Máster Interuniversitario en Seguridad de las TIC (MISTIC)

TFM – La red P2P de Bitcoin

Alumno: Juan Antonio Donet Donet

Directora: Cristina Pérez Solà

Universitat Autònoma de Barcelona

Fecha de realización: junio 2013

Dedicatòria i agraïments:

A Àngela y Maria, les meues filles per portar-se tan bé i tindre pacència amb un pare que triga tant en acabar els “deures del cole”.

A Àngels, la meua dona, per recolzar-me amb aquesta aventura de fer un Màster i que es mereix un premi per aguantar aquest friki de la informàtica

El meu agraïment també per a tots els professors del MISTIC pels valuosos coneixements que m'han aportat.

I un agraïment espécial per Cristina Pérez, pels seus encertats comentaris i correccions, i perquè no se li ha escapat si un sol accent mal posat ;)

RESUMEN:

El presente trabajo esta centrado en el análisis de la red Bitcoin. En primer lugar se ha llevado a cabo un estudio de la información disponible sobre el tema y sobre el software existente para usar en la red Bitcoin. Utilizando este software junto con una aplicación propia que se ha desarrollado se han estudiado cuestiones como el número de nodos al que nos podemos conectar desde un cliente de Bitcoin, la estabilidad de estos nodos en el tiempo, el número de distintos nodos que se van descubriendo y su procedencia geográfica. Además se ha llevado a cabo pruebas para aumentar el número de nodos simultáneos a los que estamos conectados. Por último se han analizado datos referentes a la temporalización de las transacciones que se realizan en la red Bitcoin y a los bloques que se van generando. Los datos analizados se han obteniendo datos desde un único nodo de la red o desde varios al mismo tiempo y se han contrastado con los datos existentes en la cadena de bloques.

Palabras clave: **Bitcoin, red P2P, cadena de bloques, transacciones**.

RESUM:

El present treball està centrat en l'anàlisi de la xarxa Bitcoin. En primer lloc s'ha dut a terme un estudi de la informació disponible sobre el tema i sobre el programari existent per a usar en la xarxa Bitcoin. Utilitzant aquest programari juntament amb una aplicació pròpia que s'ha desenvolupat s'han estudiat qüestions com el nombre de nodes al qual ens podem connectar des d'un client de Bitcoin, l'estabilitat d'aquests nodes en el temps, el nombre de diferents nodes que es van descobrint i la seva procedència geogràfica. A més s'ha dut a terme proves per augmentar el nombre de nodes simultanis als quals estem connectats. Finalment s'han analitzat dades referents a la temporalització de les transaccions que es realitzen a la xarxa Bitcoin i als blocs que es van generant. Les dades analitzades s'han obtingut dades des d'un únic node de la xarxa o des de diversos al mateix temps i s'han contrastat amb les dades existents en la cadena de blocs.

Paraules clau: **Bitcoin, xarxa P2P, cadena de blocs, transaccions.**

ABSTRACT:

This work is focused on the Bitcoin network. First, it has carried out a study of the information available on the subject and in the existing software for use in the Bitcoin network. This software together with the application developed in this master project, has been used in order to study issues such as the number of nodes we can connect from a client on Bitcoin, the stability of these nodes in time, the number of different discovered nodes and its geographical distribution. Furthermore several tests have been carried out to increase the number of simultaneously connected nodes. Finally, we have analyzed data regarding the timing of the transactions taking place on the Bitcoin network and of the blocks that are generated. The data analyzed were obtained from a single node of the Bitcoin network or from several nodes at the same time. All this obtained information has been compared with the existing data in the block chain.

Keywords: **Bitcoin, P2P network, block chain, transactions.**

Índice de contenido

1 INTRODUCCIÓN.

1.1 JUSTIFICACIÓN DEL TFM

Bitcoin es una moneda que en los últimos tiempos esta adquiriendo mucha notoriedad en los medios de comunicación. En marzo de 2003 por ejemplo apareció una noticia en el Telediario de la TVE comentando que "*La Bitcoin, la moneda electrónica que se creó en 2009 como método alternativo de pago, se ha popularizado en las últimas semanas, dicen que especialmente por la crisis en Chipre. Su valor se ha multiplicado por cuatro. Los bancos centrales y la reserva federal de Estados Unidos han empezado a prestarle atención.*"

Pero a pesar de esta reciente fama en los medios de comunicación generalistas, en los medios especializados Bitcoin ya lleva tiempo siendo objeto de comentarios (tanto a favor como en contra) y es fácil encontrar artículos y blogs, ya sea en medios relacionados con la computación como en los centrados en la economía, en los que se habla de esta nueva y extraña moneda.

Y digo extraña, porque si nos explican que es una moneda que no depende de una entidad reguladora, que no se basa en el oro sino en algo tan intangible como el poder de computación y que encima los nuevos bitcoins se crean de la nada, cuanto menos nos puede resultar un poco raro.

Si se quiere estudiar en que consiste el funcionamiento de esta moneda electrónica se puede encontrar fácilmente información sobre ella. Buscar bitcoin en Google nos devuelve más de 58 millones de resultados en los que se explican desde conceptos básicos (como instalar un billetero electrónico, donde comprar bitcoins, donde gastarlos,...) hasta información más especializada (formato de la cadena de bloques, opciones de minería de datos,..)

Pero sobre el propio funcionamiento de la red P2P de Bitcoin existe poca información más allá de la que aparece en la web oficial de Bitcoin. Este TFM está centrado precisamente en estudiar el flujo de información a través de la red P2P de Bitcoin. Para ello se analizarán parámetros como el número de nodos conectados, el tiempo de conexión de estos nodos, cuanto tarda en propagarse la información a través de la red, especialmente las transmisiones de nuevos bloques y transacciones, y cualquier otra información interesante sobre la red de Bitcoin y su funcionamiento.

1.2 CONTEXTO

Bitcoin es un sistema que ya lleva en marcha unos años (desde el 3 de enero de 2009), pero que puede considerarse aún como una tecnología en evolución. De hecho, existe un procedimiento para proponer y debatir ampliaciones y modificaciones de las normas básicas de la red Bitcoin. Son las llamadas BIP (Bitcoin Improvement Proposal), las cuales se proponen escribiendo a la

lista de correo de los desarrolladores de Bitcoin. Solo cuando existe un amplio consenso favorable a la propuesta, esta pasa a tener el estado de "aceptada".
Se puede ver la lista de BIP en https://en.bitcoin.it/wiki/Bitcoin_Improvement_Proposals.

1.3 Objetivos

A continuación se lista una relación de los objetivos que se pretenden alcanzar en este TFM:

a) Estudiar el funcionamiento y la topología de la red P2P de Bitcoin
b) Analizar el número de nodos que están conectados a la red de Bitcoin en un momento determinado y si su número varía significativamente en el tiempo.
c) Estudiar el número máximo de nodos a los que es posible conectarse simultáneamente.
d) A partir de los resultados obtenidos de los dos objetivos anteriores, analizar si es viable conectarse a todos los nodos simultáneamente.
e) Monitorizar las transferencias en la red para ver desde donde se están originando las transacciones y donde se originan por su parte los nuevos bloques, y comprobar si hay una relación entre ambos orígenes. Es decir, si el número de transacciones y de bloques originados en una zona concuerdan o por el contrario hay diferencias significativas y existen zonas donde se crean transacciones pero no nuevos bloques, o por el contrario, zonas en las que se crean muchos nuevos bloques pero desde donde no se realizan transacciones.
f) Medir cada cuanto tiempo se generan bloques. Comprobar como ha evolucionado este tiempo desde que se puso en marcha el sistema y extrapolar los datos obtenidos para hacer una previsión de la evolución futura.
g) Medir el tiempo entre timestamps en dos bloques consecutivos frente al tiempo de generación de dos bloques consecutivos.
h) Comprobar si hay características de las transacciones que afectan a la velocidad con la que se retransmiten por la red. Por ejemplo, si se tienen en cuenta las comisiones (fees) y la prioridad.
i) Realizar mediciones del tiempo que pasa desde que se recibe una transacción hasta que esta entre en un bloque, comprobando si hay características de las transacciones que influyen en este tiempo.

1.4 Metodología

En este apartado se describe la metodología que se seguirá durante el desarrollo del TFM.
Debido a que la documentación relacionada con la red P2P de Bitcoin es bastante precaria, en primer lugar será necesario realizar un estudio previo de su funcionamiento. Este proyecto se realizará estudiando el comportamiento y los datos obtenidos a partir del cliente de Bitcoin y otras herramientas como **BTC Sniffer, Node.py** y **Bitcoinjs**
Adicionalmente se estudiará la forma en la que se realiza la minería de bitcoins, aunque ese no sea el objetivo principal del TFM.

Por otra parte sería conveniente poder contar con una pequeña cantidad de bitcoins y al menos dos wallets o billeteras para poder iniciar transacciones entre una y otra, y así medir el tiempo de respuesta del sistema. Esta sería una forma de comprobar los tiempos obtenidos en las mediciones anteriores ya que seríamos el origen y el destino de la transacción. Las dos wallets podrían estar situadas en el mismo equipo usando una o dos máquinas virtuales, en equipos diferentes o incluso en redes diferentes. Mi idea inicial es tener una wallet en un equipo de sobremesa y otra en un portátil y así tener mayor libertad para realizar pruebas desde otra red (otra conexión ADSL en mi caso).

También sería conveniente estudiar la posibilidad de conectarnos a Bitcoin a través de la red Tor para comprobar si esto influye en los resultados obtenidos, y que tipo de ventajas puede ofrecer el uso de este tipo de red.

También mencionar que se intentará analizar cualquier nueva cuestión que pueda surgir (nuevo software, características no conocidas de la red, …) durante el desarrollo de este TFM y que por supuesto esté relacionada con los objetivos del mismo.

Por último, comentar que para los propósitos de este TFM ha sido necesario modificar el funcionamiento de algunas de las utilidades disponibles para interactuar con la red bitcoin y desarrollar una nueva.

1.5 Plan de trabajo.

A continuación se detallan las tareas planteadas al inicio del TFM para alcanzar los objetivos descritos anteriormente:

1) En primer lugar, reunir información sobre el sistema Bitcoin en general y sobre el funcionamiento de la red P2P en particular.
2) Instalar el cliente y el resto del software que se utilizará durante la realización del TFM en al menos dos equipos para realizar las pruebas y mediciones. La idea es ver si se obtienen los mismos resultados desde equipos conectados desde ubicaciones distintas. Para estas pruebas utilizaré un equipo de sobremesa y un portátil desde el que poder duplicar las pruebas partiendo de otra IP.
3) Estudio de los nodos de la red. Mediante las herramientas existentes trataremos de averiguar cuantos nodos están conectados y a continuación conectarnos a una cantidad significativa de los nodos de la red, registrando las transacciones y bloques que nos llegan de estos nodos. Para realizar todo este proceso puede que sea necesario realizar alguna pequeña modificación sobre las herramientas mencionadas.
4) Comprobar el origen de las transacciones y de los nuevos bloques que se producen en la red, estudiando si hay relación entre ambos.
5) Analizar los datos obtenidos y decidir si es necesario realizar pruebas adicionales o utilizar algún otro tipo de software. A partir de los datos obtenidos se puede plantear alguna tarea nueva o modificar alguna de las existentes.

6) Realizar nuevas pruebas, esta vez analizando si hay características y/o parámetros de las transacciones que afectan a la velocidad por la que estas se retransmiten por la red.
7) Medir los tiempos que se producen entre la generación de dos bloques consecutivos y si ha ido evolucionando a lo largo del tiempo.
8) Analizar los nuevos resultados, proponiendo nuevas pruebas en función de los resultados obtenidos.
9) Estudiar el funcionamiento básico de la red Tor, intentar medir el número de nodos que se conectan a la red P2P de Bitcoin usando Tor y ver si su número o su forma de acceder influye sobre los resultados obtenidos o hay que hacer alguna modificación o recalibración.
10) Estudiar cualquier cuestión adicional que haya podido surgir durante el desarrollo de las tareas anteriores.
11) Analizar estos datos, junto con los anteriores creando un informe con toda la información obtenida.

1.6 Descripción del resto de la memoria

En resto de la memoria está estructurado como se describe en los siguientes párrafos.

En primer lugar, el capitulo dos contiene una breve introducción del funcionamiento de la red de Bitcoin.

El tercer capítulo contiene una relación del software existente en la red Bitcoin (en los anexos se ha añadido más información sobre este software y su instalación)

En el capítulo 4, se describe el funcionamiento de la aplicación desarrollada, explicándose la estructura de la base de datos donde se guarda la información, las distintas opciones disponibles en la aplicación y como funcionan. En los anexos hay información adicional sobre los requisitos de instalación y configuración de la aplicación.

En el capítulo 5, se describen las pruebas y resultados obtenidos para aumentar el número de nodos a los que estamos directamente conectados dentro de la red P2P de Bitcoin.

En el capítulo 6, *Analizando los datos obtenidos*, se han incluido gráficos de los resultados, mediciones de tiempos, comparativas entre los datos obtenidos y los que aparecen en la cadena de bloques, ...

Una vez analizados los resultados, se plantean las conclusiones a las que se han llegado junto con algunas sugerencias de como ampliar el trabajo.

Por último se ha incluido una relación de la bibliografía utilizada.

2 La red Bitcoin

Bitcoin es una moneda electrónica descentralizada concebida en 2009 por *Satoshi Nakamoto*. El nombre se aplica también al software libre diseñado por el mismo autor para su gestión y a la red P2P de la que consta. Este TFM, como ya se ha mencionado, está centrado en el estudio de esta red P2P y en el flujo de información que circula a través de ella.

El funcionamiento de Bitcoin está basado en una base de datos distribuida en varios nodos de su red P2P, a la que se llama *blockchain* o cadena de bloques. En esta red los diferentes nodos se conectan directamente entre ellos, sin la necesidad de un servidor central, para comunicarse las transacciones y los nuevos bloques del sistema. De esta forma se evita que Bitcoin dependa de ningún emisor central, tal y como ocurre con la mayoría de monedas.

Las transacciones se registran utilizando esta red P2P y la seguridad del sistema se proporciona a través del uso de la criptografía, garantizándose que los bitcoins solo puedan ser gastados por su dueño, y nunca más de una vez. La siguiente figura ilustra el funcionamiento básico de Bitcoin.

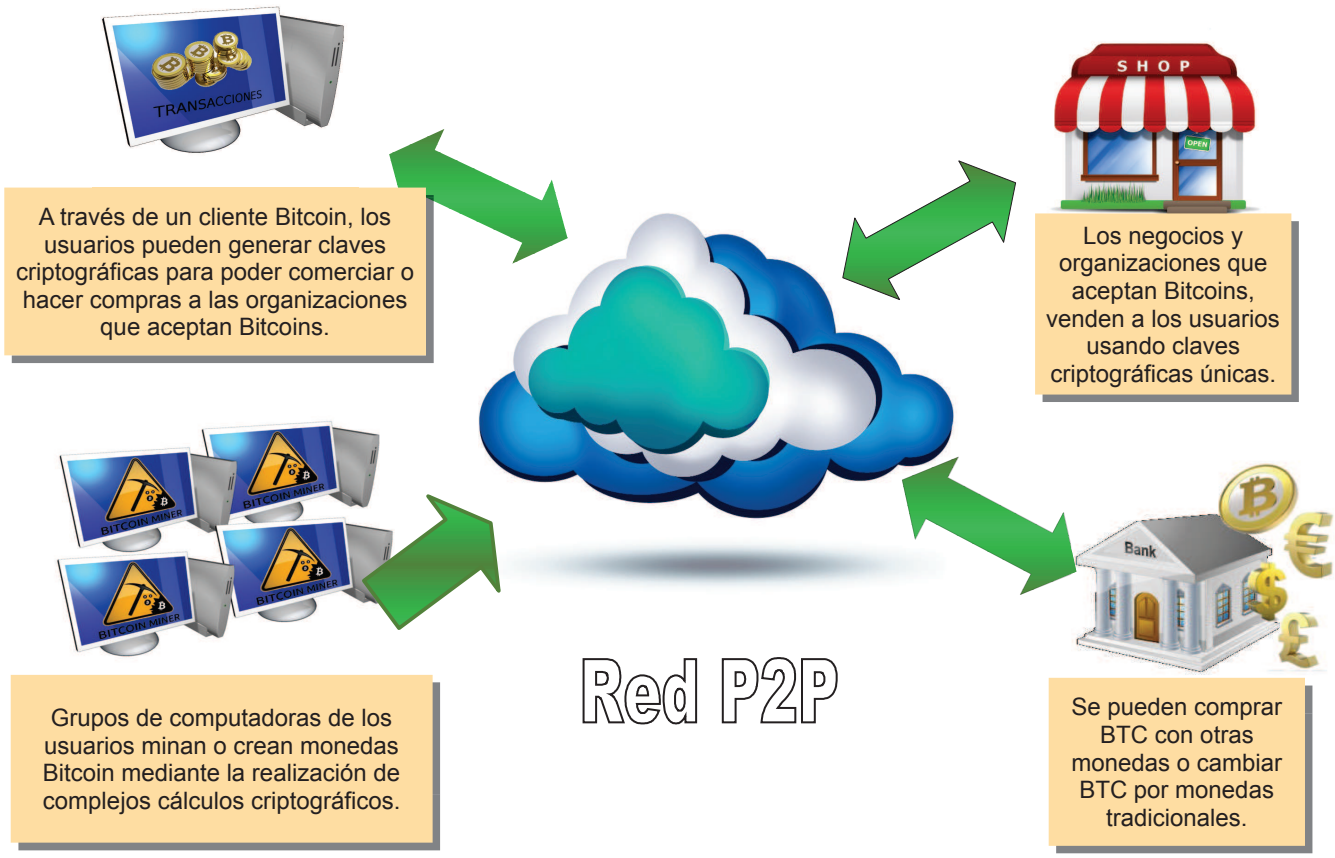


Figura 1.- Esquema del funcionamiento básico de la red Bitcoin.

El diseño de Bitcoin permite poseer y transferir valor de forma potencialmente anónima. Las monedas pueden ser guardadas en cualquier ordenador en la forma de un archivo "de bolsillo", o con un tercero que ofrezca el servicio de almacenar ese archivo. En cualquiera de los casos, los bitcoins pueden ser enviados por medio de Internet a cualquiera que tenga una "dirección Bitcoin". La estructura P2P de Bitcoin y la falta de administración central hace imposible para cualquier autoridad, gubernamental o de otra tipo, la manipulación del valor de los bitcoins o la creación de inflación produciendo más de ellos.

En cuanto a las características generales de la red, Bitcoin usa una red de difusión simple para propagar las transacciones y los nuevos bloques. Todas las comunicaciones se realizan a través de TCP en el puerto por defecto 8333. A partir de la versión 0.7 de Bitcoind / Bitcoin-Qt se soporta el uso de IPv6.

Para descubrir nodos en esta red, los clientes (la versión actual del cliente es la 0.8) pueden usar diferentes opciones: conectarse a IRC, hacer peticiones DNS, intercambiar direcciones con otros nodos, usar las direcciones proporcionadas por proveedores a través de archivos, …

En cuanto a la minería de datos, que es donde se generan los nuevos bitcoins, el constante aumento de la velocidad para obtener nuevos hash con la construcción de sistemas especializados en la minería puede llevar a aumentar la dificultad de los cálculos a realizar, lo cual es una opción ya prevista en el diseño del sistema Bitcoin.

Por último, recordar que este no es un sistema perfecto, y aún cuenta con errores y problemas. Por ejemplo, el 12 de marzo de 2013 la cadena de bloques se dividió en dos, entre los nodos que trabajaban con la versión 0.7 y los que lo hacían con la 0.8 debido a una diferencia en la forma de trabajar con la base de datos de la cadena de bloques. El error se solvento en unas pocas horas sin provocar perdidas, solo retrasos en las transacciones.

3 Software existente de Bitcoin

Una de las primeras tareas realizadas durante este TFM fue la instalación y evaluación del software existente para interactuar con la red de Bitcoin. En los siguientes apartados se describe el funcionamiento básico de este software. Para más información sobre la instalación y las características de este software consultar el *Anexo II: Descripción del software de Bitcoin.*

3.1 Bitcoin-Qt y bitcoind

Bitcoind es el programa básico que proporciona toda la funcionalidad de Bitcoin a través de la línea de comandos sin interfaz gráfica de usuario, e incorpora también una herramienta de pruebas para el mismo daemon. También proporciona una interfaz JSON-RPC, que permite controlarlo tanto en modo local como en modo remoto. Esto lo hace útil para la integración con otras herramientas de software o en sistemas de pago más amplios.

Bitcoin-Qt proporciona una interfaz amigable a través de una ventana principal usando la biblioteca de interfaz gráfica Qt4 con licencia MT. Ambos programas se distribuyen juntos desde la versión 0.5. Entre las principales características que incorpora Bitcoin-Qt se encuentran:

- Es compatible con Linux (tanto GNOME como KDE), Mac OS X y Windows y está disponible en varios idiomas.
- Interfaz amigable con pestañas.
- Solicita confirmación antes del envío de bitcoins.
- Lista de transacciones más detallada con iconos de estado y filtros en tiempo real

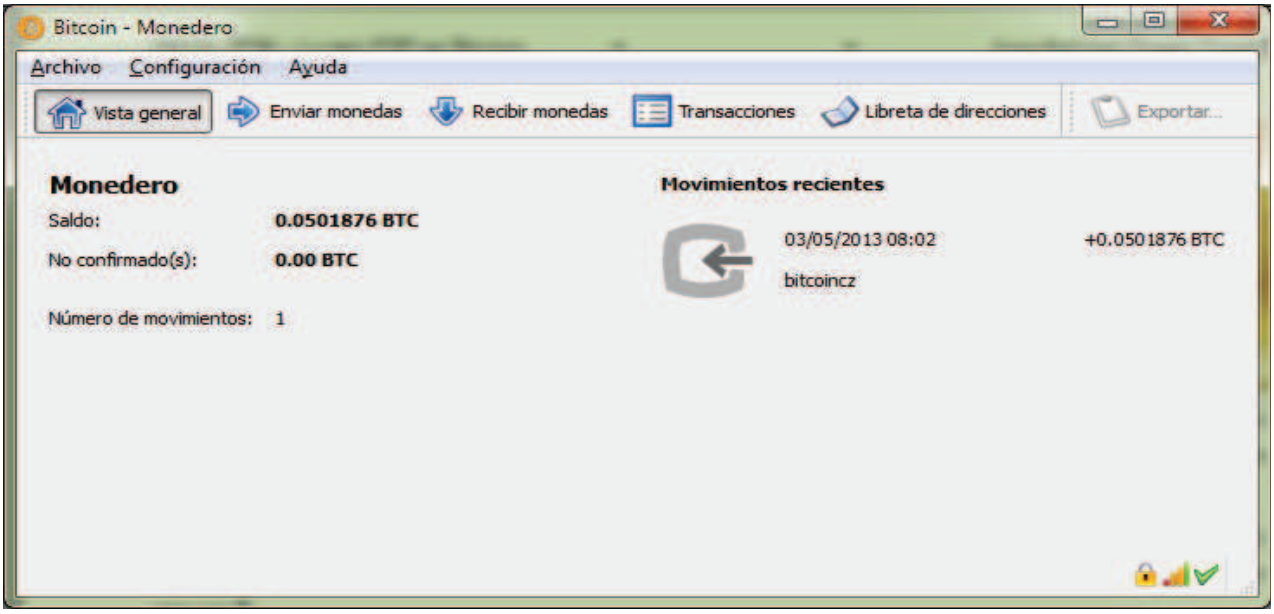


Figura 2.- Aspecto de la interfaz de Bitcoin-Qt

3.2 BITCOIN JS

BitcoinJS es un conjunto de bibliotecas de código abierto diseñado para ayudar a realizar de forma rápida proyectos personalizados de Bitcoin.

Para interaccionar con la red de Bitcoin se cuenta con **bitcoinjs-server.** Está es una implementación de un nodo Bitcoin en Node.js. Está concebido como alternativa al estándar bitcoind que se entrega con el cliente original.

A diferencia del cliente original, que contiene el nodo, la cartera, la interfaz gráfica y la minera, esta biblioteca solo contiene una versión altamente optimizada del nodo, es decir, la parte P2P de Bitcoin.

Su principal uso es funcionar como un componente de servidor para dar a los clientes más ligeros acceso en tiempo real a los datos de la cadena de bloques. Un ejemplo de estos clientes ligeros sería **bitcoinjs-gui**, que consiste en una implementación de un cliente bitcoin sobre HTML5 y JavaScript.

3.3 BITCOIN SNIFFER

Bitcoin Sniffer es un script basado en Python que se conecta a un nodo de la red bitcoin y se dispara cuando recibe alguno de los siguientes eventos:

- **new_block_event(block)**: Se activa cuando se encuentra un bloque.
- **new_tx_event(tx):** Se produce cuando se encuentra una transacción.

Figura 3.- Salida de Bitcoin Sniffer sin modificar. En la versión modificada en este TFM los datos se almacenan en una base de datos de MySQL.

Este sniffer facilita una forma fácil de escuchar eventos sin tener que modificar el cliente bitcoind.

En este TFM he modificado el código de este script para que acepte como parámetro de entrada una dirección IP de un nodo y además en vez de mostrar los resultados por pantalla los almacene en una base de datos.

3.4 Pynode

Este script en Python es un nodo cliente para la red Bitcoin. Su única función es escuchar las nuevas transacciones y bloques, lleva a acabo una comprobación de los datos y los almacena en una base de datos.

3.5 Más software de Bitcoin examinado.

Además de las aplicaciones descritas en los puntos anteriores, también se puede han examinado las siguientes aplicaciones por considerarlas interesantes, aunque no tan directamente relacionadas con los objetivos del TFM.

3.5.1 Bitcoin Wallet

Esta es una versión ligera del la billetera para Android, que se puede instalar en smartphones y tablets. Esta versión no descarga la cadena de bloques entera, lo cual no sería eficiente dado su tamaño de varios gigabytes. Pero un dispositivo con esta aplicación conectada funciona como un nodo más de la red bicoin, al que nos podemos conectar, por ejemplo usando el comando de bitcoinjs: *bitcoinjs run –connect dir_IP_dipositivo_movil:8333.* La conexión se logra tanto si el dispositivo está conectado a la red local a través de wifi como si lo está a través de la red 3G.

3.5.2 Gui Miner

GUIMiner es una interfaz gráfica para la minería de Bitcoins que proporciona una forma cómoda para manejar los “mineros” de Bitcoin. Es compatible tanto con GPU's ATI y NVIDIA, así como con la minería basada en CPU. Se puede usar tanto para la minería en solitario como para la minería en grupo, e integra una lista de grupos (o pools) de mineros para escoger.

