The Internet Protocol Journal

November 2017 Volume 20, Number 3

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

From the Editor 1

A Blockchain Tutorial............. 2

In Defence of NATs 25

Fragments 34

Thank You............................ 36

Call for Papers...................... 38

Supporters and Sponsors 39

You can download IPJ back issues and find subscription information at:
`www.protocoljournal.org`

ISSN 1944-1134

From The Editor

Publication of this journal is made possible by numerous individuals and organizations. Every year in late August we initiate a sponsorship renewal campaign, and the total funding determines our publication frequency for the following year. This *November* edition will be the third and final issue in 2017, but we hope to return to our regular quarterly publication schedule in 2018. We still need more individual and corporate sponsors, so please make a donation at **`http://tinyurl.com/IPJ-donate`** or ask your company to sign up for a sponsorship.

In our series of articles on emerging technologies we turn to *Blockchain*, a term that is now found in mainstream news outlets. We asked Bill Stallings to give us an introduction to this technology and consider some of its applications, such as *Bitcoin*.

Network Address Translation (NAT) has been widely deployed in both home and corporate networks for many years. Since the IPv4 address space is largely depleted, NATs provide an easy option for creating local networks that use private address space as defined in RFC 1918, and communicate with the public Internet through a single IP address. There are many technical problems associated with NATs, some of which have been described in other articles in this journal. This time, Geoff Huston provides an opinion piece "In Defence of NATs."

After many years on the ICANN Board, Steve Crocker has finished his term, but has agreed to continue serving on our Editorial Advisory Board. Thank you, Steve! In other news, the *Internet Society* recently celebrated its 25th anniversary, while the *Internet Engineering Task Force* (IETF) celebrated its 100th meeting. For more information on these events, visit **`http://isoc.org`** and **`http://ietf.org`**

As mentioned in our previous issue, if you have a print subscription to this journal, you will find an expiration date printed on the back cover. For the last couple of years, we have "auto-renewed" your subscription, but now we ask you to log in to our subscription system and perform this simple task yourself. This process will ensure that we have your current contact information, as well as delivery preference (print edition or PDF download). For any questions, e-mail us at: **`ipj@protocoljournal.org`**

—Ole J. Jacobsen, Editor and Publisher
`ole@protocoljournal.org`

A Blockchain Tutorial

by William Stallings, Independent Consultant

Blockchain is a recently-developed distributed digital implementation of the hardcopy transaction *ledger* that has been used throughout the world for centuries. Businesses and other organizations use ledgers in a variety of applications, such as to determine ownership, establish valuations, and document liabilities. The most common ledger applications are for tracking and chronologically recording transactions that involve an exchange of value between parties. Another common use of ledgers is to record birth and death certificates.

Blockchain first came to public notice as the technology that supports the virtual currency *Bitcoin*. And while the interest in Bitcoin has tended to wax and wane, the interest in blockchain continues to grow[1].

Distributed Ledgers

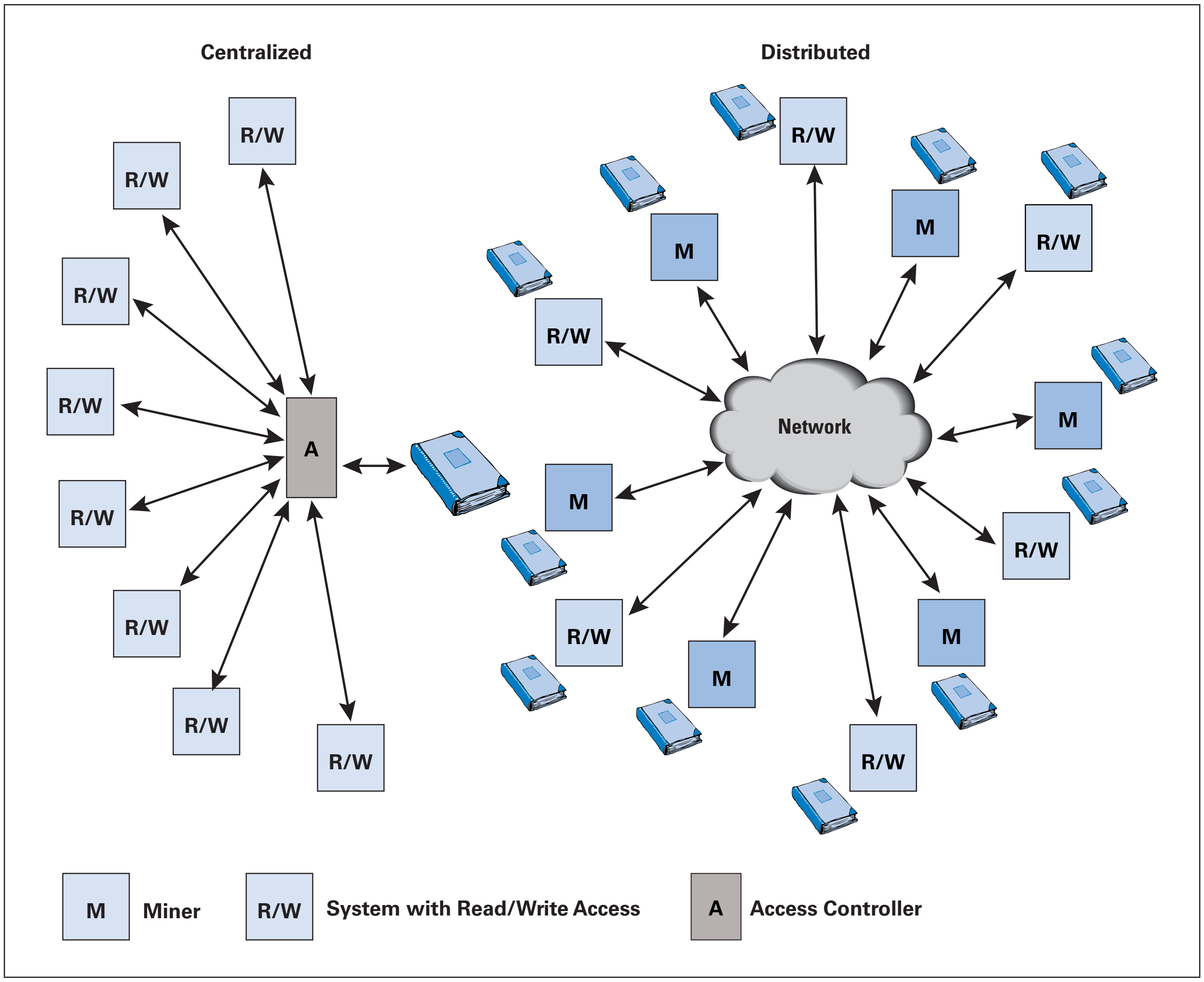
In the business context, a ledger, or *general ledger,* is defined as a central repository of the accounting information of an organization in which the summaries of all financial transactions during an accounting period are recorded. Also called the *book of final entry,* it provides all the data for preparing financial statements for the organization. As mentioned, the ledger can be used to record other types of transactions as well.

In the common business environment, a digital ledger is stored in a central server, and distributed access is provided with read and/or read/write privileges (Figure 1). To assure security, there is some sort of access control mechanism that authenticates users, enables secure access, and enforces access restrictions (for example, read-only). For a system with ongoing transactions and a heavy volume of read and write access, the central server model can be inefficient.

An alternative is a *secure distributed ledger,* which consists of an expandable list of cryptographically signed, irrevocable records of transactions that is shared by a distributed network of computers. Subject to network time delays, every participant has the same copy of the ledger. Each participant may propose a new transaction to be added to the ledger and when consensus that the transaction is valid is reached, it is added to the register.

Trust in a distributed ledger involves two concepts. First, security protocols and mechanisms, generally based on *Public-Key Cryptography,* ensure that the creator of each transaction is authenticated and validated. Transaction creators prove they are entitled to make a transaction by satisfying the particular conditions associated with this application. Meeting these conditions almost always involves the use of a secure digital signature.

Figure 1: Centralized and Distributed Ledgers



Second, a consensus mechanism is used in which computers on the network check each other to ensure records are consistent. In blockchain, this latter mechanism is implemented by systems called *miners*. Their job it is to determine that each new addition to the ledger is valid and consistent with previous entries. When the miners achieve consensus on a new entry, it is permanently added to the ledger.

Consider the use of a distributed ledger to record financial transactions or some other type of transaction that involves the exchange of value. Each transaction is a signed message that creates new outputs (transfer of value to another) while consuming old inputs (transfer of value from the transaction maker). For financial applications, each transaction is the digital equivalent of a paper check, and represents a promise by the payer to transfer control of a given amount of value to another party. The same funds or other value can be sent to only one party. An attempt at double spending, by creating two transactions that consume the same inputs, is prevented by the use of digital signatures and the trust mechanisms of the distributed ledger.

The Gartner Research document "What CIOs Should Tell the Board of Directors About Blockchain"[2] lists the following benefits of using secure distributed ledgers:

- Civilians and computerized agents govern the economic and transaction infrastructure, which is global in scale, peer-to-peer, self-regulating, secure, and reliable.
- A decentralized, shared history of activity, obligations, rights, and records ensures transparency and certainty.
- Fine-grained and diverse (not just monetary) value exchange occurs directly between participants on a network, at lower cost and higher speed compared to legacy systems.
- The system is open to everyone, both public and private, but control and openness can be customized.
- Ownership and rights are recognized broadly. Value can be natively created and exchanged with no double spending or repudiation of transactions. The system guarantees proof of existence, process, and asset provenance.
- Embedded business logic enables dynamically self-executing smart contracts linked to diverse assets.
- Distributed autonomous organizations acting as full-fledged legal entities can execute transactions with no human intervention.

General Concept

In essence, blockchain is a data structure that makes it possible to create a digital ledger of transactions and share it among a distributed network of computers. After a block of data is recorded on the blockchain ledger, it is computationally infeasible to change or remove it. When someone wants to add to the ledger, participants in the network, all of which have copies of the existing blockchain, run algorithms to validate the proposed transaction. If a majority of nodes agree that the transaction looks valid—that is, identifying information matches the history of a blockchain—then the new transaction will be approved and a new block added to the chain. The transaction is fulfilled or executed only when it has been approved for addition to the blockchain. In contrast, in a typical computerized ledger scheme, transactions are submitted to a trusted central party that is responsible for validating the transactions and posting them in the ledger.

