I. J. Computer Network and Information Security, 2018, 7, 36-43
Published Online July 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2018.07.04

Investigating and Analyzing Bitcoin Blockchain Protocol using Wireshark

Auqib Hamid Lone
Department of Computer Science and Engineering, NIT Srinagar, Jammu and Kashmir, India, 190006
E-mail: auqib92@gmail.com

Roohie Naaz Mir
Department of Computer Science and Engineering, NIT Srinagar, Jammu and Kashmir, India, 190006
E-mail: naaz310@nitsri.net

Received: 01 March 2018; Accepted: 07 June 2018; Published: 08 July 2018

Abstract—A bitcoin node needs to download the full block contents of the entire blockchain, before actually being able to send and receive transactions on bitcoin broadcast network, except simple payment verification clients which require only block headers and bloom filters to sync with others peers available on the network. Transactions/Blocks pass through a complex process at sender and receiver than it apparently looks to be. During transmission transactions/blocks are broken down into smaller chunks of data so that they can be carried on the wire. These chunks are given appropriate headers, encapsulated and then passed through several layers to reach the destination. In this paper we captured Bitcoin packets using Wireshark and deeply investigated and analyzed them. We investigated how bitcoin transaction/block messages work and what values and parameters are considered during this whole process.

Index Terms—Bitcoin, Blockchain, Wireshark.

I. INTRODUCTION

In recent years, there has been an immense growth of e-commerce across the world. Most of the internet transaction mediating systems like PayPal rely on the trusted third-Party mediators - banking institutions. These trusted third-party mediators are responsible for verification and authentication of e-transactions and thus provide a sense of assured security to its users. Unfortunately, all of these trusted third-party mediators have one severe disadvantage of having "Central Point of Failure". To overcome shortcomings of trusted third-party mediators, a concept of *Bitcoin: A Peer to Peer e-cash* was proposed by Satoshi Nakamoto in 2008 [1].

Bitcoin has revolutionized the whole financial transaction system, by introducing the very first fully decentralized digital currency, that is both secure and stable. Bitcoin can be defined in various ways; it's a protocol, a digital currency, and a platform. It is a combination of peer-to-peer network, protocols, and software that facilitates the process of creation and control of the digital currency.

The bitcoin network is a P2P network [2] where nodes exchange transactions and blocks. Wireshark can be used to visualize these transactions and blocks and thus can serve as an invaluable tool to get deep insight of the Bitcoin protocol. In this paper we capture the Bitcoin packets to deeply investigate and analyze them. We investigate how bitcoin transactions and block messages work, and what values and parameters are considered during this whole process.

The rest of the paper is organized as follows: In section II, the Bitcoin blockchain protocol is explained. In section III, basic setup methodology for experimentation is presented. In section IV, the bitcoin packets are investigated using Wireshark. Finally, the conclusion is given in section V.

II. BITCOIN BLOCKCHAIN PROTOCOL

There are many possible ways to define the Bitcoin, one way is to think of Bitcoin as a *protocol* governing the overall communication on the Bitcoin network, the other possible way is to think of it as a *digital currency* which is purely P2P electronic cash, allowing payments to be sent directly, bypassing any intermediary financial institutions. In more general sense Bitcoin represents a platform comprising of P2P network, protocols and a software that eases the creation and usage of digital currency named bitcoin.

Bitcoin uses Elliptic Curve Cryptography(ECC) [3,4]based on SECP256K1 standard. Private keys are a random 256 bit numbers, mostly encoded using Wallet Import Format(WIF). Public keys are basically x and y coordinates on a elliptic curve. Public keys can be presented in compressed or uncompressed form with 33 and 65 bytes of length respectively. A bitcoin address is actually the hash of a public key, computed as under [5].

Algorithm 1: bitcoin address generation
Input: Public Key,Version
Output: bitcoin address: 26-35 alphanumeric characters.

1 Calculate the SHA256 hash of Public key ;
2 Calculate RIPEMD-160 hash of output of the step1;
3 Key hash = Concatenation of the Version and output of step 2;
4 Calculate SHA256 two times of Key hash of step3;
5 Checksum = 1st 4 bytes of the output of step 4;
6 Concatenate Key hash with Cheksum;
7 Encoding the output of step 6 with Base58 encoding scheme;
8 Return output of step 7;

Version = 1 byte of 0 (zero); on the test network, this is 1 byte of 1

The heart and soul of Bitcoin is Blockchain: Technology that makes bitcoin a fully decentralized digital currency, which is both stable and secure. Blockchain can be defined as a immutable decentralized public ledger, which is both timestamped and ordered. Each block on the chain is identified by a hash and is linked to its previous block by referencing the previous block's hash.

