcast*". Además, en "*MiningNode*" también se deberían sobrescribir las funciones necesarias para que su comportamiento sea diferentes al de un nodo normal.

En una buena implementación, no haría falta poner los casos específicos, sino crear funciones genéricas que llamen a funcion que deban ser modificadas en función del tipo de elemento en el que se esté. Por ello, las modificaciones no deberían hacerse en los nodos, sino en funciones que se deberían de crear en los propios mensajes. Así, todos los mensajes podrían tener distintos comportamientos, pero siendo llamados de la misma forma (al igual que en el apartado anterior respondido), lo que lo haría lo más genérico posible.

Estas soluciones son las que se nos han ocurrido a nosotros, aunque por la falta de experiencia en lenguajes orientados a POO probablemente no sean las más eficientes u óptimas que podría haber.

Diagrama UML

A continuación, se adjunta una fotografía del diagrama UML resultante de la realización del proyecto. Debido a que es grande y es posible que no se pueda ver bien, también se encuentra en la raíz de la entrega el PDF con el diagrama al completo, en el que se ve perfectamente.