Blockchain provides a distributed public ledger containing transactions that are governed and maintained by specific protocols through consensus of the nodes participating in its network. The ledger consists of a linear time-sequenced chain of blocks, with each block containing one or more transactions. Each block is connected to the previous block via a hash (tamper-proof digital fingerprint). On the blockchain, users can observe transactions that have occurred, so they know which outputs are available for spending and which ones have been consumed.

Each block in the blockchain represents, in effect, the claim by someone on the network that the transactions contained inside the block are the first ones to spend the inputs involved, and therefore any transaction in the future that attempts to spend the same inputs should be rejected as invalid.

The term "blockchain" is used interchangeably to describe both the blockchain network (network of nodes) and the distributed ledger (chain of blocks). It offers a way for users who may not know or trust each other to create a record of *who* transacts *what* that will compel the assent of everyone concerned.

The blockchain ledger is not housed on a single privileged server. Rather, it is a shared data structure in which every node (user) on the network has the same copy of all other nodes (subject to propagation time delays) and can read any transaction in the ledger.

Blockchain Structure

A blockchain is a linear sequence of blocks used to store transactions. Each block contains one or more related transactions, and the blocks are ordered in increasing time sequence. Thus, each block represents a set of events that have occurred over a given time frame that is subsequent to the preceding block in the chain and prior to the following block in the chain. Users with application access to the chain can read any transaction in the sequence and can add a new block at the end of the sequence.

As shown in Figure 2, each block has a unique predecessor and successor. A block is added only at the newer, or higher end of the chain. As will be shown, there may temporarily be a branching structure as the chain grows. An essential element of blockchain is that each block is linked to its preceding block using a cryptographic algorithm. The scheme is designed such that it is computationally feasible to add a new block to the end of the chain but computationally infeasible to replace a block interior to the chain or to insert a new block between two existing blocks in the chain. After a block is added to the chain, it is read-only. Figure 3 shows the blockchain operation in general terms.

Figure 2: Block Chaining Concepts

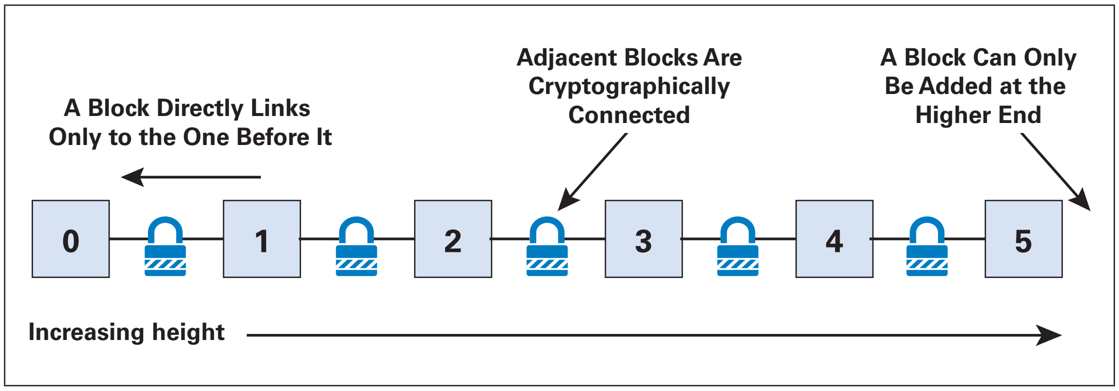
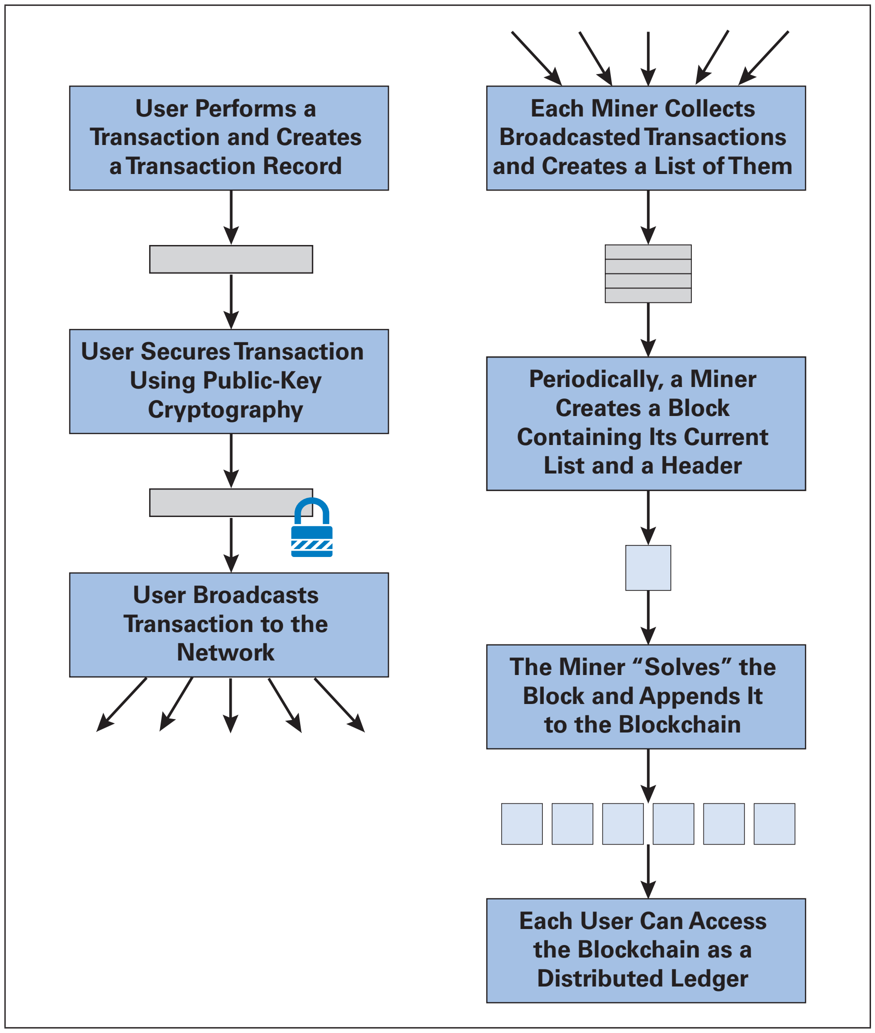

A Block Directly Links Only to the One Before It
Adjacent Blocks Are Cryptographically Connected
A Block Can Only Be Added at the Higher End
0
1
2
3
4
5
Increasing height

Figure 3: Basic Blockchain Logic


User Performs a Transaction and Creates a Transaction Record
User Secures Transaction Using Public-Key Cryptography
User Broadcasts Transaction to the Network
Each Miner Collects Broadcasted Transactions and Creates a List of Them
Periodically, a Miner Creates a Block Containing Its Current List and a Header
The Miner "Solves" the Block and Appends It to the Blockchain
Each User Can Access the Blockchain as a Distributed Ledger

The exact structure of a block may vary from one application to another. Table 1 shows the typical block format. Each block begins with a "magic number" that uniquely identifies this chain. For Bitcoin, the magic number is 0xD9B4BEF9. This number is followed by a *blocksize* field that specifies the total number of bytes in the remainder of the block. Next comes the header, consisting of multiple fields. Finally, the block contains a transaction counter (≥1) followed by one or more transactions. The internal format of each transaction is application-dependent.

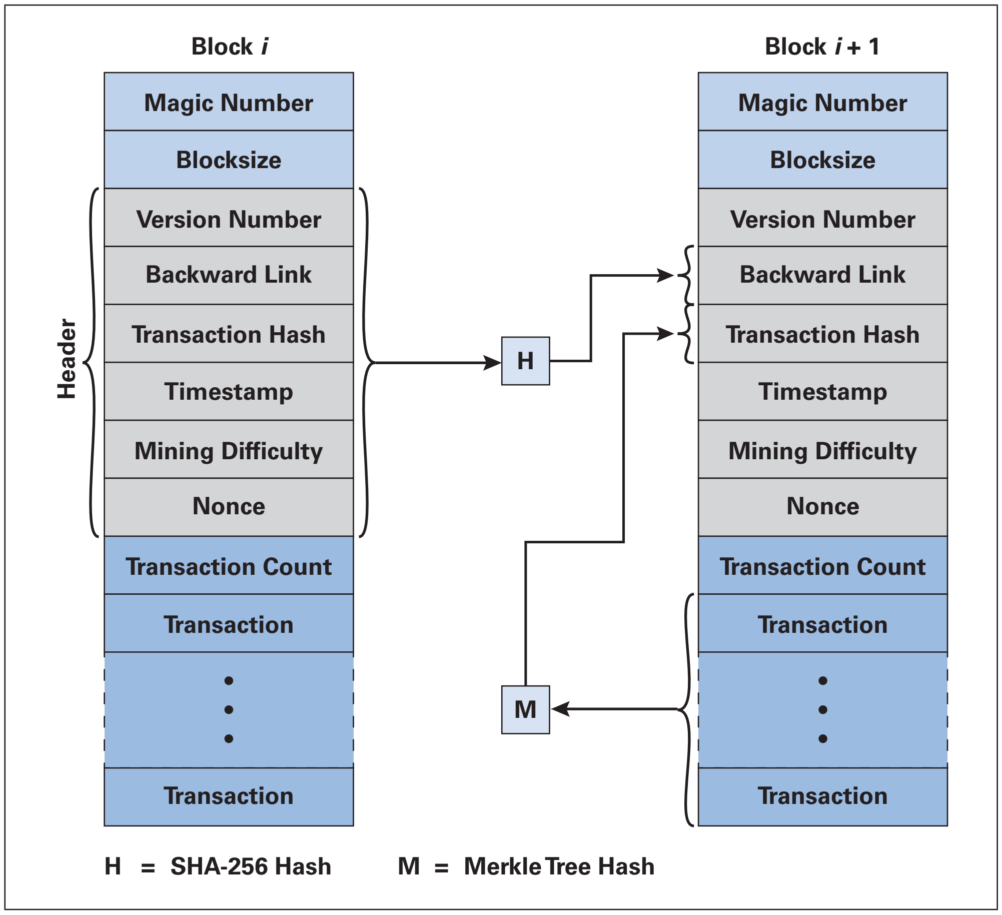
The header begins with a *Version Number,* to allow for future alterations to the block format. The blockchain application should be backward compatible so that older format versions can be processed. The foundation of the security of blockchain is found in the second field, which in effect provides a *Backward Link* to the preceding block. This backward link consists of the hash of all of the headers of the preceding block (Figure 4). By using a cryptographically strong hash function, such as SHA-256, this scheme secures the blockchain against an adversary's altering a block or inserting a block.