A. Bitcoin Blockchain core Components

a. Block and Block Header

Block is one of the important and driving component of Bitcoin Blockchain. Table 1 and Table 2 presents the structure of a block and block header with size and brief description of each entry in the the block and block header respectively.

Table 1. Structure of a Block

Bytes	Name	Description
80	Block header	This includes fields from the block header.
variable	Transaction counter	The field contains the total number of transactions in the block, including the Coinbase transaction.
variable	Transactions	All transactions in the block.

Table 2. Structure of a Block Header

Bytes	Name	Description
4	Version	The block version number that dictates the block validation rules to follow.
32	previous block header hash	This is a double SHA256 hash of the previous block's header.
32	Merkle root hash	This is a double SHA256 hash of the Merkle tree of all transactions included in the block.
4	Timestamp	This field contains the approximate creation time of the block in the Unix epoch time format. More precisely, this is the time when the miner has started hashing the header (the time from the miner's point of view).
4	Difficulty Target	This is the difficulty target of the block.
4	Nonce	This is an arbitrary number that miners change repeatedly in order to produce a hash that fulfills the difficulty target threshold.

b. Bitcoin as a State Transition System

Bitcoin platform can be thought of as a state transition system, comprising of *state* and a *statet ransaction function*. *State* consists of ownership status of all bitcoins and *transition function* takes state and transaction as input and outputs a new state [6].

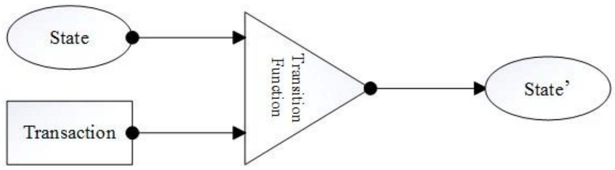

Fig.1. Bitcoin as a State Transition System.

The *state* in Bitcoin is the collection of all coins (technically, "unspent transaction outputs" or UTXO) that have been minted and not yet spent, with each UTXO having a denomination and an owner (defined by a 20-byte address which is generated as shown in algorithm 1). A transaction contains one or more inputs, with each input containing a reference to an existing UTXO and a cryptographic signature produced by the private key of the owner, and one or more outputs, with each output containing a new UTXO to be included in the state [7].

Transition function can be formally written as:
$APPLY\,(S,TX) \rightarrow S^{'}$ *or Error* defined as follows:

1. For each input in TX:

- If the referenced UTXO is not in S, return an error.
- If the provided signature does not match the owner of the UTXO, return an error.

2. If the sum of the denominations of all input UTXO is less than the sum of the denominations of all outputUTXO, return an error.
3. Return S' with all input UTXO removed and all output UTXO added.

c. Bitcoin Mining

Mining in Bitcoin is a process through which transactions within a block are validated and blocks are added to the blockchain. Mining function is performed by special nodes called as *miners*. Mining is resource intensive process in a way as enough CPU and battery power is spent for mining a block. Roughly one new block is created (mined) every 10 minute. Miners are rewarded with new coins if and when they create new blocks and are paid transaction fees in return of including transactions in their blocks. New blocks are created at an approximate fixed rate. Also, the rate of creation of new bitcoins decreases by 50%, every 210,000 blocks, roughly every 4 years.

The major advantage of mining is that it secures the bitcoin system by guarding against frauds and double spending. Algorithm 2 presents the steps required for successfully mining a block.

Algorithm 2: The Mining Algorithm

1. The hash of the previous block is retrieved from the bitcoin network;
2. Collect the set of potential transactions broadcast on the network into a block ;
3. Compute the double hash of the block header with a nonce and the hash from step 1 using SHA256 algorithm;
4. If the result of step 3 is lower than the current difficulty target then stop the process;
5. Result of step 3 is greater than the current difficulty target then repeat the process by incrementing the nonce;

d. Proof of Work

As name suggests it is the proof that enough computational resources have been spent to build the valid block. Proof of Work(PoW) involves searching for a value that when hashed, such as with SHA256 , the resulting hash begins with number of zero bits or is less than a specific target. The following equation sums up the Proof of Work requirement in bitcoin:

$$H(N||P_{hash}||Tx||Tx||...Tx) < Target$$

Where N is a nonce, P_{hash} is a hash of the previous block, Tx represents transactions in the block, and Target is the target network difficulty value. This means that the resulting hash should be less than the target hash value. The only way to find this nonce is the brute force method.

