Blockchain-based NAT Management for 5G Age

Younchan Jung

School of Information, Communications
and Electronics Engineering
Catholic University of Korea
Bucheon-si, Gyeonggi-do, Republic of Korea
Email: ycjung@catholic.ac.kr

Marnel Peradilla

Computer Technology Department,
College of Computer Studies
De La Salle University - Manila
Manila, Philippines
Email: marnel.peradilla@dlsu.edu.ph

***Abstract*—Full deployment of IPv6 addressing fails because nowadays Network Address Translation (NAT) devices are commonly used to extend internal private addressing from the global public IP addressing. The existing phone system uses the vertical model to solve issues relating to the NAT and mobility management. Also, the horizontal model has been studied, where a centralized Software-Defined network (SDN) controller is in charge of handling network functions such as NAT and mobility management. The goal of this paper is to propose a blockchain-based NAT management (BNATM) scheme to overcome the limitation that both the horizontal model as well as the vertical model face in relation with NAT and mobility management. Our proposal focuses on the idea that, if we use the blockchain technologies, each peer can easily obtain the necessary parameters required to handle the complicated NAT and mobility management procedures. Finally, this paper analyzes the latency comparisons among the proposed BNATM scheme, existing vertical model and centralized controller-based horizontal model.**

Keywords–NAT management; SDN horizontal model; Blockchain; Blockchain-based management; Hash address; Transaction access.

I. INTRODUCTION

The explosive growth of the Internet during 1990s signaled the danger of IP address exhaustion and also created an instant demand on IP addresses. The Internet Engineering Task Force (IETF) simultaneously introduced the IPv6 and Network Address Translation (NAT) [1] [2]. However, full deployment of IPv6 addressing fails because of the NAT's widespread use. Currently, NAT devices are commonly installed at network edges to modify the addresses of packets crossing the NAT.

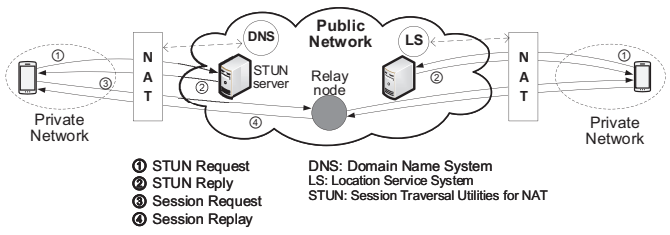

Figure 1. Vertical model for NAT management

NAT has one accessible public address which will be shared among End Nodes (ENs) inside the private network. NAT essentially extends internal addressing from the global IP addressing used over the Internet. NAT provides network resources to get over a shortage of the address space by mapping relatively public IP addresses to private IP addresses [3] [4]. However, the non-standardized characteristics of NAT cause traversal problems. Different NAT network products are available with different proprietary specifications. Therefore, NAT devices start to cause problems especially with the development of peer-to-peer applications [5]–[7].

Three issues are raised in implementing these application systems. First, a smart NAT management is needed in order to manage the private addressing of the local ENs in the private region and solve the NAT traversal issues. Second, the issue of mobility management, which focuses on ENs that use private IP addresses, should be solved. Most of the existing mobility management schemes only deal with the tracking of the location of the EN (that is, the addresses that are closely related to their locations) but not the use of private address [8] [9]. So, NAT management and mobility management functions need to collaborate in order that the application systems be operational. Lastly, for the joint operation of mobility management over the heterogeneous network, a significant portion of the existing work uses vertical model for network functions [10] [11]. As depicted in Figure 1, the existing vertical model for network functions for NAT management and mobility management has limitations in handling an integrated operation of heterogeneous network functions. The current trend for the NAT management and mobility management is to utilize the horizontal model of network functions [12]–[14]. The difference between the horizontal model and the vertical model depends on whether the processing of the network

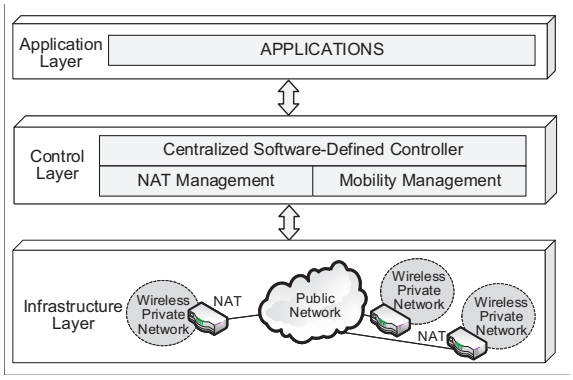

Figure 2. Horizontal model for 5G NAT management

 ISBN: 978-1-61208-644-6

functions takes place on the common control plane. In the horizontal model, all network functions are performed on the control plane, while in the vertical model, data processing and network function processing are performed in the same plane. Software-Defined Networking (SDN) is an emerging networking paradigm change from the vertical model to the horizontal model [15]. As shown in Figure 2, by separating the network's control plane from data plane, the control plane is implemented in a logically centralized controller. The centralized network controller in the control plane manages the intelligence and state of the entire network. However, the current network legacy devices, which operate based on the vertical model, are not yet ready to implement the horizontal model of network functions [16].