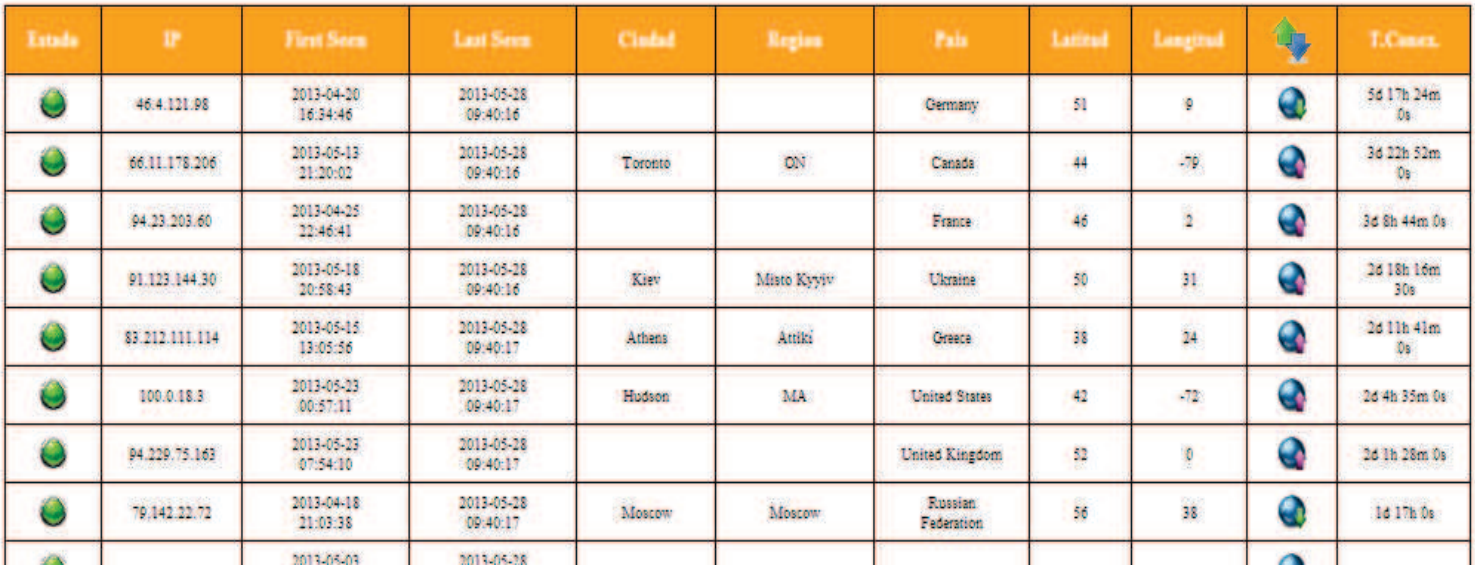
Las webs de algunos de estos grupos de minería muestran una gran cantidad de datos y estadísticas sobre los bloques de Bitcoin encontrados.

3.5.3 Blockchain

Este sitio web (http://blockchain.info/es/) ofrece una gran cantidad de datos sobre casi cualquier aspecto de Bitcoin. Todos estos datos nos servirán para contrastar y/o verificar los datos que se vayan obteniendo en el TFM.

4 Software desarrollado

Además de todo el software descrito en el apartado anterior, he ido desarrollando una aplicación, a la que he llamado BTCdoNET[1], para recoger y analizar datos sobre la red Bitcoin. Empezó simplemente como una aplicación en PHP que registraba información sobre los nodos de la red Bitcoin a los que me iba conectando, almacenándola en una base de datos y mostrando los resultados a través de una página web tal y como se puede apreciar en la siguiente imagen.



Estado	IP	First Seen	Last Seen	Ciudad	Region	Pais	Latitud	Longitud		T.Conex.
	46.4.121.98	2013-04-20 16:34:46	2013-05-28 09:40:16			Germany	51	9		5d 17h 24m 0s
	66.11.178.206	2013-05-13 21:20:02	2013-05-28 09:40:16	Toronto	ON	Canada	44	-79		3d 22h 52m 0s
	94.23.203.60	2013-04-25 22:46:41	2013-05-28 09:40:16			France	46	2		3d 8h 44m 0s
	91.123.144.30	2013-05-18 20:58:43	2013-05-28 09:40:16	Kiev	Misto Kyyiv	Ukraine	50	31		2d 18h 16m 30s
	83.212.111.114	2013-05-15 13:05:56	2013-05-28 09:40:17	Athens	Attiki	Greece	38	24		2d 11h 41m 0s
	100.0.18.3	2013-05-23 00:57:11	2013-05-28 09:40:17	Hudson	MA	United States	42	-72		2d 4h 35m 0s
	94.229.75.163	2013-05-23 07:54:10	2013-05-28 09:40:17			United Kingdom	52	0		2d 1h 28m 0s
	79.142.22.72	2013-04-18 21:03:38	2013-05-28 09:40:17	Moscow	Moscow	Russian Federation	56	38		1d 17h 0s

Figura 4.- Ejemplo de la lista de nodos de la aplicación desarrollada para el presente TFM.

A esta funcionalidad básica se le han ido añadiendo más características y funcionalidades, las cuales se describen en los siguientes apartados.[2]

4.1 Características generales de BTCdoNET.

La aplicación esa dividida en dos secciones:

- **BTCdoNET-Recorder,** se encarga de guardar periódicamente información básica sobre los nodos a los que nos estamos conectando, como dirección IP, tiempo de conexión, geolocalización del nodo, cuando se realizó la primera y la última conexión con un nodo,...
  También se guardan estadísticas como el número medio, mínimo y máximo de nodos conectados simultáneamente durante una hora, así como a cuantos diferentes nodos nos hemos conectado.
  Por pantalla muestra un par de tablas con estadísticas de las conexiones y una lista de nodos conectados a la red Bitcoin.
- **BTCdoNET** maneja toda la información guardada y además puede recoger mucha más información. Esta aplicación está organizada en las siguientes cuatro opciones principales:

1 Se trata de un juego de palabras (muy malo, lo reconozco) entre mi apellido y la abreviatura de network.
2 En la dirección *juadodo.no-ip.org/bitcoin* hay una versión de prueba on-line de la aplicación.

- **Listas**: Permite listar los nodos que están activos, todos los encontrados o los más estables. Además permite realizar varias comprobaciones del estado de los nodos descubiertos, como por ejemplo si están en funcionamiento o si tienen abierto el puerto 8333.
- **Estadísticas**: permite ver estadísticas según el país de origen de los nodos, tanto para todos los nodos como solo para los conectados actualmente. También se pueden ver estadísticas de los nodos según el tiempo que han permanecido conectados.
- **Añadir conexiones.** Uno de los objetivos del TFM es estudiar el número de nodos conectados simultáneamente a nuestro equipo. En esta opción se incluyen diversas herramientas para modificar la configuración del cliente de bitcoin para aumentar el número de conexiones.
- **Bitcoin Sniffer:** usando una versión modificada de Bitcoin Sniffer, podemos capturar las transacciones y bloques registrados por un nodo remoto y guardar la información junto con un timestamp del momento en que se recibió.

  Una de la opciones disponibles permite abrir varias instancias al mismo tiempo del Bitcoin Sniffer modificado, cada una conectada a un nodo distinto. Las transacciones y bloques se van registrando en la base de datos junto con el momento en que cada uno de los nodos los va recibiendo. Toda esta información se va mostrando en pantalla al mismo tiempo que se van guardando en la base de datos.

  La información recogida desde Bitcoin Sniffer se puede revisar en cualquier momento.

A continuación se detalla el funcionamiento de la aplicación, empezando por la parte de recogida de información sobre los nodos (BTCdoNET-RECORDER) y la parte del análisis de datos (BTCdoNET).

Figura 5.- Aspecto general del formato en que se muestran los datos en BTCdoNET-Recorder

4.2 Estructuras de datos utilizadas.

Toda la información manejada por la aplicación se guarda en una base de datos de MySQL llamada **bitcoin**. Dentro de esta base de datos la información se distribuye en las tablas que se muestran en la siguiente imagen. Junto al esquema de las tablas se ha incluido un breve comentario sobre la información que guardan. El reloj que aparece junto a algunas de las tablas indica que se van actualizando regularmente cada cierto intervalo de tiempo.

En los siguientes puntos se explica en detalle toda la estructura de datos y el significado de cada uno de los campos de las tablas que forman parte de esta base de datos.

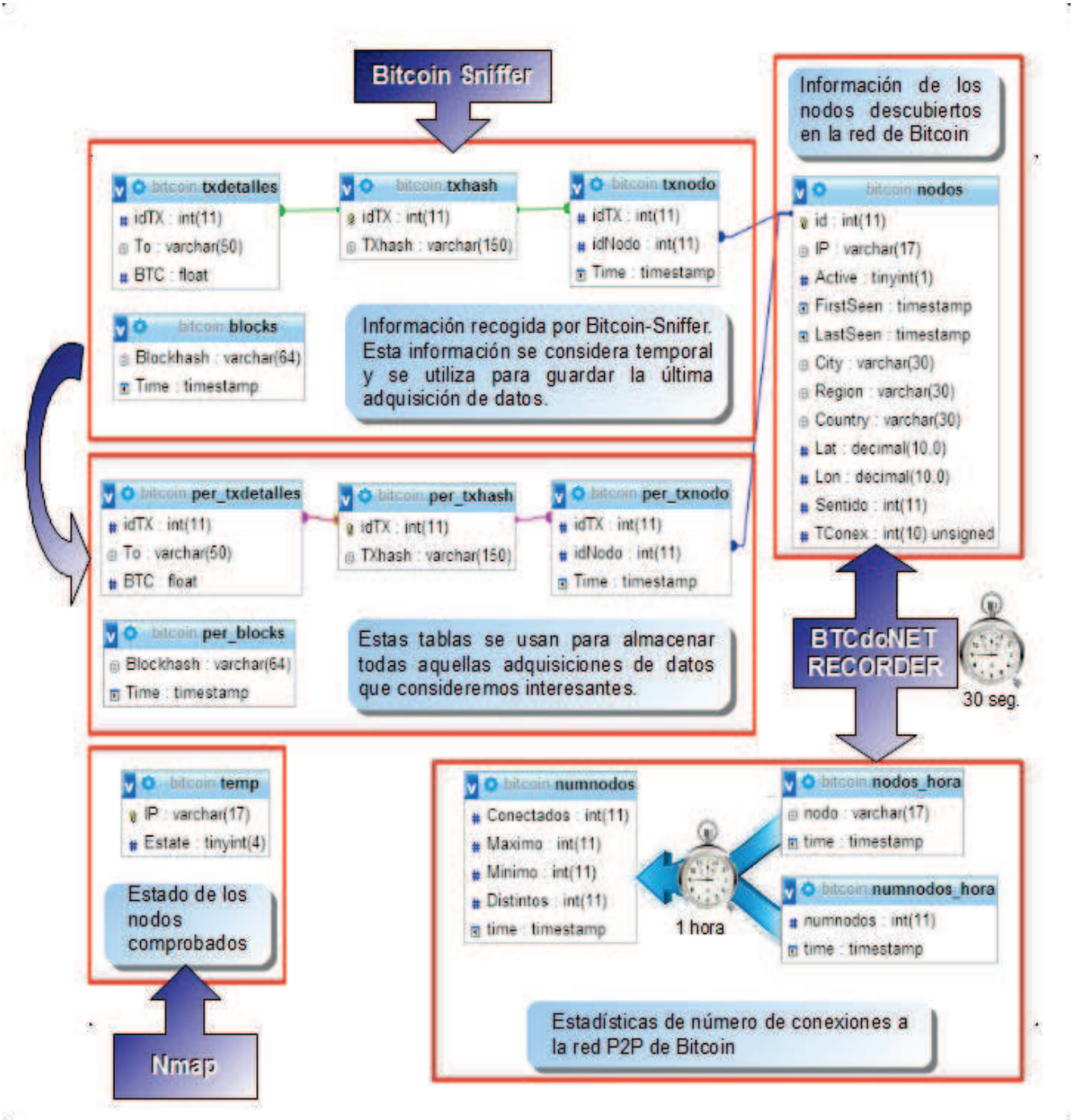


Figura 6.- Estructura de la base de datos bitcoin donde se almacena la información recogida por la aplicación BTCdoNET.

4.3 BTCDONET-RECORDER. RECOGIENDO DATOS DE LOS NODOS.

Esta es la primera parte de la aplicación que desarrollé y es la que se encarga de guardar los datos de los nodos en la base de datos. Solo debería estar abierta una instancia de este código por cada equipo en el que se quieran registrar las aplicaciones.

La aplicación genera una página web mediante PHP que se va actualizando cada 30 segundos, aunque se puede modificar este tiempo variando el parámetro *$refresco*. Su funcionamiento básico consiste en ejecutar el comando ***netstat***[3] para ver las conexiones que hay realizadas en nuestro equipo. En el código PHP se analiza el resultado de este comando para extraer aquellas direcciones asociadas al puerto 8333 (puerto por defecto de Bitcoin). En caso de que se este utilizando otro puerto hay que modificar la variable *$puerto.*

Además se comprueba que la dirección IP asociada al puerto 8333 no sea la nuestra (se configura a través de la variable *$miIP*) o 127.0.0.1. Si se diera este caso se almacenaría la otra dirección de la conexión.

En la siguiente imagen se muestra un ejemplo de la salida del comando netstat. Se han coloreado en verde las conexiones al puerto 8333 de equipos remotos, y en azul las conexiones de equipos remotos a nuestro puerto 8333. Se usa la opción -n por dos motivos: no nos interesa que se resuelvan los nombres de las direcciones, ya que se trabaja sobre direcciones IP, y además de esta forma la ejecución del comando es muchísimo más rápida.

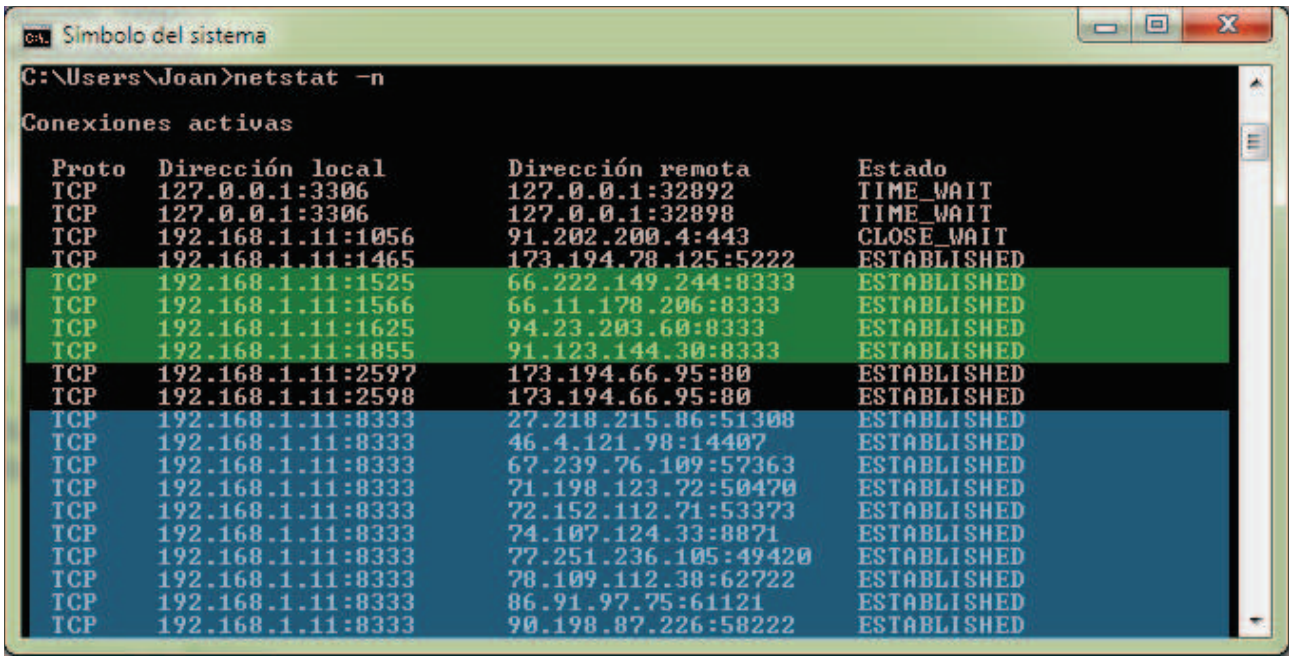


Figura 7.- Ejemplo de la ejecución de netstat en un equipo con un cliente de Bitcoin activo.

3 **Netstat** (network statistics) es una herramienta de línea de comandos que muestra un listado de las conexiones activas de una computadora, tanto entrantes como salientes. Existen versiones de este comando en varios sistemas como Unix, GNU/Linux, Mac OS X, Windows y BeOS.
La información obtenida del comando incluye el protocolo en uso, las tablas de ruteo, las estadísticas de las interfaces y el estado de la conexión

Antes de añadir la dirección IP de un nodo a la base de datos de MySQL se comprueba si ya está registrada. Si es un nodo que todavía no se ha encontrado se guarda en la base de datos ***bitcoin*** en la tabla ***nodos,*** si el nodo ya estaba registrado sólo se actualizan sus datos. Los campos que se tienen en la tabla ***nodos*** son los siguientes:

- **id**: clave primaria de la tabla. Es un dato de tipo entero con el atributo AUTO_INCREMENT
- **IP**: Dirección IP del nodo
- **Active**: Indica si la conexión con el nodo esta activa o no. Este campo se pone a false para todos los nodos cada 30 segundos. Aquellos nodos que forman parte de una conexión activa hacia o desde el puerto 8333 se vuelven a cambiar a true.
- **FirstSeen**: TimeStamp de la primera vez que se vio el nodo.
- **LastSeen**: TimeStamp de la última vez que se vio el nodo. Este valor se actualiza cada vez que se encuentra una conexión activa con el nodo.
- **City**, **Region**, **Country**, **Lat** y **Lon**: valores de geolocalización de la IP del nodo. Se obtienen a través de una consulta a la dirección http://www.geoplugin.net/php.gp
- **Sentido**: El significado de este campo es el que se indica en la siguiente tabla:

Icono	Valor	Descripción
	1	La conexión con el nodo está activa y nuestro equipo está actuando como cliente.
	2	La conexión está activa y nuestro nodo está actuando como servidor.
	-1	Indica conexión no activa y que en la última conexión realizada con ese nodo estábamos actuando como cliente.
	-2	Indica conexión no activa y que en la última conexión realizada con ese nodo estábamos actuando como servidor.

- **Tconex**: Cada vez que se encuentra un nodo conectado se suman 30 segundos al tiempo de conexión, excepto cuando se encuentra por primera vez. En realidad lo que se suma es el valor del parámetro $refresco, así que se si asigna un valor distinto, Tconex también cambiará en función de este valor.

Estado	IP	First Seen	Last Seen	Ciudad	Region	Pais	Latitud	Longitud		T.Conex.
	94.23.203.60	2013-04-25 22:46:41	2013-06-13 00:10:45			France	46	2		39d 16h 53m 0s
	66.11.178.206	2013-05-13 21:20:02	2013-06-13 00:10:44	Toronto	ON	Canada	44	-79		22d 18h 28m 0s
	91.123.144.30	2013-05-18 20:58:43	2013-06-13 00:10:45	Kiev	Misto Kyyiv	Ukraine	50	31		21d 8h 22m 0s

Figura 8.- Detalle de como se muestra toda esta información en la página web generada.

Esta información también se podría obtener ejecutando el comando *getpeerinfo* del propio bitcoind, pero de esta forma se evita que sea dependiente de la aplicación que se está ejecutando. De hecho detectará también las conexiones que realicemos a través de Bitcoin Sniffer.

Además de la información guardada en la tabla nodos, se van registrando cada 30 segundos en la tabla ***numnodos_hora*** el número de nodos conectados en ese momento junto con un timestamp. Igualmente cada 30 segundos, en la tabla **nodos_hora** se van guardando las direcciones IP de los distintos nodos a los que conectamos.

Un resumen de la información de estas tablas se guarda cada hora en la tabla **numnodos**. Esta tabla contiene la siguiente información calculada de las dos tablas anteriores:

- **Conectados**: Es la media aritmética del número de nodos conectados durante una hora.
- **Maximo**: El máximo número de nodos que se han conectado durante una hora
- **Minimo**: El mínimo número de nodos que se han constado durante esa hora
- **Distintos**: Los distintos nodos a los que nos hemos conectado. Esto puede dar una idea de si van variando mucho o poco.
- **Time**: Timestamp del momento en que se han realizado estos cálculos. Hay que tener en cuenta que los datos se referirán a la hora anterior de la indicada en el timestamp.

Toda esta información se muestra en la misma página web en una serie de tablas. En primer lugar se muestran datos con información de los nodos encontrados y conectados, además de estadísticas de las conexiones durante la última hora y las últimas 24 horas.

Resumen conexiones con nodos bitcoin Sun 9-Jun-2013 7:55:26			
Total nodos encontrados	7965	Total nodos conectados / conexiones:	47 / 47
Total conexiones entrantes (puerto local 8333):	8	Total conexiones salientes (puerto remoto 8333):	39
Última hora			
Media conexiones	51	Distintos nodos conectados	57
Máximo conectados	54	Mínimo conectados	46
Últimas 24 horas			
Media conexiones	37	Distintos nodos conectados (por hora)	43
Máximo conectados	52	Mínimo conectados	23

Figura 9.- Imagen de la tabla resumen de las conexiones actuales, de las de la última hora y las últimas 24 horas.

4.4 BTCdoNET. Opciones disponibles.

Esta parte de la aplicación no registra la información de los nodos en la base de datos, por lo que se pueden tener tantas instancias abiertas como se necesite sin adulterar los datos. Además incorpora una serie de utilidades para mostrar información, analizar el estado de los nodos conocidos y obtener datos sobre las transacciones realizadas en la red Bitcoin.

La página principal de este esta aplicación es similar a la anterior pero ahora se incluye un pequeño menú de opciones en la parte superior izquierda del navegador. Desde este menú se pueden elegir las diversas opciones de la aplicación. En primer lugar hay un pequeño icono de un circulo con puntos. Si van cambiando de color significa que la página se ira actualizando cada pocos segundos (30 seg. en la mayoría y 3 seg en las relacionadas con Bitcoin Sniffer). Si por el contrario esta completamente en gris, la actualización se ha detenido. De esta forma podemos examinar los datos tranquilamente al mismo tiempo que se recogen más.

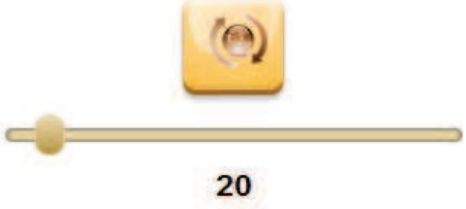
4.4.1 Listas

Dentro de la opción Listas tenemos los siguientes apartados:

- **Nodos activos.** Es la vista por defecto, muestra una lista de los nodos de la red Bitcoin actualmente conectados a nuestro equipo.
- **Todos los nodos.** Muestra la totalidad de los nodos encontrados.
- **Nodos más estables.** Busca los nodos a los que nos hemos conectado en algún momento durante los últimos 7 días ordenados de forma descendente según el tiempo de conexión.

  Esta lista de nodos más estables se utiliza en otras partes de la aplicación para generar listas de direcciones IP sobre las que realizar otras pruebas.

- **Comprobar nodos**. A partir de la lista de nodos estables, se comprueba algunos aspectos de la conectividad de estos nodos. Si aún no se ha ejecutado nunca la comprobación de nodos nos aparecerá solamente un botón y una barra deslizante (imagen derecha)



Con la barra deslizante seleccionamos el número de nodos a analizar de entre la lista de nodos más estables. Al pulsar el botón, se escanea el puerto 8333 de cada uno de los nodos objetivo mediante el comando **nmap**[4].

Los nodos se van escaneando uno a uno. Este es un proceso relativamente lento que puede tardar unos pocos minutos. La información se muestra en una tabla donde aparece la dirección IP de cada uno de los nodos escaneado con un color distinto según su estado:

- **Gris**: El nodo parece que esté desconectado.
- **Rojo**: El nodo está conectado, pero tiene el puerto 8333 cerrado. Puede que esté utilizando otro puerto para conectarse a la red bitcoin o simplemente no tiene activa ninguna aplicación para conectarse a la red. De cualquier forma, no nos podremos conectar a él.
- **Naranja**: El nodo está activo, pero el puerto 8333 está filtrado. Hay algún router o cortafuegos filtrando las conexiones a ese puerto. Esto puede resultar en dificultades a la hora de conectarnos a estos nodos.
- **Verde**: El nodo está activo y además tiene el puerto 8333 abierto.

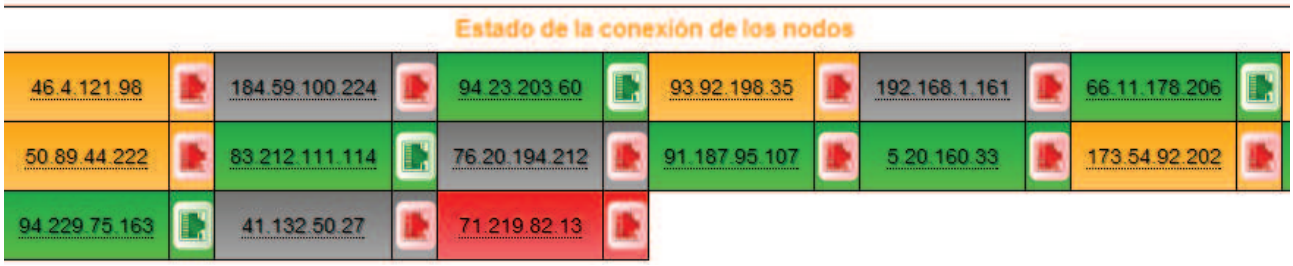


Figura 10.- Ejemplo de la comprobación del estado de los nodos.

Además, a la derecha de cada dirección IP aparece un icono de un conector RJ-45.

En caso de que el interior del conector este en verde, significa que estamos conectados a este nodo (o lo estábamos cuando se escaneo el nodo).

Si por el contrario aparece en rojo eso significa que no estamos conectados a este nodo.

Los mejores candidatos para “esnifar” sus transacciones son cualquiera de los nodos con fondo verde. Los mejores candidatos para añadir a la lista de conexiones son estos, pero si se añaden usando JSON, los mejores son los que tienen el fondo verde, pero el icono RJ-45 en rojo, es decir, los nodos con el puerto 8333 abierto a los que aún no nos hemos conectado.

4 **Nmap** es una aplicación de código abierto que sirve para efectuar rastreo de puertos escrita originalmente por *Gordon Lyon* (más conocido por su alias *Fyodor Vaskovich*). Se usa para evaluar la seguridad de sistemas informáticos, así como para descubrir servicios o servidores en una red informática.
Ha llegado a ser una de las herramientas imprescindibles para todo administrador de sistema, y es usado para pruebas de penetración y tareas de seguridad informática en general.
En Linux se puede instalar con *apt-get install nmap*. También existe una versión para Windows que se puede descargar desde: http://nmap.org/download.html.

Además en esta vista si nos situamos encima de una de las direcciones IP se nos mostrarán algunos de los datos de ese nodo en un menú emergente como el que se ve en la siguiente imagen:

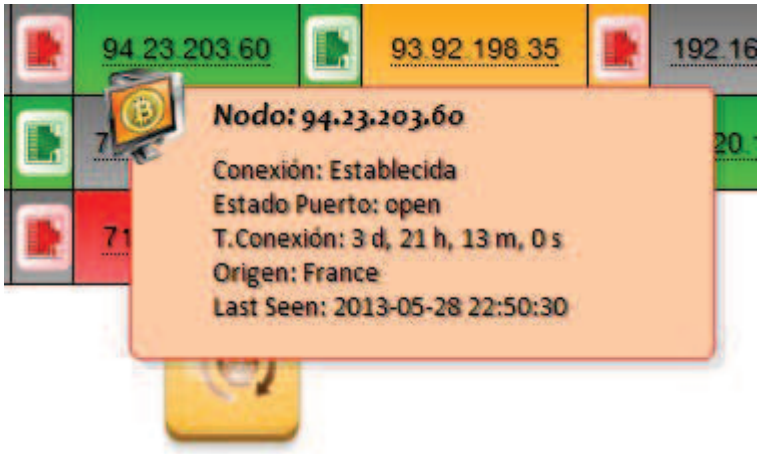


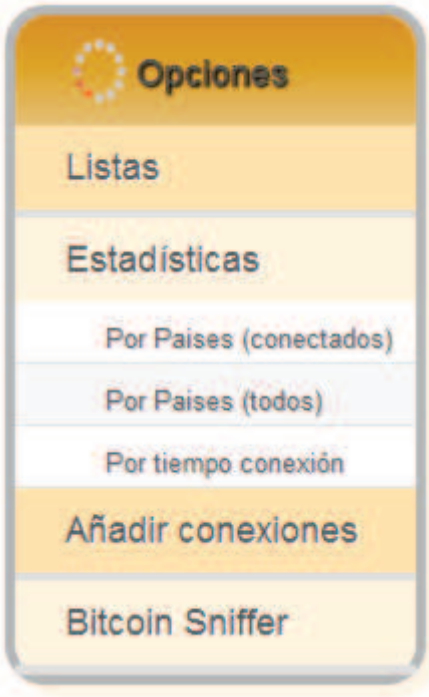
Figura 11.- Ejemplo de la información emergente que se muestra de los nodos comprobados.