In either case, the adversary would have to create a block with a header whose hash value equals a given value, and this creation is computationally infeasible for SHA-256.

Table 1: Contents of a Block

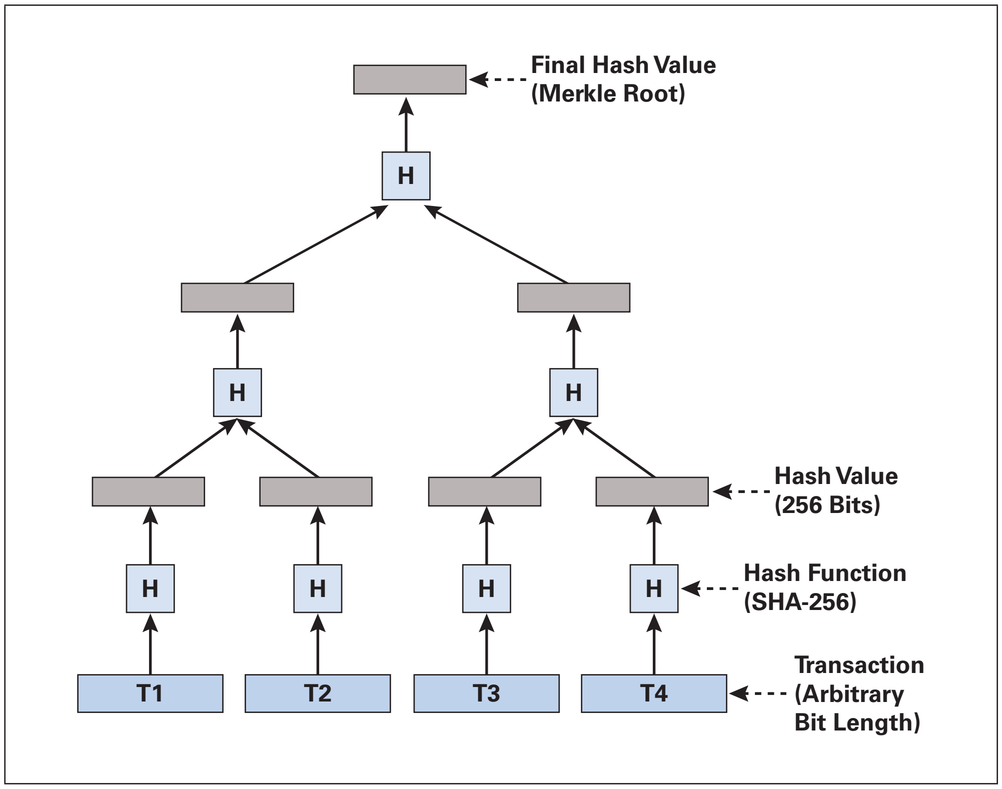
Item	Description
Magic Number	A unique identifier for the blockchain; remains constant for all subsequent blocks
Blocksize	Number of bytes following up to end of block
Version Number	Block format version
Link to Previous Block	Hash of preceding block header
Transaction Hash	The root node of a Merkle Tree, a descendant of all the hashed pairs in the tree. The root node is a 256-bit hash based on all of the transactions in the block.
Timestamp	When block was created
Mining Difficulty	A relative measure of how difficult it is to find a new block. The difficulty is adjusted periodically as a function of how much hashing power has been deployed by the network of miners.
Nonce	Used to calculate proof-of-work
Transaction Counter	Number of transactions in this block
Transactions	The (nonempty) list of transactions

Figure 4: Linkage Between Blocks



The next header field is the *Transaction Hash*. This hash value is computed from the set of data blocks that comprise the list of transactions. Rather than a single hash over this entire set, a *Merkle Tree* technique is used, illustrated in Figure 5. The transaction blocks are processed in pairs; if there is an odd number of transactions, the last transaction on the list is duplicated. Then, each pair of blocks is concatenated to form a binary block that is input to a hash function, typically SHA-256, which produces a 256-bit hash value. The resulting hash values are again paired, and each 512-bit pair is used as input to the hash function. This process continues until a single hash value results, known as the *Merkle Root*.

Figure 5: Example of a Merkle Tree



Following the transaction hash in the header is the *Timestamp* field, which indicates the relative time that this block was created, using a scheme specific to the application.

The next field, *Mining Difficulty*, is a measure of how difficult it is to find a new block. This procedure is explained subsequently. Finally, a one-time *Nonce* value is generated that is used for the proof-of-work concept, described subsequently.

Blockchain Mining

Consider an application that requires the storage of time-sequenced transactions for a distributed group of users. Numerous security issues arise, including authenticating users and ensuring the integrity of the sequence of stored transactions.

The latter includes the need for mechanisms to protect against malicious altering or insertion of transactions. Traditionally, these requirements are met by one or more trusted third parties that act as middleman. In a distributed environment with a large population of users, a peer-to-peer approach becomes more attractive as an efficient method for meeting these requirements. Such an approach is used in blockchain.

The distributed blockchain environment has the following characteristics:[3]

- Each user has a copy of the blockchain.
- Each user running the blockchain client is part of the network.
- New blocks are broadcast to the network.
- Each user updates its local copy of the blockchain.
- If a user is behind the current height of the chain, it can ask other nodes for copies of the blocks needed to catch up.
- If every user has a copy of the blockchain, when the blockchain is queried, every user gets the same answer.

Within a given application, blocks are created periodically to be added to the chain. The linking of a new block to the end of the chain is most commonly done by a process called *mining*.

Each block in the blockchain is required to have evidence that a costly, nonreversible sacrifice of time and energy has been dedicated to that particular block and no others. This evidence is known as *Proof-of-Work*. The important characteristics of proof-of-work include that it represents a true sacrifice: the actions performed are absolutely useless for any purpose other than producing the proof; and that it is nonreversible: no matter what happens, the resources used to produce the proof cannot be recovered. When Bitcoin clients encounter two valid but different blockchains, they choose to accept the one that represents the highest total proof-of-work.

Some entities within a blockchain network act as miners. It is the task of the miners to add new blocks to the chain and, in effect, miners compete to do this task. Any user can create a set of transactions that are to be formed into a block and added to the end of the chain. The miners are a distributed, pooled resource that create the blocks and add them to the chain.

In a typical open, distributed blockchain application, there are no designated miners. The entity adding the next block to the chain is selected on a per-block basis based on whoever in the world chooses to produce the most proof-of-work. In effect, miners compete for the right to add the next block. The incentive for doing so is a reward based on the application, such as earning Bitcoins for adding to the Bitcoin blockchain.

Miners can enter the system without asking for or requiring anyone's permission, and the network will continue to operate seamlessly when any particular miner leaves the system. The system is kept stable by virtue of the nonrecoverable sacrifice and its ability to discourage non-cooperating miners.

The operation of the miners is governed by a *consensus protocol.* In general, a consensus protocol takes as an input the requests of the components and decides upon one of these requests[4]. The blockchain consensus protocol ensures that among multiple conflicting proposed transactions, only one gets approved, preventing for example a double spending of the same coins.

A miner constructs a new block in the following fashion: Users broadcast transactions onto the network to be added to a new block. A miner collects these transactions to form a pool of transactions that are not yet part of a block.

Periodically, the miner constructs a new block with the pool of transactions it currently has. The miner validates all the transactions and decides on an ordering within the block. The miner then invests considerable computational effort to construct a new block; this process is called *solving* a block. This block is then broadcast to all the miners on the network and tentatively added to the end of the blockchain. The application requires that each block prove a significant amount of work was invested in its creation to ensure that untrustworthy peers who want to modify past blocks have to work harder than honest peers who only want to add new blocks to the blockchain.

In effect, the consensus mechanism for blockchain is a lottery race, in which the winner is rewarded in some fashion. The winner is the miner that is able to add a new block to the chain that is accepted by other miners.

The technique that is used for the proof-of-work may differ for different applications. Bitcoin uses a *cryptographic hash* technique that works as follows[5]: The cryptographic hash value of the block header is calculated to form the backward link used by the next block in the chain. If any hash value is allowed, this operation is a simple one. To make the process more resource intensive, a *mining difficulty* is established, which defines how many leading zeros the header hash value must have. Thus, with a mining difficulty of 1, there must be one leading zero. The miner can vary the hash value of the header by varying the value of the nonce field. Typically, a miner will begin with the nonce equal to 1, calculate the hash, and see if it satisfies the difficulty requirement. If not, it increments the nonce and tries again, repeating the process until a hash value is produced that satisfies the difficulty measure.

This difficulty measure is simple to express and effective. For example, if a single leading zero is required, then half of the possible hash values meet the requirement; thus, on average every other hash attempt will result in a hit. If ten leading zeros are required, the level of effort is on the order of one thousand hash attempts. If twenty leading zeros are required, the level of effort is on the order of one million hash attempts. For the Bitcoin blockchain, the target time for solving a block is 10 minutes.

We can express the mining function as follows: For a difficulty level of *alpha,* the hash value *H* must satisfy the following inequality:

H(version number, backward link, transaction hash, timestamp, alpha, nonce) < alpha

The miner must choose a value of nonce that satisfies this inequality. For a secure hash function such as SHA-256, it is effectively impossible to guess a value of nonce that works. Instead, the miner must try out many different values of nonce (using much computing power) until the condition is satisfied. Moreover, the lower the value of alpha, the harder it is to satisfy the condition. A proposed solution, however, can easily be verified. That is, once the nonce value is fixed, it is easy to determine if *H(version number, backward link, transaction hash, timestamp, alpha, nonce)* is less than *alpha*.