The goal of this paper is to develop a Blockchain-based Network Address Translation Management (BNATM) system which performs better in NAT management as well as mobility management than the horizontal model of network functions on the control layer. Figure 3 compares the BNATM system with vertical (or horizontal) model system from the viewpoint of naming and addressing for ENs. A BNATM address is a hash of the public key which is similar to the Bitcoin address in the blockchain-based payment system [17]. In terms of naming, the BNATM system uses the hash addresses differently from the existing system where the domain name is used. From the BNATM addressing point of view, public IP addresses and private IP addresses are used in the public network and a variety of private networks, respectively. However, in the IPv6-based horizontal model, addressing is based only on 128-bit public IP addresses.

		Identity Management	
	Network Service Protocol	**NAME**	**IP Address (Location Address)**
Vertical and Horizontal Model	IPv4 + NAT System	Domain Name	32-bit Public IP Address
		Domain Name	32-bit Private IP Address
	IPv6	Domain Name	128-bit Public IP Address
Proposed System	IPv4 + BNATM*	Hash Address	32-bit Public IP Address
		Hash Address	32-bit Private IP Address

BNATM: Blockchain-based NAT Management

Figure 3. Comparisons of naming and addressing

The BNATM structure is closed to the administrative control that SDN horizonal system operates with. However, a centralized Software-Defined controller on the control plane is in charge of the essential role in order to implement the application-based NAT and mobility management. The BNATM scheme utilizes one of the most innovative features of the blockchain where there is no central server running. It operates through a peer-to-peer network of connected computers or nodes. So, this idea gives significantly advantageous effects on NAT and mobility management by reducing the complexity of the system deployment and latency taken for the end-to-end session set up.

The rest of this paper is organized as follows. Section II proposes the blockchain-based architecture for NAT management. In Section III, this paper explains how to process a transaction to create a block and query/reply mechanism needed to access the transaction information from the blockchain. Section IV describes the improvement effects of the proposed management system. This paper concludes in Section V.

II. BLOCKCHAIN-BASED NETWORK ARCHITECTURE FOR NAT MANAGEMENT

A. Proposed network architecture

Together with explaining the BNATM network architecture (see Figure 4), the steps to run the network are as follows:

1) New transactions are sent to the nearest super node (SN). After a SN receive the transaction message, it broadcasts the message to all SNs. Each transaction message contains several data fields for NAT and mobility management, which will be described in the next section.
2) The SN collects new transactions into a block and performs on solving the proof-of-work for its block. Here, the SN maintains the full blockchain. There are two kinds of blockchains: full blockchain and block-header chain. The EN usually maintains the block-header chain. Later, when the EN needs a certain transaction information, it uses the query/reply mechanism by sending a query message to the nearest SN. Then, the SN searches the corresponding transaction data from the blockchain and returns the requested transaction information to the EN.
3) When an SN finds a proof-of-work, it broadcasts the block to all SNs and ENs.
4) SNs accept the block only if all transactions in it are valid.
5) SNs imply their acceptance of the block by working on creating the next necessary block in the chain, using the hash of the accepted block as the previous hash. SNs will always keep working on extending it. The main role of the EN is to update its NAT-related information by pushing it into the blockchain.
6) ENs accept the new block and extend the next necessary block header in the chain. This means that every EN maintains the block header chain rather than the full blockchain.

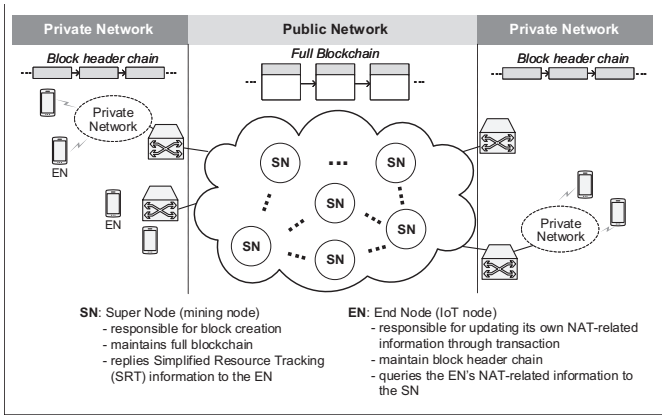

Figure 4. Proposed network architecture for Blockchain-based NAT management scheme

B. Time to get in the blockchain

An SN is responsible for the following functions:

- maintains the entire blockchain to store the entire transaction history,
- verifies incoming transactions by checking digital signatures and confirming the validity of the transaction,
- creates a block using recently collected valid transactions,
- finds a valid nonce to create a valid block header (the proof-of-work part) and,
- hopes that its created block is accepted by other nodes and not defeated by a competitor block created by other SNs.

If the proof-of-work is well designed, this price will be a minor inconvenience (like a short delay) for legitimate ENs but an economic deterrent to attackers of the service. Here, we define the user's waiting time from the moment a new transaction is announced to the network until the transaction successfully gets in the blockchain as Time-to-Get-in-Blockchain (TGinB). It is easy to adjust the average TGinB value by controlling the block creation difficulty. Our BNATM scheme adjusts this difficulty to target 5 seconds between blocks. The period of 5 seconds for TGinB means that it only takes 5 seconds for the blockchain to provide the NAT and mobility control to a certain EN since it moves and joins the new private network.

C. Obtaining public NAT address from the DHCP reply

Private IP addresses must be configured automatically for new ENs that moved from one network to another. Dynamic Host Configuration Protocol (DHCP) enables this entire process to be managed centrally. The DHCP server maintains a pool of private IP addresses and leases an address to any DHCP-enabled EN when it starts up on the network.