La información obtenida al escanear el puerto 8333 de los distintos nodos más estables se almacena en un tabla de la base de datos bitcoin llamada **temp** que contiene los siguientes campos:

- **IP**: Dirección IP del nodo escaneado.
- **Estate**: Es un campo de tipo integer que indica el estado de la conexión:
  - 1 → down. El nodo no está activo
  - 2 → closed. Activo, pero con el puerto 8333 cerrado.
  - 3 → filtered. Activo, pero con el puerto 8333 filtrado.
  - 4 →open. Activo y con el puerto 8333 abierto.

4.4.2 Estadísticas

Es esta opción aparecen varias tablas con información de los datos distribuidos según diversos parámetros. Las opciones disponibles son:



- **Por países (conectados)**. Muestra el número de nodos conectados agrupados por el país.
- **Por países (todos)**. Muestra el total de nodos encontrados y el total de nodos conectados para cada uno de los países donde se ha encontrado al menos un nodo.
- **Por tiempo de conexión.** Muestra tres tablas con el número de nodos según el tiempo que han estado conectados desde que se empezaron a guardar datos, en los últimos 7 días y en las últimas 24 horas.

Ejemplo Listas por países (conectados)

Nodos conectados desde:	16 países

Distribución de nodos por países

País	Conectados
United States	13
Germany	4
Canada	4
United Kingdom	3
Netherlands	3
Australia	2
Ukraine	2
Argentina	1
Brazil	1
France	1
Austria	1
Hungary	1
Venezuela	1
Russian Federation	1
Finland	1
Greece	1
Total	40

Ejemplo Listas por países (todos)

Nodos encontrados en:	118 países
Nodos conectados desde:	16 países

Distribución de nodos por países

País	Encontrados	Conectados
United States	1984	13
China	873	0
United Kingdom	373	3
Germany	337	4
Russian Federation	294	1
Canada	285	4
Australia	194	2

Ejemplo Listas por tiempo de conexión

Resumen nodos por tiempo conexión

T. Conexión	Encontrados	Conectados
Menos de 1 min	1796	
Entre 1 y 30 min	1996	
Entre 30 min y 60 min	617	
Entre 1 y 5 horas	1403	
Entre 5 y 12 horas	554	
Entre 12 y 24 horas	239	
Entre 1 y 2 días	88	
Entre 2 y 7 días	36	
Más de una semana	6	

Resumen nodos por tiempo conexión con actividad en la última semana

T. Conexión	Encontrados	Conectados
Menos de 1 min	214	215
Entre 1 y 30 min	345	376
Entre 30 min y 60 min	132	140
Entre 1 y 5 horas	279	298
Entre 5 y 12 horas	96	143
Entre 12 y 24 horas	61	99
Entre 1 y 2 días	23	66
Entre 2 y 7 días	8	31

Resumen nodos por tiempo conexión con actividad en las últimas 24 horas

T. Conexión	Encontrados	Conectados
Menos de 1 min	70	70
Entre 1 y 30 min	32	42
Entre 30 min y 60 min	13	16
Entre 1 y 5 horas	19	29
Entre 5 y 12 horas	12	20
Entre 12 y 24 horas	10	23

Figura 12.- Ejemplos de las tablas mostradas en el apartado de Estadísticas. En la última versión se han incorporado gráficos en el apartado de países.

4.4.3 AÑADIR CONEXIONES.

El apartado de añadir conexiones permite añadir nodos a la lista de nodos de BitcoinQt utilizando varios métodos:

- **Crear archivo estables**: Busca en la base de datos el número de nodos y el máximo intervalo de tiempo desde que nos conectamos por última vez a un nodo ordenándolos por el tiempo de conexión.

  Con los nodos encontrados se crea un archivo de configuración llamado ***Bitcoin.conf***. En mi caso al usar Windows, este archivo se almacena en:

  "C:\Users\Joan\AppData\Roaming\Bitcoin\"

  pero se puede usar otra ruta cambiando la variable ***$directorio***.

El archivo creado tiene un formato como el que se muestra a continuación:

```
# bitcoin.conf configuration file.
# Created by BTCdoNET

rpcuser=juadodo
rpcpassword=4ng**********************************************************
rpcallowip=127.0.0.1
rpcport=8332
server=1
dbcache=50
dns=1
maxconnections=1024
logtimestamps=1

addnode=192.168.1.161
addnode=46.4.121.98
addnode=93.92.198.35
……
```

Figura 13.- Detalle de la pantalla de opciones para seleccionar los nodos estables.

- **Crear archivo comprob.**: Igual que el anterior, pero las direcciones IP que se añaden en este caso son las de la tabla temp con un valor Estate de 3 ó 4, es decir, aquellos nodos comprobados mediante el menú ***Listas*** → ***Comprobar nodos*** que estaban conectados y con el puerto 8333 abierto o filtrado.

  Tanto en este caso como en el anterior, para que los parámetros se carguen hay que reiniciar el programa Bitcoin-Qt o bitcoind, dependiendo de cual estemos usando.

- **JSON add estables** y **JSON add comprobados**: tienen la misma funcionalidad que las dos opciones anteriores, pero se llevan a cabo de distinta forma. En vez de crear un archivo de configuración los datos se envían directamente a la aplicación usando JSON. Esto evita tener que reiniciar el cliente de Bitcoin.

- **JSON compro. + try:** Realiza una comprobación similar a la que se realiza desde ***Listas*** → ***Comprobar nodos*** pero ahora además trata de conectarse a los nodos que tienen abierto el puerto 8333. Cuando se encuentra un nodo que cumple las condiciones anteriores tratamos de conectarnos al nodo (p.e. a 83.212.111.114) enviando por JSON[5] el comando:

  addnode 83.212.111.114 onetry

  En la interfaz de BTCdoNET aparecerá un mensaje adicional mientras se comprueba el estado de los nodos indicando que se ha enviado el comando addnode.

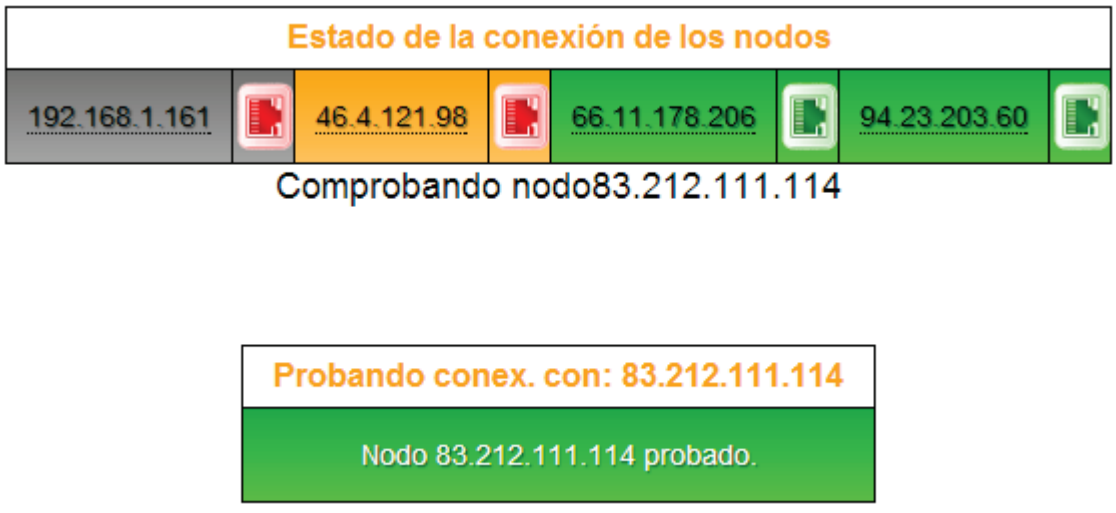


Figura 14.- Ejemplo de como se comprueban los nodos y se intenta la conexión con aquellos que reúnen las condiciones.

5 **JSON**, acrónimo de *JavaScript Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.
La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX.
Cada vez hay más soporte de JSON mediante el uso de paquetes escritos por terceras partes. La lista de lenguajes soportados incluye ActionScript, C, C++, C#, ColdFusion, Common Lisp, Delphi, E, Eiffel, Java, JavaScript, ML, Objective-C, Objective CAML, Perl, PHP, Python, Rebol, Ruby, Lua y Visual FoxPro.

- **JSON try:** En este caso no se comprueban los nodos, sino que se utilizan los datos de comprobaciones anteriores, los cuales están almacenados en la tabla ***temp*** de la base de datos ***bitcoin***.

Las pruebas y resultados obtenidos con todas estas opciones se describen en el capítulo 5.

4.4.4 Bitcoin Sniffer

Las opciones dentro del apartado de BitcoinSniffer hacen uso de una versión modificada de esta aplicación. Las modificaciones que se han realizado son las siguientes:

- **Entrada de parámetros**: Por defecto Bitcoin Sniffer aceptaba como único posible parámetro un archivo de configuración. Ahora los parámetros que acepta son dos:
  - Dirección IP del nodo del que se van a obtener las transacciones. Si no se indica ninguna se utiliza 127.0.0.1.
  - Puerto: Puerto al que nos intentaremos conectar. Por defecto se utiliza el 8333.
- **Salida de datos**: Los datos no se envían a la salida estándar, sino que se escriben directamente en la base de datos. En esta versión se registran las transacciones y los bloques.

Los datos de los últimos bloques registrados se almacenan en la tabla blocks de la base de datos bitcoin. La tabla **blocks** contiene los siguientes campos:

- **Blockhash**: hash calculado sobre ese bloque.
- **Time**: Timestamp del momento en que se recibe comunicación por parte del primer nodo de que lo registra.

Los datos de las últimas transacciones registradas se almacenan en las siguientes tablas de la base de datos ***bitcoin***:

- **txhash**: Esta tabla contiene los siguientes campos:
  - **idTX**: clave primaria de la tabla. Es un campo autonumérico que se asigna a cada una de las transacciones registradas.
  - **TXhash**: hash de las distintas transacciones que se registran.

Los datos de las transacciones solo se registran en la tabla txhash una vez, aunque se

estén obteniendo datos desde distintos nodos.

- **txdetalles**: En esta tabla se almacenan los detalles de las transacciones. Igual que con la tabla anterior, solo se registra una vez aunque la información la envíen varios nodos. Los campos que se almacenan son los siguientes:
  - **idTX**: clave ajena que identifica a que transacción pertenecen los datos.
  - **To**: a donde se envía la transacción.
  - **BTC**: cantidad de bitcoins que aparece en la transacción.
- **Txnodo**: Cada vez que se obtienen los datos de una transacción desde uno de los nodos examinados se registran los siguientes: campos:
  - **idTX**: clave ajena que identifica a que transacción pertenecen los datos.
  - **IdNodo**: clave ajena que identifica al nodo que nos ha informado de la transacción.
  - **Time**: timestamp del momento en que un determinado nodo nos ha enviado los datos de una transacción.

El funcionamiento de las distintas opciones disponibles en este apartado es el siguiente:

- **Desde un solo nodo:** De esta forma se inicia una única instancia de Bitcoin Sniffer. En primer lugar nos aparece un pequeño menú donde podemos seleccionar distintas direcciones IP de la lista de nodos comprobados que tienen el puerto 8333 abierto.

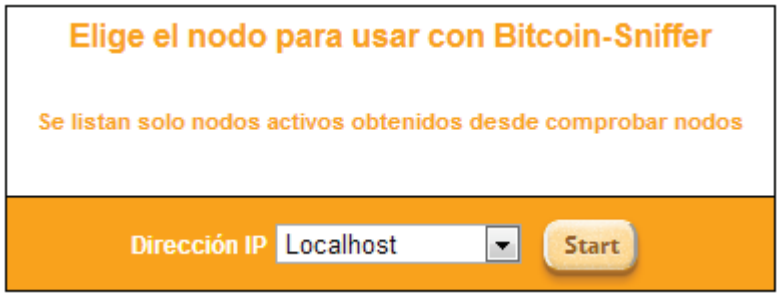


Figura 15.- Opciones para seleccionar un nodo para usar con Bitcoin Sniffer.

Una vez seleccionado el nodo, al pulsar el botón Start se inicia el programa de Bitcoin Sniffer en segundo plano que va guardando la información de bloques y transacciones en la base de datos. Al mismo tiempo se van leyendo los datos desde la aplicación en PHP y se van mostrando en pantalla tal y como se muestra en la siguiente imagen. La lista de datos se actualiza cada 3 segundos, pero se puede parar en cualquier momento con el botón para examinar los datos obtenidos mientras se continua con la adquisición de más bloques y transacciones.

Últimos 5 bloques encontrados

	Block hash	Time
	0000000000000081fe2cbf020aece7e63e0d1ebf332c26bf25340aefa5c70695	2013-06-07 14:44:51
	0000000000000818a53960017039dd2e429749e1955cf91f96c0ff313ebc32a	2013-06-07 14:32:26
	00000000000000fd47650ad7ef2fb8f205ba975a357b20eef7ce436aaf889f6c	2013-06-07 14:21:41
	00000000000000fb9c3ca666ea16a752e5b2c17e51b952fc88f555c5a0da799a	2013-06-07 14:03:31
	0000000000000295369f5efaae4eb0e5c53e71363eb5346807bee78ba21f54a	2013-06-07 13:58:30

Bitcoin-Sniffer funcionando

Dirección IP: 127.0.0.1

	TXhash	Time
	a4c20253a5ecb5e39be6dcfb58b0d8d47ff3f7ba32091025c0501cef6c1c74f4	2013-06-07 15:04:02
	4c0eb5946f9a87c0eeba8cd4bbbcdd6e0ad7a9ee1b68d8c794d29db9ea0d4f06	2013-06-07 15:04:02
	22d1a59b45547ca3cce5786d25e7ac0a9208f1859c8ad4d46801baac50d8ee97	2013-06-07 15:04:02
	909026071461e120c8b036aa611c5990278d8b1bed85e5d4690b471aaee3ca0f	2013-06-07 15:04:00
	f4f9ff25855f8d62190be11707c4c382610fd8f2e278af3a839398a4133a4bd1	2013-06-07 15:03:53
	bb1f1462c8ad65b6b03e231125d82a523f091c81c92a47b2662d3078c6586dc7	2013-06-07 15:03:53

Figura 16.- Detalle de los datos obtenidos por la versión modificada de Bitcoin Sniffer.

Si se quiere detener Bitcoin sniffer hay que pulsar sobre el botón Stop que aparece en la ventana emergente que se muestra en la *figura 17,* imagen izquierda. Se puede volver a iniciar Bitcoin Sniffer sobre el mismo nodo pulsando en el botón Start de esta misma ventana (*figura 17*, imagen derecha).

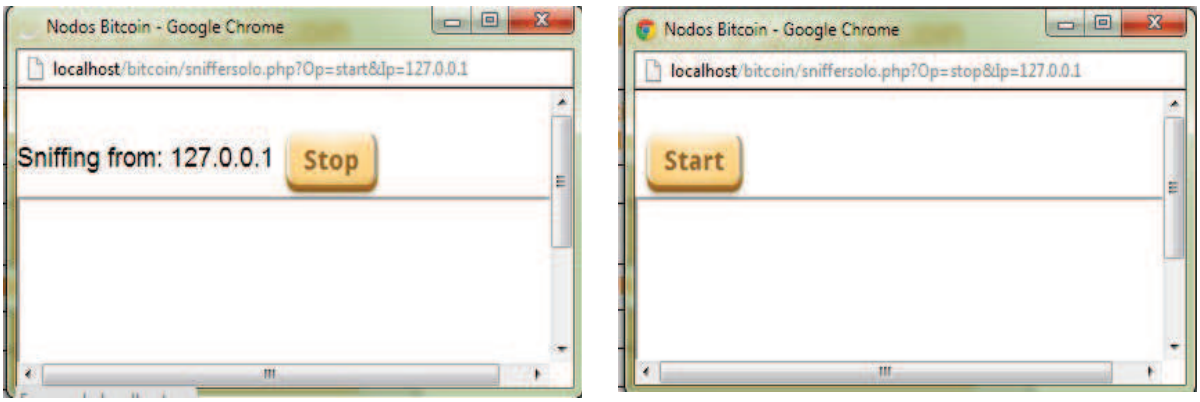


Figura 17.- Ventana para detener y reiniciar la adquisición de datos desde Bitcoin Sniffer.

Además, si se sitúa el ratón sobre el icono del ojo que aparece a la izquierda de cada una de los bloques o de las transacciones se muestran los detalles del bloque o de la transacción correspondiente.

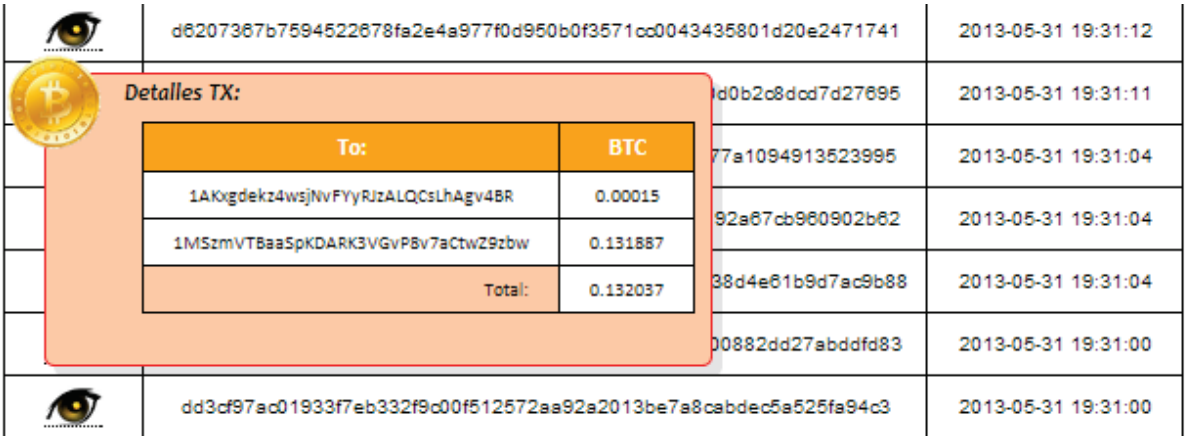


Figura 18.- Ejemplo de la ventana emergente con detalles de una de las transacciones.

- **Desde varios nodos**. Esta opción permite iniciar varias instancias de Bitcoin Sniffer. Igual que en la opción anterior se selecciona la dirección IP de la lista de nodos comprobados que tienen el puerto 8333 abierto. En este caso pulsando sobre el botón con el signo + podemos añadir más direcciones:

  Una vez seleccionados los nodos para iniciar la obtención de datos hay que pulsar sobre el botón *Start*. Esto iniciará una instancia de Bitcoin Sniffer por cada nodo seleccionado. Cada una de estas instancias irá procesando por separado la información sobre las transacciones que se van generando en la red de Bitcoin.

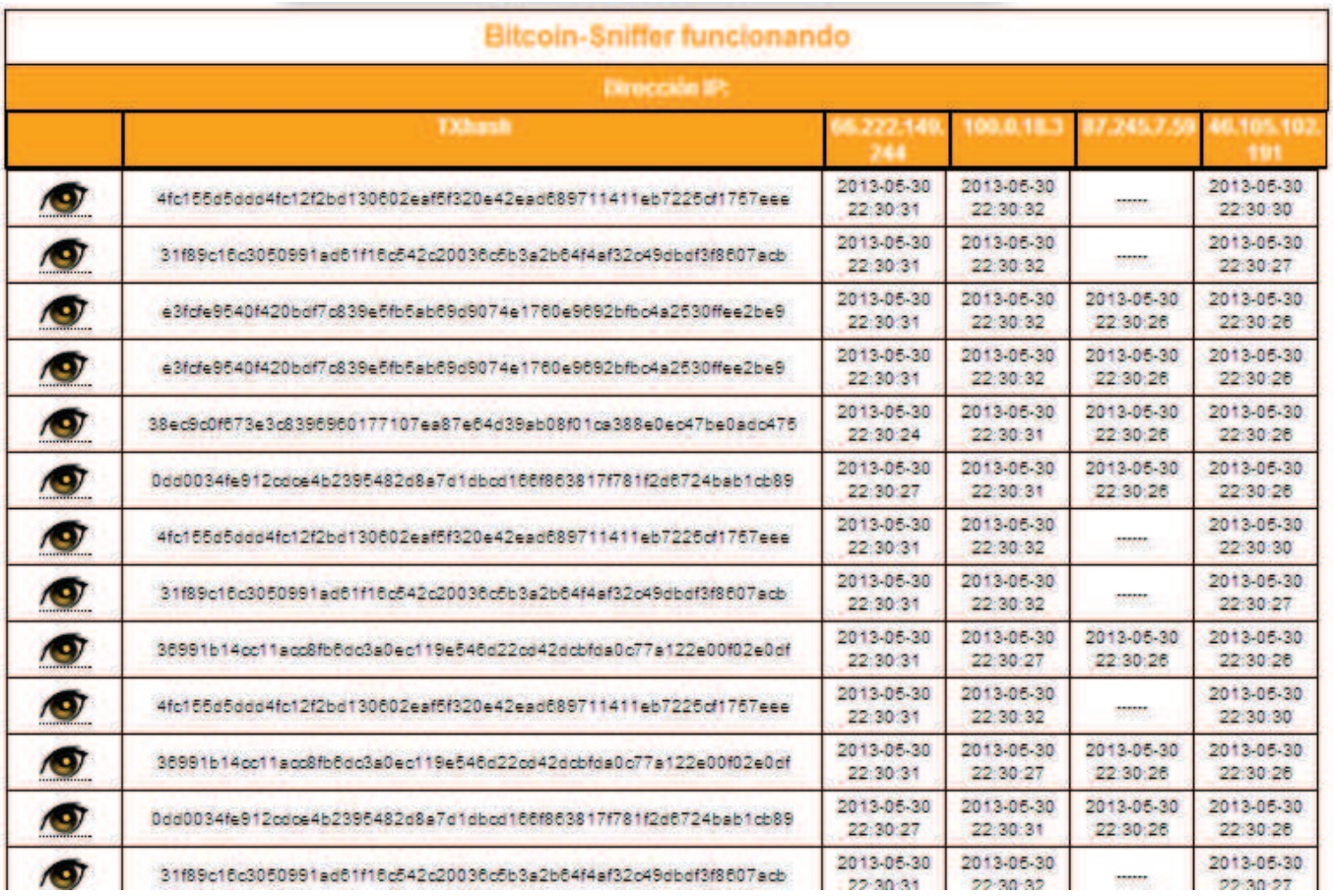
Figura 19.- Opciones para seleccionar varios nodos como objetivos de Bitcoin Sniffer.

Al recibir una transacción desde un nodo, primero se comprueba si ya existe información sobre esa misma transacción. Si no aparece en la tabla quiere decir que somos el primer nodo que ha recibido información de esa transacción y se almacenará en la base de datos el hash, los detalles de la transacción y un timestamp para marcar cuando se han recibido los datos desde ese nodo.

El resto de nodos, tras comprobar que una transacción ya está almacenada solo almacenarán en la base de datos el timestamp del momento en que se han recibido los datos. En realidad se guardan la id del nodo y de la transacción junto con el timestamp para poder identificar a quien y a qué pertenece ese timestamp.

Con los bloques se actúa de forma más sencilla, solo se guarda el timestamp del primer nodo que transmite información sobre un nuevo bloque encontrado.

Todos estos datos se muestran en tiempo real en dos tablas. La primera muestra información de los nodos y es similar a la que aparece en la *Figura 16*. En cuanto a la tabla de transacciones difiere en que ahora hay una columna por cada nodo desde el que escuchamos. Cuando en la columna de un nodo aparecen una linea punteada, significa que no se ha recibido todavía información de esa transacción desde el nodo.



Bitcoin-Sniffer funcionando

	TXhash	66.222.149.244	100.0.18.3	87.245.7.59	46.105.102.191
	4fc155d5ddd4fc12f2bd130602eaf5f320e42ead689711411eb7225cf1757eee	2013-05-30 22:30:31	2013-05-30 22:30:32	------	2013-05-30 22:30:30
	31f89c16c3050991ad61f16c542c20036c5b3a2b64f4ef32c49dbdf3f8607acb	2013-05-30 22:30:31	2013-05-30 22:30:32	------	2013-05-30 22:30:27
	e3fcfe9540f420bdf7c839e5fb5ab69d9074e1760e9692bfbc4a2530ffee2be9	2013-05-30 22:30:31	2013-05-30 22:30:32	2013-05-30 22:30:26	2013-05-30 22:30:26
	e3fcfe9540f420bdf7c839e5fb5ab69d9074e1760e9692bfbc4a2530ffee2be9	2013-05-30 22:30:31	2013-05-30 22:30:32	2013-05-30 22:30:26	2013-05-30 22:30:26
	38ec9c0f673e3c8396960177107ea87e64d39ab08f01ca388e0ec47be0adc475	2013-05-30 22:30:24	2013-05-30 22:30:31	2013-05-30 22:30:26	2013-05-30 22:30:26
	0dd0034fe912cdce4b2395482d8a7d1dbcd166f863817f781f2d6724bab1cb89	2013-05-30 22:30:27	2013-05-30 22:30:31	2013-05-30 22:30:26	2013-05-30 22:30:26
	4fc155d5ddd4fc12f2bd130602eaf5f320e42ead689711411eb7225cf1757eee	2013-05-30 22:30:31	2013-05-30 22:30:32	------	2013-05-30 22:30:30
	31f89c16c3050991ad61f16c542c20036c5b3a2b64f4ef32c49dbdf3f8607acb	2013-05-30 22:30:31	2013-05-30 22:30:32	------	2013-05-30 22:30:27
	36991b14cc11acc8fb6dc3a0ec119e546d22cd42dcbfda0c77a122e00f02e0df	2013-05-30 22:30:31	2013-05-30 22:30:27	2013-05-30 22:30:26	2013-05-30 22:30:26
	4fc155d5ddd4fc12f2bd130602eaf5f320e42ead689711411eb7225cf1757eee	2013-05-30 22:30:31	2013-05-30 22:30:32	------	2013-05-30 22:30:30
	36991b14cc11acc8fb6dc3a0ec119e546d22cd42dcbfda0c77a122e00f02e0df	2013-05-30 22:30:31	2013-05-30 22:30:27	2013-05-30 22:30:26	2013-05-30 22:30:26
	0dd0034fe912cdce4b2395482d8a7d1dbcd166f863817f781f2d6724bab1cb89	2013-05-30 22:30:27	2013-05-30 22:30:31	2013-05-30 22:30:26	2013-05-30 22:30:26
	31f89c16c3050991ad61f16c542c20036c5b3a2b64f4ef32c49dbdf3f8607acb	2013-05-30 22:30:31	2013-05-30 22:30:32	------	2013-05-30 22:30:27

Figura 20.- Detalle de la tabla de transacciones cuando se están usando varios nodos como objetivos de Bitcoin Sniffer.

