# Proyecto Bitcoin

*Final*


Integrantes:

- Lucas Aldazabal 107705
- Bautista Boselli 107490
- Juan Francisco Gulden 107985

# Conexión entre cliente y servidor

# Tcp stream loop

Creamos un TcpListener y lo asociamos a nuestra direccción IP

Usamos el método  incoming para obtener conexiones establecidas de peers a nuestro servidor

Leemos los mensajes “version” y “ verack” del socket para poder hacer el handshake

# Broadcasteo de headers y bloques

# Broadcasteo de headers

A los clientes que envíen el mensaje “sendheaders” se les broadcastea el header directamente.

A los clientes que NO hayan envíado el mensaje “sendheaders” al momento de broadcastear el header, se envía un  mensaje “inv” con el hash del bloque, para que el cliente lo solicite.

Antes de broadcastear los headers se chequea que el cliente haya solicitado headers al servidor mediante el mensaje “getheaders”. De no ser así, se ignora a dicho cliente.

Cada header nuevo que es descargado por el servidor, se broadcastea una vez que **se haya descargado su bloque correspondiente**.

# Respetar linea temporal

El envío de un header nuevo ocurre cuando recibimos el bloque del mismo para poder tenerlo una vez nos pidan el getdata.

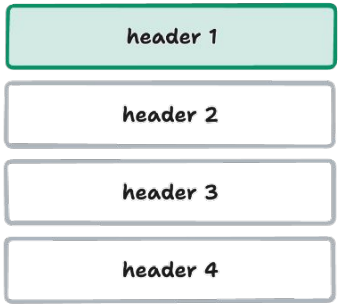
Esto puede traer un **problema**, que los bloques no lleguen en orden como los headers.

Para esto cada vez que llega un bloque, revisamos si el header anterior fue marcado como broadcasteado previamente para no saltearnos alguno, en caso de no haberlo sido, este tampoco se manda.

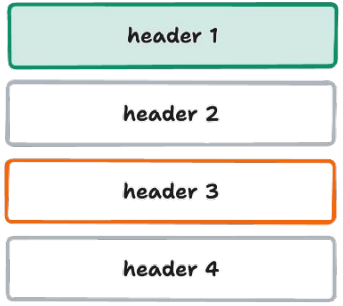
Cuando el bloque recibido es el siguiente a broadcastear, también se revisa si los siguientes consecutivos están descargados

y esperando a este bloque para mandarlos todos juntos y volver a estar al dia.

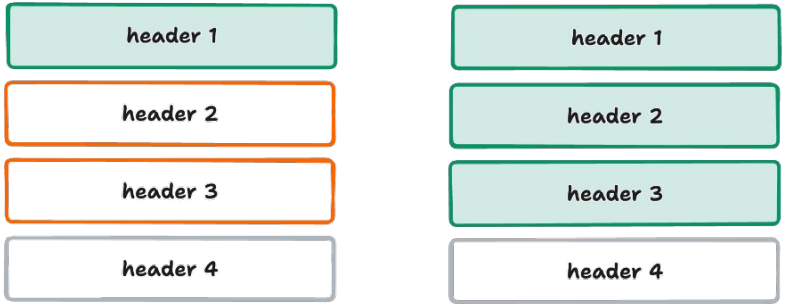
Solo header 1 enviado, los demas pendientes de descarga



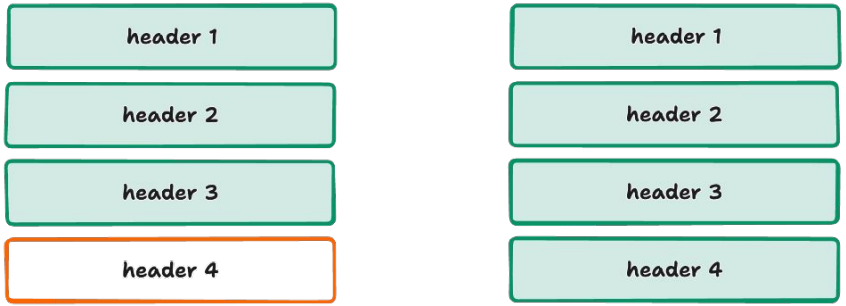

llega el bloque 3, se marca como descargado pero sin envios




llega el bloque 2, se marca como descargado y se puede enviar este y el siguiente




llega el bloque 4, se marca como descargado y se puede enviar



# Calculo de porcentaje para headers

Cuando se descarga una gran cantidad de headers (superior al 5% de todos los headers de la testnet), se imprime el estado de la descarga de una forma distinta:

- Indicando la cantidad de headers descargados por segundo, el porcentaje de headers descargados con respecto a la testnet y el total descargados hasta el momento.