A new block can be added to the Bitcoin blockchain only if its header hash is at least as challenging as a difficulty value expected by the consensus protocol. Every 2,016 blocks, the network uses timestamps stored in each block header to calculate the number of seconds elapsed between generation of the first and last of those last 2,016 blocks. The ideal value is 1,209,600 seconds (2 weeks). That is, if blocks are generated at a rate of once per 10 minutes (600 seconds), then 2,016 blocks should be generated in 2,016 × 600 = 1,209,600 seconds. If it took fewer than 2 weeks to generate the 2,016 blocks, the expected difficulty value is increased proportionally (by as much as 300%) so that the next 2,016 blocks should take exactly 2 weeks to generate if hashes are checked at the same rate. If it took more than 2 weeks to generate the blocks, the expected difficulty value is decreased proportionally (by as much as 75%) for the same reason.

Returning to a general discussion of blockchain, not specific to Bitcoin, we can now see how the interlocking values of the nonce, transaction hash, and header hash protect the blockchain. An adversary who wishes to successfully alter the transaction list is faced with two alternative challenges: (1) modify the transaction list in such a way that the transaction hash is unchanged, also leaving the header hash unchanged; this modification is computationally infeasible for a secure hash function such as SHA-256; or (2) allow the transaction hash to change but modify the nonce so that the header hash is unchanged; again, this modification is computationally infeasible. Similarly, to insert a new block interior to the chain, the adversary would have to find a nonce value so that the header hash of the inserted block equals that of the preceding block.

Note that the computational effort of the adversary is far greater than that of the miner. Using the Bitcoin difficulty measure, for example, a difficulty value of 30 means that 2^{226} possible hash values can satisfy the requirement and the level of effort is on the order of 2^{30} hash attempts. For an adversary, it is necessary to find a nonce value that will produce a given unique hash value out of the 2^{256} possible values. On average, this discovery will take about 2^{128} hash attempts.

Miner Selection

The *Proof-of-Work* mechanism discussed previously is a way of selecting which miner gets to append a block to the chain. As we have seen, in this scheme the miner is essentially chosen at random through the competition among miners to produce a proof-of-work that is costly to produce but easy to verify.

Other than proof-of-work, numerous alternative methods have been considered or implemented, including the following[6]:

Proof-of-Stake grants mining rights to participants in proportion to their holding of the currency within the blockchain network. Miners must demonstrate that they hold more than a threshold amount of currency to be able to mine blocks. Proof-of-stake blockchains provide protection from a malicious attack because executing an attack would require the attackers to own a large amount of currency, which is very expensive. Besides, the miners owning a large stake most probably won't attack the system, for example, through double spending. Over time, such attacks will decrease the value of the cryptocurrency and the value of their stake.

The *Proof-of-Burn* process involves destroying Bitcoins by consuming them in a way that does not generate new Bitcoins[7]. The idea is that miners should show proof that they burned some coins—that is, sent them to a verifiably unspendable address. This process is expensive from their individual point of view, just like proof-of-work; but it consumes no resources other than the burned underlying asset. To date, all proof-of-burn cryptocurrencies work by burning proof-of-work-mined cryptocurrencies, so the ultimate source of scarcity remains the proof-of-work-mined "fuel."

Permacoin has proposed a modification to Bitcoin[8], which uses *Proof-of-Retrievability* (POR) to re-purpose the mining resource of Bitcoin to distributed storage of archival data. A POR proves that a node is investing memory or storage resources to store a target file or file fragment. This approach provides additional incentives to contribute resources to the network.

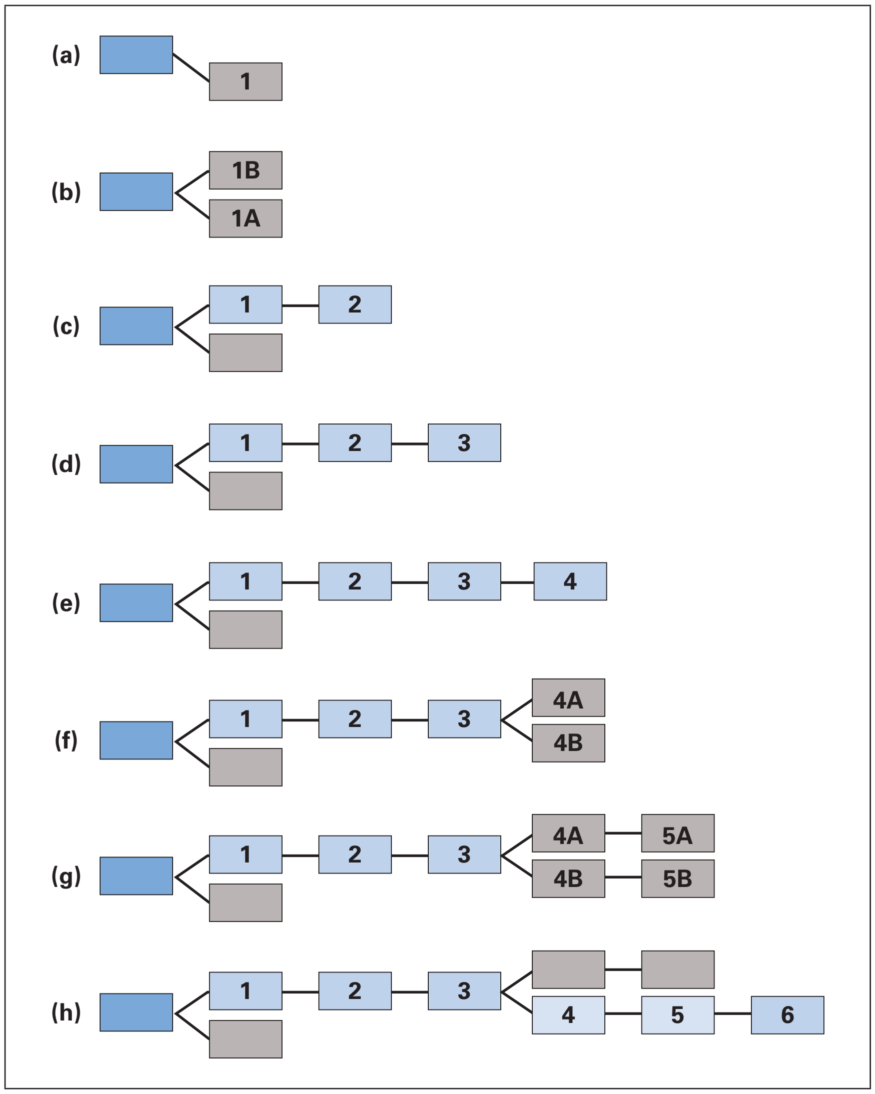
Building the Chain

After a miner has successfully hashed a block header, by finding a nonce value that satisfies the difficulty requirement, the miner can add the block to the end of the chain. It can happen that multiple blocks are created with the same height (distance from the origin block).

This situation occurs if more than one miner, working on different transactions, each produces a block at roughly the same time. This situation creates an apparent fork in the blockchain. Because the blockchain must be a linear, time-ordered sequence, the blocks in the branches following the fork are only provisional. The conflict is resolved when one of the branches exceeds the length of any competing branches.

Figure 6, based on [9], illustrates a typical sequence of events that resolves conflicts. There is an initial block at height 0. All miners try to solve the next block, and one miner solves a block at height 1 (see row *a* in the figure). But at almost the same time, another miner solves a block (row *b*). Block 1A may contain different transactions from 1B, and the users and miners in the network don't know which block should be the accepted one. So, both blocks are considered as provisional, and some miners work on adding to the chain at 1A, and some at 1B.

Figure 6: Adding Blocks to a Chain



Eventually, some miner creates a new block attached to 2B (row *c*). Because all miners must work at the highest height, those miners working on finding a successor to 1A stop that work. All miners are now working on creating a block to attach to the accepted block 2. At row *d* in Figure 6, one miner has successfully created and attached a block at height 3, and broadcasts this update to the network. All of the miners abandon their work at height two and now try to attach a block to the new block 4.

Next, a miner adds a block at height 4 and broadcasts it to the network (row *e*). Other miners, as soon as they receive this information, begin to work at block 4. However, at least one miner, before it receives this update, creates a block at height 4, creating a fork in the chain (row *f*), as happened back at row *c* in the figure. This situation creates a race condition that may continue. It is possible that both forks of the chain solve another block at about the same time (row *g*). In this example, the miners working on 5B solve a block, first adding a new block 6 (row *g*). This new chain, with block 6, is broadcast to all users and miners on the network. With block 6 in place, users are assured the blocks 4A and 5A are "locked in" to the blockchain. And miners who were working to add a block 6 realize they have lost the race. Now all miners begin working to try to append a new block after block 6. This activity continues as the chain grows, with occasional forks that are eventually discarded.

Confirming Transactions

As a miner is collecting transactions, it validates each one. The nature of the validation depends on the application but generally it depends on the use of public-key cryptography to authenticate the parties to the transaction and assure the integrity of the content of the transaction record[10]. The miner then assembles its current pool of validated transactions into a block. When the block is established as the next block in the chain, it is referred to as a *confirmation*. If there is a fork in the chain, then this confirmation is only provisional until the fork is resolved.

The deeper a block is embedded in a chain (that is, farthest from the current height of the change) the more difficult it would be for an adversary to alter the block. Thus, in any given application, a user of the distributed ledger can decide how many confirmations to wait for before acting with full confidence in a particular ledger entry.

Scalability

A blockchain in active use grows over time and never shrinks, raising the question of the scalability of a blockchain application. For example, Bitcoin allows a maximum block size of 2 MB and as of November 6, 2017, Bitcoin blockchain activity had the following characteristics (`https://blockchain.info`):

- Blockchain size (total size of all block headers and transactions, not including database indexes): 140.295 GB
- Average block size: 1.03 MB

- Transactions per day (most recent day): 333,161
- Aggregate size of transactions waiting to be confirmed: 40.31 MB

To perform all the Bitcoin functions and store the entire blockchain requires considerable processing and memory resources. For many blockchain applications, however, it is not necessary for all users to perform all the blockchain tasks, which include mining management, *Peer-to-Peer* (P2P) network communication and blockchain management, key management, and virtual asset management. For many blockchain applications, systems can be configured that provide only a subset of the tasks of a full implementation, with the handling of public-private key pairs as the most common core feature.