Hay que destacar que después de realizar diversas pruebas estos “huecos” en la tabla de datos solo suelen aparecen al inicio y al final de la toma de muestras. Es decir mientras se inician y paran las distintas instancias de Bitcoin Sniffer. Salvo desconexión en los nodos todos ellos suelen enviar información de las mismas transacciones.

Por el motivo expuesto en el párrafo anterior, en el capítulo de resultados para obtener las estadísticas se suelen descartar los primeros y los últimos datos obtenidos.

Igual que en el caso de adquirir los datos desde un solo nodo, aquí se nos permite parar la actualización de los datos en pantalla, pero seguir adquiriéndolos en background. También podemos parar y reiniciar la propia adquisición de datos en todas las instancias.

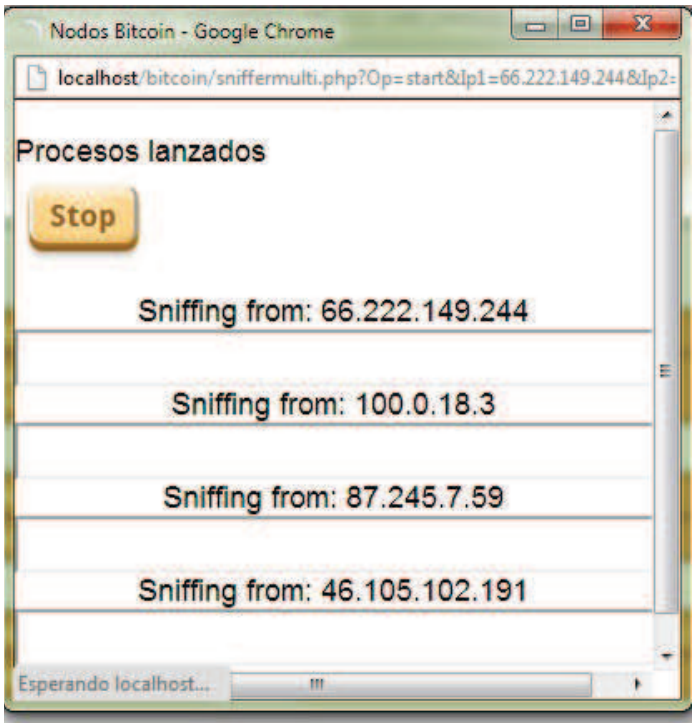


Figura 21.- Ventana desde la que iniciar y parar varias instancias de Bitcoin Sniffer.

- **Revisar datos última.** Esta opción vuelve a cargar en pantalla los datos de las últimos bloques y transacciones registrados. Los datos se visualizan de forma muy similar a cuando se están capturando los datos, con dos excepciones. Primero, que la información del momento en el que se ha encontrado cada transacción por parte de los distintos nodos no aparece directamente. Para acceder a ella solo hay que situar el ratón sobre el icono con una red de ordenadores.

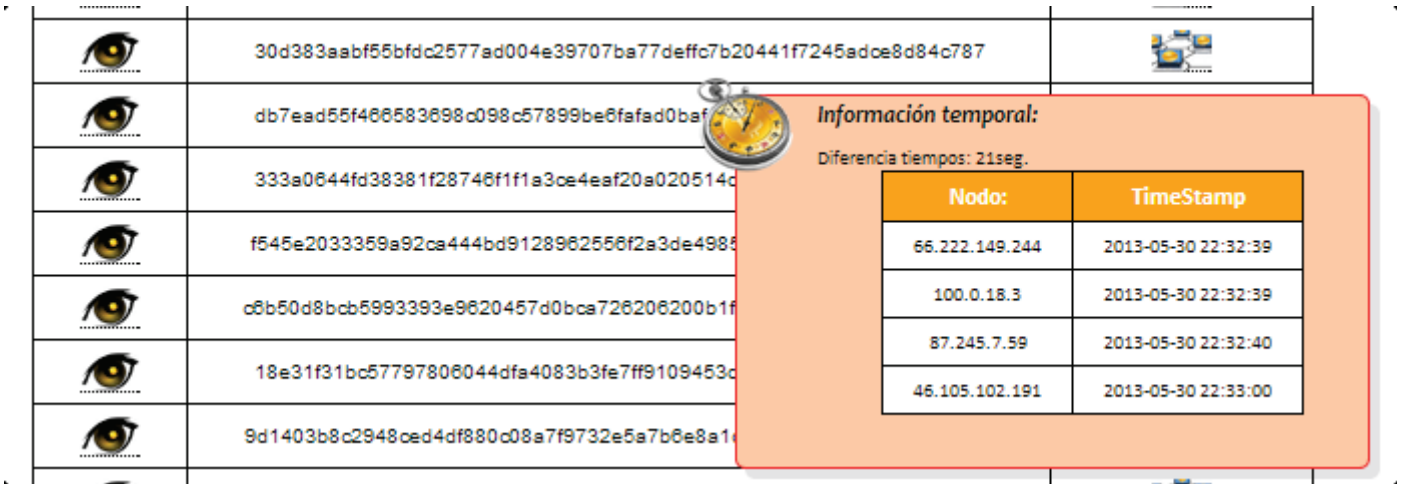


Figura 22.- Ejemplo de la información mostrada sobre un transacción recibida desde varios nodos.

Y en segundo lugar, se muestran más datos sobre los bloques. Estos datos no se muestran en tiempo real, ya que requieren muchos accesos a la base de datos. La siguientes imágenes muestra un ejemplo de la información mostrada para uno de los bloques:

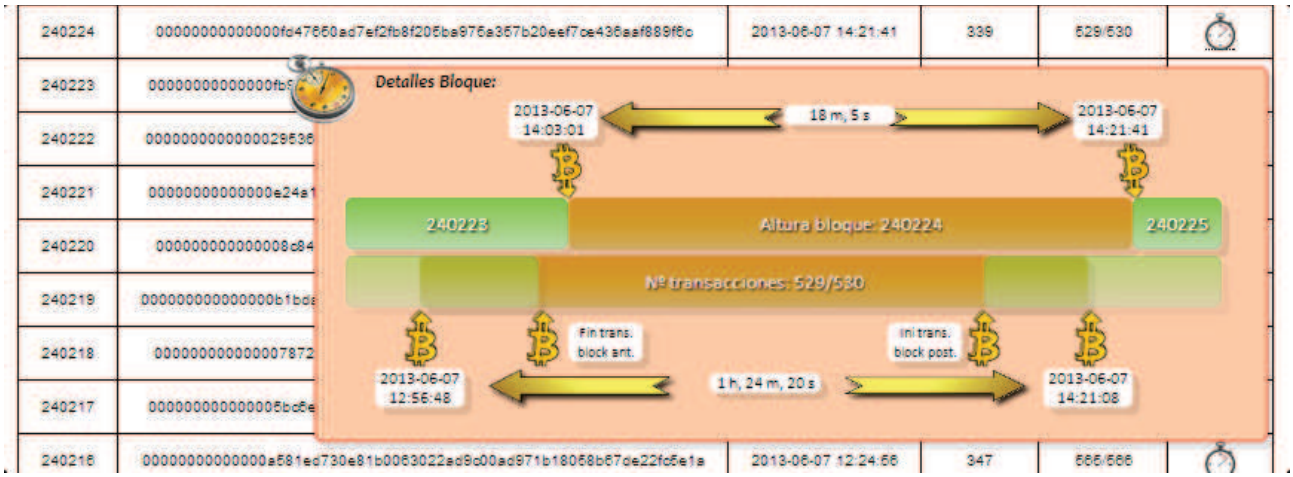


Figura 23.- Información mostrada sobre cada bloque encontrado.

Los tiempos que aparecen en la parte superior muestran cuando se encontró el bloque anterior, cuando se encontró el bloque actual y el intervalo entre ambos tiempos, que se corresponde con el tiempo que ha tardado en generarse este bloque.

En la parte inferior aparecen cuando se recibió información sobre la primera y la última transacción perteneciente a ese bloque y el tiempo transcurrido entre ambas. En la imagen se destacan dos zonas en las que se producen solapamiento entre transacciones pertenecientes a más de un bloque.

- **Revisar datos guardados**. Las tablas donde se almacenan los datos de los bloques y transacciones, debido al volumen que ocupan y a que en ocasiones los nodos elegidos no están conectados, se pueden descartar después de analizarse. La aplicación permite guardar estos datos en otras tablas más "permanentes", junto con otras series de datos seleccionadas. Mediante este apartado se permite revisar esta tablas permanentes que contienen datos de varias sesiones de adquisición.

  El aspecto de la tabla donde se muestran los datos es exactamente la misma que en el apartado anterior, lo único que cambia son las tablas origen de los datos.

- **Copiar datos temporales**. Guarda los datos almacenados en las tablas temporales en otras más permanentes. En la base de datos los nombres de estas tablas llevan delante el prefijo "*per_*".

  Una vez copiados los datos, se borran de las tablas temporales.

- **Borrar datos temporales.** Borra los datos de las transacciones de las tablas temporales.

Figura 24.- Mensaje que aparece cuando se ha copiado los datos correctamente en las tablas permanentes (izquierda) y cuando se han borrado correctamente los datos temporales (derecha).

4.4.5 Posibles ampliaciones.

La aplicación desarrollada solo esta preparada para usarse de forma local. Aunque funciona sobre un servidor web y simplemente sería dar acceso a este desde el exterior, al crear la aplicación no tenia en mente la seguridad de la misma y cuando lo consideré me di cuenta de que no tendría suficiente tiempo para llevar a cabo las modificaciones en el plazo de entrega. Entre estas modificaciones para permitir el acceso externo a la aplicación:

- Comprobar que solo se este ejecutando una instancia de la aplicación que recoge la información de los nodos.
- Sustituir todas los pasos de parámetros usando GET por POST y comprobar todos los parámetros que se pasan para evitar técnicas de SQL-Injection.
- Añadir un módulo de autenticación y autorización a la aplicación distinguiendo entre distintos roles de usuario. Unos posibles roles serían usuarios que pueden generar nuevas adquisiciones de datos y usuarios que solo pueden consultar datos.

Y en cuanto al aspecto estético (que también es importante) tenía pensado incluir la generación de gráficos en los resultados junto con las tablas de datos (ya tengo una versión que incluye algunos gráficos) y mostrar los nodos conectados y encontrados sobre un mapamundi en vez de sobre una tabla .

4.4.6 Versión de evaluación on-line.

En la dirección http://juadodo.no-ip.org/moodle se ha dejado una versión de prueba limitada (solo lista los datos sin poder realizar nuevas adquisiciones) para comprobar el funcionamiento de la aplicación desarrollada sin necesidad de instalarla.

En realidad se trata de una copia de la aplicación sin la parte que registra los nodos instalada en una máquina virtual. Desde aquí se accede (con un usuario limitado solo a SELECT) a la base de datos bitcoin en la máquina real. Como en la máquina real sí se está ejecutando la parte de la adquisición de datos, los datos se siguen actualizando cada 30 segundos.

5 Probando el número de conexiones simultaneas.

Sin tener el puerto 8333 abierto en el cortafuegos o en el router que nos da acceso a Internet, el número máximo de nodos a los que nos podemos conectar en la red Bitcoin es de ocho.

Si se tiene este puerto abierto el número de nodos a los que nos conectamos va aumentando hasta estabilizarse en torno a los 50 nodos. Entre los datos que nos muestra la aplicación BTCdoNET están las siguientes mediciones referidas a los datos del número de nodos conectados a nuestro equipo:

Ultima hora			
Media conexiones	55	Distintos nodos conectados	55
Máximo conectados	55	Mínimo conectados	55
Últimas 24 horas			
Media conexiones	50	Distintos nodos conectados (por hora)	59
Máximo conectados	58	Mínimo conectados	41

Figura 25.- Detalle de la información mostrada por BTCdoNET sobre los nodos conectados.

Estos datos son un resumen de los almacenados para la última hora y para las últimas 24 horas.

En la siguiente gráfica se muestra la evolución del número de nodos a lo largo del día 30 de mayo de 2013.

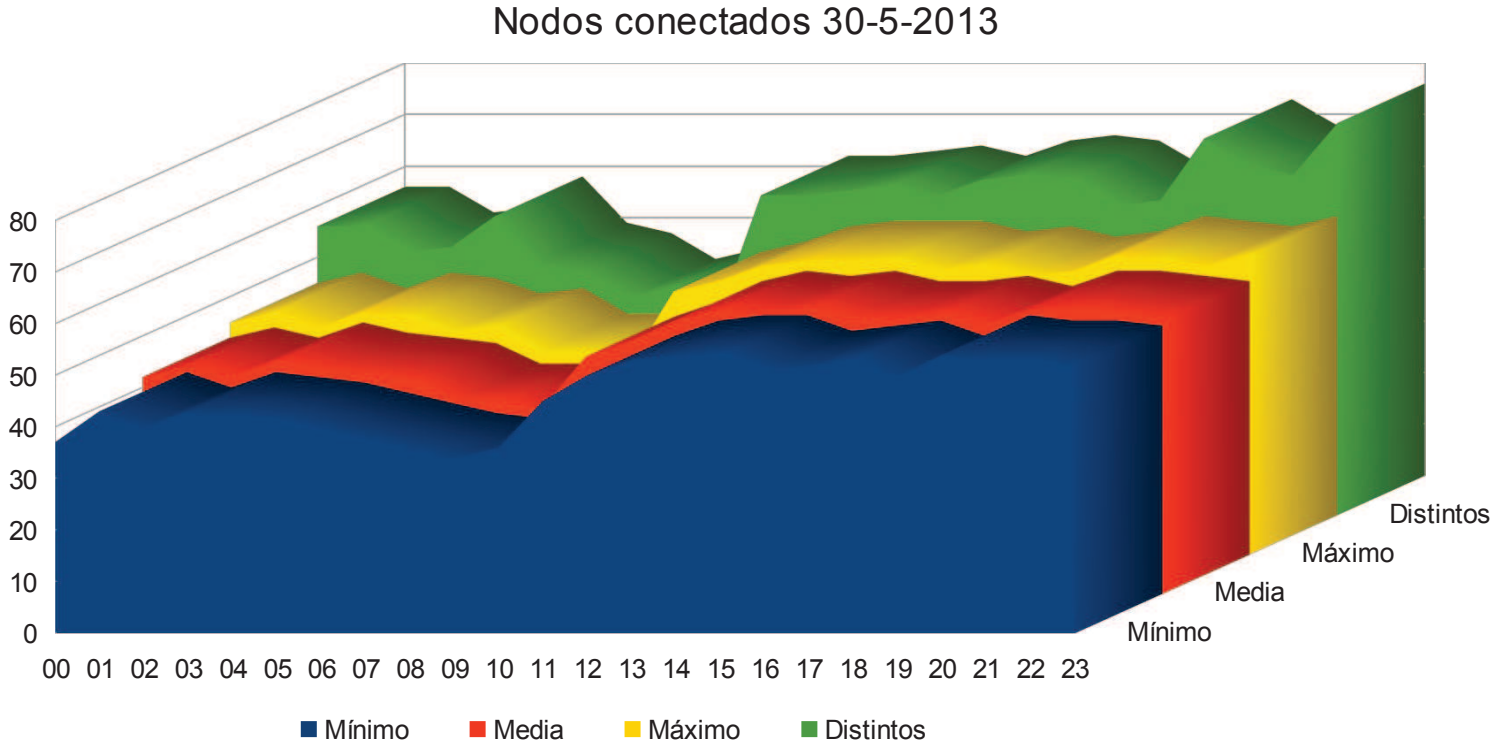


Figura 26.- Gráfico del número de nodos conectados durante el día 30/05/13.

5.1 Modificando el archivo de configuración Bitcoin.conf

Para intentar aumentar el número de conexiones simultaneas hacia la red Bitcoin se han modificado los siguientes parámetros del archivo de configuración de *bitcoin.conf*.

- Aumentar el parámetro *timeout* a 15000 y 30000. Este parámetro controla el tiempo limite sin recibir información de un nodo antes de desconectarlo. Se especifica en milisegundos y por defecto vale 5000.
- Aumentar el parámetro *maxconnections* (por defecto 125) hasta 1000.
- Aumentar el parámetro *banscore*. Este es el umbral para la desconexión de pares con mal comportamiento (por defecto: 100)
- También se han realizado cambios en los parametros maxreceivebuffer y maxendbuffer, que modifican el tamaño máximo del buffer de datos de recepción y envío respectivamente.

Pero cambiando estos parámetros en el archivo de configuración no parece haber tenido mucho efecto en los nodos, solo un aumento de menos del 10% sobre los nodos conectados. Sin modificar el archivo y tras varias horas de ejecución, el número de nodos se estabiliza en torno a 50, y con los parámetros modificados se estabiliza en torno a los 58 nodos conectados.

Otra prueba realizada consiste en aumentar el número de nodos conocidos mediante el comando del cliente bitcoind

addnode dir_IP add

Pero después de crear varios fichero de configuración añadiendo desde 50 hasta 500 de los nodos encontrados (sabiendo que están conectados la mayoría de ellos) no se produjo ningún aumento significativo del número de nodos conectados.

Lo único apreciable fue un pequeño aumento de la velocidad inicial en conectarnos a los nodos cuando se reinicia el cliente bitcoind o bitcoin-Qt.

Diversas pruebas modificando otros parámetros del fichero de configuración Bitcoin.conf tampoco resultaron en un aumento del número de nodos conectados.

5.2 Añadir nodos mediante addnode.

Un segundo intento fue el de enviar este mismo comando *addnode*, pero por JSON, de esta forma se pueden añadir nuevos nodos a la lista del cliente sin tener que reiniciarlo, pero tampoco se produjo un aumento de nodos conectados.

El tercer intento si que produjo algunos resultados positivos. Este consistió en enviar por JSON el comando *addnode* sobre una lista de direcciones IP, pero usando la opción *onetry* en vez de la opción *add*.

El método seguido es el siguiente, primero se buscan nodos candidatos, ya que no tiene sentido intentarlo con nodos que están desconectados o tienen el puerto 8333 cerrado porque no nos van a contestar a las peticiones . A continuación se envía un mensaje por JSON a Bitcoin-Qt por cada nodo candidato. Una vez ejecutado el resultado es el que se muestra en la siguiente tabla:

Resumen conexiones con nodos bitcoin Fri 31-May-2013 18:00:17			
Total nodos encontrados	6776	Total nodos conectados:	115
Total conexiones entrantes (puerto local 8333):	81	Total conexiones salientes (puerto remoto 8333):	34
Ultima hora			
Media conexiones	55	Distintos nodos conectados	118
Máximo conectados	115	Mínimo conectados	50
Últimas 24 horas			
Media conexiones	50	Distintos nodos conectados (por hora)	59
Máximo conectados	58	Mínimo conectados	41

Figura 27.- Detalle de nodos encontrados justo después del envío masivo de un par de cientos de direcciones IP mediante la opción JSON+try de la aplicación BTCdoNET.

Se ha aumentado hasta un total de 115 los nodos conectados. Pero este es un aumento temporal, a los pocos minutos de realizar la prueba, el número de nodos se vuelve a estabilizar en torno a los 58 nodos conectados.

La prueba anterior se realizó buscando primero de entre los nodos conocidos más estables, aquellos que estuvieran funcionando y tuvieran el puerto 8333 abierto o, si no había muchos, con el puerto 8333 filtrado. Esta lista de nodos se puede obtener con la opción *Listas → Comprobar nodos* de la aplicación desarrollada.

Con la lista creada solo hay que enviarla desde el menú *Añadir conexiones → JSON try*. Como ya se ha comentado el número de nodos conectados se duplica, pero en poco tiempo vuelve a bajar hasta el nivel anterior.

Otro intento, por si el problema era la alta velocidad a la que se añadían los nuevos nodos a probar consistió en ir comprobando los nodos y si cumplen los requisitos enviarles el comando JSON para conectarnos (*Añadir conexiones → JSON compro. + try*), pero no se consiguieron mejores resultados en el aumento de nodos.

Después de probar diversas opciones de envío de datos, seleccionando las IP de entre todas las conocidas, o enviándolas todas de forma masiva solo se ha aumentado un poco más el máximo de nodos conectados hasta 138. Pero lo que si se ha casi triplicado es el número de distintos

nodos a los que nos hemos conectado como resultado de todas las pruebas realizadas, tal como se muestra en la tabla de la *figura 28.*

Conexiones realizadas a la red Bitcoin

Resumen conexiones con nodos bitcoin Fri 31-May-2013 18:49:38			
Total nodos encontrados	6787	Total nodos conectados:	78
Total conexiones entrantes (puerto local 8333):	43	Total conexiones salientes (puerto remoto 8333):	35
Ultima hora			
Media conexiones	84	Distintos nodos conectados	334
Máximo conectados	126	Mínimo conectados	54
Últimas 24 horas			
Media conexiones	50	Distintos nodos conectados (por hora)	61
Máximo conectados	115	Mínimo conectados	41

Figura 28.- Captura de pantalla después de realizar varías pruebas de envío de nodos. Se ha destacado el gran aumento de número de nodos distintos a los que nos hemos conectado.

No se ha podido aumentar la cantidad de nodos conectados más allá de los 138, aunque esta cantidad es el doble de lo que teníamos hasta ahora.

5.3 Velocidad de conexión a nodos.

Un efecto colateral de los intentos anteriores ha sido que si bien no se ha obtenido un gran efecto en el número de nodos conectados, si se consigue aumentar rápidamente la cantidad de nodos conectados al iniciarse Bitcoin-Qt.

Normalmente al iniciarse Bitcoin-Qt se tardan varias horas en llegar al máximo al que nos podemos conectar. Pero si enviamos varios comandos addnode por JSON en pocos minutos se consigue llegar hasta más de 50 nodos conectados.

5.4 Número de conexiones vs número de nodos conectados.

Otro aspecto interesante que ha surgido de estas pruebas es comprobar que el número de conexiones activas indicado por Bitcoin-Qt no era el mismo que el número de nodos conectados detectados por BTCdoNET. Después de descartar que se tratará de algún error por parte de la aplicación, comprobé que la razón es que había varias conexiones activas desde una misma dirección IP. Este fenómeno aparece sobre todo después de realizar pruebas de inyección de nodos conectados mediante el envío a través de JSON del comando addnode. La razón es que debido a este comando se establecen varias conexiones simultaneas con algunos nodos.

Debido a esta discrepancia entre el número de conexiones y el número de nodos modifiqué la tabla resumen de conexiones para que incluyera las dos cifras como se ve en la siguiente imagen.

Resumen conexiones con nodos bitcoin Sun 9-Jun-2013 18:38:46			
Total nodos encontrados	8072	Total nodos conectados / conexiones:	58 / 62
Total conexiones entrantes (puerto local 8333):	55	Total conexiones salientes (puerto remoto 8333):	3
Última hora			
Media conexiones	57	Distintos nodos conectados	67
Máximo conectados	59	Mínimo conectados	56

Figura 29.- Instantánea donde se aprecia la diferencia entre nodos conectados y conexiones.

Lo habitual es que la discrepancia entre los dos números aparezca después de enviar comandos para añadir nodos, pero después de descubierto el fenómeno se ha apreciado también cuando no se ha enviado ningún tipo de comando al cliente Bitcoin-Qt.

Además de deberse a que un nodo establezca varias conexiones con nuestro equipo, otra posibilidad es que haya varios nodos detrás de un mismo router, por lo que tendrían la misma dirección IP.

Por comprobar que se pudieran dar los dos casos conecte un par de equipos a través de un router con NAT (un Mikrotik RB450G) e intente conectar varias veces con la dirección del router. El resultado fue tener varías conexiones abiertas con esa dirección, pero no se podía distinguir desde la aplicación cuando las conexiones eran con el mismo cliente de Bitcoin o cuando era desde dos clientes distintos.

Figura 30.- Si hay varios nodos detrás de un router no podremos saber en caso de tener dos o más conexiones con la dirección del router, si detrás de este router hay uno o más nodos.

6 ANALIZANDO DATOS OBTENIDOS

En los siguientes apartados se analizan los datos obtenidos desde la red P2P de Bitcoin. Algunos datos sobre los nodos descubiertos. A 30 de mayo de 2013 habían más de 6700 nodos descubiertos en la red Bitcoin. Siendo los datos más antiguos registrados por la aplicación BTCdoNET del día 18 de abril del 2013.

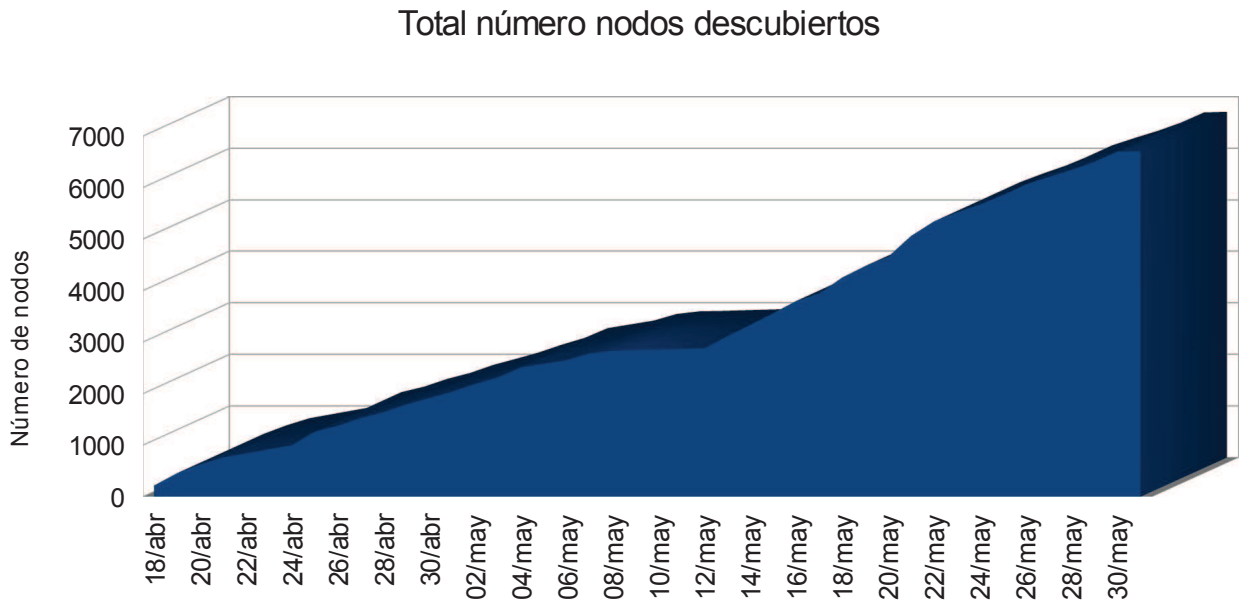


Figura 31.- Gráfico del número de nodos descubierto desde el 18/04/13 y 30/05/13.