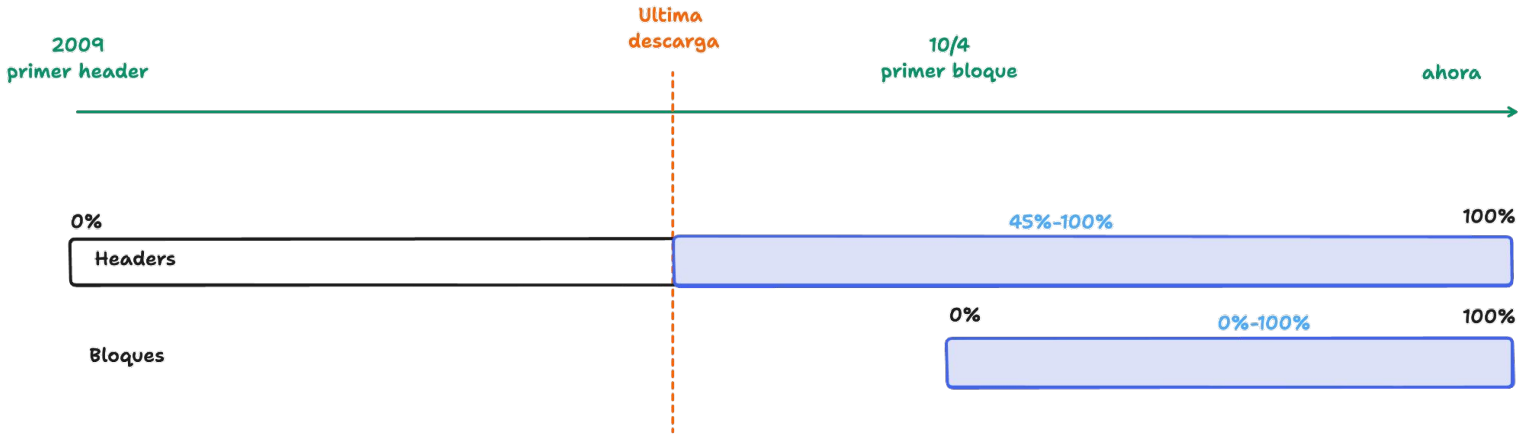
El porcentaje se mide de la siguiente forma:

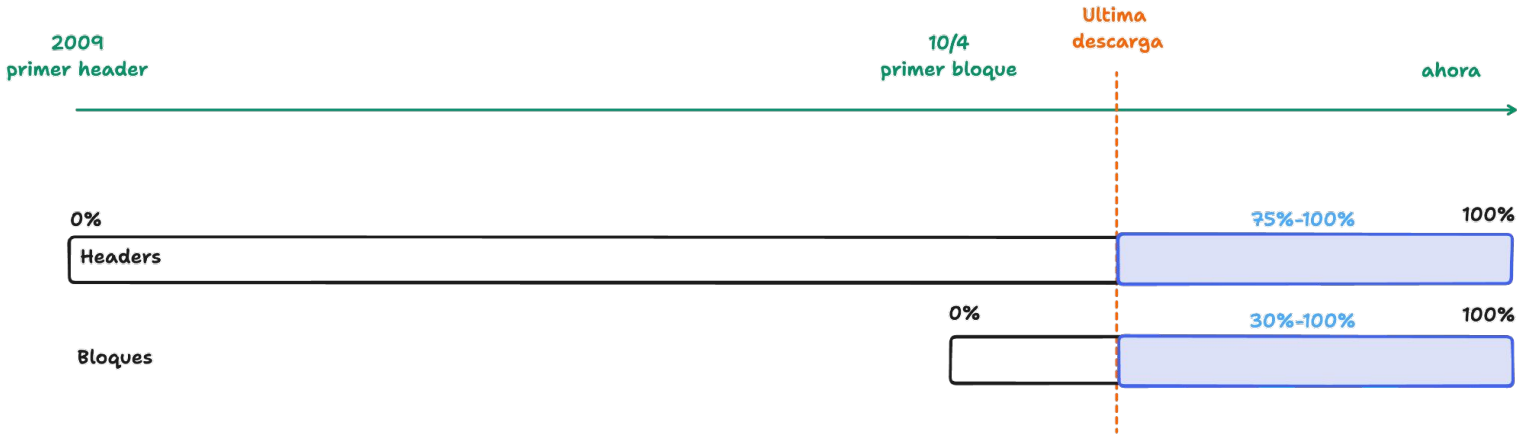
(timestamp del último header descargado - timestamp del primer header descargado de la testnet) * 100 / ( timestamp actual - timestamp del primer header descargado de la testnet)

# Cálculo de porcentaje para bloques

Cuando se descarga una gran cantidad de bloques (superior al 2% de todos los bloques a descargar), se imprime el estado de la descarga de una forma distinta:

- Indicando la cantidad de bloques descargados por segundo, el porcentaje de bloques descargados con respecto a los totales a descargar (del 10/4 en adelante), y el total descargados hasta el momento.


2009
primer header
Ultima
descarga
10/4
primer bloque
ahora
0%
Headers
45%-100%
100%
0%
0%-100%
100%
Bloques


2009
primer header
10/4
primer bloque
Ultima
descarga
ahora
0%
Headers
75%-100%
100%
0%
30%-100%
100%
Bloques

# Nice to have

**Guardado de UTXO:** El Utxo una vez generado, es guardado con el hash del último bloque como “checkpoint” para siguiente reinicio y optimizar el inicio del programa luego de la primera vez.

**Calcular hash una vez:** Al realizar el IBD se guardan los hashes de los headers haciendo que solo se tengan que calcular la vez que fue descargado el header.

**Claridad de transacciones de la wallet:** Se muestran como los movimientos afectan a la wallet de turno. Se podría comparar con un extracto bancario o un historial.

**No duplicado de datos entre interfaz y node:** En todo el programa hay una única fuente de verdad, el “Node State”, haciendo que no sólo no haya diferencia entre lo que tiene cada parte, sino que usa mucha menos memoria

# Nice to have

**Verificar integridad del store:** Al inicio del programa, se revisa que todos los headers posteriores al 10/4 tengan su bloque descargado en la carpeta, si faltase se agrega en el pending block state

**Descartar peers:** Cuando un peer nos responde mal, o directamente falla es eliminado de la lista de Peers

**Elegir peer más rápido:** Cuando se realizan los handshakes, se registra cuánto tarda y con esto elegir al peer más rápido para la descarga de Headers

**Mantener orden de la blockchain:** Los headers solo se almacenan cuando su hash del bloque anterior coincide con el hash de nuestro último header guardado, manteniéndose así ordenada la blockchain en todo momento.

# ¡MUCHAS GRACIAS!