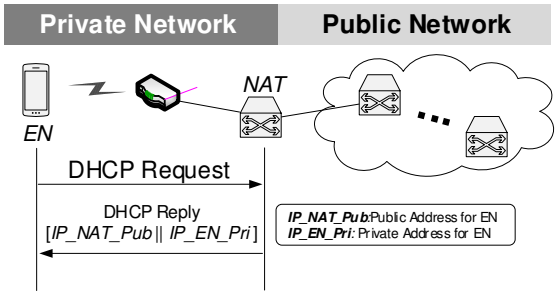

Figure 5. Obtaining public NAT address during private address assignment stage

A DHCP-enabled EN, upon accepting a lease offer, receives a valid private IP address for the private network to which it is currently connecting. There are additional parameters that a DHCP server is configured to assign to ENs. Some examples are router configuration (default gateway), Domain Name System (DNS) servers, and DNS domain name. In the proposed BNATM scheme, instead of using the DNS domain name, the NAT address, which is one accessible public address that will be shared among ENs inside the private network, is included in those parameters DHCP server offers. Figure 5 shows that the DHCP reply message contains the offered private address and the NAT address which will be used as the EN's source address when its packet enters the public network.

D. NAT port number as a function of the private address and port number

When EN sends a packet using its source private IP address and port number (IP_EN_Pri : $Port\_EN\_Pri$) to destination IP address and port number (IP_Dest : $Port\_Dest$), the NAT creates a map for EN's private address and port number (IP_EN_Pri : $Port\_EN\_Pri$) by assigning public $IP\_NAT_{EN}\_Pub$ and $Port\_NAT_{EN}\_Pub$ as public address and port number, respectively. So, incoming packets from [IP_Dest : $Port\_Dest$] destined to [$IP\_NAT_{EN}\_Pub$: $Port\_NAT_{EN}\_Pub$] are forwarded to [IP_EN_Pri : $Port\_EN\_Pri$]. As depicted in Figure 6, the BNATM scheme requires the important condition that $Port\_NAT_{EN}\_Pub$ should be derived from the hash function of IP_EN_Pri and $Port\_EN\_Pri$. EN is aware that NAT devices use the NAT port assignment function of **H16** where the first 16 bits are taken from the hash value.

E. Hash address used in BNATM scheme

Currently, the Long Term Evolution (LTE) communication system uses telephone numbers to identify each user while Voice over Internet Protocol (VoIP) applications use email addresses or domain names. However, in this paper, we propose to use the address derived from the public key to identify either a user or thing, that is, the hash address. A hash address is a hash of the Elliptic Curve Cryptography (ECC) public key. The hash address is the public part of a public-private cryptographic key. The private part of the key is under the control of the user. For example, when an EN moves and changes its private address, the EN creates a new transaction which contains its hash address and sends the transaction to the network. At this moment, the EN also uses its private key to sign the transaction, which results in the signature.

III. TRANSACTION PROCESS TO CREATE A BLOCK AND TRANSACTION ACCESS FROM BLOCKCHAIN

The wallet of EN monitors its state changes, such as private IP address changes. When any changes are found, the EN creates a new transaction and forwards it to the SNs. Then

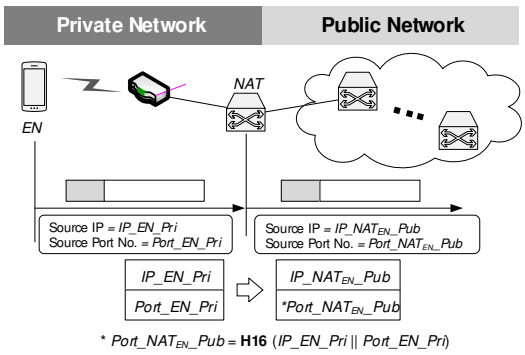

Figure 6. Public NAT port number determined as a function of EN's private IP address and port number

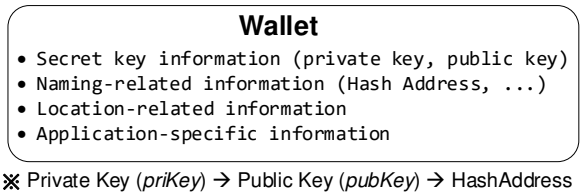

Figure 7. BNATM wallet information

SNs gather the transactions and compete to create the new block, which contains all the transactions after the previous block. Once an SN succeeds to create a block, it forwards the block to the other SNs and ENs. All the SNs maintain the full blockchain. On the other hand, when an EN receives the latest block, it updates the block header chain because the EN maintains chains of block header information excluding the body part of a block. When the SN receives the query from a certain EN, it searches the latest Tx information for the EN from the blockchain and replies to the EN with the information. The full details from transaction creation to the use of transaction information will be discussed in the following subsections.

A. Wallets in BNATM scheme

As shown in Figure 7, it is important for a BNATM user to have some knowledge of how a BNATM wallet software works. The wallet contains the following information:

- Secret key: private key ($priKey$), public key ($pubKey$)
- Naming-related: hash address ($HashAddress$)
- Location-related: IP_EN_Pri, $IP\_NAT_{EN}\_Pub$
- Application-specific: $Port\_NAT_{EN}\_Pub$, $AudioPort\_NAT_{EN}\_Pub$, encoder

The tasks performed by the wallet software also include:

- generates the corresponding public key ($pubKey$) and the hash address ($HashAddress$),
- updates its own location information that is, current private IP address (IP_EN_Pri) and current public NAT address ($IP\_NAT_{EN}\_Pub$) and,
- updates the necessary information for applications.