The authors of [11] define five categories of configuration:

- *Basic Client:* A client that runs on a user-controlled device and can perform key management operations, but cannot perform any P2P network communication. It is not a stand-alone solution. Examples include some dedicated hardware clients/wallets and cold-storage (offline storage) clients that require a second online device for transaction processing.
- *Thin Client:* A client that runs on a user-controlled device and can perform key management operations. It performs some P2P tasks related to network communication and blockchain verification but does not keep a copy of the full blockchain.
- *Thick Client:* A client that runs on a user-controlled device and performs all P2P tasks related to network communication and blockchain verification, keeps a copy of the full blockchain, and can perform all key management-related operations. This type of client is sometimes referred to as a *full node*.
- *Fully Functional Basic Client:* A node that performs all of the functions of a thick client, and executes the mining algorithm.
- *Hosted Client:* A client that does not run on a user-controlled device and all tasks are performed by a trusted third party on behalf of the user. This type of client is sometimes referred to as *hosted* or *web client/wallet*. In this case, it is not relevant whether key management is handled in the browser (for example, via JavaScript) because this requirement would in turn require users to download and verify the script code from the website of the third party every time they want to use it.

Depending on the blockchain application, even a full client may not need to store the full chain going back to the genesis block. That task can be reserved for a few archival nodes. The archival nodes can be used to bootstrap fully validating nodes from the beginning but are otherwise not active.

An example of a thin client is the *Simple Payment Verification* (SVP) client used in the Bitcoin application[12]. The majority of nodes on the Bitcoin network are SVP clients.

An SVP client stores only the portions of a blockchain needed to verify specific transactions of interest to this client. The node downloads the block headers and transactions that represent payments to its addresses. An SPV node doesn't have the security level of a fully validating node. Since the node has block headers, it can check that the blocks were difficult to mine, but it can't check to see that every transaction included in a block is actually valid because it doesn't have the transaction history and doesn't know the set of unspent transactions outputs. SPV nodes can validate only the transactions that actually affect them. The SPV nodes trust the fully validating nodes to have validated all the other transactions that are out there. The cost savings of being an SPV node are substantial. The block headers are only about 0.1% the size of the block chain. So instead of storing tens of gigabytes, the SPV node stores only a few tens of megabytes. Even a smartphone can easily act as an SPV node in the Bitcoin network.

Blockchain Types

Broadly speaking, there are three types of blockchains[13]. A *Public Blockchain* can be accessed and mined by anyone with Internet access. Access includes not only reading but also posting transactions, and they will be included if they are valid. Nodes participating in a public blockchain network do not have to obtain permission to access the ledger or add transactions. These blockchains are generally considered to be fully decentralized. Public blockchains have the benefit of information transparency and auditability, but they sacrifice information privacy.

A *Consortium Blockchain* is used across multiple organizations. The consensus process is controlled by authorized nodes. For example, one might imagine a consortium of 15 financial institutions, each of which operates a node and of which 10 must sign every block in order for the block to be valid. The right to read the blockchain may be public or restricted to the participants, and there are also hybrid variations such as the root hashes of the blocks being public together with an *Application Programming Interface* (API) that allows members of the public to make a limited number of queries and get back cryptographic proofs of some parts of the blockchain state. These blockchains may be considered partially decentralized.

A *Fully Private Blockchain*, or *Permissioned Blockchain,* limits write permissions to within a single organization that owns the blockchain. Read permissions may be public or restricted to an arbitrary extent. Likely applications include database management and auditing internal to a single company, so public readability may not be necessary in many cases at all, though in other cases public auditability is desired.

Currently, and projected for the foreseeable future, the majority of blockchain applications by market share are public. Most of the remainder are fully private[14].

Bitcoin

The original application of blockchain, for which it was invented, is *Bitcoin*. Bitcoin is perhaps the most widely used alternative (non-state-issued) currency in the world. Bitcoin is a digital currency scheme[15]. The network of miners literally creates money out of computer processing cycles. Currency within the system is given value (as it is in any money system) by its scarcity; in this system, the scarcity is created by requiring that money be processed by computationally intensive procedures. Having a great deal of computational power enables a miner to create Bitcoin value more quickly.

Blockchain provides the ledger for recording all of the digital currency transactions. Each transaction is potential only until it is recorded in a block that is accepted as valid and added to the chain. Recording requires the cooperative effort of the miners to achieve. As in incentive, miners are paid in Bitcoins for successfully adding a block to the blockchain.

Other Blockchain Applications

Great interest is being shown in applying blockchain technology to a wide variety of commercial and government applications. The following applications are listed in[16]:

- *Nasdaq* is using its Linq blockchain technology to complete and record private securities transactions.
- *Depository Trust & Clearing Corporation,* working with market participants and technology firm Axoni, is managing post-trade events for credit default swaps.
- *Factom* is providing blockchain technology for the Honduran land registry project. The focus is data security.
- *Everledger*'s focus is on the identity and legitimacy of objects. Blockchain works well here because its history cannot be changed and it enables trust by consensus. The company's initial work provides a distributed ledger of diamond ownership and transaction history verification for owners, insurance companies, claimants, and law enforcement agencies. The system assists with prevention of fraud in the supply chain, but also helps consumers decide whether to buy particular diamonds. The ultimate goal is to track diamonds from mine to market, so that consumers can see if correct duties and taxes have been paid and whether a diamond is a "blood diamond" that has been mined and traded in a war zone and contributed to human atrocity.

Blockchain technology has also caught the interest of numerous government agencies dealing with national security and homeland security, including *Defense Advanced Research Projects Agency* (DARPA) and the U.S. Air Force. The *Department of Homeland Security* (DHS) has awarded contracts for five projects that will use distributed ledger technology to develop new solutions for identity management and privacy protection[17]:

- *Digital Bazaar* is developing a Linked Data ledger format and architecture to demonstrate how to publish identity credentials.
- *Respect Network Corporation* is developing a decentralized registry and discovery service to integrate with the public blockchain.
- *Narf Industries* is developing an identity management solution built on a permission-less blockchain, with a focus on confidentiality (with selective information disclosure), integrity, availability, non-DHS repudiation, provenance, and pseudo-anonymity.
- *Xcelerate Solutions* is researching blockchain solutions to enable users to establish and maintain trusted identity transactions with public and private organizations.
- *Factom* is studying possible blockchain-based advancements for the security of digital identities for the *Internet of Things* (IoT). The project will create an identity log that captures the identification of a device, who manufactured it, lists of available updates, known security issues, and granted authorities while adding the dimension of time for added security. The goal is to limit would-be hackers' abilities to corrupt the past records for a device, making it more difficult to spoof.

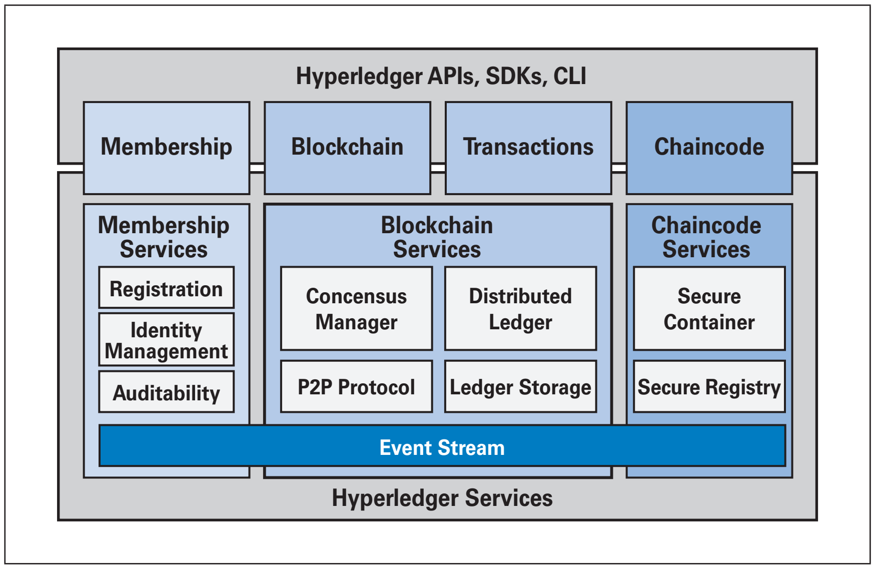
Another interesting development, one that indicates the growing and widespread popularity of the blockchain technology, is the *Initial Coin Offering* (ICO). With the ICO model, instead of selling ownership shares to investors to finance the start of the company, the startup sells digital coins, or tokens, that have value within the application or service the company offers. Sales of ICO tokens exceeded US$250 million in 2016 and are estimated to exceed US$1 billion in 2017[18].

Open-Source Blockchain

In 2016, the Linux Foundation, a nonprofit that champions open-source technologies, announced the *Hyperledger* project, an effort to create an enterprise-grade distributed blockchain ledger framework (`https://www.hyperledger.org`)[19, 20]. Participants in the group include R3, Cisco Systems, IBM, Intel, and VMware, among others. The objective of this project is to develop a standardized, production-grade digital ledger fabric. The project focuses on identifying and addressing important features for an enterprise-class, cross-industry open standard for distributed ledgers that can transform the way business transactions are conducted globally.

Figure 7 illustrates the current Hyperledger reference architecture within which open-source code is being developed.