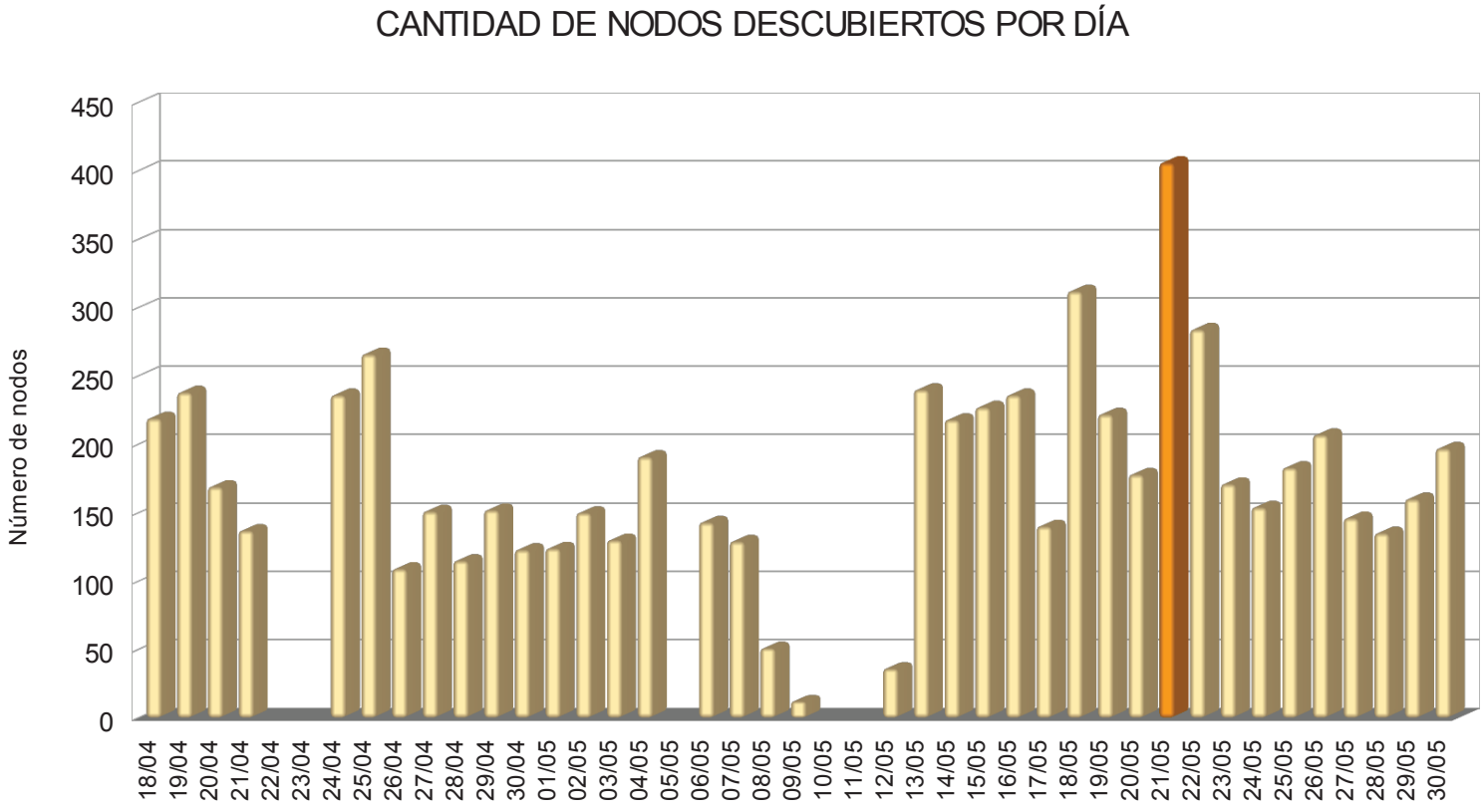


Figura 32.- Gráfico con nodos descubiertos por día.

El mayor número de nodos descubiertos se dio el día 21 de mayo con un total de 406 nuevos nodos encontrados en la red bitcoin. (ver gráfico *Figura 32*). Los días que aparecen sin nodos descubiertos se corresponden con los días en los que el equipo no estuvo funcionando.

El 14 de junio de 2013, tras dos semanas de funcionamiento ininterrumpido de la aplicación después de obtener estas gráficas el número de nodos descubiertos es ya de más de 8790.

6.1 LOCALIZACIÓN DE LOS NODOS

En cuanto a la distribución de nodos por países el 31,18% de los nodos (2092 nodos) se encuentran en Estados Unidos, seguido por China con un 13,61% (913 nodos). Los nodos españoles solo representan un 1,92% del total, solo 129 del total de 6709 encontrados (datos del 30 de mayo de 2013).

Agrupados dentro de "Otros" en el siguiente gráfico hay 1162 nodos, un 17,32% del total de nodos encontrados. Estos nodos están distribuidos entre un total de 101 países que van desde los 52 nodos de Israel o los 49 de Finlandia hasta países como Ecuador, Montenegro, Nepal y Iraq, por citar solo algunos, con solo uno o dos nodos conectados.

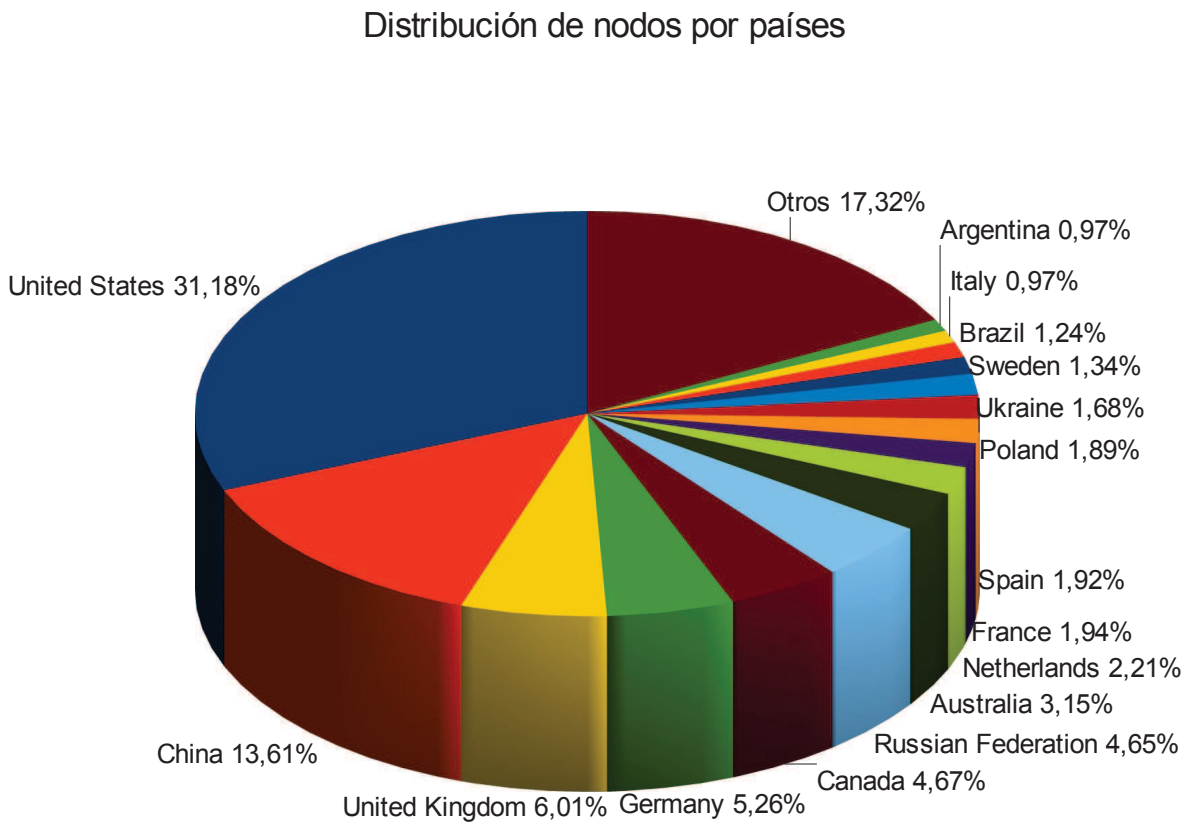


Figura 33.- Gráfico de la distribución del número de nodos por país.

En el siguiente gráfico (*Figura 34*) se agrupan los países según el número de nodos encontrados en cada uno de ellos. Como se puede ver hay 53 países con menos de 5 nodos y solo 2 (Estados Unidos y China) con mas de 500. España estaría en el grupo de entre 100 y 250 junto con otros 5

países (Australia, Países Bajos, Francia, Polonia y Ucrania)

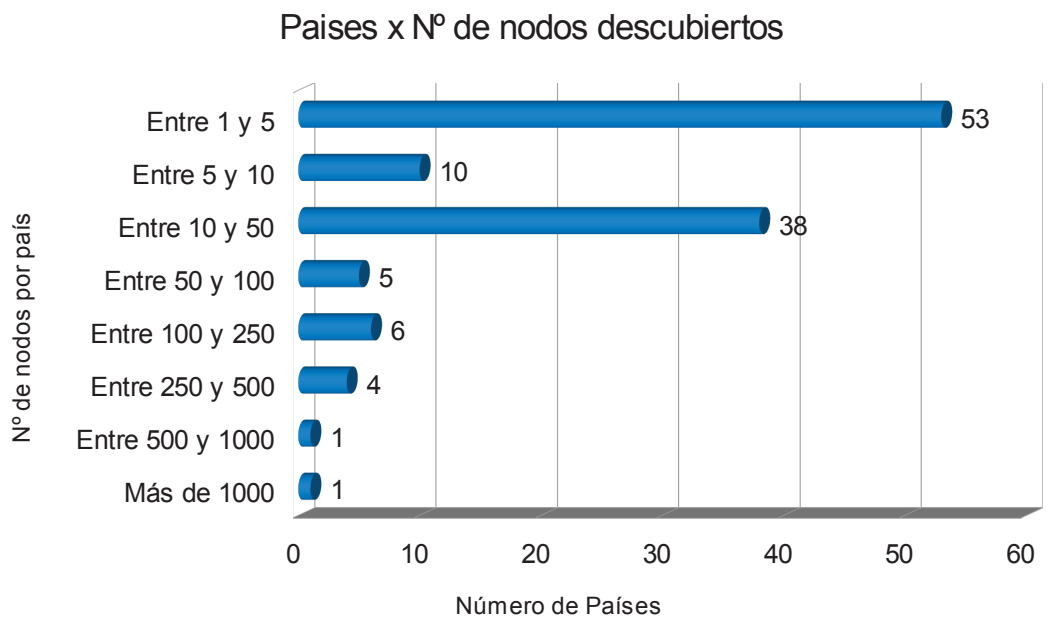


Figura 34.- Gráfico del número de países clasificados según los nodos encontrados.

6.2 Estudiando el tiempo de conexión

En el siguiente gráfico se muestran los nodos distribuidos según el tiempo que han permanecido conectados desde que se puso en marcha la aplicación por primera vez. Como se puede apreciar la mayoría son conexiones muy cortas, aproximadamente el 56% duran menos de media hora.

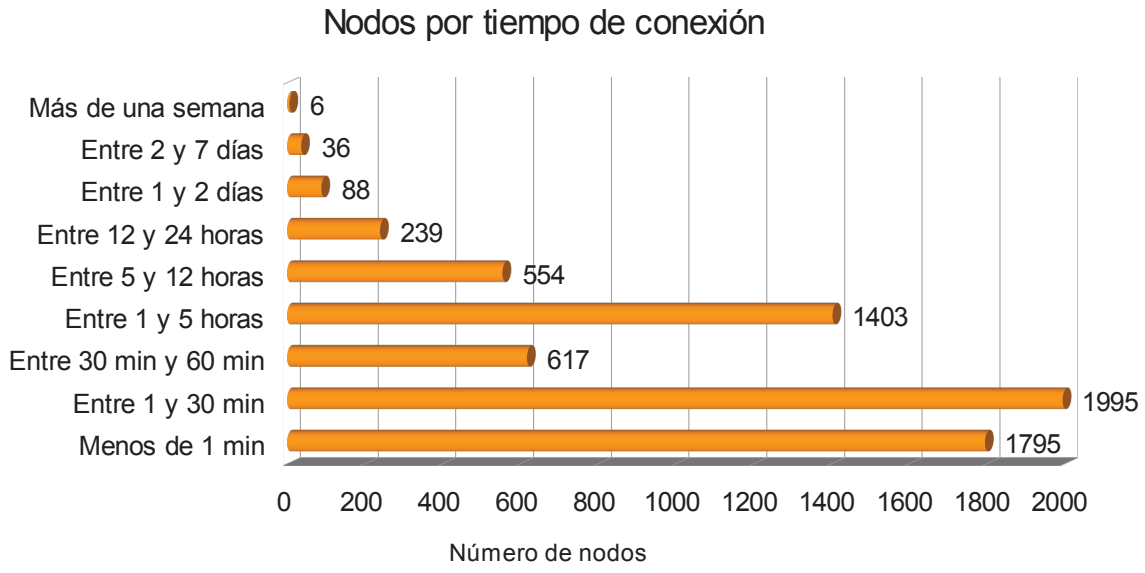


Figura 35.- Gráfico del número de nodos clasificados según el tiempo de conexión.

Otro 41% son nodos que han estado conectados al nuestro durante un tiempo de entre media hora y un día. Por otra parte hay nodos que se han conectado durante una gran cantidad de tiempo. Un total de 6 de ellos han estado conectados más de una semana.

El que más tiempo ha estado conectado con un total de 1.089.840 segundos (más de 12 días) es un nodo con la IP 192.168.1.161. Este es un equipo de la red local en el que también se estaba ejecutando un versión del monedero de Bitcoin, por lo que es normal que se conectaran muy a menudo.

El segundo nodo que más tiempo ha estado conectado, un total de 1.021.530 segundos (casi 12 días) es el nodo con la IP 94.23.203.60, que según la geolocalización está situado en Francia

Si nos limitamos a examinar los nodos que han tenido actividad en los últimos 7 días, los resultados son los que se muestran en el siguiente gráfico:

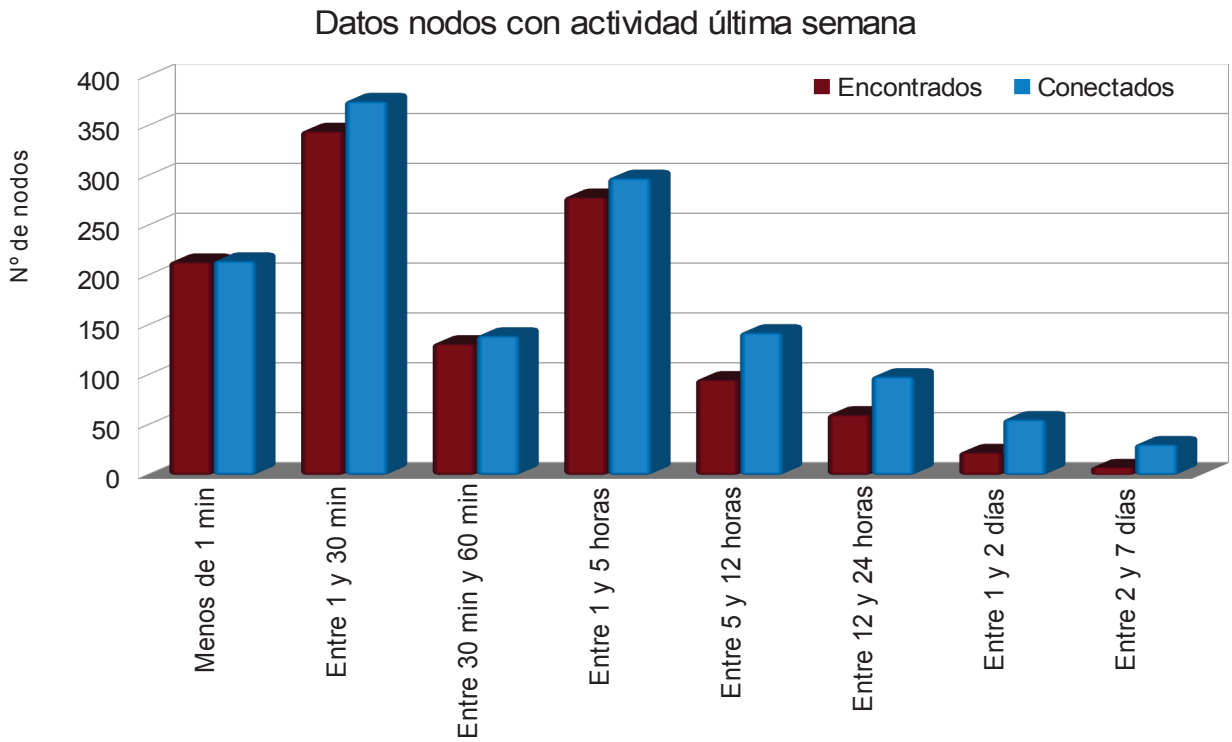


Figura 36.- Gráfico del número de nodos según el tiempo de conexión durante una semana de funcionamiento ininterrumpido.

En rojo aparece el tiempo de conexión de los nodos encontrados durante los últimos 7 días. En este caso como mucho el tiempo de conexión será evidentemente de 7 días. Pero eso no es cierto para la columna azul que indica el tiempo de conexión para los nodos conectados durante esa semana. Es decir, el nodo se puede haber encontrado hace más de una semana y su tiempo de conexión puede incluir periodos más antiguos de conexión, lo que estamos midiendo pues en esta columna es el tiempo de conexión total de los nodos que han tenido al menos unos segundos de actividad durante los últimos 7 días.

6.3 Estudiando las transacciones

Utilizando la opción de BTCdoNET para lanzar una o varias instancias simultáneamente de Bitcoin Sniffer y registrar los tiempos, se han realizado una serie de pruebas para medir la diferencias de tiempo en que cada nodo publica las transacciones que se realizan.

6.3.1 Algunas medidas sobre los tiempos entre transacciones.

Usando la versión modificada de Bitcoin Sniffer y cogiendo los datos directamente sobre localhost (es decir, sobre 127.0.0.1) he ido guardando información sobre el momento en que se reciben las transacciones.

A partir de una muestra de 1500 transacciones examinadas se han obtenido los siguientes resultados: el tiempo medio entre transacciones es de 2,31 segundos, siendo el mínimo tiempo entre ellas de 0 segundos (de hecho se suelen recibir en ráfagas) y el máximo de 43 segundos. Entre la primera de las transacciones examinada y la última han transcurrido un total de 3460 segundos.

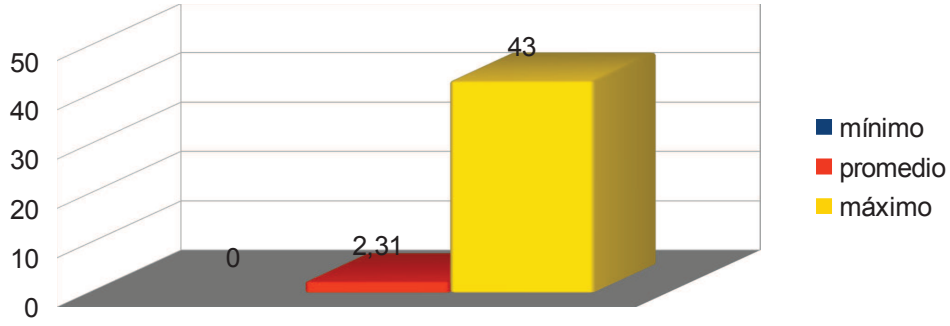


Figura 37.- Tiempo mínimo, medio y máximo entre transacciones consecutivas enviadas por un mismo nodo.

6.3.2 Diferencias de tiempos de registro de transacciones entre nodos

Se han estudiado diversas muestras de capturas de transacciones desde varios nodos, variando los propios nodos y el número de ellos utilizados. Los resultados son los que se muestran en la siguiente tabla:

Nodos analizados	Nº transacciones	Media	Máximo
3	500	10,94	57
4	500	10,27	37
5	500	15,15	141
5	1100	12,74	134

En la tabla anterior, nodos analizados son el número de nodos sobre los que se aplicó Bitcoin Sniffer al mismo tiempo, número de transacciones las transacciones utilizadas para realizar los cálculos (se desecharon las primeras y las últimas cogiendo el tramo central de transacciones), la media es el promedio en segundos de las diferencias entre el mayor y el menor timestamp para cada transacción, y máximo, la mayor diferencia entre timestamps de la misma transacción.

Como se puede apreciar, las transacciones se suelen recibir en todos los nodos examinados dentro de un rango de tiempo situado entre 0 y 141, aunque la media esta mucho más cerca del 0, aproximadamente entre 10 y 15 segundos.

6.4 ESTUDIANDO LOS BLOQUES.

Cada vez que se registra un nuevo bloque se almacena el timestamp de cuando se recibido información sobre ese nodo por primera vez. Además, los bloques guardan internamente también un timestamp de cuando se generó ese bloque.

La siguiente gráfica (*Figura 38*) muestra la diferencia en segundos entre estos dos tiempos en una pequeña muestra de 30 bloques. Los tiempos varían entre un mínimo de 1 segundo y un máximo de 183. La media son aproximadamente 32 segundos.

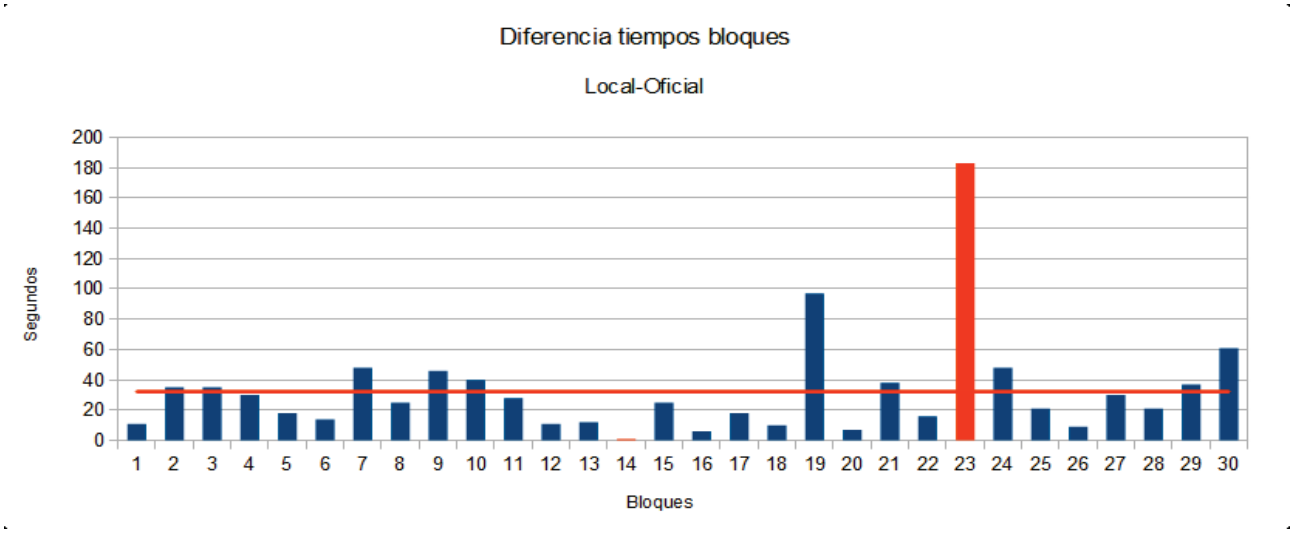


Figura 38.- Gráfica de la diferencia en segundos entre el tiempo de un bloque y cuando se registro localmente ese nodo.

La gráfica de la *figura 39* muestra el tiempo en segundos entre bloques consecutivos. El tiempo máximo ha sido de 56 minutos y el mínimo de sólo 3 segundos. En promedio se generan bloques cada 10 minutos aproximadamente, lo cual concuerda con la información que aparece en la wiki de bitcoin.

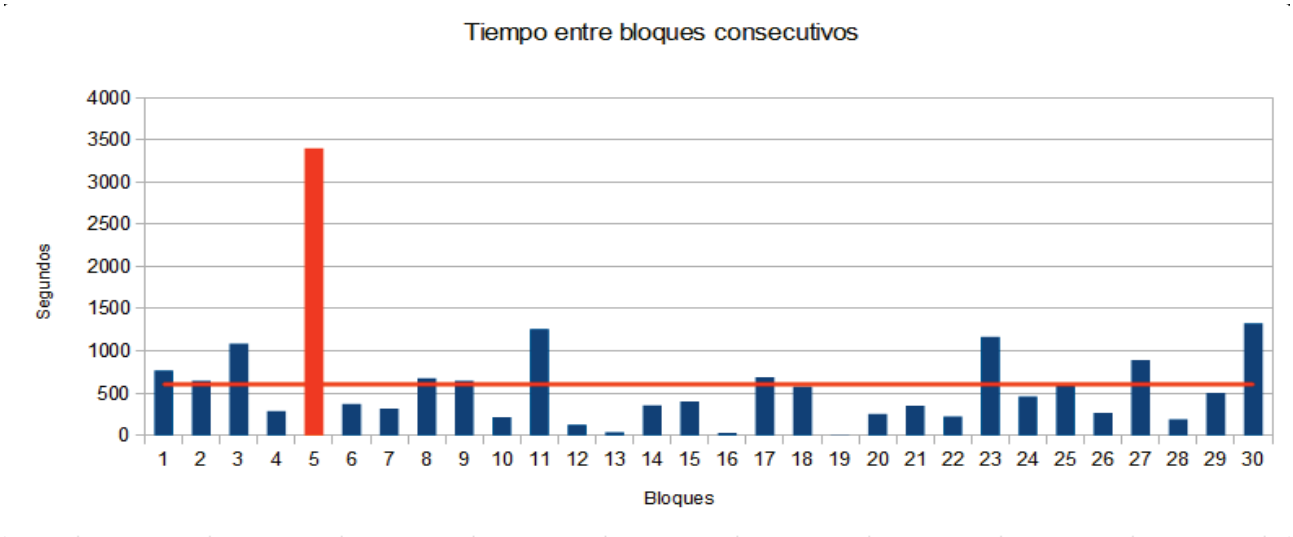


Figura 39.- Gráfica de diferencia de tiempo entre bloques generados.

6.4.1 Los bloques y sus transacciones.

Veamos los datos obtenidos de tres nodos consecutivos que aparecen en la siguiente imagen.

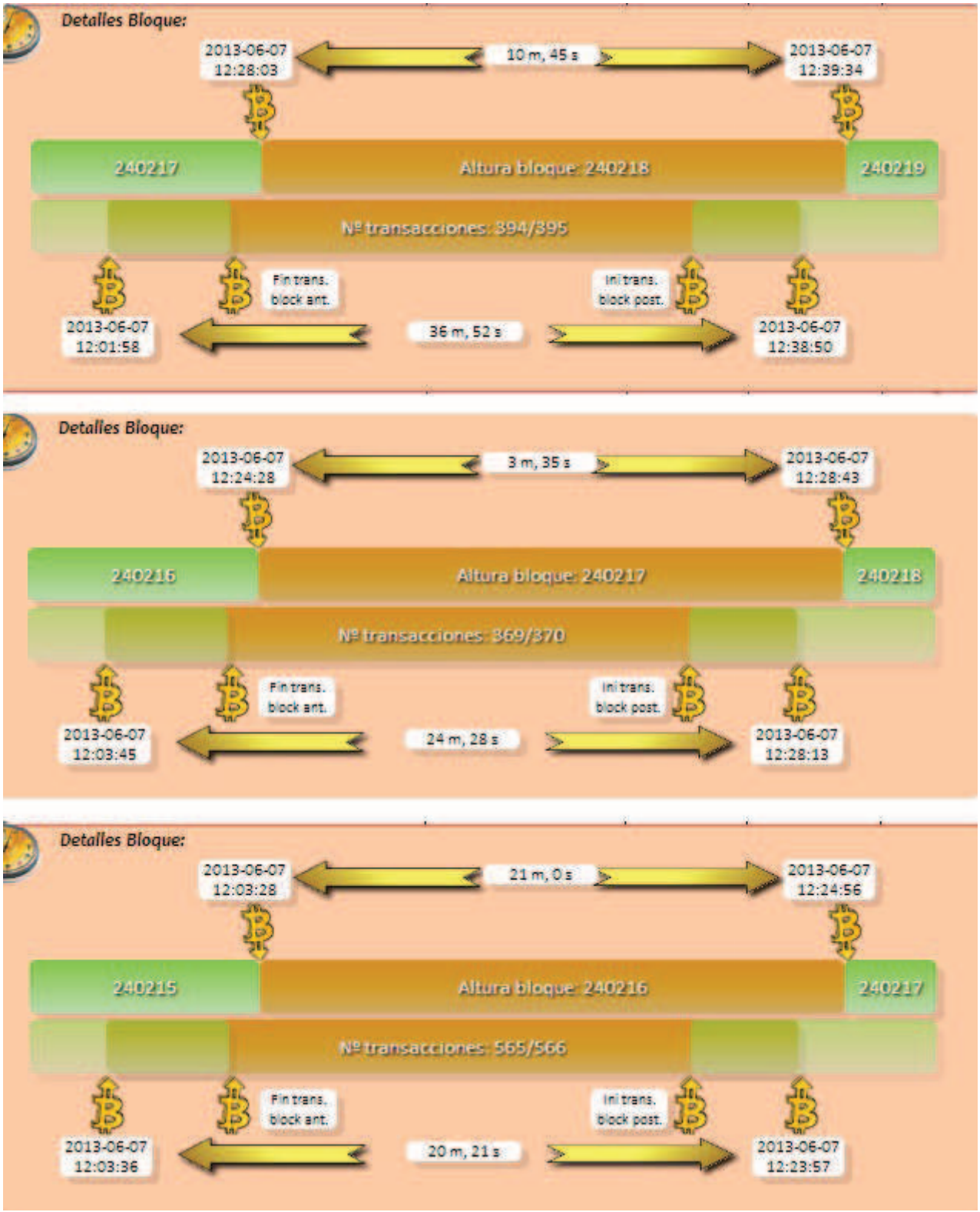


Figura 40.- Detalles de los tiempos mostrados por la aplicación BTCdoNET de tres nodos consecutivos.