B. BNATM Transactions

EN's latest state information resides in the user's wallet. Their history is stored into a distributed database called the blockchain. Unlike centralized SDN controller, the blockchain stores a secure list of all transactions. A BNATM transaction is defined as the EN's state information during the period of dynamically assigned private IP address. So, the transaction change rate is the same as private IP address change rate. This means that EN updates its transaction when it moves and obtains a new private IP address. The transaction consists of Transaction Input (TxIn) and Transaction Output (TxOut). The TxIn contains the signature and the public key computed from the EN's private key which creates the transaction. The first field of TxOut contains the hash address that identifies the owner of this transaction.

As illustrated in Figure 8, the TxOut holds three types of fields:

1) Naming-related field: the $HashAddress$ is used for the purpose of searching a certain transaction from the blockchain. $TimeStamp$ is used to find the latest transaction for a given $HashAddress$. It is because among a series of transactions for a certain EN, only the latest transaction residing in the blockchain contains valid state information for the EN.
2) Location-related field: current public NAT address ($IP\_NAT_{EN}\_Pub$), which is important for NAT management and mobility management, indicates the EN's current location for the life of the transaction. The life will expire when the next transaction is issued.
3) Application-specific field: information such as application port number, audio port number and encoder type are included.

Once the EN creates a transaction (Tx) at the circumstance of location change, it sends the new transaction to the network. The first SN in the network that receives the Tx verifies the sent Tx if it is a valid Tx. If the Tx is correct, the SN relays it to other SNs in the network. Figure 9 explains the Tx verification process. To verify that a Tx is valid, an SN follows these steps:

- The script engine evaluates the $<scriptSig>$ of the $TxIn$. This $<scriptSig>$ just places two pieces of data into the stack, those are $<sig>$ and $<pubKey>$.
- The protocol now evaluates the $<scriptPubKey>$ of the $TxOut$ in the previous Tx. **OP_Duplicate** is a command that duplicates the last element of the stack, $<pubKey>$ of the $TxIn$ in the new Tx.
- Then, the **OP_HASHAddress** command computes the $HashAddressIN$ from the last element $<pubKey>$ on the stack.
- The command of **Place_HashAddress** places $<HashAddressOUT>$ onto the stack. This hash

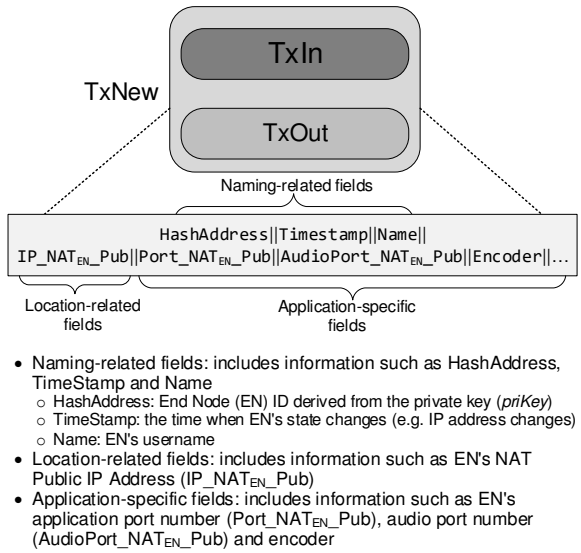

Figure 8. BNATM transaction architecture

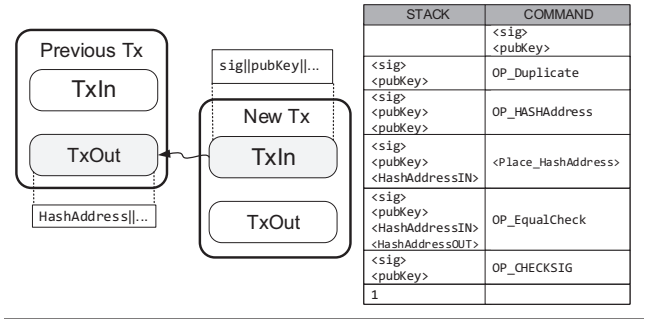

Figure 9. Transaction verification process

address is extracted from the $TxOut$ in the previous Tx.

- The next command is **OP_EqualCheck**. This command checks that the last two elements of $<HashAddressIN>$ and $<HashAddressOUT>$ are equal. If they are not, the Tx is tagged as invalid. After this verification, the two elements are withdrawn from the stack.
- The last command, **OP_CHECKSIG** checks that the Tx signature of $<sig>$ is correct. First, it hashes New Tx and checks that $<sig>$ is the correct signature for this hash. If the signature is correct, the Tx is valid. Otherwise, the Tx is rejected.

C. Blockchain and Proof-of-work

The blockchain is a distributed database holding all the BNATM transactions and keeping a secure list of all the transactions. The EN software that uses the blockchain has to send a query to an SN and receive the corresponding reply for a certain transaction. So, the SN is always ready to parse the blockchain to extract the relevant Tx information. This Tx information returned from the SN is used for the NAT and mobility management.

The blockchain uses proof-of-work to secure the distributed database. An attacker wishing to change the blockchain would have to apply a computational power equivalent to all the computational power spent from that point in time to the present. The blockchain is an ever-growing series of blocks where a certain block has a link to its previous block. Each block contains a group of new Txs created by the ENs. New Txs in the network are collected into a block which is appended to the blockchain. The mining SN is responsible for creating a new block. The scope of this paper does not include the selection protocol of the mining SN. Note that old blocks are never removed from the blockchain, thus the blockchain can only increase in length. The new block is secured with a partial hash inversion proof-of-work.