III. METHODOLOGY

The bitcoin network is a P2P network where nodes exchange transactions and blocks. Analyzing contents of transactions, blocks and messages exchanged by peers requires capturing them, while nodes communicate with each other. For experimentation purposes, we installed Bitcoin Core(Testnet) client running on Testnet3 network [8]. The reason for using Testnet is that it allows application developers or bitcoin testers to experiment, without having to use real bitcoins or worrying about breaking the main bitcoin chain. There are few notable differences between bitcoin mainnet and testnet which are summarized in table 3.

Table 3. Bitcoin Mainnet and Testnet

Mainnet	Testnet
The real Bitcoin network, with the block chain that everyone uses.	test network that runs in parallel with mainnet, except that the value of these coins is negligible.
Costs real money not only to buy bitcoins but also as transaction fees	Essentially free to acquire testnet bitcoins.
Bitcoin protocol works on TCP port 8333 for the main network	Works on TCP port 18333 for testnet.

For capturing Bitcoin packets, we used network protocol analyzer called Wireshark[9,10].Wireshark allowed us to visualize messages exchanged between peers and thus served as an invaluable tool to deeply investigate about the Bitcoin protocol contents. We used [11] website to recieve some free coins and then transfer some coins back to it and record the whole process for investigation with Wireshark.

IV. INVESTIGATING BITCOIN PROTOCOL USING WIRESHARK

Typically a full bitcoin node can perform four functions: wallet, miner, blockchain and network routing node. The first thing that a bitcoin core node does when it starts up is initiation of discovery of all peers. This is achieved by querying DNS seeds that are usually hardcoded into the bitcoin core client and are maintained by bitcoin community members. This lookup returns a number of DNS A records. The process of node discovery, block or header exchanges and other network functionality are summarized below:

1. Client sends a protocol message *Version* containing various fields such as version, services, timestamp, network address, nonce and some other fields.
2. Remote node responds with it's own version.
3. After exchanging *version* messages both nodes then exchange *verack* messages, indicating that connection has been established.
4. After connection establishment *Getaddr* and *addr* messages are exchanged to find about unknown peers. Nodes can test liveliness of connection by sending ping messages. Active nodes respond with pong messages.
5. Now node can start downloading blocks to sync with network using *GetBlock* and *GetData* messages. If the node already has all blocks fully synchronized then it listens for new blocks using the *Inv* protocol message.

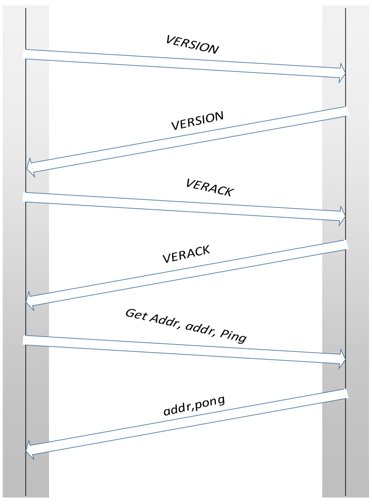

Fig.2. Bitcoin Node Discovery until version 0.9.3.

Until bitcoin core client version 0.9.3, a method called blocks-first was used for synchronization by nodes. The blocks-first method is very slow and some time takes days to complete synchronization. Consequently it was discontinued from bitcoin core client version 0.10.0. New initial block download(IBD) method named headers-first was introduced from client version 0.10.0. The headers-first method drastically reduces synchronization time to few hours. In headers-first method, when the client starts up, it checks whether the blockchain is fully synchronized already if the header chain is already synchronized; if not, which is the case the first time the client starts up, it requests headers from other peers using the getHeaders message. If the block chain is fully synchronized, it listens for new blocks via Inv messages, and if it already has a fully synchronized header chain, then it requests blocks using Getdata protocol messages. The IBD node also checks whether there is trade-off between headers and blocks: if headers are more in header chain; then it requests blocks by issuing the Getdata protocol message.