Figure 7: Hyperledger Reference Architecture



Four main elements make up a Hyperledger-based application:

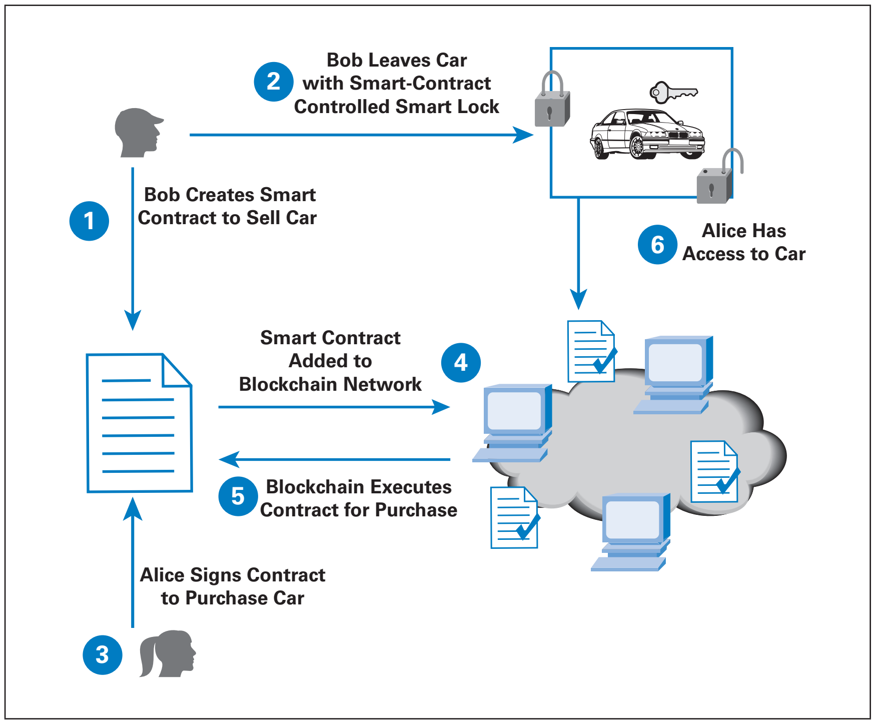
- *Membership:* Deals with registering, identifying, and auditing the activity of the peers who will use this particular ledger. The system distinguishes between two kinds of peers. A *validating peer* is a node on the network responsible for running consensus, validating transactions, and maintaining the ledger. A *non-validating peer* is a node that functions as a proxy to connect clients (issuing transactions) to validating peers.
- *Blockchain:* Consists of all the functions associated with building, storing, and providing access to a blockchain ledger.
- *Chaincode:* Implemented in Go, Chaincode is the realization of a smart contract. Each chaincode is encapsulated in a Docker container.
- *Transactions:* Examples of transaction types include the following: A *deploy* transaction takes a chaincode as a parameter; the chaincode is installed on the peers and is ready to be invoked. An *invoke* transaction invokes a transaction of a particular chaincode that has been installed earlier through a deploy transaction; the arguments are specific to the type of transaction. A *query* transaction returns an entry of the state directly from reading the peer's persistent state.

Smart Contracts

Smart Contracts (also called *Self-Executing Contracts, Blockchain Contracts,* or *Digital Contracts*) are computer programs that act as agreements, in which the terms of the agreement can be preprogrammed with the ability to self-execute and self-enforce itself. The main goal of a smart contract is to enable two anonymous parties to trade and do business with each other, usually over the Internet, without the need for a middleman.

The concept of smart contracts predates blockchain technology, but it is blockchain that has enabled the sudden growth in the use of smart contracts. One of the most prominent platforms for blockchain-based smart contracts is the open-source platform *Ethereum*[21].

An example of the use of smart contracts is shown in Figure 8[22].



2 Bob Leaves Car with Smart-Contract Controlled Smart Lock
1 Bob Creates Smart Contract to Sell Car
6 Alice Has Access to Car
Smart Contract Added to Blockchain Network 4
5 Blockchain Executes Contract for Purchase
Alice Signs Contract to Purchase Car
3

Figure 8: Smart Contract Example

The steps involved follow:

1. Bob creates a digital contract to sell his car. He identifies himself with his blockchain address (757382), which is his public key, uses a smart contract to define the terms of the sale, and signs the contract with his private key. The terms might read as follows:

```
IF $20,000 is sent to my account number 757382
THEN Transfer car ID 73849Z to account number that transferred the money
     Grant smart lock access to account number that transferred the money
```

   That English language description corresponds to code embedded in the digital contract. When the contract is added to the blockchain ledger, the code is automatically executed.

2. Bob leaves his car and car key in a garage locked with a smart lock controlled by the smart contract. The car has its own blockchain address 73849Z, which is a public key stored on the blockchain.

3. Alice wants to buy the car and searches for a suitable contract via a web browser. She finds Bob's car listed and signs the contract with her private key, which triggers an automatic transfer of $20,000 from her blockchain address (389157), which is her public key, to Bob's blockchain address 757382.
4. The signed smart contract is verified by each node in the blockchain network to verify that Bob is the owner of the car and that Alice has sufficient funds for the purchase.
5. If the network verifies the conditions, Alice automatically gets the access code to the smart lock for the garage (encrypted with Alice's public key), Alice is registered as the new owner, and $20,000 is transferred to Bob.
6. Alice can obtain the access code using her private key and then use the access code to pick up her car.

The whole process is distributed, automated, and does not require a central authority. In general, any blockchain-based smart contract employs the following steps:

1. A contract is defined. For some applications, there is a pre-defined contract specifying the terms of the contract and conditions for execution.
2. An event triggers the execution of the contract. An event could be the initiation of a transaction or the receipt of information.
3. When the contract is added to the blockchain ledger, the contract is executed, a process that typically involves movement of some value based on conditions met.

4a. For digital assets on the blockchain, such as cryptocurrency, accounts are automatically settled.

4b. For assets that are not part of the blockchain, such as stocks, changes to accounts in the ledger will match settlement instructions off the blockchain.

The smart contract model is very flexible and can be used in a wide variety of applications; some of them are listed in Table 2[23] on the following page.

Table 2: Blockchain Use Cases

Use Case	Description
Trade Clearing and Settlement	Manages approval workflows between counterparties, calculates trade settlement amounts, and transfers funds automatically
Coupon Payments	Automatically calculates and pays periodic coupon payments and returns principle upon bond expiration
Insurance Claims Processing	Performs error checking, routing, and approval workflows, and calculates payout based on the type of claim and underlying policy
Micro-insurance	Calculates and transfers micropayments based on usage data from an IoT-enabled device (for example, pay-as-you-go automotive insurance)
Electronic Medical Records	Provides transfer and/or access to medical records upon multi-signature approvals between patients and providers
Population Health Data Access	Grants health researchers access to certain health information; micropayments are automatically transferred to the patient for participation
Personal Health Tracking	Tracks patients' health-related actions through IoT devices and automatically generates rewards based on specific milestones
Royalty Distribution	Calculates and distributes royalty payments to artists and other associated parties according to the contract
Autonomous Electric Vehicle Charging Station	Processes a deposit, enables the charging station, and returns remaining funds when complete
Record Keeping	Updates private company share registries and capitalization table records, and distributes shareholder communications
Supply Chain and Trade Finance Documentation	Transfers payments upon multi-signature approval for letters of credit and issues port payments upon custody change for bills of lading
Product Provenance and History	Facilitates chain-of-custody process for products in the supply chain where the party in custody is able to log evidence about the product
Peer-to-Peer Transacting	Matches parties and transfer payments automatically for various peer-to-peer applications: lending, insurance, energy credits, etc.
Voting	Validates voter criteria, logs vote to the blockchain, and initiates specific actions as a result of the majority vote

Summary

Four elements characterize blockchain:

Blockchain is a Replicated Ledger. The ledger provides a history of all transactions that is immutable and is changed only by appending at the newest end of the linear chain of blocks that implements the ledger. The ledger is distributed and readable by all participants.

Blockchain operates by Consensus. Consensus is implemented by means of a shared, decentralized, peer-to-peer protocol. This shared control of the blockchain tolerates disruption, in that from time to time there may be temporary forks in the otherwise linear chain. The consensus mechanism is a means for validating transactions.

Fundamental to the operation of blockchain is *Cryptography.* Various cryptographic algorithms ensure the integrity of the ledger, the authenticity of transactions, the privacy of transactions, and the identity of participants.

Finally, blockchain is a versatile framework for implementing *Business Logic* that is embedded in the ledger. This logic is application-dependent and is reflected in the format and content of the transactions.

References

[1] "Blockchain: The next big thing," *The Economist,* May 9, 2015.

[2] "What CIOs Should Tell the Board of Directors About Blockchain," Gartner Research, February 14, 2017.

[3] Giles, A., "Blockchains 101," Slide presentation from the #StartingBlock2015 tour by @blockstrap, available at: `slideshare.net`

[4] Lamport, L. "The Part-Time Parliament," ACM *Transactions on Computer Systems* (TOCS), May 1998.

[5] Bitcoin Developer Guide, `bitcoin.org/en/developer-guide`

[6] Xu, X. et al., "The Blockchain as a Software Connector," 2016 13th Working IEEE/IFIP Conference on Software Architecture, April 2016.

[7] Kaye, M., "How to Secure a Blockchain with Zero Energy," *Bitcoin Magazine,* January 16, 2014.

[8] Miller, A. et al., "Permacoin: Repurposing Bitcoin Work for Data Preservation," IEEE Symposium on Security and Privacy, May 2014.

[9] Giles, A., "Blockchains 102," Slide presentation from the #StartingBlock2015 tour by @blockstrap, available at: `slideshare.net`

[10] Stallings, W., *Cryptography and Network Security,* Seventh Edition, ISBN-13: 978-0134444284, Pearson, 2017.

[11] Judmayer, A., Stifter, N., Krombholz, K., and Weippl, E., *Blocks and Chains: Introduction to Bitcoin, Cryptocurrencies, and Their Consensus. Synthesis Lectures on Information Security, Privacy, and Trust,* ISBN-13: 978-1627057165, Morgan & Claypool, 2017.

[12] Narayanan, A., Bonneau, J., Felton, E., Miller, A., and Goldfeder, S., *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction,* ISBN-13: 978-0691171692, Princeton University Press, 2016.

[13] Buterin, M.V., "On Public and Private Blockchains," Ethereum Blog, August 7, 2015, **`https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/`**

[14] Transparency Market Research, "Blockchain Technology Market (Type - Public Blockchain, Private Blockchain, and Consortium Blockchain; Application - Financial Services and Non-financial Sector) - Global Industry Analysis, Size, Share, Growth, Trends, and Forecast 2016–2024," 2017.

[15] Velde, F., "Bitcoin: A primer," Chicago Fed Letter, December 2013, **`http://www.chicagofed.org/digital_assets/publications/chicago_fed_letter/2013/cfldecember2013_317.pdf`**

[16] Underwood, S., "Blockchain Beyond Bitcoin," *Communications of the ACM,* November 2016.

[17] Prisco, G., "Department of Homeland Security Awards Blockchain Tech Development Grants for Identity Management and Privacy Protection," *Bitcoin Magazine,* August 18, 2016.

[18] Regnier, P., "ICO Is the New IPO," *Bloomberg BusinessWeek,* June 19, 1917.