Si nos fijamos en los tiempos que transcurre entre que se encuentra un bloque y el tiempo entre la primera y la última transacción registrada correspondiente un bloque (tiempo entre transacciones para abreviar), vemos que en los dos primeros casos el tiempo entre transacciones es mucho

mayor que el tiempo entre bloques. Un poco más de 20 minutos en ambos casos, pero si lo vemos en proporción representa más del triple en el primer caso y ocho veces más en el segundo. En el tercer caso sin embargo nos encontramos con el resultado de que el tiempo entre transacciones es más pequeño que el tiempo entre generación de bloques (unos40 segundos).

En la secuencia de bloques utilizada para crear las gráficas de las figuras anteriores existen 3 bloques en los que sucede lo mismo, los marcados en la gráfica como 6, 11 y 30.

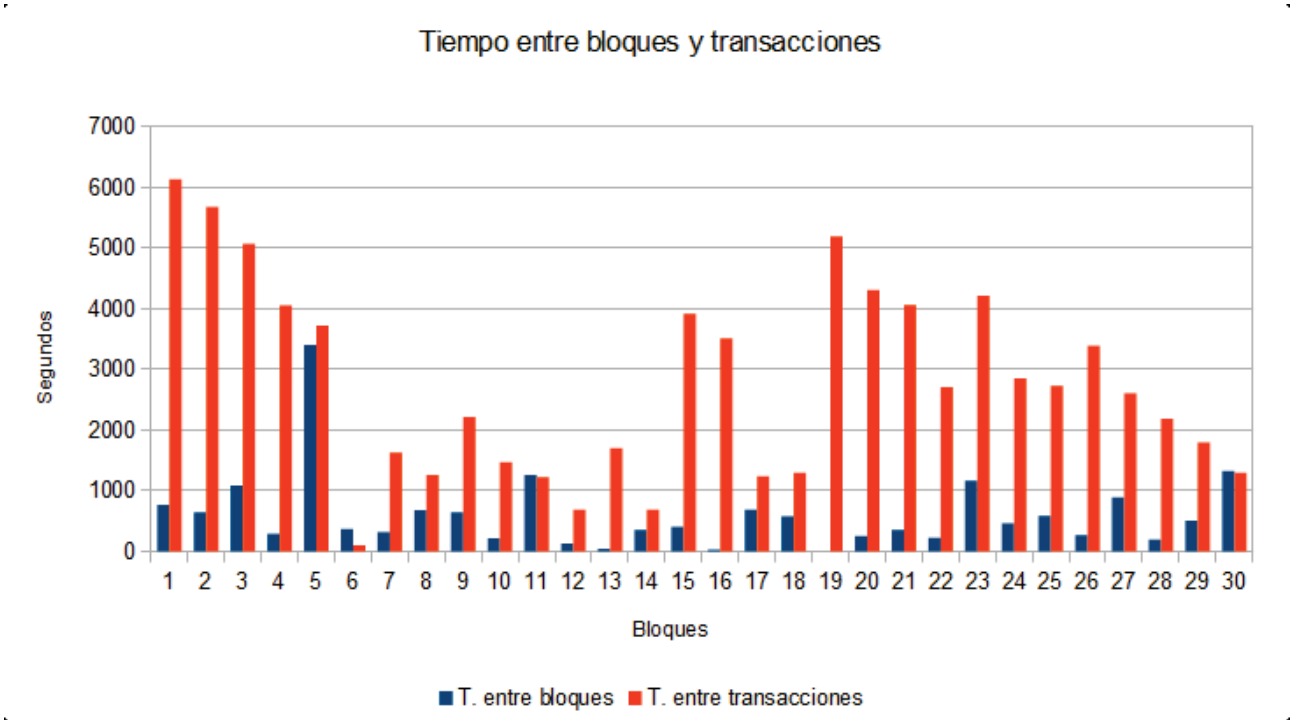


Figura 41.- Gráfica comparativa del tiempo que tarda en generarse un bloque y el tiempo entre la primera y la última transacción que forman parte de ese bloque.

Además, estos casos no se dan solo en la muestra que estoy usando para ilustrar los ejemplos, se puede comprobar que estos mismos resultados se repiten en otras muestras tomadas.

6.4.2 Solapamiento de transacciones.

Otra cuestión que me preocupó al analizar los datos es observar que algunas transacciones pertenecientes a varios bloques se solapan en el tiempo. ¿Puede darse esta situación?

Si no atenemos a lo que dice la documentación de Bitcoin acerca de los bloques si que lo son, o al menos cumplen las reglas:

> "*A block is a record of some or all of the most recent Bitcoin transactions that have not yet been recorded in any prior blocks.*" [6]

Es decir, un bloque almacena algunas o todas las transacciones de Bitcoin mas recientes, que no se hayan registrado ya en bloques anteriores. Puede darse el caso de bloques con transacciones solapadas en el tiempo y también aunque sean pocos, bloques generados a partir de un subconjunto de las transacciones que se han registrado. Lo que no puede darse nunca es que la

misma transacción aparezca en más de un bloque.

Otra cuestión interesante es ver si tienen relación el número de transacciones de un bloque con el tiempo en que tarda en generarse. La siguiente gráfica compara la forma de la gráfica del tiempo que tarda en generarse un bloque con el número de transacciones que contiene. Una es una medida de tiempo y la otra del número de transacciones, pero se puede apreciar que la forma de la curva resultante es muy similar, por lo que podemos deducir que ambos parámetros sí están relacionados, es decir, por lo general cuanto más tiempo tarda en generarse un bloque, más transacciones contiene.

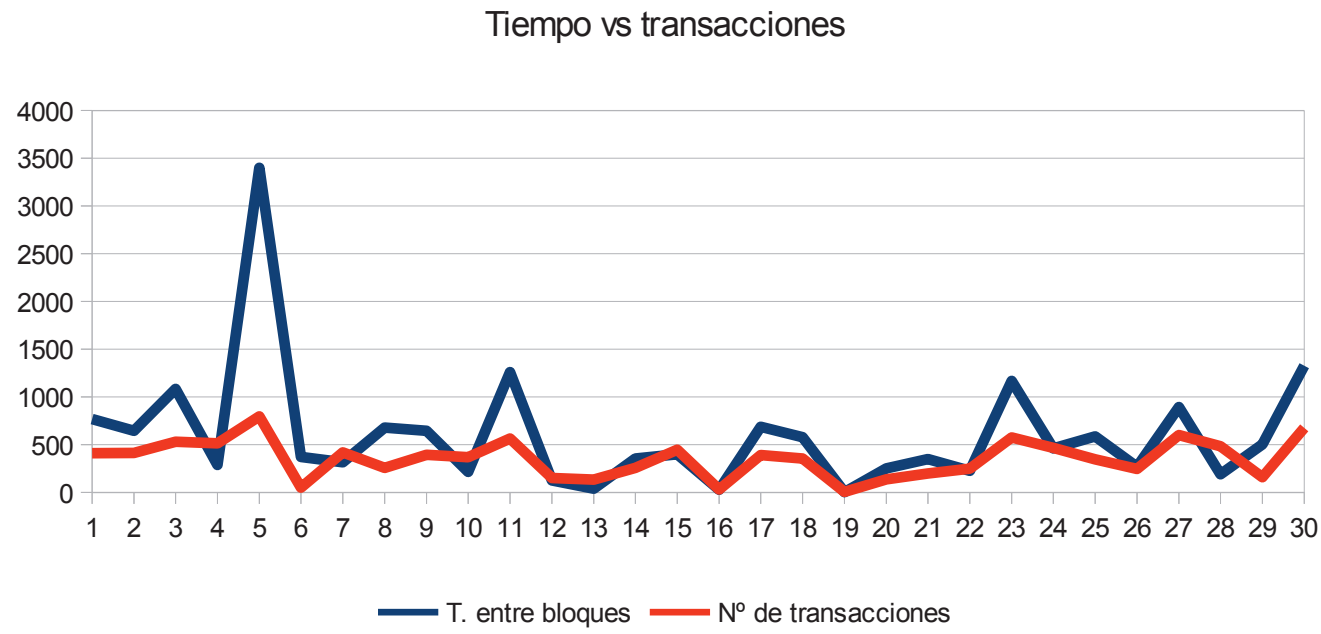


Figura 42.- Gráfica del número de transacciones que contiene un bloque comparado con el tiempo que ha tardado en generarse.

6.4.3 La transacción perdida.

Otro dato curioso, es que nunca hemos registrado todas las transacciones de un bloque. Hasta cierto punto, puede ser normal que nos perdamos la transmisión de una transacción, pero que en todos los bloques falta al menos una es extraño. En realidad, el no detectar esta transacción se debe al propio funcionamiento de la red Bitcoin y de la minería de datos asociada. Esa transacción que falta se corresponde con el pago de la recompensa (actualmente 25 BTC) para el primero que encuentra un hash válido para el bloque. Por lo tanto esa transacción no se llega a transmitir nunca por separado, se transmite junto con el bloque.

> *"Because there is a reward of brand new Bitcoins for solving each block, every block also contains a record of which Bitcoin address is entitled to receive the reward. This record is known as a generation transaction, or acoinbase transaction, and is always the first transaction appearing in every block. "*

6 Texto extraído de https://en.bitcoin.it/wiki/Blocks#What_is_the_maximum_number_of_blocks.3F

7 Conclusiones.

El funcionamiento de la red P2P de Bitcoin que en apariencia es algo tan sencillo como que los nodos conectados compartan la información que circula por esta red, es en realidad un sistema mucho más complicado y al mismo tiempo interesante. Dentro la red de Bitcoin hay que tener en cuenta aspectos como el descubrimiento de otros nodos por parte de los clientes, la transmisión de nuevas transacciones, la generación de nuevos bloques, que transacciones se incluyen dentro de un bloque u cuanto tardan en incluirse.

Con este TFM se han empezado a obtener y analizar datos sobre su funcionamiento y características, al mismo tiempo que se ha desarrollado una herramienta que puede servir para realizar futuros estudios.

A continuación se plantean algunas posibles ampliaciones del trabajo realizado:

- Realizar más medidas de transacciones, originando las transacciones y midiendo el tiempo en que se origina localmente la transacción, cuando la registran nodos a los que nuestro cliente de bitcoin está conectado, cuando la registran nodos a los que no estamos conectados directamente desde el cliente (pero si desde bitcoin sniffer), cuando se incluye la transacción originada en un bloque y cuando este bloque es confirmado de forma segura y pasa a formar parte de la cadena de bloques más larga.
- Medir la influencia de las comisiones en la reducción del tiempo de inclusión de una transacción dentro de un bloque. Se trataría de medir el tiempo de las transacciones que incluyen comisión y compararlo con el tiempo de las transacciones que no las tienen.
- También habría que medir la cantidad de comisiones que se incluyen en las transacciones, su porcentaje con respecto a la cantidad enviada y comprobar si a mayor comisión efectivamente hay mayor rapidez.
- Estudiar la influencia usar Bitcoin junto con redes como Tor. Se trataría por una parte de ver si se produce un retraso importante en la recepción y el envío de transacciones y bloques si se está utilizando este tipo de red junto con la de bitcoin.
- Otra cuestión sería tratar de identificar esos nodos que están detrás de una red Tor. Una idea inicial sería, en caso de que el estudio propuesto en el punto anterior arrojara que efectivamente hay un retraso en el envío de las transacciones si se utiliza Tor, estudiar el timestamp de las transacciones enviadas por un nodo para ver si contienen este tipo de retraso.

8 Bibliografía.

- **Satoshi Nakamoto** "*Bitcoin: A Peer-to-Peer Electronic Cash System*" satoshin@gmx.com www.bitcoin.org
- **Bitcoin.it**. "*Blocks*", [en línea]. December 2010, mayo 2013.https://en.bitcoin.it/wiki/Blocks
- **Bitcoin.it**. *"Block chain*", [en línea]. Diciembre 2010, mayo 2013. https://en.bitcoin.it/wiki/Block_chain
- **Bitcoin.it**. "*Current Network Status*", [en línea]. Septiembre 2011, mayo 2012. https://en.bitcoin.it/wiki/Current_Network_Status
- **Bitcoin.it**. "*Protocol specification*", [en línea]. Diciembre 2010, mayo 2013. https://en.bitcoin.it/wiki/Protocol_specification
- **Castro, Sebastian** "*Bitcoin P2P Network Sniffer v0.0.2*", [en línea]. https://github.com/sebicas/bitcoin-sniffer
- **Sáenz Higueras, Nita; Vidal Oltra, Rut.** P08/89018/00445 "*Redacción de textos científico-técnicos*" Versión en PDF disponible en: http://materials.cv.uoc.edu/cdocent/_D_CBBU62JZTHQ9CQQGJ.pdf

Máster Interuniversitario en Seguridad de las TIC (MISTIC)

ANEXOS

TFM – La red P2P de Bitcoin

Alumno: Juan Antonio Donet Donet

Directora: Cristina Pérez Solà

Universitat Autònoma de Barcelona

Fecha de realización: junio 2013

ÍNDICE DE CONTENIDO

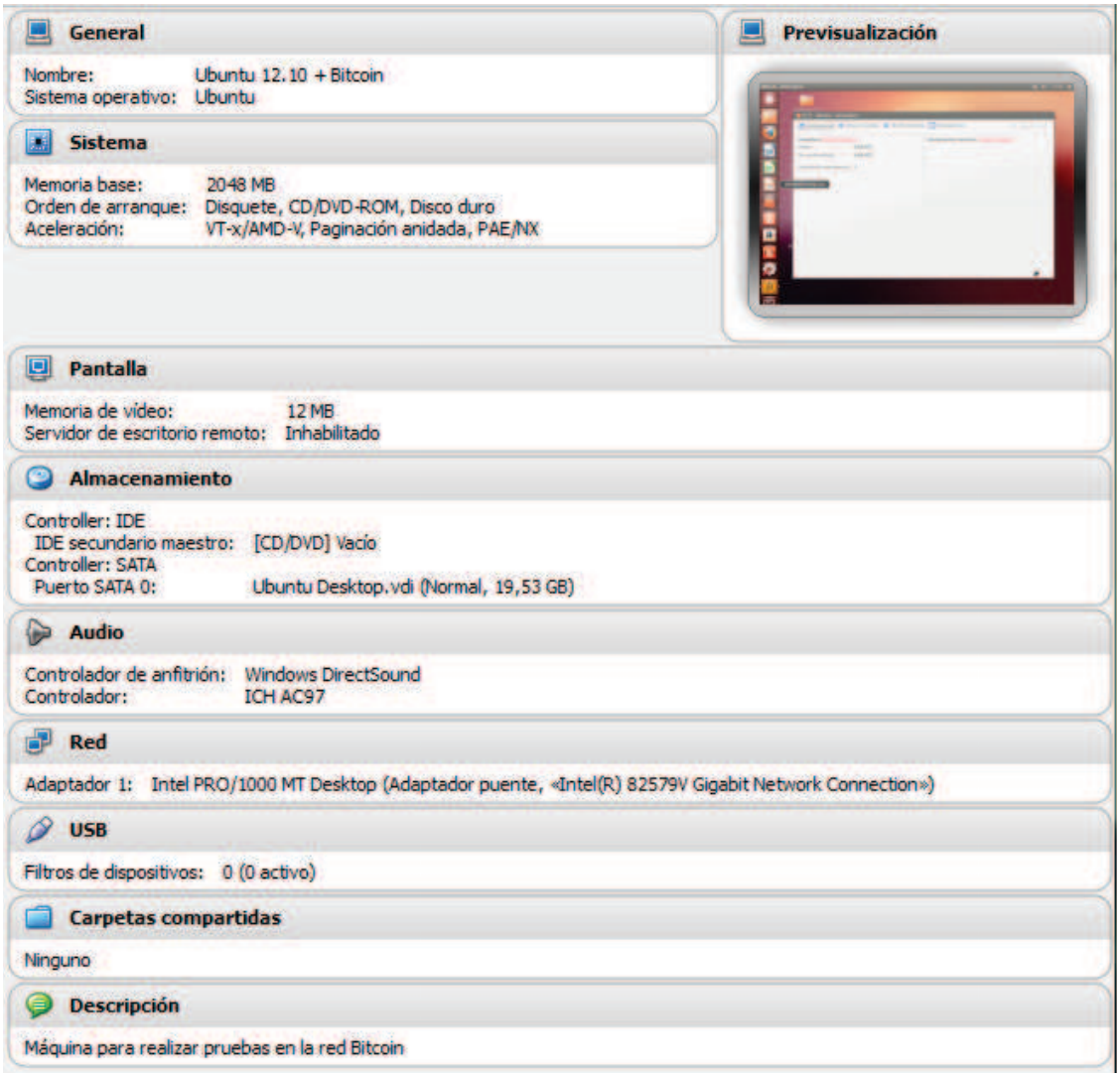
ANEXO I - CARACTERÍSTICAS GENERALES DEL ENTORNO DE PRUEBA.

A continuación se describen algunas de las características del hardware utilizado para la realización de este TFM:

Equipo 1		Equipo 2.		Portátil	
Procesador	i7-3820 3.60 GHz	Procesador	Quad Q8400 3.6GHZ	Procesador	i7 Q720 1.6GHz
RAM	16 GB	RAM	4 GB	RAM	4 GB
Gráfica:	NVIDIA GTX 670	Gráfica:	GeForce 9600 GT	Gráfica:	ATI HD 5650
S.O.	Windows 7 64 bits		Windows 7 64 bits	S.O.	Windows 7 64 bits

En cuanto a la conexión a Internet es una ADSL de 10 MB. Los test de velocidad sobre esta línea dan los siguientes resultados:

El software utilizado se instaló sobre Windows o sobre una máquina virtual con Ubuntu. La principales características de la configuración de la máquina virtual son las que aparecen en la siguiente imagen:



ANEXO II - DESCRIPCIÓN DEL SOFTWARE DE BITCOIN

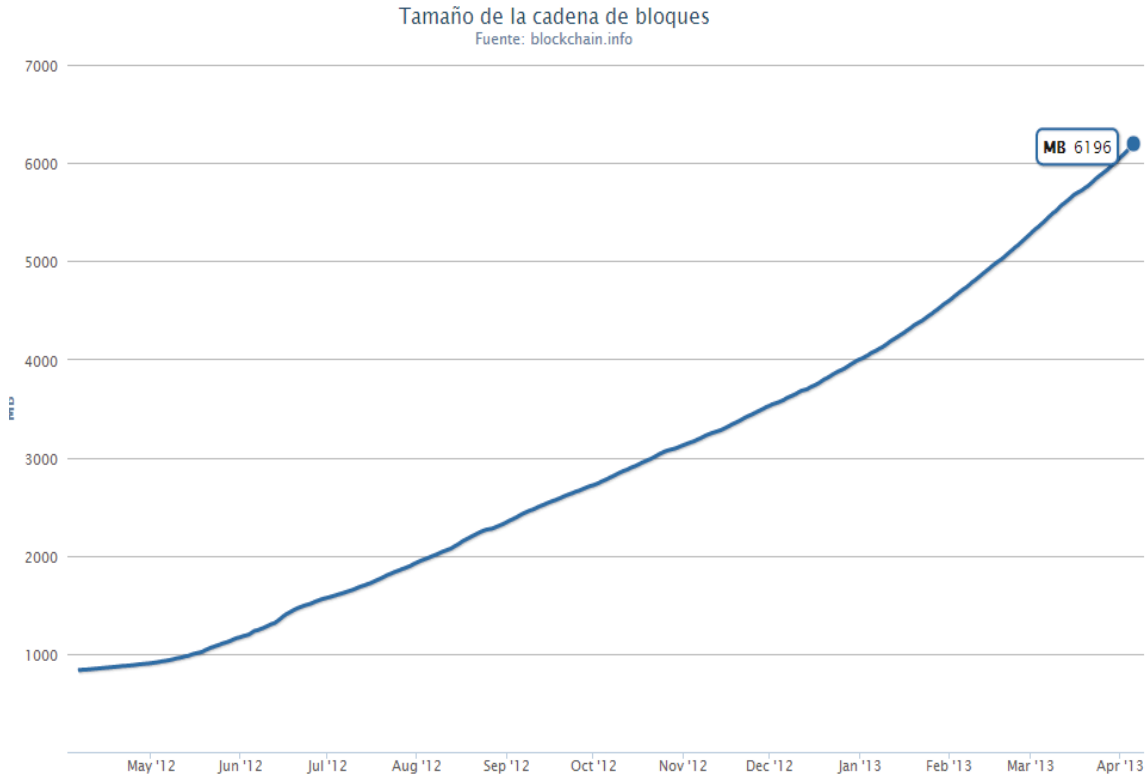
II.A - Bitcoin-Qt y Bitcoind.

Este es el cliente y la billetera oficial de Bitcoin.

II.A.I - CARACTERÍSTICAS

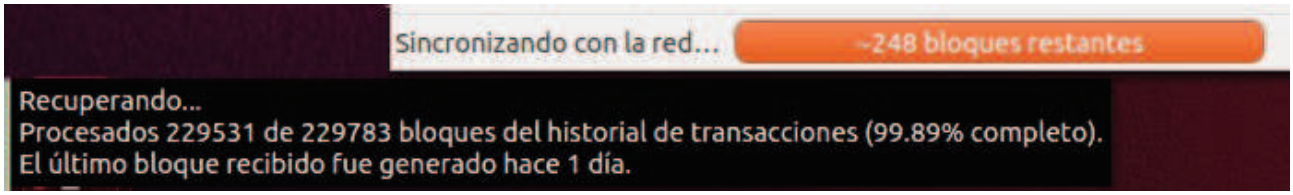
Se han instalado la versión 0.8.1 de bitcoind y Bitcoin-Qt. Mientras que bitcoind es el programa básico que proporciona toda la funcionalidad de Bitcoin a través de la línea de comandos, Bitcoin-Qt proporciona una interfaz amigable a través de una ventana principal. Bitcoin-Qt está basado en la biblioteca de interfaz gráfica Qt4 con licencia MT. Ambos programas se distribuyen juntos desde la versión 0.5.

La primera vez que se ejecuta este programa es necesario descargarse todos los bloques para sincronizarse con la red de Bitcoin. Este es un proceso largo que puede ocupar varias horas. Además hay que tener en cuenta el tamaño de la cadena de bloques, que se va haciendo más grande a medida que se añaden transacciones al conjunto. En el último año, el tamaño de la cadena de bloques a pasado de un poco menos de 1 GB en mayo del año 2012 hasta los casi 6,2 GB que tenemos a principios de abril de 2013.

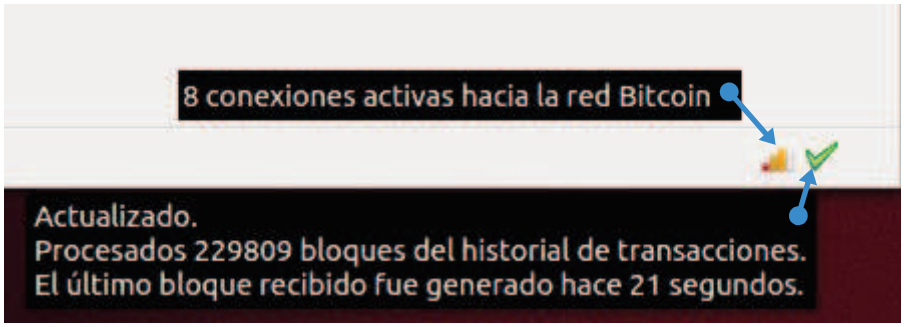


Cada vez que se ejecuta BitcoinQt, se realiza una comprobación de los bloques que se tiene almacenados y se comprueba en la red si existe alguno nuevo. En caso de que efectivamente

existan bloques nuevos, se van descargando hasta que se vuelva a estar sincronizado con la red.

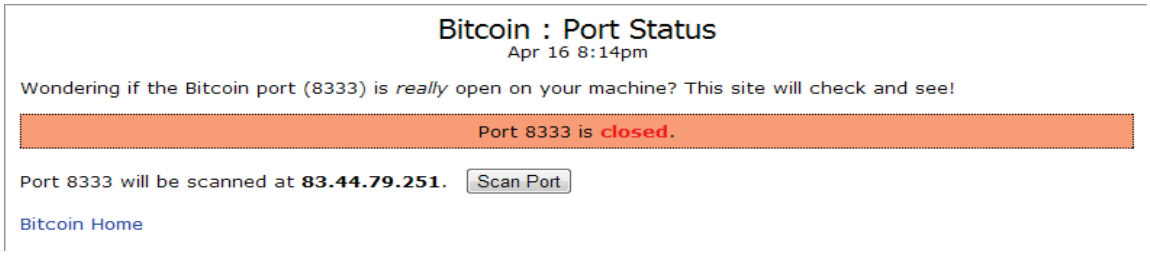


Una vez sincronizados no aparecerá el icono en la esquina inferior derecha de la ventana de la aplicación. Es icono , situado al lado del anterior, nos indica la cantidad de conexiones a la red de Bitcoin que mantenemos. En caso de tener el puerto de Bitcoin cerrado, tendremos solo hasta un máximo de ocho conexiones.