As shown in Figure 10, each block includes a group of valid Txs and block header information of the hash of the previous block, timestamp, a nonce and the root hash of all Txs. All the Txs in the block body are leaves in the Merkle Tree. Each Tx is hashed and hashes are hashed together to form a binary tree of hash pointers. So, the block header contains two hashes. One is related to the hash of the previous block and the other is the top hash in the binary tree of hash pointers, that is, the root hash. The nonce in a block solves the partial hash inversion problem. That is, the nonce is a number such that

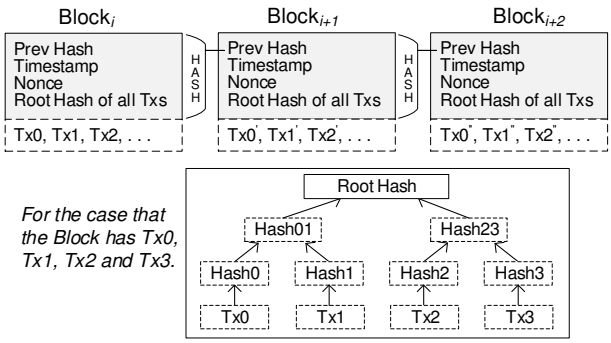

Figure 10. BNATM Blockchain

the hash of the entire block (including the nonce) starts with a certain number of zero bits. So, the block mining difficulty can be controlled by increasing the number of starting zero bits in the hash. The target block mining difficulty can be adjusted to control the TGinB period. This paper assumes that BNATM blocks are generated every 5 seconds. The target of 5 seconds means that on average it takes 5 seconds for an EN to be able to accept the session raised from the other ENs since the EN moves and joins the new private network. Reducing the average TGinB period increases the effective number of blocks updated globally per second, that is, block-creation rate. Our challenge is also to increase the block-creation rate as much as possible. This issue is beyond the scope of this paper.

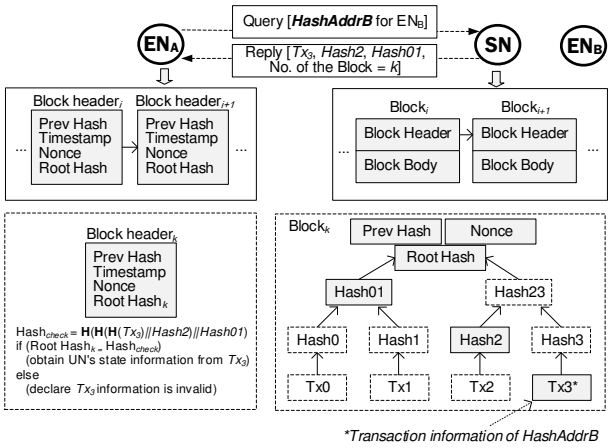

Figure 11. Transaction information access using query/reply mechanism

D. Query/reply mechanism to access transaction information

ENs only keep the block header chain while each SN maintains full blockchain. As shown in Figure 11, when an EN needs the transaction information for a particular hash address, that is, $HashAddrB$, it sends a Query to the nearest SN. Then, the SN parses the corresponding transaction $Tx3$ (it is assumed that $Tx3$ is the latest Tx for the hash address $HashAddrB$) from the blockchain and sends a Reply message to EN. The Reply message contains $Tx3$, $Hash2$, $Hash01$ and No. of the Block. When EN receives these pieces of information, it checks the validity of $Tx3$. It first calculates the

$Hashcheck$, that is, $\mathbf{H}(\mathbf{H}(\mathbf{H}(Tx3)||Hash2)||Hash01)$ where **H** means the hash function. Then, using the block number information contained in the Reply message, EN extracts the $RootHash_k$ at the $Blockheader_k$ in the block header chain. Lastly, the EN compares the extracted $RootHash_k$ to the calculated $Hashcheck$. If the two values are equal, the EN obtains the reliable Tx information for the $HashAddrB$. Otherwise, the EN declares the $Tx3$ information is invalid.

IV. BLOCKCHAIN-BASED NAT MANAGEMENT OPERATION AND IMPROVEMENT EFFECTS

A. Blockchain-based session establishment through NAT and mobility management

In Session Initiation Protocol-based (SIP) VoIP call operation, an end user sends SIP requests to initiate a session. Figure 12 shows a series of steps to complete a session set up between two ENs where they are located within the public network. This means that both of them use public IP addresses. Here, each EN changes its location dynamically. This dynamic feature of the EN requires to include the name and location resolution procedures. So, the DNS and Location Search (LS) system need to be involved to cause the latency problem. Because of the involvement of these two procedures, the vertical model, which passes through a series of these procedures, suffers from a relatively large amount of latency to complete a call set up.

