IIC3272 — Criptomonedas y contratos inteligentes

# Tarea 4: Red de Bitcoin

Entrega: Junio 24, 2020

## 1. Cosas administrativas

Cada tarea en este curso equivale a 5 % de la nota final. En total tendremos 5 tareas, pero la peor tarea que solucionan no cuenta para la nota final. Quiere decir que pueden botar una tarea.

**La solución se entrega usando la plataforma Canvas en un zip.**

El uso de materiales (o soluciones) encontrados en Internet está permitido. Solo se les pide citar la fuente que utilizaron. No se aplica ninguna penalidad para su uso.

## 2. La tarea

En esta tarea simularemos lo que hace un full node de Bitcoin: conseguir bloques enteros.

Usando el código explicado en clases (`block.py` y `network.py`), deberían agregar soporte para los mensajes que nos permiten conseguir primeros 20 bloques de la testnet de Bitcoin (el genesis block ya está definido como una constante en nuestro código). Aquí se los pide descargar los bloques enteros, y no solo los headers (entonces tendrán que incluir todas las transacciones).

Para esto, necesitan hacer dos cosas. Primero, necesitan definir el mensaje que les permite conseguir un bloque entero desde un nodo. Como la respuesta a este mensaje, el nodo les enviará la serialización de un bloque. Para manejar este objeto, tendrán que implementar la clase `FullBlock` en `block.py`. En el código entregado ya existe un borrador de la clase con los métodos necesarios para su funcionamiento correcto. Para poder definir el método `parse` de la clase `FullBlock`, necesitan parsear todas las transacciones que ocurren en este bloque. Para parsear una transacción pueden ocupar el código disponible en `txP2PKH.py`.

Todos los detalles de serialización y de la lógica de mensajes que se mandan por la red de Bitcoin pueden encontrar en `https://en.bitcoin.it/wiki/Protocol_documentation`. Si no entienden cómo definir a un mensaje o cómo mandarlo a un nodo tienen que revisar el código entregado en la clase sobre la red de Bitcoin.

Una vez implementado, deberían hacer lo siguiente:

1. [**6 puntos**] Usando su implementación, deberían descargar los primeros 20 bloques de la testnet de Bitcoin y imprimir sus hashes, y el MerkleRoot.

2. [**1 punto**] Una vez que tienen los bloques, intentan validar las transacciones de cada bloque usando la clase `Tx` definida en `txP2PKH.py`. La ejecución resultaría en un error. Expliquen por qué ocurre este error. No es necesario corregirlo, solo se les pide por qué la validación falla.

**Entrega:** Para esta tarea, tienen que entregar el código que utilizaron para realizar la tarea, junto con una explicación de como usar su código para conseguir los bloques, y una explicación del error que ocurre cuando intentan validar las transacciones de cada bloque.

**Bonus 1. [5 % de la nota final] Este bonus reemplaza una tarea entera!** El objetivo de este problema es replicar lo mismo de la tarea, pero conectándose a tres nodos de Bitcoin distintos. El bonus tiene dos partes:

1. [**5 puntos**] Usando la implementación disponible en `network.py`, deberían conectarse a tres nodos distintos, y de cada uno descargar los primeros 20 bloques de **mainnet** de Bitcoin. *Ojo de nuevo: de mainnet, y no de testnet.* Después de esto deberían verificar que los datos conseguidos de distintos nodos realmente coinciden.

2. [**2 puntos**] Consigan la dirección de al menos un nodo usando el mensaje `getaddr`. Ojo que la respuesta que recibirán será de tipo `addr`, definido en `https://en.bitcoin.it/wiki/Protocol_documentation`, así que tendrán que deserializar esta secuencia de bytes.

Igualmente cómo en la tarea principal, para este bonus hay que entregar el código, y una explicación de como usarlo para resolver la tarea.