[19] Androulaki, E., "Cryptography and Protocols in Hyperledger Fabric," Real-World Cryptography Conference, 2017.

[20] Cachin, C., "Architecture of the Hyperledger Blockchain Fabric," Workshop on Distributed Cryptocurrencies and Consensus Ledgers, July 2016.

[21] Bogatyy, I., "A Next-Generation Smart Contract and Decentralized Application Platform," Ethereum White Paper, **`https://github.com/ethereum/wiki/wiki/White-Paper`**

[22] BlockchainHub, "Smart Contracts," **`https://blockchainhub.net/smart-contracts/`**

[23] Odini, M., "Understanding Blockchain," Slideshare presentation, February 16, 2017, **`slideshare.net`**

WILLIAM STALLINGS is an independent consultant and author of numerous books on security, computer networking, and computer architecture. His latest book is the forthcoming *Effective Cybersecurity: A Practical Guide to Standards and Best Practices* (Pearson, 2018). He maintains a computer science resource site for computer science students and professionals at ComputerScienceStudent.com and is on the editorial board of *Cryptologia*. He has a Ph.D. in computer science from M.I.T. He can be reached at **`ws@shore.net`**

In Defence of NATs

by Geoff Huston, APNIC

N*etwork Address Translation* (NAT) has often been described as an unfortunate aberration in the evolution of the Internet, and one that will be expunged with the completion of the transition to *Internet Protocol Version 6* (IPv6). I think that this view, which appears to form part of today's conventional wisdom about the Internet, unnecessarily vilifies NATs. In my opinion, NATs are far from being an aberration; instead I see them as an informative and positive step in the evolution of the Internet, particularly as they relate to possibilities in the evolution of name-based networking. Here's why.

Background

It was in 1989, some months after the US National Science Foundation-funded IP backbone network had been commissioned, and at a time when there was a visible momentum behind the adoption of IP as a communications protocol of choice, that the first inklings of the inherent finite nature of the IPv4 address became apparent in the *Internet Engineering Task Force* (IETF)[1].

Progressive iterations over the IP address consumption numbers reached the same general conclusion: that the momentum of IP deployment meant that the critical parts of the 32-bit address space would be fully committed within 6 or so years. It was predicted that by 1996 we would have fully committed the pool of Class B networks, which encompassed one quarter of the available *Internet Protocol Version 4* (IPv4) address space. At the same time, we were concerned about the pace of growth of the routing system, so stop-gap measures that involved assigning multiple Class C networks to sites could have staved off exhaustion for a while, but perhaps at the expense of the viability of the routing system[2].

The IETF considered other forms of temporary measures, and the stop-gap measure that was adopted in early 1994 was the dropping of the implicit network/host partitioning of the address in classful addressing in favour of the use of an explicit network mask, or *classless* addressing. This change directly addressed the pressing nature problem of the exhaustion of the Class B address pool, as the observation at the time was that while a Class C network was too small for many sites given the recent introduction of the personal computer, Class B networks were too large, and many sites were unable to realise reasonable levels of address use with Class B addresses. This move to classless addressing (and classless routing, of course) gained some years of breathing space before the major impacts of address exhaustion, and the time gained was considered enough to complete the specification and deployment of a successor IP protocol[3].

In the search for a successor IP protocol, several ideas were promulgated. The decisions around the design of IPv6 related to a desire to make minimal changes to the IPv4 specification, while changing the size of the address fields and changing some of the encoding of control functions by using the ex-tension header concept, and changing the fragmentation behaviour to stop routers from performing fragmentation in real time[4].

The common belief at the time was that the adoption of classless addressing in IPv4 bought sufficient time to allow the deployment of IPv6 to proceed. It was anticipated that IPv6 would be deployed across the entire Internet well before the remaining pools of IPv4 addresses were fully committed. This assumption, together with a deliberate approach for hosts to prefer to use IPv6 for communication when both IPv4 and IPv6 was available for use, would imply that the use of IPv4 would naturally dwindle away as more IPv6 was deployed, and that no "flag day" or other means of coordinated action would be needed to complete this Internet-wide protocol transition[5].

In the flurry of documents that discussed a successor protocol was work that explored the concepts behind "address realms" where one single unique address realm could be replaced by a number of distinct address realms, where the addresses in a packet header could be rewritten when the packet passed across a realm boundary[6]. One paper at that time described the concept of source address sharing[7]. If a processing unit was placed on the wire, it was possible to intercept all outbound *Transmission Control Protocol* (TCP) and *User Datagram Protocol* (UDP) packets and replace the source IP address with a different address and change the packet header checksum, and then forward the packet on towards its intended destination. As long as this unit used one of its own addresses as the new address, then any response from the destination would be passed back to this unit. The unit could then use the other fields of the incoming IP packet header, namely the source address and the source and destination port addresses, to match this packet with the previous outgoing packet and perform the reverse address substitution, this time replacing the destination address with the original source address of the corresponding outgoing packet. This scenario allowed multiple internal end systems to use a "public" address, provided that they were not all communicating simultaneously. More generally, a pool of public addresses could be shared across a larger pool of internal systems.

It may not have been the original intent of the inventors of this address-sharing concept, but the approach was enthusiastically adopted by the emerging *Internet Service Provider* (ISP) industry in the 1990s. ISPs were seeing the emergence of the home network and were unprepared to respond to it. With the previous deployment model, using dial-up modems, each active customer was assigned a single IP address as part of the session start process.

A NAT in the gateway to the home network could extend this "single IP address per customer" model to include households with home networks and multiple attached devices. To do so efficiently a further refinement was added, namely that the source port was part of the translation. That way up to 65,535 simultaneous TCP sessions could theoretically share a single external address, provided that the NAT could rewrite the source port along with the source address[8].

For the ensuing decade NATs were deployed at the edge of the network, and ISPs have used them as a means of externalising the need to conserve IP addresses. The address-sharing technology was essentially deployed by, and operated by, the end customer, and within the ISP network each connected customer still required just a single IP address.

But perhaps that role is underselling the value of NATs in the evolution of the Internet. NATs provided a *firewall* between the end customer and the carrier. The telephony model shared the same end-to-end service philosophy, but it achieved this protection by exercising overarching control over all components of the service. For many decades, the telephone network was a controlled monopoly that was intolerant of any form of competitive interest in the customer. The Internet did not go down this path, and one of the reasons why is that NATs allowed end customers to populate their home network with whatever equipment they chose, and via a NAT, present to the ISP carrier as a single "termination" with a single IP address. This effective segmentation of the network created a parallel segmentation in the market, which allowed the consumer services segment to flourish without carrier-imposed constraint. And at the time that was critically important. The Internet wasn't the next generation of the telephone service. It was an entirely different utility service operating in an entirely different manner.

More recently, NATs have appeared within the access networks themselves, performing the address-sharing function across a larger set of customers. This function was first associated with mobile access networks but has been used in almost all recent deployments of access networks, as a response to the visible scarcity in the supply of available IPv4 addresses.

NATs have not been universally applauded. Indeed, in many circles within the IETF, NATs were deplored.

It was observed that NATs introduced active middleware into an end-to-end architecture, and divided the pool of attached devices into clients and servers. Clients (behind NATs) had no constant IP address and could not be the target of connection requests. Clients could communicate only with servers, not with each other. It appeared to some to be a step in a regressive direction that imposed a reliance on network middleware with its attendant fragility, and imposed an asymmetry on communication[9].

For many years, the IETF did not produce standard specifications for the behaviour of NATs, particularly in the case of handling of UDP sessions. Because UDP has no specific session controls, such as session opening and closing signals, how was a NAT meant to maintain its translation state? In the absence of a specific standard specification, different implementations of this function made different assumptions and implemented different behaviour, introducing another detrimental aspect of NATs: *variability*.

How could an application operate through a NAT if the application used UDP? The result was the use of various NAT discovery protocols that attempted to provide the application with some understanding of the particular form of NAT behaviour that it encountered[10].

NATs in Today's Internet

Let's now look at the situation today in the Internet of late 2017. The major hiatus in the supply of additional IPv4 addresses commenced in 2011 when the central *Internet Assigned Numbers Authority* (IANA) pool of unallocated IPv4 addresses was exhausted. Progressively the *Regional Internet Registries* (RIRs) ran down their general allocation address pools: *Asia Pacific Network Information Centre* (APNIC) in April 2011, *Réseaux IP Européens Network Coordination Centre* (RIPE NCC) in September 2012, *Latin America and Caribbean Network Information Centre* (LACNIC) in 2014, and *American Registry for Internet Numbers* (ARIN) in 2015. The intention from the early 1990s was that the impending threat of imminent exhaustion of further addresses would be the overwhelming impetus to deploy the successor protocol. By that thinking then the Internet would have switched to use IPv6 exclusively before 2011. Yet, that has not happened.

Today a minimum of 90% of the connected device population of the Internet still uses IPv4 exclusively, while the remainder use IPv4 and IPv6[11]. This network is an all-IPv4 network with a minority proportion also using IPv6. Estimates vary of the device population of today's Internet, but they tend to fall within a band of 15 to 25 billion connected devices[12]. Yet only some 2.8 billion IPv4 addresses are visible in the Internet routing system. This reality implies that on average each announced public IPv4 address serves from 3 to 8 hidden internal devices.

Part of the reason why estimates of the total population of connected devices are so uncertain is that NATs occlude these internal devices so effectively that no conventional Internet census can expose these hidden internal device pools with any degree of accuracy.

And part of the reason why the level of IPv6 deployment is still so low is that users, and the applications that they value, appear to operate perfectly well in a NATed environment. The costs of NAT deployment are offset by preserving the value of existing investment, both as a tangible investment in equipment and as an investment in knowledge and operational practices in IPv4.

NATS can be deployed incrementally, and they do not rely on some ill-defined measure of coordination with others to operate effectively. They are perhaps one of the best examples of a piecemeal incremental deployment technology where the incremental costs of deployment directly benefit the entity who deployed the technology. This situation is in direct contrast to IPv6 deployment, where the ultimate objective of the deployment, namely the comprehensive replacement of IPv4 in the Internet, can be achieved only after a significant majority of the population of the Internet are operating in a mode that supports both protocols. Until then the deployments of IPv6 are essentially forced to operate in a dual-stack mode, and also support IPv4 connectivity. In other words, the incremental costs of deployment of IPv6 generate incremental benefit only when others also take the same decision to deploy this technology. Viewed from the perspective of an actor in this space, the pressures and costs to stretch the IPv4 address space to encompass an ever-growing Internet are a constant factor. The decision to complement that factor with a deployment of IPv6 means an additional cost that in the short term does not offset any of the IPv4 costs.