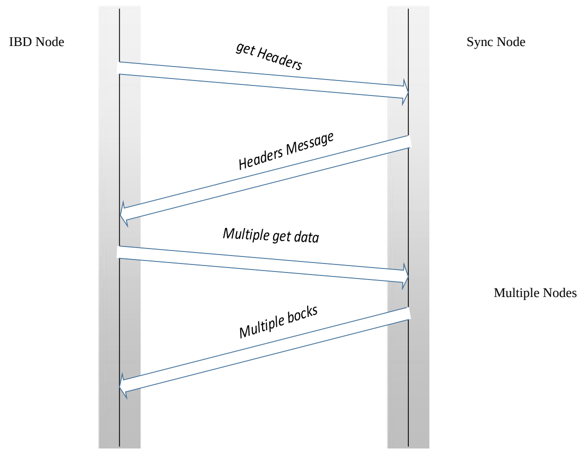

Fig.3. Header and Block Synchronization in Bitcoin Core Client>= 0.10.0.

Fig.4. Flow graph between two Peers on Bitcoin Network.

A. *Analyzing Bitcoin Protocol Messages in Wireshark*

Bitcoin dissector[12] plays an important role in analyzing the traffic and identification of Bitcoin protocol commands. Bitcoin dissector in Wireshark also provides valuable information such as the packet type, command name, and results of the protocol messages. Figure 4 provides protocol graph showing the flow of data between the two peers, providing clear picture on what messages are exchanged when a nodes starts up.

At present there are 27 types of protocol messages in total, but most likely they will increase over time as the protocol grows [4]. Table 4 lists the main ingredients of Bitcoin protocol messages with short description of each.

Table 4. Structure of a Bitcoin Protocol Message

Field Name	Bytes	Description
magic	4	Value indicating message origin network
command	12	String identifying messagecontent.
length	4	Length of payload in number of bytes
checksum	4	First 4 bytes of *sha*256(*sha*256(*payload*)).
payload	variable	The actual data.

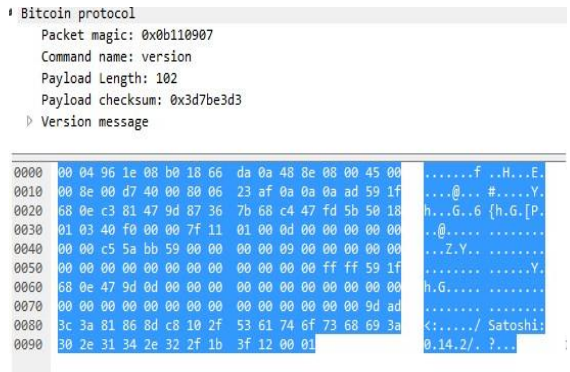

Fig.5. Bitcoin Protocol Message in Wireshark.

a. Version

This is the first message that a node sends out to the network, advertising its version and block count. The remote node then replies with the same information and the connection is then established.

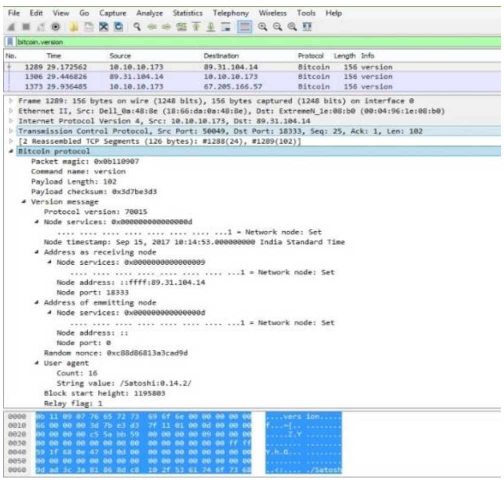

Fig.6. Version message in Wireshark.

Table 5. Payload contents of Version Message

Field Name	Bytes	Description
version	4	Identifies protocol version being used by the node.
services	8	bitfield of features to be enabled for this connection.
timestamp	8	standard UNIX timestamp in seconds.
addr_recv	26	network address of the node receiving this message.
addr_from	26	network address of the node emitting this message.
nonce	8	Randomly generated every time a version packet is sent and is to detect connections to self.
user_agent	variable	User Agent (0x00 if string is 0 bytes long).
start_height	4	The last block received by the emitting node.
relay	1	Whether the remote peer should announce relayed transactions or not

If version packet is accepted, *verack* packet will be sent.

b. Verack

This message is sent in reply to version. It typically consists of only a message header with the command string "veracK".