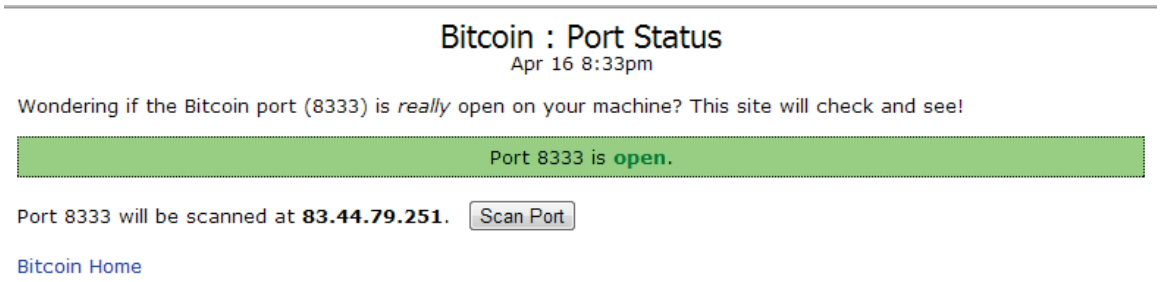


Para comprobar si tenemos el puerto de Bitcoin cerrado (por defecto el 8333), podemos usar una aplicación para comprobar los puertos que tenemos abiertos. Hay varias disponibles para Windows, Linux y Android. Aunque otra opción sencilla y cómoda es realizar la comprobación desde alguna página web, como por ejemplo desde:

http://www.alloscomp.com/bitcoin/portStatus.php)



En caso de tener el puerto cerrado como se muestra en la imagen anterior, tendremos que comprobar el cortafuegos de nuestro equipo y abrir el puerto en el router que estemos utilizando para conectarnos a Internet. Una vez realizados los cambios, volvemos a comprobar el estado del puerto 8333



Con el puerto ya abierto, como se ve en la imagen anterior, las conexiones activas hacia la red Bitcoin pueden superar el limite de 8 conexiones activas.

II.A.II - DESCUBRIENDO NODOS EN BITCOIN.

Cada conexión activa en la red P2P de Bitcoin implica que estamos conectados con una serie de nodos de los que obtenemos información y al mismo tiempo se la proporcionamos. Pero, ¿cómo se descubren estos nuevos nodos? Bitcoin tiene tres métodos para encontrar otros nodos:

- **addr**: cada nodo difunde un mensaje addr que contiene su propia dirección IP cada 24 horas. Los nodos transmiten estos mensajes a un par de nodos conocidos. Cuando los nodos reciben algún mensaje de este tipo, si la información es nueva para ellos la almacenan. Con este sistema, cada nodo tiene una imagen razonable de quien está conectado a la red en un momento dado. Después de conectarnos a la red, nuestra dirección se añadirá a la base de datos de todo el mundo casi al instante debido al mensaje *addr* inicial que se envía.

  Los datos de estos pares se almacenan (a partir de la versión 0.7.0) en el archivo ***peers.dat***. Este archivo utiliza un formato específico de Bitcoin e independiente de cualquier sistema de base de datos. Hay que comentar que la aplicación que se menciona en el punto anteriormente obtiene los datos de forma completamente independiente de bitcoin. Lo que hace es usar el comando *netstat* para observar las conexiones realizadas que tienen como origen o destino el puerto 8333.

- **DNS**. Algunos host actúan como servidores DNS y mantienen un registro de nodos de la red. Estos registros sirven para que los nuevos nodos puedan encontrar otros nodos rápidamente durante el arranque. Una lista de estos DNS aparece en el archivo ***net.cpp***, como se muestra en el siguiente extracto de su código fuente:

```
1170  // DNS seeds
1171  // Each pair gives a source name and a seed name.
1172  // The first name is used as information source for addrman.
1173  // The second name should resolve to a list of seed addresses.
1174  static const char *strMainNetDNSSeed[][2] = {
1175      {"bitcoin.sipa.be", "seed.bitcoin.sipa.be"},
1176      {"bluematt.me", "dnsseed.bluematt.me"},
1177      {"dashjr.org", "dnsseed.bitcoin.dashjr.org"},
1178      {"xf2.org", "bitseed.xf2.org"},
1179      {NULL, NULL}
1180  };
1181
```

- **IRC**: Se utilizaba para descubrir nodos durante el arranque, hasta que fue sustituido desde la versión 0.6.x por el método DNS.

II.A.III - ALGUNOS COMANDOS DE BITCOIND

Bitcoind cuenta con toda una serie de comandos para realizar y recibir transacciones, listar información, gestionar las cuentas, … A continuación se describen algunos de los comandos que sirven para obtener información del estado de la aplicación:

```
$ bitcoind
$ bitcoind stop
```

Las instrucciones anteriores sirven para poner en marcha o parar el cliente bitcoind.

```
$ bitcoind getaccountaddress juadodo
```

Devuelve la dirección de bitcoin para recibir pagos de la cuenta especificada (juadodo)

```
$ bitcoind getaddressesbyaccount juadodo
```

Devuelve la lista de direcciones de la cuenta especificada.

```
$ bitcoind getaccount 191t8fnut9TrVhpAww3rgucPNB4ED71a5Y
```

Devuelve la cuenta asociada a una dirección.

```
$ bitcoind getblockcount
```

Devuelve el número de bloques de la cadena de bloques más larga.

```
$ bitcoind getblockhash 232278
```

Devuelve el hash de un bloque (siempre en la cadena más larga), indicando su indice (232278 en el ejemplo).

```
$ bitcoind getblock 000000000000005c8a0d929ac02b482b737743b799b75539cc3407459bbba38b
```

Devuelve la información del bloque cuyo hash se ha proporcionado.

```
$ bitcoind getconnectioncount
```

Devuelve el número de conexiones a otros nodos.

```
$ bitcoind getdifficulty
```

Devuelve la dificultad de prueba de trabajo como un múltiplo de la dificultad mínima.

```
$ bitcoind getgenerate
```

Devuelve verdadero si bitcoind está generando hashes y falso si no los está generando.

```
$ bitcoind setgenerate true
$ bitcoind setgenerate false
```

Activa o desactiva la generación de hashes, es decir, la minería.

```
$ bitcoind getinfo
```

Devuelve información del estado. En la imagen inferior izquierda se muestra un ejemplo de su ejecución:


```
{
    "version" : 80000,
    "protocolversion" : 70001,
    "walletversion" : 60000,
    "balance" : 0.00000000,
    "blocks" : 232279,
    "connections" : 20,
    "proxy" : "",
    "difficulty" : 8974296.01488785,
    "testnet" : false,
    "keypoololdest" : 1363519226,
    "keypoolsize" : 99,
    "paytxfee" : 0.00000000,
    "unlocked_until" : 0,
    "errors" : ""
}
```



```
{
    "blocks" : 232281,
    "currentblocksize" : 224994,
    "currentblocktx" : 246,
    "difficulty" : 8974296.01488785,
    "errors" : "",
    "generate" : true,
    "genproclimit" : -1,
    "hashespersec" : 4215479,
    "pooledtx" : 449,
    "testnet" : false
}
```


```
$ bitcoind getmininginfo
```

Devuelve información relativa a la minería. La imagen superior derecha muestra un ejemplo de su ejecución.

```
$ bitcoind getpeerinfo
```

Lista información de los pares a los que estamos conectados actualmente. En la imagen de la derecha ve un ejemplo de la información devuelta de uno de los nodos a los que estamos conectados.

```
$ bitcoind getmemorypool
```

Devuelve los datos necesarios para trabajar en la construcción de un bloque.

```
$ bitcoind getrawmempool
```

Devuelve todos los ids de las transacciones en la memoria pool. Es una lista como la que se ve en la imagen inferior.


```
    "fc20afca40cdabcf34ddd24f4bae6782963534db65a6012663c944acd6f445fd",
    "fcb8e10f71645a7cf3c6d9cc2c4b261c00d43d1ac29bff8818420f24c05d22c1",
    "fd049f56f8ce050152796ed7fa88d5123561f3a0768002d7626a90dbcb343226",
    "fd290162dee63c223af152fcdc233f0a924580cb4537d795b0f36b0b5a04f950",
    "fd39f3c1abc7c72468d0ab44d3815f342f2036ec8b9e7f531021eb122d7bb600",
    "fdc563e16621f33a34efd19186501e6799f9b4e8d5bb5800b2768e96abc80f9b",
    "fedcd6959537975ef245f53f10de9e3c4f397d56f079320ccb6543ed6b41c54b",
    "ff5104576a3b9ecfc842d0e8b8587ca1975fbe1272a2ce8133ec41676316a95f",
    "ff58b64aedc4e31a4be4e2f5692536f5900ff4631bb5893f4ff4c25a2ac446e7"
```

II.B - BitcoinJS

BitcoinJS es un conjunto de bibliotecas de código abierto diseñado para ayudar a realizar de forma rápida proyectos personalizados de Bitcoin.

En el Anexo II se muestran detalles de la instalación de BitcoinJS.

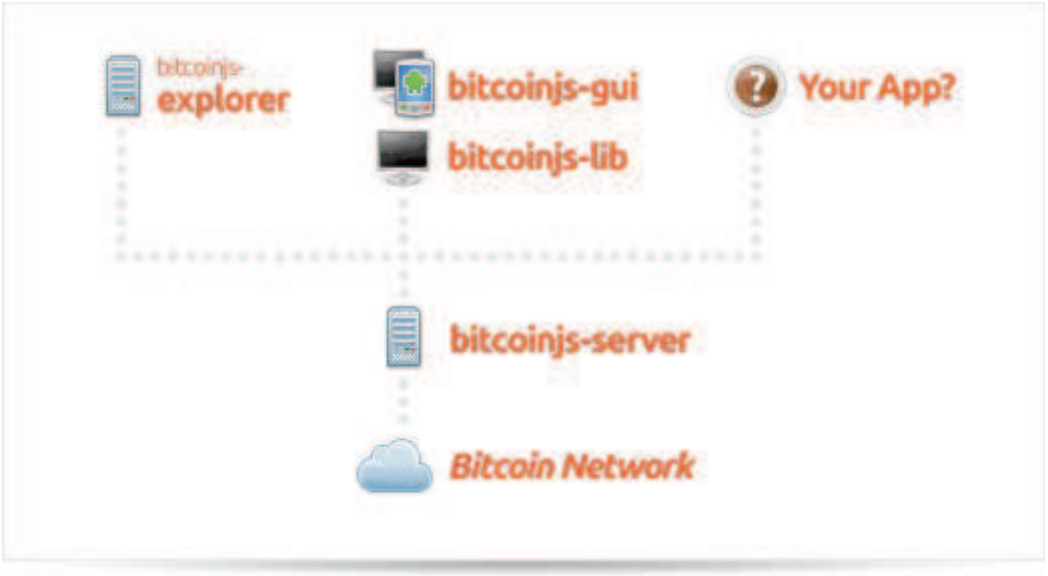
II.B.I - BITCOINJS-SERVER

Para interaccionar con la red de Bitcoin se cuenta con bitcoinjs-server. Está es una implementación de un nodo Bitcoin en Node.js. Está concebido como alternativa al estándar bitcoind que se entrega con el cliente original.

A diferencia del cliente original, que contiene el nodo, la cartera, la interfaz gráfica y la minera, esta biblioteca solo contiene una versión altamente optimizada del nodo, es decir, la parte P2P de Bitcoin.

Su principal uso es funcionar como un componente de servidor para dar a los clientes más ligeros acceso en tiempo real a los datos de la cadena de bloques. Pero también se puede utilizar para escribir otro software que requiere datos en tiempo real de la cadena de bloques o que requiera ejecutar consultas en ella.

Como se ve en la siguiente imagen, sobre bitcoinjs-server se sitúan el resto de proyectos, es decir, que bitcoinjs-server actúa de interfaz con la red Bitcoin para el resto de aplicaciones.



II.B.II - INSTALANDO BITCOINJS SERVER

La instalación de Bitcoinjs en Ubuntu es sencilla, pero hay que tener en cuenta una serie de detalles, los cuales se comentan a continuación:

En primer lugar, es necesario instalar una serie de pre-requisitos, entre los que están ***python***, ***openSSL*** y ***pkg-config***. Con el siguiente comando instalamos todos estos pre-requisitos:

```
# Install prerequisites
$ sudo apt-get install build-essential python2.7 pkg-config libssl-dev
```

A continuación es necesario instalar ***Node.js***. Aquí viene uno de los problemas, ya que si instalamos la última versión, esta no es compatible con la instalación de Bitcoinjs. Al proceder a instalar bitcoinjs después de instalar la última versión de Node.js (v0.10.2) se produce el siguiente warning, y la instalación termina en error:

```
npm http 304 https://registry.npmjs.org/leveldb
npm http 304 https://registry.npmjs.org/mkdirp
npm http 304 https://registry.npmjs.org/mongodb
npm WARN engine forever@0.10.0: wanted: {"node":"0.8.x"} (current: {"node":"v0.1
0.2","npm":"1.2.15"})
npm WARN engine leveldb@0.7.1: wanted: {"node":">=0.6.13 <0.9.0"} (current: {"no
de":"v0.10.2","npm":"1.2.15"})
```

La solución a este problema es descargar una versión anterior. En mi caso he usado la versión 0.8.4 que se puede descargar desde http://nodejs.org/dist/v0.8.4/node-v0.8.4.tar.gz. Una vez descargada esta versión solo hay que instalarla ejecutando estos comandos:

```
# Install Node.js
$ tar xzvf node-v0.8.4.tar.gz
$ cd node-v0.8.4
$ ./configure
$ make && sudo make install
```

En unos pocos minutos tendremos la instalación de Node.js realizada y con ello todos los prerequisitos necesarios para instalar BitcoinJS Server.

```
# Install BitcoinJS Server
$ sudo npm install -g bitcoinjs
```

Un último comentario, si la instalación de BitcoinJS Server falla y aparece un warning en el que se dice que el usuario root no tiene permisos en alguna de las carpetas, se puede solucionar ejecutando el siguiente comando:

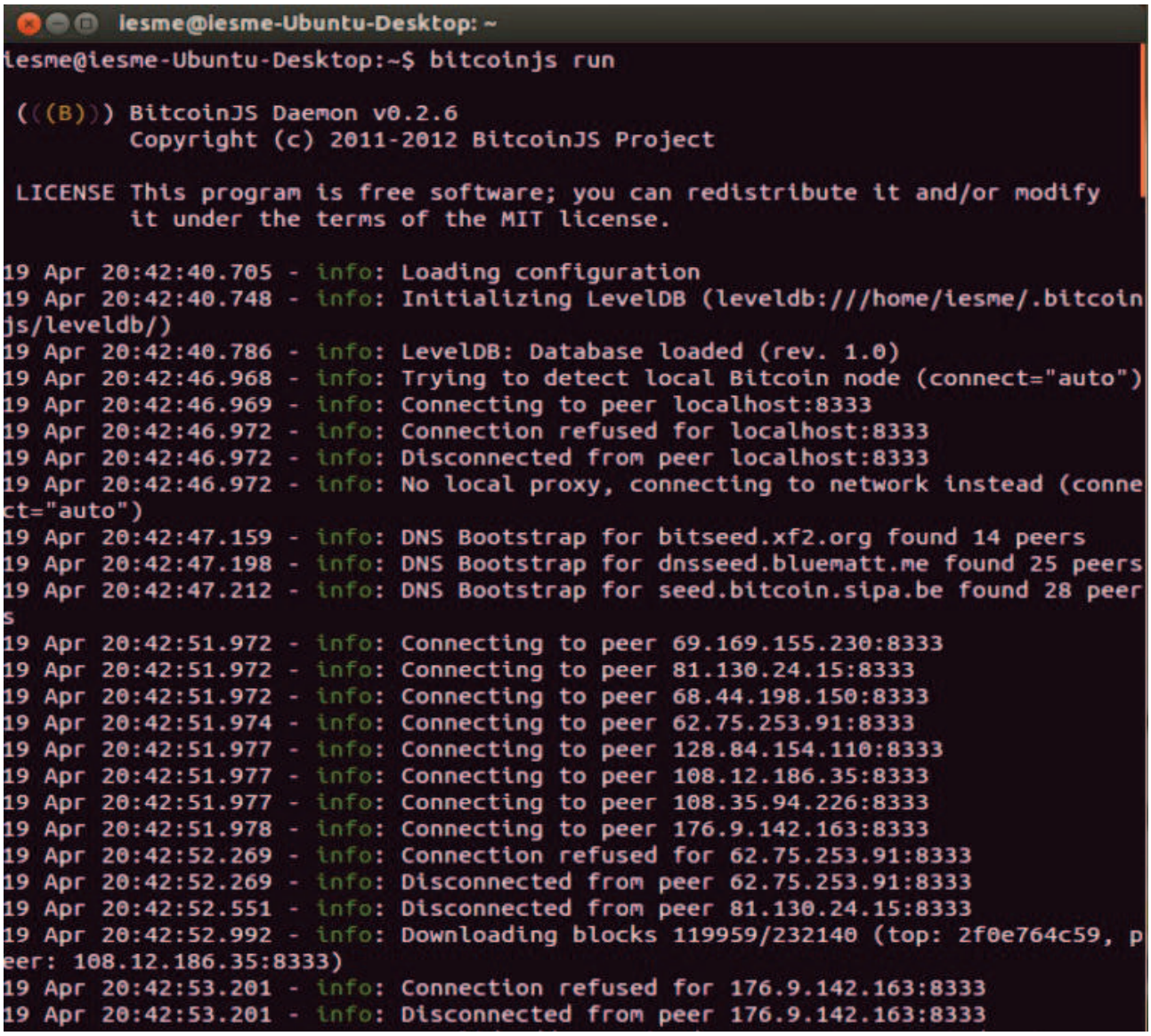
```
$ sudo npm install -g bitcoinjs --unsafe-perm
```

II.B.III - PROBANDO BITCOINJS-SERVER.

Después de instalar esta aplicación, para iniciarla hay que ejecutar el comando:

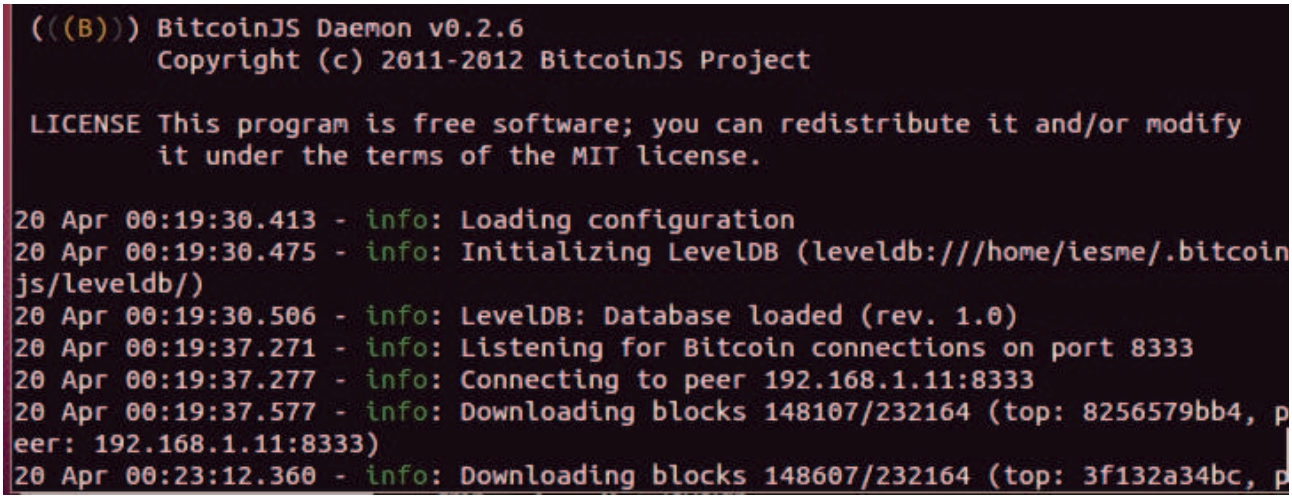
```
$ bitcoinjs run
```

Ahora ya estamos conectados a la red Bitcoin y se empezaran a descargar bloques (ver la siguiente imagen) hasta que se haya descargado la cadena de bloques entera.



Con la opción *run* vamos buscando nodos igual que con el cliente BitcoinQt, pero si tengo ya descargada la base de datos en otro equipo de la red local (o en la máquina anfitrión, ya que como se ha comentado estas aplicaciones están instaladas sobre una máquina virtual) puedo conectarme a ese nodo y evitar volver a descargar toda la información desde Internet:

```
$ bitcoinjs run --connect 192.168.1.11:8333
```



II.B.IV - OPCIONES DE BITCOINJS

Entre las opciones de consola de bitcoinjs están las siguientes:

- ***run***: Ejecuta bitcoinjs-server en primer plano. El formato del comando es:

```
$ bitcoinjs run [--testnet | --livenet] [<args>]
```

A *run* se le pueden pasar los siguientes argumentos:

--testnet : esta opción es para conectarse a la red de pruebas. Nos servirá para probar nuevos proyectos sin interferir en la red real.

--livenet: con esta opción nos conectamos a la red real. Es la opción por defecto.

--connect=<setting>: Cambia el modo en que este nodo se conecta a la red. Los valores que se pueden usar son:

<direccionIP>:<puerto> por ejemplo localhost:8333 o 192.168.1.11:8333

p2p: se conecta a la red peer-to-peer de bitcoin. En el siguiente apartado se muestra como descubre bitcoinjs los nodos a los que conectarse.

none: no se conecta a nada

auto: comprueba si se puede conectar a localhost:8333, y si no se puede intenta conectarse a la red p2p. Esta es la opción por defecto.

***--addnode*:** añade un nodo al grupo de nodos conocidos

***--forcenode*:** intenta conectarse a este nodo siempre. Esta opción predomina sobre cualquier configuración de connect. Por ejemplo, si ejecutamos:

```
$ bitcoinjs run – connect = p2p -- addnode 192.168.1.11:8333
```

se conectará al nodo 192.168.1.11 y después también a los nodos que encuentre en la red p2p.

- ***start***: ejecutamos bitcoinjs-server en modo gestionado, es decir, se inicia la aplicación pero no hay salida en la pantalla.
- ***stop*** y **restart:** para y reinicia bitcoinjs-server respectivamente.
- ***db-reset*** y ***db-drop****:* resetea y borra la base de datos respectivamente.

- ***bch-import*** y ***bch-export***: importan/exportan la cadena de bloques desde/a ficheros.
- ***verify***: chequea los datos de la cadena de bloques.
- ***test***: Ejecuta bitcoinjs-server en modo prueba.
- ***help***: muestra la ayuda de bitcoinjs

II.B.v - DESCUBRIENDO NODOS EN BITCOINJS-SERVER

Observando la salida del comando bitcoinjs-server encontramos líneas como las siguientes:

```
20 Apr 16:28:40.195 - info: DNS Bootstrap for bitseed.xf2.org found 14 peers
20 Apr 16:28:40.324 - info: DNS Bootstrap for dnsseed.bluematt.me found 25 peers
```

Bitcoin busca en varios servidores DNS de bitcoin y se descarga direcciones de pares a los que poder conectarse. Esta es una de las tres formas que se tienen de descubrir nodos en la red p2p de bitcoin y que ya se han descrito en el apartado "*Descubriendo nodos en bitcoin*" *(pág. 8)*

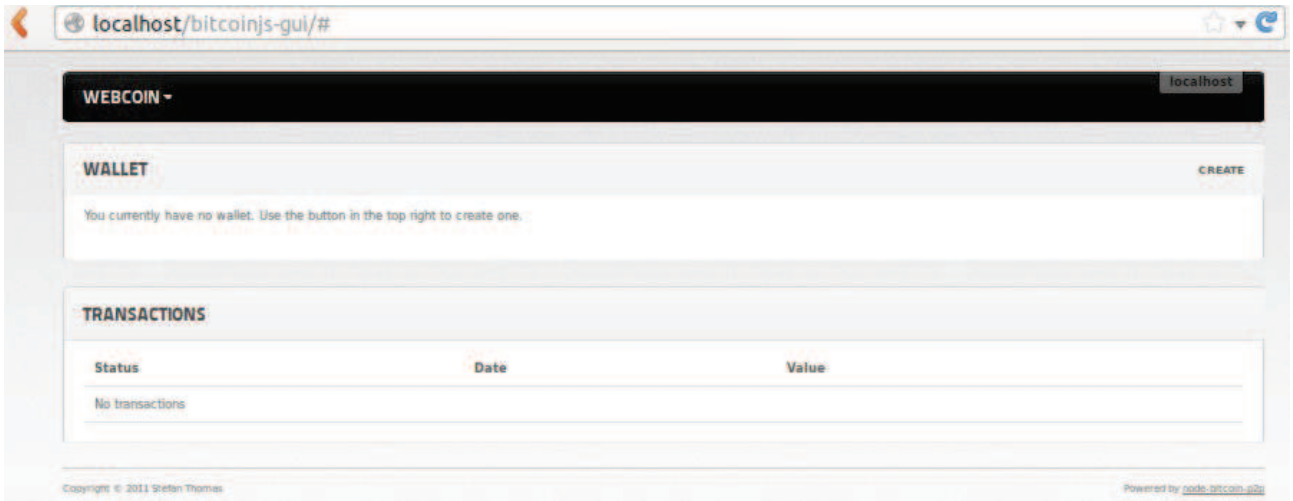
La configuración de este mecanismo de "siembra" con los servidores DNS a los que nos conectaremos aparecen en el archivo ***setting.js:***

```
97   // List of bootstrapping mechanisms
98   this.network.bootstrap = [
99     new DnsBootstrapper([
100      "bitseed.xf2.org",
101      "dnsseed.bluematt.me",
102      "seed.bitcoin.sipa.be",
103      "dnsseed.bitcoin.dashjr.org",
104    ]),
105    new IrcBootstrapper('irc.lfnet.org', '#bitcoin')
106  ];
```

II.B.vi - BITCOINJS GUI

Como ya se ha comentado, sobre bitcoinjs se pueden desarrollar otros proyectos. Uno de ellos es bitcoinjs-gui. Esta aplicación es una implementación de un cliente bitcoin sobre HTML5 y JavaScript.

El aspecto que tiene es siguiente:



II.C - Pynode

Este script es un nodo cliente para la red bitcoin. Su única función es escuchar las nuevas transacciones y bloques, lleva a acabo una comprobación de los datos y los almacena en una base de datos.

II.C.I - INSTALANDO PYNODE

Antes de proceder en la instalación de pynode tenemos que tener en cuenta las dependencias que tiene y que según se nos informa en la página web de github sobre este script son:

- ***python-bitcoinlib***: Esta librería se puede descargar desde la misma página web de github: https://github.com/jgarzik/python-bitcoinlib.
- ***Gevent***: Podemos encontrar información de esta librería de networking en la página http://www.gevent.org/ . Para instalarla en Ubuntu hay que realizar los siguientes pasos:

```
# El siguiente comando es solo para el caso de no tener instalados
# los paquetes que se mencionan. Para averiguar que versión de python-dev
# instalar podemos usar python -V

$sudo apt-get install python-setuptools build-essential python2.7-dev

# Instalamos primero greenlet, que es usado por esta librería
$ easy_install greenlet

# Instalamos por último gevent
$ easy_install gevent
```

- ***Python-leveldb***: Podemos ver más información de esta base de datos desarrollada por Google en: http://code.google.com/p/py-leveldb/

```
# La siguiente línea de código solo sera necesaria en caso de no tener ya
# instalados los paquetes que se indican en el propio comando.
$sudo apt-get install subversion automake libtool

# El siguiente comando descarga el código desde SVN
$ svn checkout http://py-leveldb.googlecode.com/svn/trunk/ py-leveldb-read-only

# Generamos la librería leveldb
$ cd py-leveldb-read-only
$ ./compile_leveldb.sh

# Generamos las extensiones de Python
$ sudo python setup.py build

# Por último las instalamos
$ sudo python setup.py install
```

Ahora ya tenemos todos los pre-requisitos cumplidos, solo hay que descargar los scripts desde https://github.com/jgarzik/pynode. No hay que olvidar copiar la carpeta bitcoin que hemos descargado de python-bitcoinlib dentro de la carpeta donde tenemos los scripts de pynode. Adicionalmente será necesario crear un archivo de configuración con los siguientes valores:

```
# nombre del host o IP del nodo al que nos vamos a conectar
host=127.0.0.1

# Puerto del nodo al que nos vamos a conectar (por defecto: 8333)
port=8333
```

```
# Usuario y contraseña del servidor JSON-RPC. Se usa autenticación básica de HTTP.
rpcuser=XXXX
rpcpass=YYYY

# Puerto TCP de entrada del servidor JSON-RPC (por defecto: 9332)
rpcport=9332

# Directorio de la base de datos
db=/tmp/chaindb

# fichero de log, o '-' o dejar sin valor para usar la salida estandar
log=/tmp/chaindb/node.log

# Si estan presentes, importar estos bloques en la base de datos
loadblock=/tmp/blk0001.dat

      # if present, disable all signature checking in new blocks
      # (disabled by default)
      nosig=1

      # if present, force signature checking on all blocks,
      # even those normally skipped because they were prior
      # to a checkpoint.
      # (disabled by default)
      forcesig=1
```

Después de instalarlo, modificamos el archivo de configuración indicando un directorio donde se almacenará la base de datos (o donde ya tenemos una) y el nodo al que nos conectaremos. Hay que tener en cuenta que solo nos podemos conectar a un nodo. Por defecto este es 127.0.0.1, pero podemos incluir cualquier otro. En mi caso lo he ejecutado en una máquina virtual con Ubuntu y lo he conectado a la maquina huésped en la que estaba ejecutándose BitcoinQt para realizar pruebas.