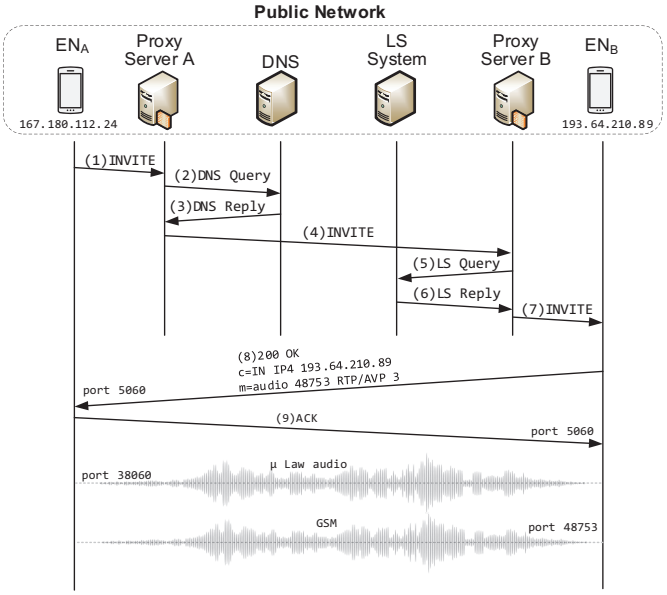

Figure 12. Existing SIP-based VoIP call operation

Figure 13 shows the proposed BNATM-based VoIP call operation. Here, we deal with the case that two ENs are located behind NAT devices. EN_A as well as EN_B belong to the private network. EN_A with the $HashAddrA$ wants to establish a session with EN_B with the $HashAddrB$. EN_A needs to obtain EN_B's state information. Then, EN_A sends (a) QUERY message which contains $HashAddrB$ to the nearest SN. When an SN, which has the full blockchain information, receives the QUERY, it seeks the corresponding Tx for $HashAddrB$. The SN sends back (b) REPLY message containing the Tx information for EN_B. This query/reply mechanism was explained in Subsection III. The query/reply procedure of ((a) and (b)) enables EN_A to obtain EN_B's state information: $HashAddrB$, Timestamp, Name, $IP\_NAT_{ENB}\_Pub$, $Port\_NAT_{ENB}\_Pub$, $AudioPort\_NAT_{ENB}\_Pub$ and encoder type. Here, EN_A resolves the current location of EN_B. Then, EN_A sends (c) INVITE message to $IP\_NAT_{ENB}\_Pub$. This INVITE message contains EN_A's hash address of $HashAddrA$. NAT_A translates the EN_A's private IP address and port number as $IP\_NAT_{ENA}\_Pub$ and $Port\_NAT_{ENA}\_Pub$. When NAT_B receives the packet, it translates the destination IP address and destination port number as $IP\_EN_B\_Pri$ and $Port\_EN_B\_Pri$. When EN_B receives the INVITE message from EN_A, it extracts the EN_A's hash address of $HashAddrA$. Now, the EN_B sends (d) QUERY message which contains the $HashAddrA$ to the nearest SN. When an SN receives the QUERY, it seeks the corresponding Tx for the $HashAddrA$. The SN sends back (e) REPLY message containing the Tx information for EN_A. The transaction access procedure of (d) and (e) enables EN_B to obtain EN_A's state information: $HashAddrA$, Timestamp, Name, $IP\_NAT_{ENA}\_Pub$, $Port\_NAT_{ENA}\_Pub$, $AudioPort\_NAT_{ENA}\_Pub$ and encoder type. At this moment, EN_B sends a Binding Request message toward its NAT device (NAT_B). Acknowledgement for this Binding Request is not necessary. The purpose of the Binding Request is to force the NAT_B to create a mapping entry of $[AudioPort\_NAT_{ENB}\_Pub : AudioPort\_EN_B\_Pri]$. At this moment, EN_B sends (f) 200 OK message to EN_A. After EN_A receives 200 OK message, it sends the Binding Request toward NAT_A. Similarly, the Binding Request from EN_A enables the NAT_A to create a mapping entry of $[AudioPort\_NAT_{ENA}\_Pub : AudioPort\_EN_A\_Pri]$. Finally, EN_A send (g) ACK message to EN_B. As a result, each side can reach the agreement on other session parameters such as audio encoder and others. Then, bidirectional session traffic travels through the established audio channels.

As shown in Figure 13, our blockchain-based session establishment scheme easily solves the problem of handling complex issues of NAT and mobility management. This advantage results from the fact that each peer can obtain the necessary parameters for peer-to-peer session establishment via simple query/reply mechanism between an EN and its nearest SN.

B. Improvement effects of BNATM scheme to complete network management

This paper calls the existing vertical model shown in Figure 12 and BNATM model in Figure 13 as "Vertical Model" and "BNATM Model", respectively. The following assumptions have been made to perform the comparative analysis for NAT and mobility management with respect to total latency to complete this network management between EN_A and EN_B, where each of them are located in different domains.

- The vertical model operates with the public IP addresses for ENs. On the other hand, the BNATM model operates with private addresses for ENs.
- Three types of delays exist, that is,
  1) T_I: intra-domain delay caused in intra-domain links,