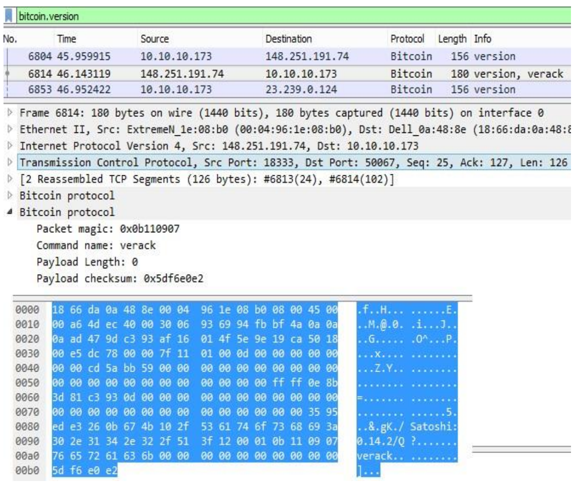

Fig.7. Veracк Message in Wireshark.

c. Addr

This message is used by bitcoin nodes to provide information on known nodes of the network. If nodes don't advertise for 3-hours they are automatically forgotten. Table 6 summarizes the contents of payload of *addr* message.

Table 6. Payload Contents of addr Message

Field Name	Bytes	Description
count	More than 1	Number of address entries (max: 1000).
address list	Multiple of 30	Address of other nodes on the network.

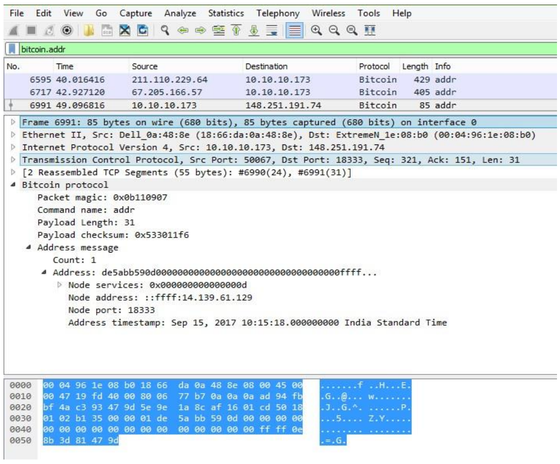

Fig.8. Addr Message in Wireshark.

d. Inv

This message is used by nodes to advertise their knowledge of one or more objects. It can be received unsolicited or in reply to *getblock* message. Payload of this message can have upto 50,000 entries corresponding to approximately 1.8 megabytes. Table 7 summarizes the contents of payload of *inv* message.

Table 7. Payload Contents of inv Message

Field Name	Bytes	Description
count	variable	Number of inventory entries.
inventory	Multiple of 36	Inventory vectors.

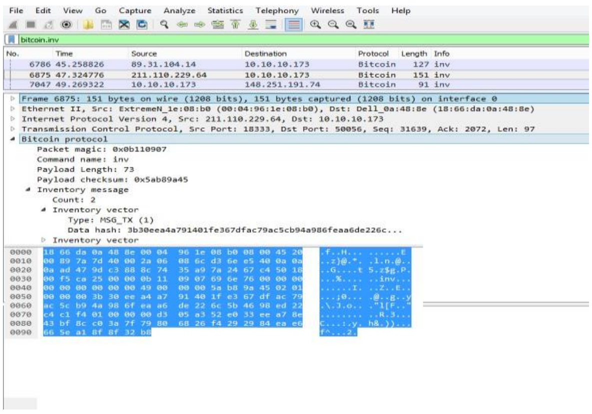

Fig.9. Inv message in Wireshark

e. Getdata

This message is used in response to *inv* for retrieving the contents of specific object. Usually *getdata* is sent after receiving *inv* packet, after filtering known objects. Payload of this message can have upto 50,000 entries corresponding to approximately 1.8 megabytes. Table 8 summarizes the contents of payload of *getdata* message.

Table 8. Payload Contents of Getdata Message

Field Name	Bytes	Description
count	variable	Number of inventory entries.
inventory	Multiple of 36	Inventory vectors.