Para iniciar la aplicación hay que ejecutar el siguiente comando:

```
$ ./node.py config-file
```

La siguiente imagen muestra una captura de pantalla de la ejecución de pynode.

```
PeerManager: connecting to 192.168.1.11:8333
connecting
send msg_version(nVersion=60002 nServices=1 nTime=Sun Apr 21 00:31:53 2013 addrT
o=CAddress(nTime=0 nServices=1 ip=192.168.1.11 port=8333) addrFrom=CAddress(nTim
e=0 nServices=1 ip=0.0.0.0 port=0) nNonce=0xB52D8778E9134C76 strSubVer=/pynode:0
.0.1/ nStartingHeight=27044)
192.168.1.11 connected
recv msg_version(nVersion=70001 nServices=1 nTime=Sun Apr 21 00:32:07 2013 addrT
o=CAddress(nTime=0 nServices=1 ip=0.0.0.0 port=0) addrFrom=CAddress(nTime=0 nSer
vices=1 ip=80.29.84.67 port=8333) nNonce=0x1CDFF2D5AEA6DFCE strSubVer=/Satoshi:0
.8.0/ nStartingHeight=232310)
send msg_verack()
send msg_getaddr()
send msg_getblocks(locator=CBlockLocator(nVersion=60002 vHave=[14911539109534584
14661151292797986221001800135945086936951107199820OL]) hashstop=0000000000000000
0000000000000000000000000000000000000000000000)
recv msg_verack()
ChainDb: height 27045, block 000000003372929e5bd3f7dfcc265dc50d6924655a338230910
b1e9041541acb
MemPool: blk.vtx.sz 1, neverseen 1, poolsz 0
ChainDb: height 27046, block 00000000217fe5a71681cc5a9c36e2bb4e17b9d5a54dba54742
```

II.D - Bitcoin Sniffer

Bitcoin Sniffer es un script basado en Python que se conecta a un nodo de la red bitcoin y se dispara cuando recibe alguno de los siguientes eventos:

- **new_block_event(block)**: Se activa cuando se encuentra un bloque.
- **new_tx_event(tx):** Se produce cuando se encuentra una transacción.

Este sniffer facilita una forma fácil de escuchar eventos sin tener que modificar el cliente bitcoind. Solo puede conectar a un nodo de la red. Lo ideal es conectarlo a un nodo local, pero teóricamente puede conectarse a cualquiera.

Para realizar pruebas con Bitcoin Sniffer, lo he instalado en una máquina virtual y, como en otros casos que ya he comentado, lo he conectado al nodo que tengo instalado en la máquina real. Para ello he modificado el código de sniffer.py cambiando la dirección IP por defecto por la de mi equipo:

```
# Default Settings if no configuration file is given
settings = {
        "host": "192.168.1.11",
        "port": 8333,
        "debug": False
}
```

Para iniciar la aplicación solo hay que ejecutar:

```
$ ./sniffer.py
```

```
Bitcoin Network Sniffer v0.0.2
------------------------------------------------------------------------
Connecting to Bitcoin Node IP # 192.168.1.11:8333
Connected & Sniffing :)


- Valid TX: df7e59591f83762fbe9fa906d162858a597202007c3d63fba4d5dcca2fe5b69b

    To: 19Kszoy6nUPrQkd5rh9Zg5KxNco1ngdvG2 BTC: 0.00000001
    To: 1LFqoC8AnydKCFFKvsi1KkFL3Qapki9Kb5 BTC: 0.01104269

- Valid TX: fdc14f08a26bfa6c10938e4703ad5c22b23292882e044bd51e75f14e5143a310

    To: 1LM943uV9g9NurxaDNjg5sbzBtR8P1yYGz BTC: 0.01000000
    To: 12dyLXxBnXKcQC9NHfHKk36Fassm9R8AiU BTC: 17.88339007
    To: 1K7v3v25ffiy5kFgfEjEYZW9Y6YYbdMDYE BTC: 0.01458211

- Valid TX: 4253f7d885d79ad4d8f925f9bad7fea4fed0eca09ccc2dda024605b2cc8d3ed7

    To: 1HBwTme99Doufg8upiQcin2BoWbfrpsQnK BTC: 0.40657098
    To: 17LhMGALXT2Q2ZWNrVYzNRaDwsRFKbG5eT BTC: 21.72912902

- Valid TX: 101666da42df9bc93526365fd19b01df6f35f343424ef810a1aeee5b36db0659
```

II.E - Bitcoin Wallet

También existe una versión ligera de la billetera para Android llamada *Bitcoin Wallet* que se puede instalar en smartphones y en tablets. Entre las características de esta aplicación están:

- Visualización del saldo de la cartera en Bitcoin y en otras divisas.
- Envío y recepción de bitcoins a través de NFC, códigos QR o URL Bitcoin
- Libreta de direcciones para guardar las direcciones Bitcoin utilizadas habitualmente.
- Las transacciones se pueden llevar a cabo sin conexión, ejecutándose al recuperar la conexión.
- Notificaciones de sistema para la conectividad a la red Bitcoin y para la recepción de bitcoins.
- En cuanto a las características de la red, no usa cloud servers ni servidores web, sino que se conecta a la red p2p directamente. Podemos comprobarlo viendo que tiene abierto el puerto 8333 (a través de nmap por ejemplo). Además podemos observar que nos podemos conectar a este nodo (p.ej. usando: *bitcoinjs run –connect dir_IP_dipositivo_movil:8333*) tanto si está conectado a través de wifi a la red local, como si lo está a través de la red 3G.
- Evidentemente, Bitcoin Wallet no descarga la cadena de bloques completa ya que esto no sería factible en los dispositivos móviles debido a su gran tamaño. Para ello utiliza una implementación en java llamada bitcoinj que implementa un modo ligero con verificación de pagos simplificada.

Se puede encontrar más detalles de esta aplicación en la dirección:

https://code.google.com/p/bitcoin-wallet/

II.F - GUI Miner

II.F.I - CARACTERÍSTICAS.

GUIMiner es una interfaz gráfica para la minería de Bitcoins que proporciona una forma cómoda para manejar los “mineros” de Bitcoin. Es compatible tanto con GPU's ATI y NVIDIA, así como con la minería basada en CPU. Se puede usar tanto para la minería en solitario como para la minería en grupo, e integra una lista de grupos (o pools) de mineros para escoger.

La versión actual de GUIMiner es la 20121203, que se puede descargar desde:

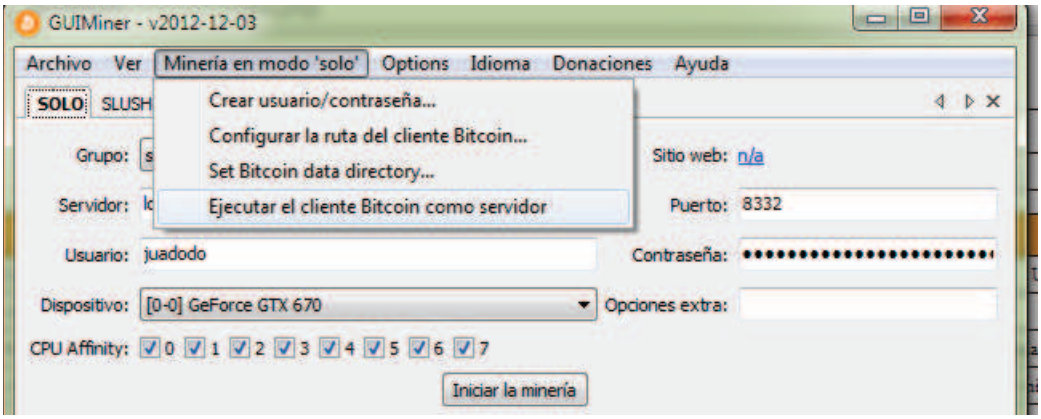
https://github.com/downloads/Kiv/poclbm/guiminer-20121203.exe

El aspecto de esta interfaz es el siguiente:

Para realizar una prueba del funcionamiento del programa, he estado probando por una parte la minería en modo 'solo' y la minería en grupo con uno de los grupos de minería que aparecen configurados en el programa: *slush's pool.*

II.F.II - MINERÍA EN MODO 'SOLO'.

Para realizar la minería en modo solo hay que crear el archivo ***bitcoin.conf*** (en caso de que no exista) y configurar como mínimo los parámetros ***rpcuser*** y ***rpcpassword***. Además, hay que ejecutar el cliente Bitcoin como servidor (se puede hacer desde el menú *Minería en modo 'solo'*).



Por último hay que seleccionar *solo* en la opción grupo e introducir en usuario y contraseña los mismos datos que hemos puesto en *rpcuser* y *rpcpassword.* Si el equipo en el que se está ejecutando BitcoinQt es el mismo, usaremos como Servidor localhost y como puerto el que tengamos en *bitcoin.conf* en el parámetro ***rpcport*** (si no hay ninguno, el puerto por defecto es el 8332).

Un último detalle es que si minero y servidor no están en el mismo equipo hay que configurar el parámetro **rpcallowip** en bitcoin.conf. Con la opción *rpcallowip=** se permiten las conexiones desde cualquier dirección IP, con: *rpcallowip=192.168.1.** se permiten conexiones desde la red local indicada, y con *rcpallowip=192.168.1.156* se permite el acceso solo desde un solo host.

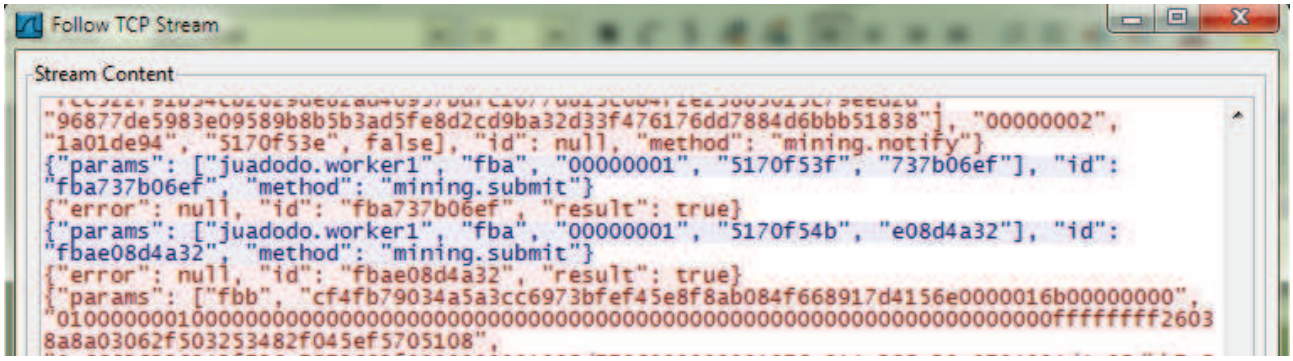
II.F.III - TRANSMISIÓN BÁSICA DE LOS DATOS EN LA MINERÍA EN GRUPO.

Lo primero es seleccionar un grupo o pool (las pruebas las he realizado con *slush's pool*) y a continuación introducir los datos del usuario y la contraseña que previamente deberemos haber configurado en la web del grupo. El mismo programa nos proporciona un enlace a las páginas web de los distintos grupos registrados.

Hay que tener en cuenta que esos datos no son los de la cuenta de minero, sino los datos del trabajador, ya que en caso de tener varios equipos disponibles para la minería podemos asociar varios trabajadores a la misma cuenta.

Una vez puesto en marcha la minería, comprobamos si se han abierto más puertos hacia la red bitcoin. Notar que la opción puerto que teníamos disponible en la minería en solitario ya no aparece. Con el comando *tasklist* o desde el administrador de tareas podemos comprobar el PID del proceso encargado de la minería (***poclbm.exe***) y seguidamente comprobamos las conexiones realizadas usando el comando ***netstat -n -a -o***, que nos muestra las conexiones establecidas en nuestro equipo y el PID del proceso asociado a ese conexión. De esta forma comprobamos que se esta usando el puerto 3333 para conectarse con el pool de minería.

Con el programa Wireshark podemos ver el contenido de lo que se está transmitiendo, la mayoría de lo cual parecen datos de hashes calculados, parámetros de configuración del trabajador minero (en mi caso se llama juadodo.worker1) y el resto serán las “tareas” que nos encargan desde el pool. Se puede ver una muestra de estos datos en la siguiente imagen:



II.F.IV - ESTADÍSTICAS DE LA MINERÍA EN GRUPO.

Actualmente cada bloque descubierto implica una ganancia de 25 BTC más las comisiones que se apliquen a las transacciones. Y teniendo en cuenta que un bitcoin tiene un valor de alrededor de 100 $, es una recompensa significativa para que mucha gente se dedique a la minería.

Gráfico de la evolución del precio del bitcoin. A mediados de Abril se llego al superar los 225$, aunque casi inmediatamente se desplomo a unos 80$.

Hay que tener en cuenta que esta recompensa es para una sola persona, pero es muy difícil de conseguir en solitario. Una opción distinta que asegura recompensas más estables es la minería en grupo. Aquí la recompensa de 25 BTC se reparte entre todos los mineros del grupo. Existen diferentes métodos de calcular el porcentaje que se lleva cada minero: DGM, proporcional, PPS, SMPPS, POT, Score, … Además suele ser habitual en la mayoría de los grupos de minería quedarse un porcentaje de las comisiones de las transacciones en concepto de mantenimiento del sitio.

En concreto, el grupo con el que he realizado pruebas (*slush's pool*) utiliza la siguiente formula para calcular las recompensas:

$$\frac{(25\ \text{BTC} + \text{comisiones del bloque} - 2\%\ \text{comisiones}) * (\text{shares encontrados por el usuario})}{(\text{total shares en la actual ronda})}$$

Las páginas web de los grupos de minería suelen mostrar toda una serie de datos y estadísticas sobre el estado del proceso de minería que llevan a cabo. Entre todos estos datos se suelen encontrar los siguientes:

- Datos sobre la ***cuenta del minero***: nombre, dirección de bitcoin donde recibir las

recompensas, la recompensa acumulada (tanto la confirmada como la estimada).

- ***Estadísticas sobre los trabajadores*****,** es decir, los equipos que tenemos dedicados a la minería de bitcoins. Aquí nos muestran datos como los bloques encontrados, shares actuales, Mhash/s, …
- ***Datos del grupo*** o pool y el estado de la minería en general: Cuando comenzó la actual ronda, es decir, cuando se empezó a buscar un bloque, dificultad del bloque (se va ajustando dinámicamente), recompensa estimada, …
- ***Historial de bloques encontrados***, se almacenan datos como el momento en que se ha encontrado un bloque, el total de shares que se han calculado sobre ese bloque, la recompensa en BTC, la validez (es decir, si ya se ha confirmado que ha sido realmente el pool el que ha encontrado el bloque o aún hay un número de confirmaciones pendientes). La siguiente tabla es un ejemplo del historial de bloques encontrado por el pool Slush.

Historia del bloque Bitcoin

Todas las horas están en UTC.

#	Bloque encontrado en	Duración	El total de shares	Your shares	Su recompensa BTC	Bloque #	Block value	Validez
17569	19/04/13 14:40	04:44:15	30587867	388	0.00032930	232113	25.20058629	97 confirmaciones pendientes
17568	19/04/13 09:56	00:24:12	2661482	39	0.00039348	232089	25.09070005	73 confirmaciones pendientes
17567	19/04/13 09:32	00:58:26	6304037	73	0.00019180	232087	26.01421547	71 confirmaciones pendientes
17566	19/04/13 08:33	01:05:26	6979514	85	0.00027908	232080	25.29409567	64 confirmaciones pendientes
17565	19/04/13 07:28	00:57:29	6071632	71	0.00010412	232071	25.33231061	55 confirmaciones pendientes
17564	19/04/13 06:30	01:11:58	7396406	86	0.00022882	232066	25.36642000	50 confirmaciones pendientes
17563	19/04/13 05:18	02:03:08	12266465	162	0.00033250	232059	25.31245000	43 confirmaciones pendientes
17562	19/04/13 03:15	06:23:36	37782193	394	0.00053451	232049	25.31240000	33 confirmaciones pendientes
17561	18/04/13 20:51	00:58:24	5699987	67	0.00039407	232025	25.01200000	9 confirmaciones pendientes
17560	18/04/13 19:53	02:09:27	12261909	168	0.00040576	232021	25.40652674	5 confirmaciones pendientes
17559	18/04/13 17:44	00:42:53	4003454	46	0.00029854	232010	26.23772591	confirmado

II.F.v - Blockchain

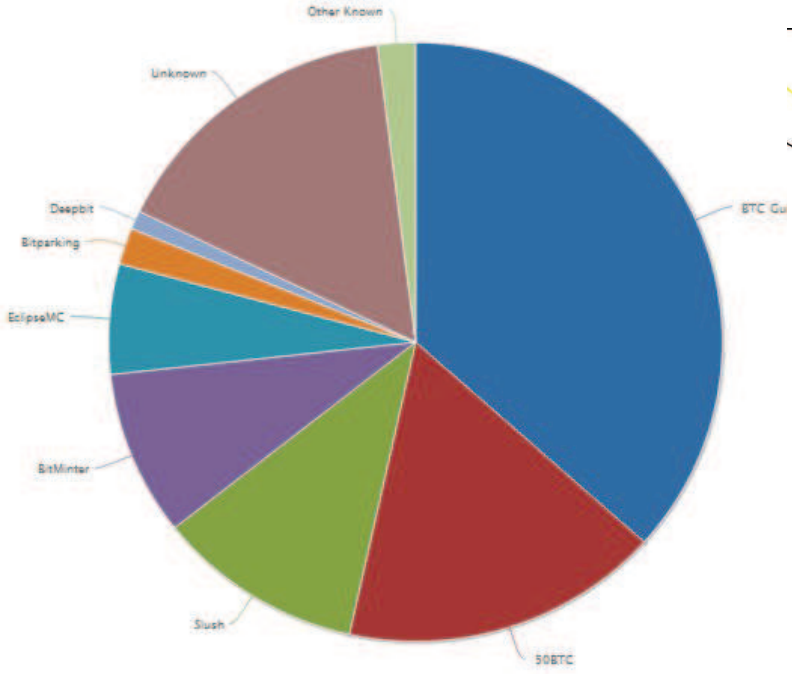
Dirección: http://blockchain.info/es/

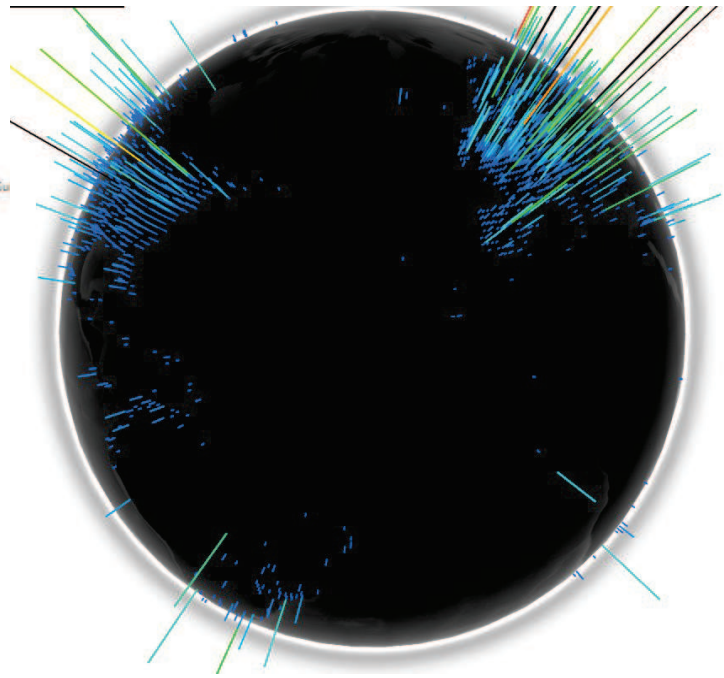
Está no es una aplicación, es un sitio web que ofrece una gran cantidad de datos sobre casi cualquier aspecto de Bitcoin. Todos estos datos nos servirán para contrastar/verificar los datos que se vayan obteniendo en el TFM.

Entre toda la información que contiene la web de blockchain, quizá la más interesante desde el punto de vista de este TFM sea que muestra las transacciones que se van realizando en tiempo real (ver imagen de la derecha) y una lista de los bloques minados recientemente.

También muestra una lista de todos los nodos conectados a blockchain.info, estadísticas de los pools de minería (ver imagen inferior izquierda), listas de bloques huérfanos, registros de transacciones extrañas, un gráfico que muestra los nodos conectados sobre un mapa (imagen inferior derecha).

Además, se pueden consultar una gran variedad de estadísticas y gráficos monetarios de Bitcoin.





ANEXO III - INSTALANDO LA APLICACIÓN BTCDONET

Requisitos para instalar la aplicación desarrollada en este TFM. En primer lugar es necesario instalar un **servidor de páginas web**, **PHP** y **MySQL server**. Para instalarlo de forma comoda instalé el paquete XAMPP.

En concreto he instalado la versión XAMPP 1.8.1 que incluye:

- Apache 2.4.3
- MySQL 5.5.27
- PHP 5.4.7
- phpMyAdmin 3.5.2.2
- FileZilla FTP Server 0.9.41
- Tomcat 7.0.30
- Strawberry Perl 5.16.1.1 Portable
- XAMPP Control Panel 3.1.0

También es necesario instalar la herramienta **nmap**. La versión que he instalado es la que incluye zenmap (aunque no es necesario instalar esta interfaz) y que se puede descargar desde http://nmap.org/download.html

Además para el envío de comandos **JSON** es necesario tener la extensión para PHP que implementa el formato de intercambio de datos JavaScript Object Notation (JSON) . Esta ya esta incluida (con unas pequeñas modificaciones) en el directorio de la aplicación.

También es necesario tener instalado **Python** para Windows y una versión de **pynode** (descrito en el anexo anterior), para que funcione la versión modificada de Bitcoin Sniffer (ya incluida dentro del directorio de la aplicación)

Además para eliminar los procesos creados de Bitcoin Sniffer, como no se podían detener a través del comando taskkill de Windows ni de las opciones disponibles para el control de procesos desde PHP, se ha incluido el comando **pskill.exe** dentro del directorio *sniffer* de la aplicación.

III.A - Ajustes de configuración de la aplicación.

Para que funcione la aplicación hay además que ajustar los siguientes parámetros de configuración dentro del archivo index.php. Tanto para la aplicación BTCdoNET como para BTCdoNET-RECORDER.

Para obtener datos de los nodos a los que se conecta el cliente bitcoin hay que configurar los

siguientes parámetros:

- $miIP, es la IP del equipo donde se va a ejecutar un cliente de bitcoin desde el que obtendré los datos.
- $puerto que utiliza el cliente de bitcoin del que obtenemos los datos, generalmente es el : 8333
- $directorio, directorio donde se encuentra el archivo de configuración del cliente de bitcoin bitcoin.conf.

Los siguientes parámetros sirven para realizar la conexión con la base de datos MySQL donde se almacena la información obtenida por la aplicación:

- $dir_MySQL, por defecto suele se localhost si la base de datos está en nuestro equipo, pero puede otra IP si la base de datos está en otro equipo. Como por ejemplo sucede en la versión limitada de la aplicación que he puesto on-line y que se explica en el siguiente punto.
- $user_MySQL, es el usuario de la base de datos de MySQL. Necesita ser un usuario con permisos suficiente para seleccionar e insertar datos en las tablas de la base de datos utilizada. En el caso de la aplicación on-line (ver siguiente punto) se usa un usuario que solo tiene permisos para ejecutar selects por seguridad.
- $pass_MySQL, password del usuario de la base de datos utilizado.
- $bd_MySQL, nombre de la base de datos de MySQL donde se guarda la información. Por defecto se llama bitcoin.

Además se necesita configurar una serie de parámetros para poder conectarnos a través de JSON con el cliente de Bitcoin. El contenido de estos parametros debe de coincidir con los que se han puesto en el archivo de configuración bitcoin.conf.

- $rpcdir, indica donde esta el archivo PHP que se utiliza para realizar la conexión, por defecto es "./json/jsonRPCClient.php". Este parámetro NO aparece en bitcoin.conf.
- $rpcuser y $rpcpassword, son el usuario y contraseña utilizados para conectarnos por JSON al cliente bitcoin.
- $rpcallowip son las direcciones a las que se permite conectarse. Si queremos dar acceso por ejemplo a la red local 192.168.1.0/24 hemos de configurar este parametro como “192.168.1.*”.
- $rpcport, es el puerto a través del cual se envían mensajes JSON al cliente, por defecto es el 8332.

En la siguiente imagen se muestra un ejemplo del contenido del archivo index.php con la configuración de estos parámetros:

```php
<?php

/*********************************************************************
        Parámetros para configurar la aplicación
*********************************************************************/

//IP del equipo donde se va a ejecutar bitcoin
$miIP="192.168.1.11";
// puerto de cuyas conexiones se extraera la IP
$puerto=":8333";
//Directorio donde esta bitcoin.conf
$directorio="C:\\Users\\Joan\\AppData\\Roaming\\Bitcoin\\";

//Parámetros de la conexión con la base de datos
$dir_MySQL="localhost";
$user_MySQL="root";
$pass_MySQL="";
$bd_MySQL="bitcoin";

//Parámetros para la conexión con Bitcoin-Qt a través de JSON
//Estos parámetros deben coincidir con los que se indican
//en el archivo bitcoin.conf
$rpcdir="./json/jsonRPCClient.php";
$rpcuser="juadodo";
$rpcpassword="BITCOINisAMAZING,amazingISbitcoin-->J04n4nt0n1.";
$rpcallowip="127.0.0.1";
$rpcport="8332";
```

III.B - Probando la aplicación en linea.

Siendo un poco engorrosa la instalación al depender de varias aplicaciones de terceros (Python, PHP, MySQL, nmap, pynode) al final he decidido crear una versión para poderla probar en línea desde la dirección http://juadodo.no-ip.org/bitcoin/

Para no permitir que cualquiera pueda borrar datos o manipularlos se ha utilizado el siguiente sistema. En una máquina virtual se ha instalado una copia de BTCdoNET, sin la parte que registra los nodos. El archivo index.php se ha modificado para que se conecte a la base de datos de la máquina real con solo permiso para usar consultas de selección sobre las tablas de bitcoin.

En la máquina real sí se está ejecutando la versión de BTCdoNET-RECORDER por lo que los datos a los que accedemos van a estar actualizados.