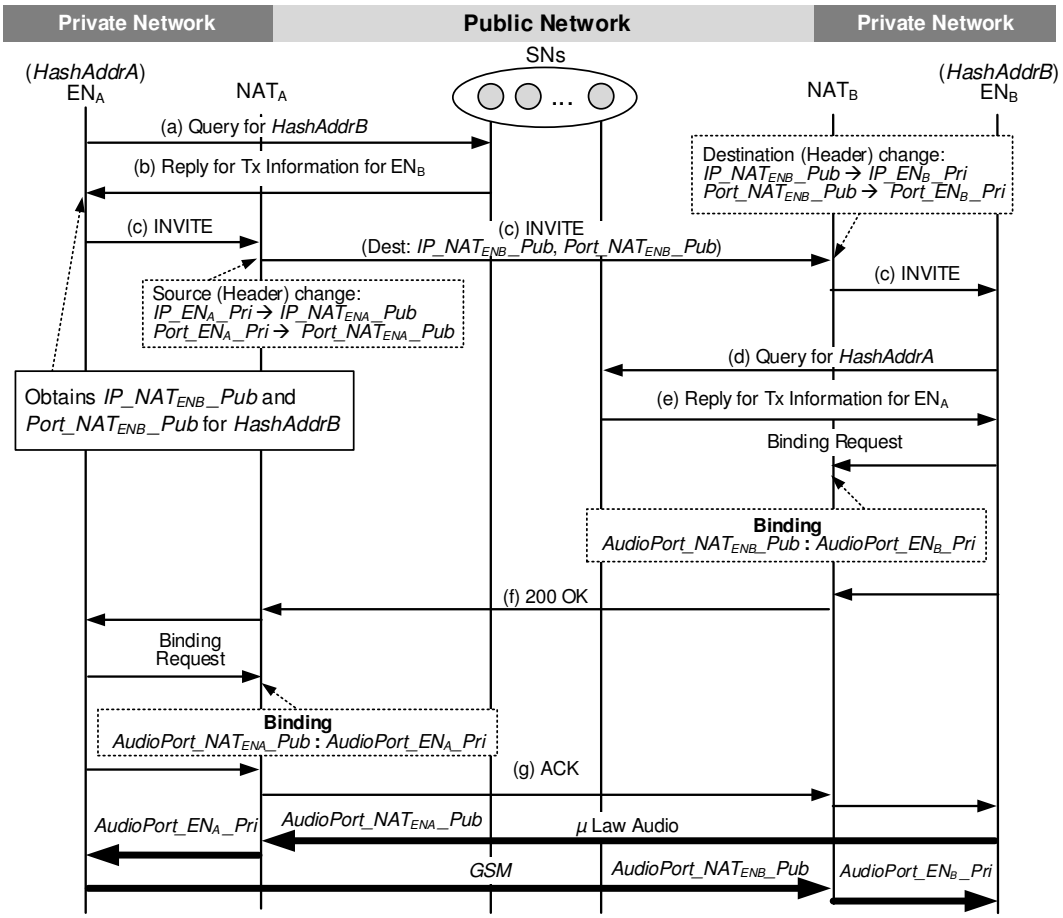

Figure 13. BNATM-based VoIP call operation

2) T_{II}: end-to-end delay caused in end-to-end path,
3) T_{III}: delay caused to collaborate with the distributed servers, which are spread in inter-domain regions.

- $T_{II} = 5T_I$ and $T_{III} = 10T_I$.

We summarize the comparison of the vertical model and BNATM model in Table I. The most interesting point is that the BNAT model does not have Type III delay components. This means that in the BNATM system, there is no need to collaborate with the query/reply procedures of (a), (b), (d) and (e) in Figure 13 to agree on necessary parameters to solve the issues relating to NAT and mobility management. Table I shows that, to reach an agreement on those parameters, the BNATM model requires $19T_I$ compared to existing SIP phone system's $57T_I$. It is very surprising that the BNATM system performs better by 300% than the existing vertical model. Recall that we assume the vertical model operates with the public IP addresses while BNATM model with private addresses for ENs. So, we argue that this 300% improvement on latency is lower bound because the vertical model excludes all the steps necessary for NAT management.

The 300% improvement on latency results from the simple query and reply mechanism to obtain parameters necessary to set up a session to the other side. When a source EN wants to establish a session to destination EN, the source EN sends a query to its neighbor SN to obtain the Tx information for the destination EN. When the SN receives the query from the source EN, it searches the latest Tx information for the destination EN from the blockchain and replies the information to the source EN. As shown in Figure 13, the reply messages of (b) and (e) enable to easily obtain the information of [$IP\_NAT_{ENB}\_Pub$, $Port\_NAT_{ENB}\_Pub$, $AudioPort\_NAT_{ENB}\_Pub$, encoder type, $\cdots$] and [$IP\_NAT_{ENA}\_Pub$, $Port\_NAT_{ENA}\_Pub$, $AudioPort\_NAT_{ENA}\_Pub$, encoder type, $\cdots$], respectively.

C. Performance tradeoff among vertical model, horizontal model and proposed BNATM system

Mid-call mobility management implies the handover process to provide seamless connection when an EN moves to a new network during an on-going session. Mid-call mobility management requires strict conditions on latency to maintain session quality when a handover occurs. According to the existing studies for latency comparisons for vertical and horizontal mobility management models [12], the vertical model yields a latency of 2,850 milliseconds for pre-call mobility management, assuming that the intra-domain delay of T_I is 50 milliseconds. Mid-call mobility solutions that are based on vertical model such as MIPv4 and MIPv6 produce latency performance of 268 milliseconds and 1,128 milliseconds, respectively. Here, the vertical model assumes

the ENs only use public IP addresses and operate without NAT devices. However, our BNATM system assumes that ENs are assigned with private IP addresses and operating behind NAT devices. In the BNATM model, a pre-call mobility management needs the latency of 950 milliseconds and a mid-call mobility management requires an average latency of 5 seconds.

In the proposed BNATM system, the blockchain extends a new block every 5 seconds. This is the reason why the BNATM system suffers from relatively high latency compared to the vertical model, especially in the case of the real-time network function of mid-call mobility management. Note again that the pre-call mobility management latency improves by 300% in the BNATM system compared to the vertical model. Such significant improvement inevitably needs to pay the price of real-time management issues such as the mid-call mobility management cases. As a result, the proposed BNATM system will show better performance over most of the network functions except for the real-time mid-call mobility management.