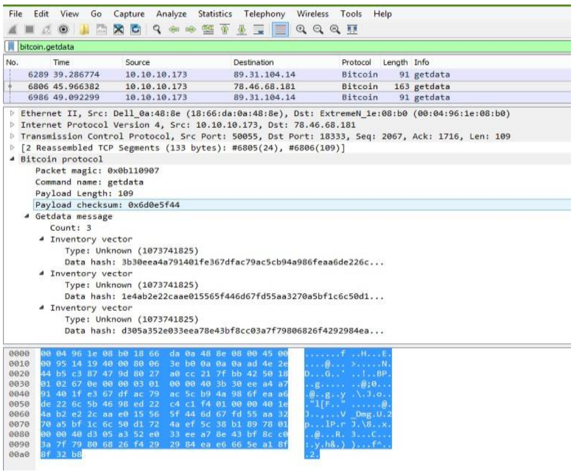

Fig.10. Getdata Message in Wireshark.

f. Notfound

This message is used in response to *getdata* when contents of specific object can't relayed. Table 9 summarizes the contents of payload of *notfound* message.

Table 9. Payload Contents of Notfound Message

Field Name	Bytes	Description
count	variable	Number of inventory entries.
inventory	Multiple of 36	Inventory vectors.

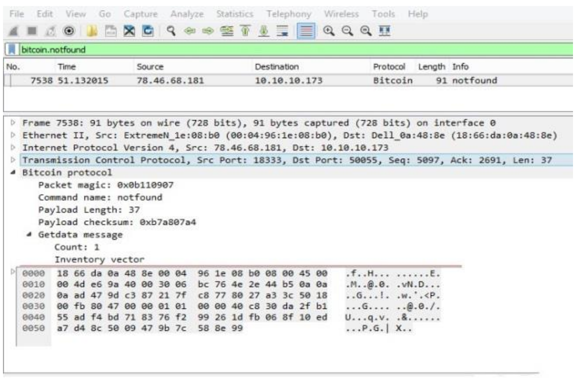

Fig.11. Notfound Message in Wireshark.

g. Getblocks

This message is sent by a node requesting other node to return *inv* packet containing list of blocks starting from the last known hash in the block locator object to stop_hash or 500 blocks whichever is earlier. Table 10 summarizes the contents of payload of *getblocks* message.

Table 10. Payload Contents of Getblocks Message

Field Name	Bytes	Description
version	4	The protocol version.
hash count	1 or more	Number of block locator hash entries.
Block locator hashes	32 or more	Block locator objects from newest block to genesis block.
hash _stop	32	Hash of the last desired block. Set 0 to get as many blocks possible (500).

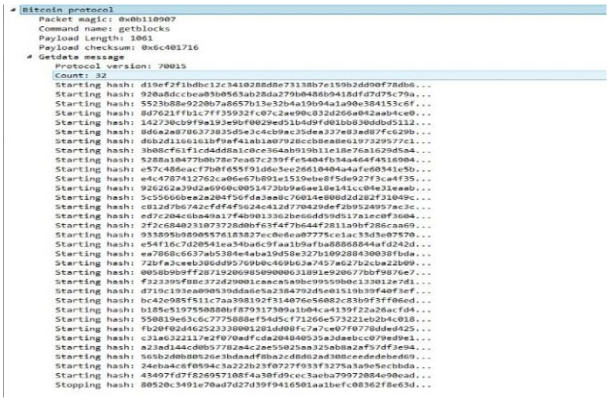

Fig.12. Getblocks Message in Wireshark.

h. Getheaders

This command is used by thin clients to quickly download the block chain to sync with the network. The response to *getheaders* message is a header packet containing the headers of the block starting right after the last known hash in the block locator object, hash stop or 2000 blocks, whichever comes first. Table 11 summarizes the contents of payload of *getheaders* message.

Table 11. Payload Contents of Getheaders Message

Field Name	Bytes	Description
version	4	The protocol version.
hash count	1 or more	Number of block locator hash entries.
Block locator hashes	32 or more	Block locator objects from newest block to genesis block.
hash _stop	32	Hash of the last desired block. Set 0 to get as many blocks possible (2000).

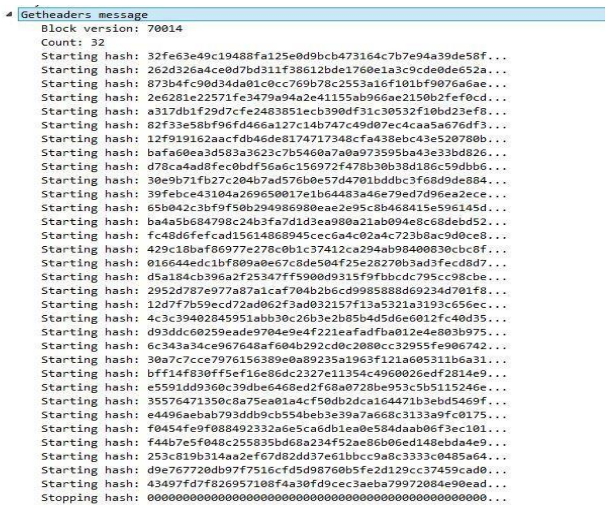

Fig.13. Getheaders Message in Wireshark.

i. Block

This message is send in response to *getdata* message requesting transaction information from a block hash. Table 12 summarizes the contents of payload of *block* message.

Table 12. Payload Contents of Block Message

Field Name	Bytes	Description
version	4	Signed Block version information.
prev_block	32	The hash value of the previous block this particular block references.
merkle_root	32	The reference to a Merkle tree collection which is a hash of all transactions related to this block.
timestamp	4	A Unix timestamp recording when this block was created.
bits	4	The calculated difficulty target being used for this block .
nonce	4	The nonce used to generate this block.
txn_count	variable	Number of transaction entries.
txns	variable	Block transactions, in format of "tx" command.

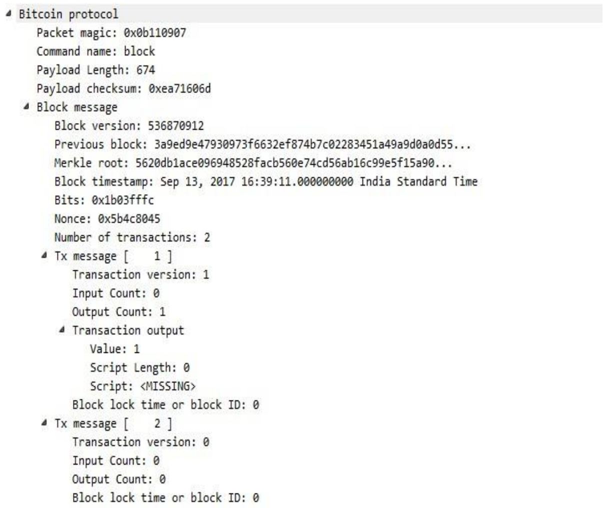

Fig.14. Block Message in Wireshark.

j. Headers

This message is send in response to *getheader* message returning desired block headers. Table 13 summarizes the contents of payload of *headers* message.

Table 13. Payload Contents of Header Message

Field Name	Bytes	Description
count	variable	Number of Block headers.
headers	More than 81 per header * No. of head-ers	Block header.
txns	variable	Block transactions, in format of "tx" command.

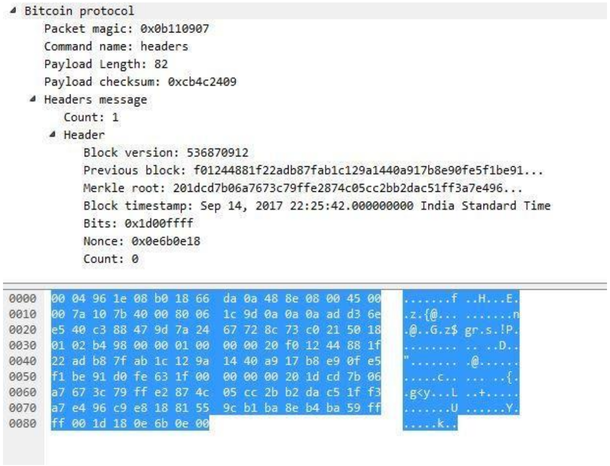

Fig.15. Headers Message in Wireshark.

k. Ping

This message is send by nodes for testing TCP/IP connection liveliness. Any connection error is presumed to be a closed connection and address is removed as current peer. Table 14 summarizes the contents of payload of *ping* message.

Table 14. Payload Contents of Header Message

Field Name	Bytes	Description
nonce	8	Random nonce.