TABLE I. COMPARISON OF BNATM AND SIP PHONE SYSTEM

	BNATM Model (Proposed BNATM system)	**Vertical Model** (Existing SIP phone system)
Delay components	Type I: (a), (b), (d), (e) Type II: (c), (f), (g) Type III: None	Type I: (1), (7) Type II: (4), (8), (9) Type III: (2), (3), (5), (6)
Latency	$4T_I + 3T_{II} = 19T_I$	$2T_I + 3T_{II} + 4T_{III} = 57T_I$

$Note: T_{II} = 5T_I$ and $T_{III} = 10T_I$

V. CONCLUSION

Existing phone systems, such as the SIP-based VoIP call system, use the vertical model to solve issues related to the NAT and mobility management. As one candidate for future network architecture, the SDN horizontal model has been explored to control network functions such as NAT and mobility management. However, it is very difficult for the centralized SDN controller to replace all existing vertical model-based distributed servers, which are spread in inter-domain regions.

The goal of this paper was to propose a blockchain-based NAT management system to overcome the limitations that both the horizontal model and the vertical model face in solving issues related to NAT management as well as mobility management. Our idea focuses on the fact that, if we use blockchain technologies, each peer can easily reach agreement on the necessary parameters required to handle NAT and mobility management procedures. It is because our BNATM system works without either existing distributed servers in the vertical model or network controller in the horizontal model.

In this paper we proved that, from the latency viewpoint, the BNATM system performs better by 300% than the existing vertical model. As a result, the proposed BNATM system will show better performance over most of the network functions except for the real-time control cases such as mid-call mobility management.

ACKNOWLEDGMENT

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2017R1A2B4006086).

REFERENCES

[1] P. Srisuresh and G. Tsirtsis, "Network Address Translation - Protocol Translation (NAT-PT)," RFC 2766, Feb. 2000. [Online]. Available: https://rfc-editor.org/rfc/rfc2766.txt, accessed February 1, 2018

[2] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 8200, Jul. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8200.txt, accessed February 1, 2018

[3] R. Ghafouri, A. Ashrafi, and B. V. Vahdat, "Security consideration of migration to IPv6 with NAT (Network Address Translation) methods," in 2015 23rd Iranian Conference on Electrical Engineering, May 2015, pp. 746–749.

[4] S. Kalwar, N. Bohra, and A. A. Memon, "A survey of transition mechanisms from IPv4 to IPv6; Simulated test bed and analysis," in 2015 Third International Conference on Digital Information, Networking, and Wireless Communications (DINWC), Feb 2015, pp. 30–34.

[5] P. Leppaho, N. Beijar, R. Kantola, and J. L. Santos, "Traversal of the customer edge with NAT-unfriendly protocols," in 2013 IEEE International Conference on Communications (ICC), June 2013, pp. 2933–2938.

[6] Y. Wang, S. Xu, J. Wang, Y. Xue, J. Fu, and B. Hu, "Research of NAT traversal based on RTP relay server under mobile internet environment," in 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), vol. 01, Dec 2015, pp. 1370–1374.

[7] W. K. Jia, G. H. Liu, and Y. C. Chen, "NAT-Aware Peer Grouping and Chunk Scheduling for Mesh-Pull P2P Live Streaming Systems," in 2015 IEEE 39th Annual Computer Software and Applications Conference, vol. 2, July 2015, pp. 387–392.

[8] C. E. Perkins, "IP Mobility Support for IPv4, Revised," RFC 5944, Nov. 2010. [Online]. Available: https://rfc-editor.org/rfc/rfc5944.txt, accessed February 1, 2018

[9] D. B. Johnson, J. Arkko, and C. E. Perkins, "Mobility Support in IPv6," RFC 6275, Jul. 2011. [Online]. Available: https://rfc-editor.org/rfc/rfc6275.txt, accessed February 1, 2018

[10] Y. Jung and M. Peradilla, "Host mobility management using combined MIPv6 and DNS for MANETs," in 2013 International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT), Aug 2013, pp. 100–105.

[11] M. B. Yassein, S. Aljawarneh, and W. Al-Sarayrah, "Mobility management of Internet of Things: Protocols, challenges and open issues," in 2017 International Conference on Engineering MIS (ICEMIS), May 2017, pp. 1–8.

[12] M. Peradilla and Y. Jung, "Combined Operations of Mobility and NAT Management on the Horizontal Model of Software-Defined Networking," in Proceedings of the International Conference on Internet of Things and Cloud Computing, ser. ICC '16. ACM, 2016, pp. 31:1–31:10.

[13] Y. Jung, M. Peradilla, and A. Saini, "Software-defined naming, discovery and session control for iot devices and smart phones in the constraint networks," Procedia Computer Science, vol. 110, 2017, pp. 290 – 296, 12th International Conference on Future Networks and Communications (FNC 2017).

[14] K. Tantayakul, R. Dhaou, and B. Paillassa, "Impact of sdn on mobility management," in 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), March 2016, pp. 260–265.

[15] J. H. Cox, J. Chung, S. Donovan, J. Ivey, R. J. Clark, G. Riley, and H. L. Owen, "Advancing Software-Defined Networks: A Survey," IEEE Access, vol. 5, 2017, pp. 25 487–25 526.

[16] S. Azodolmolky, P. Wieder, and R. Yahyapour, "Performance evaluation of a scalable software-defined networking deployment," in 2013 Second European Workshop on Software Defined Networks, Oct 2013, pp. 68–74.

[17] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," URL: https://bitcoin.org/bitcoin.pdf, accessed February 1, 2018.

 ISBN: 978-1-61208-644-6