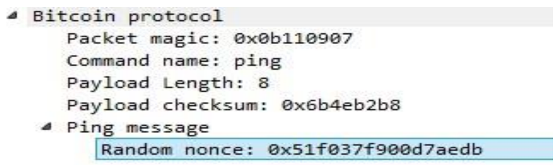

Fig.16. Ping Message in Wireshark.

l. Pong

This message is send in response to *ping* message by a node to prove its liveliness on network. Modern protocol versions, a *pong* response is generated by using a nonce included in the *ping* message. Table 15 summarizes the contents of payload of *pong* message.

Table 15. Payload contents of header message

Field Name	Bytes	Description
nonce	8	nonce from ping.

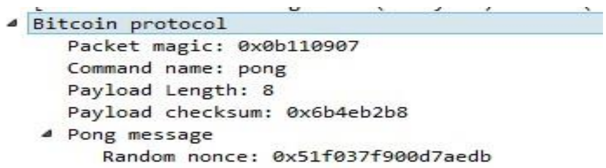

Fig.17. Pong Message in Wireshark.

V. Conclusion

In this research work, we have deeply studied and investigated what exactly happens in the back screen when new peer joins the bitcoin network and starts syncing with network by downloading the blockchain. Firstly, we introduced core components of Bitcoin and then by using computer network protocol analyzer tool i.e. Wireshark, we have captured Bitcoin packets that are exchanged when peers communicate and deeply observed and analyzed their contents to make the whole communication process easy to understand for the readers.

References

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.

[2] K.. A and J.. J. S, "Recent Replica Placement Algorithms in P2P Networks – A Review", *International Journal of Computer Network and Information Security*, vol. 5, no. 5, pp. 55-63, 2013.

[3] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, vol. 48, no. 177, p. 203, 1987.

[4] S. V, H. Shanavas.I, N. V and B. M, "Hardware Implementation of Elliptic Curve Cryptography over Binary Field", International Journal of Computer Network and Information Security, vol. 4, no. 2, pp. 1-7, 2012.

[5] “Protocol documentation.” [Online]. Available: https://en.bitcoin.it/wiki/Protocol_documentation

[6] V. Buterin, “Ethereum: A next-generation smart contract and decentralized application platform,” https://github.com/ethereum/wiki/wiki/White-Paper, 2013.

[7] G. Wood, “Ethereum: a secure decentralised generalised transaction ledger,” http://gavwood.com/paper.pdf, 2014.

[8] “Testnet.” [Online]. Available: https://en.bitcoin.it/wiki/Testnet

[9] “the wireshark wiki.” [Online]. Available: https://wiki.wireshark.org/

[10] A. Turner and A. Irwin, "Bitcoin transactions: a digital discovery of illicit activity on the blockchain", Journal of Financial Crime, vol. 25, no. 1, pp. 109-130, 2018.

[11] “Tp’s testnet faucet.” [Online]. Available: http://tpfaucet.appspot.com/

[12] “Wireshark display filter.” [Online]. Available: https://www.wireshark.org/docs/dfref/b/bitcoin.html

Authors’ Profiles

Auqib Hamid Lone I did my Bachelors in Information Technology and Engineering with distinction, post B. Tech my interest and passion towards information security took me into master’s and I completed M. Tech in Information Security & Cyber Forensics from Jamia Hamdard University, New Delhi with university rank. Currently I am Research Scholar at NIT Srinagar J&K. My areas of interest are Blockchain Technology, Cryptography, Network Security, Web Application Security and Digital Forensics.

Roohie Naaz Mir is professor and Head of Department Computer Science & Engineering at National Institute of Technology Srinagar, India. She received B.E. (Hons) in Electrical Engineering from University of Kashmir (India), M.E. in Computer Science & Engineering from IISc Bangalore (India) and PhD from University of Kashmir, (India). She is a Fellow of IEI, IETE, a senior member of IEEE, member of IACSIT and IAENG. She is the author of many scientific publications in international journals and conferences. Her current research interests include Network Security, Reconfigurable computing, mobile computing, security and routing in networks.

How to cite this paper: Auqib Hamid Lone, Roohie Naaz Mir,"Investigating and Analyzing Bitcoin Blockchain Protocol using Wireshark", International Journal of Computer Network and Information Security(IJCNIS), Vol.10, No.7, pp.36-43, 2018.DOI: 10.5815/ijcnis.2018.07.04