So, for many actors the question is not "Should I deploy IPv6 now?" but "How far can I go with NATs?" By squeezing some 25 billion devices into 2 billion active IPv4 addresses, we have used a compression ratio of around 14:1, or the equivalent of adding 4 additional bits of address space. These bits have been effectively "borrowed" from the TCP and UDP port address space. In other words, today's Internet uses a 36-bit address space in aggregate to allow these 25 billion devices to communicate.

Each additional bit doubles this pool, so the theoretical maximum space of a comprehensively NATed IPv4 environment is 48 bits, fully accounting for the 32-bit address space and the 16-bit port address space. This number is certainly far less than the IPv6 128 bits of address space, but the current division of IPv6 into a 64-bit network prefix and a 64-bit interface identifier drops the available IPv6 address space to 64 bits. The prevalent use of a /48 as a site prefix introduces further address use inefficiencies that effectively drop the IPv6 address space to span the equivalent of some 56 bits.

NATs can be pushed harder. The "binding space" for a NAT is a 5-tuple consisting of the source and destination IP address, a source and destination port address, and a protocol identifier. This 96-bit NAT address space is a highly theoretic ceiling, but the pragmatic question is how much of this space can be exploited cost-effectively such that the marginal cost of exploitation is lower than the cost of an IPv6 deployment.

NATs as Architecture

NATs appear to have pushed applications to a further level of refinement and abstraction that were at one point considered to be desirable objectives rather than onerous limitations.

The maintenance of both a unique fixed-endpoint address space and a uniquely assigned name space for the Internet could be regarded as an expensive luxury when it appears that only one of these spaces is strictly a necessity in terms of ensuring integrity of communication.

The IPv4 architecture made several simplifying assumptions—one of which was that an IPv4 address was overloaded with both the unique identity of an endpoint and its network location. In an age where computers were bolted to the floor of a machine room, this assumption seemed very minor. However, in today's world it appears that the overwhelming number of connected devices are portable devices that change their location constantly, both in a physical sense and in terms of network-based location. This paradigm places stress on the IP architecture, and the result is that IP is variously tunnelled or switched in the final-hop access infrastructure in order to preserve the overloaded semantics of IP addresses.

NATs deliberately disrupt this relationship, and the presented client-side address and port have a particular interpretation and context only for the duration of a session.

In the same way that clients now share IP addresses, services now also share addresses. Applications cannot assume that the association of a name to an IP address is a unique 1:1 relationship. Many service-identifying names may be associated with the same IP address, and in the case of multihomed services the name could be associated with several IP addresses.

With this change comes the observation that IP addresses are no longer the essential "glue" of the Internet. They have changed to a role of ephemeral session tokens that have no lasting semantics. NATs are pushing us to a different network architecture that is far more flexible—a network that uses names as the essential glue that binds it together.

We are now in the phase of the Internet evolution where the address space is no longer unique, and we rely on the name space to offer coherence to the network.

From that perspective, what does IPv6 really offer?

More address bits? Well perhaps not all that much. The space created by NATs operates from within a 96-bit vector of address and port components, and the usable space may well approach the equivalent of a 50-bit conventional address architecture. On the other hand, the IPv6 address architecture has stripped off some 64 bits for an interface identifier and conventionally uses a further 16 bits as a site identifier. The resulting space is of the order of 52 bits. It's not clear that the two pools of address tokens are all that much different in size.

More flexibility? IPv6 is a return to the overloaded semantics of IP addresses as being unique endpoint tokens that provide a connected device with a static location and a static identity. This situation appears to be somewhat ironic in view of the observation that increasingly the Internet is largely composed of battery-powered mobile devices of various forms.

Cheaper? Possibly, in the long term, but not in the short term. Until we get to the "tipping point" that would allow a network to operate solely using IPv6 without any visible impact on the user population of the network, then every network still must provide a service using IPv4.

Permanent address-to-endpoint association? Well, not really. Not since we realised that having a fixed interface identifier represented an unacceptable privacy leak. These days IPv6 clients use so-called *privacy addresses* as their interface identifier, and regularly change this local identifier value.

Perhaps we should appreciate the role of NATs in supporting the name-based connectivity environment that is today's Internet. It was not a deliberately designed outcome, but a product of incremental evolution that has responded to the various pressures of scarcity and desires for greater flexibility and capability. Rather than eschewing NATs in the architecture as an aberrant deviation in response to a short-term situation, we may want to contemplate an Internet architecture that embraces a higher level of flexibility of addressing. If the name space is truly the binding glue of the Internet, then perhaps we might embrace a view that addresses are simply needed to distinguish one packet flow from another in the network, and nothing more.

Appreciating NATs

When NATs were first introduced to the Internet, they were widely condemned as an aberration in the Internet architecture. And in some ways NATs have directly confronted the model of a stateless packet switching network core and capable attached edge devices.

But that model has been a myth for decades. The Internet as it is deployed is replete with various forms of network "middleware," and the concept of a simple stateless packet switching network infrastructure has been relegated to the status of an historical, but now somewhat abstract, concept.

In many ways, this condemnation of NATs was unwarranted, as we can reasonably expect that network middleware is here to stay, irrespective of whether the IP packets are formatted as IPv4 or IPv6 and irrespective of whether the outer IP address fields in the packets are translated or not.

Rather than being condemned, perhaps we should appreciate the role that NATs play in the evolution of the architecture of the Internet.

We have been contemplating what it means to have a name-based data network, where instead of using a fixed relationship between names and IP addresses, we eschew this mapping and perform network transactions by specifying the name of the desired service or resource[13]. NATs are an interesting step in this direction, where IP addresses have lost their fixed association with particular endpoints, and are used more as ephemeral session tokens than endpoint locators. This step certainly appears to be an interesting one in the direction of named data networking.

The conventional wisdom is that the endpoint of this current transitioning Internet is an IPv6 network that has no further use for NATs. But it may not be true. We may find that NATs continue to offer an essential level of indirection and dynamic binding capability in networking that we would rather not casually discard. It may be that NATs are a useful component of network middleware and that they continue to have a role in the Internet well after this transition to IPv6 has been completed, whenever that may be!

References

[1] Frank Solensky, "Continued Internet Growth," Proceedings of the 18th Internet Engineering Task Force Meeting, August 1990.

[2] Hans Werner Braun, Peter Ford, and Yakov Rekhter, "CIDR and the Evolution of the Internet," SDSC Report GA-A21364, Proceedings of INET'93, Republished in *ConneXions—The Interoperability Report,* Volume 7, No. 9, September 1993.

[3] Vince Fuller, Tony Li, Jessica Yu, and Kannan Varadhan, "Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy," RFC 1519, September 1993.

[4] Scott Bradner and Allison Mankin, "The Recommendation for the IP Next Generation Protocol," RFC 1752, January 1995.

[5] Dan Wing and Andrew Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts," RFC 6555, April 2012.

[6] Lixia Zhang, "A Retrospective View of Network Address Translation," *IEEE Network,* September/October 2008.

[7] Paul Tsuchiya and Tony Eng, "Extending the IP Internet Through Address Reuse," ACM SIGCOMM *Computer Communications Review,* Volume 23, No.1, January 1993.

[8] Pyda Srisuresh and Der-hwa Gan, "Load Sharing Using IP Network Address Translation (LSNAT)," RFC 2391, August 1998.

[9] Tony Hain, "Architectural Implications of NAT," RFC 2993, November 2000.

[10] Geoff Huston, "Anatomy: A Look Inside Network Address Translators," *The Internet Protocol Journal,* Volume 7, No. 3, September 2004.

[11] IPv6 Deployment Measurement,
`https://stats.labs.apnic.net/ipv6`

[12] Internet of Things connected devices 2015–2025:
`https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`

[13] L. Zhang, et Al., "Named Data Networking," ACM SIGCOMM *Computer Communication Review,* Volume 44, No. 3, July 2014.

[14] Daniel Karrenberg, Yakov Rekhter, Elliot Lear, and Geert Jan de Groot, "Address Allocation for Private Internets," RFC 1918, February 1996.

GEOFF HUSTON, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for building the Internet within the Australian academic and research sector in the early 1990s. He is author of numerous Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005. He served on the Board of Trustees of the Internet Society from 1992 until 2001. At various times Geoff has worked as an Internet researcher, an ISP systems architect, and a network operator. E-mail: **`gih@apnic.net`**

Kimberly C. Claffy Honored with Postel Award

The Internet Society, a global non-profit dedicated to ensuring the open development, evolution and use of the Internet, recently announced that Dr. Kimberly C. Claffy, founder and director of the *Center for Applied Internet Data Analysis* (CAIDA) is this year's recipient of the prestigious *Jonathan B. Postel Service Award*.

© Stonehouse Photographic/Internet Society

Dr. Claffy is a pioneer in the field of measuring and understanding the Internet, not only through her research contributions, but her commitment to establishing and operating infrastructure to support large-scale data collection, curation, and sharing with the scientific research community.

The Postel Award was established by the Internet Society to honor individuals or organizations that, like Jon Postel, have made outstanding contributions to the data communications community. The award is focused on sustained and substantial technical contributions, service to the community, and leadership.

Dr. Claffy was selected by an international award committee comprised of former Postel Award winners. The committee placed particular emphasis on candidates who have supported and enabled others in addition to their own contributions. The committee noted that the award is being presented to Dr. Claffy in recognition for: "her pioneering work on Internet measurement through the development of infrastructure and methodologies for data collection, analysis, and sharing around the world."

The first of Dr. Claffy's many papers on Internet traffic measurement and analysis was published in 1992, years before the Internet transitioned to the global, private sector led network it is today. Since then, she has published dozens of papers and received numerous grants and awards for her work.

In 1997 Dr. Claffy founded CAIDA, based at the University of California's San Diego Super-computer Center, as a center which conducts network research and builds research infrastructure to support large-scale data collection, curation, and data distribution to the scientific research community.

"Simply put, Dr. Claffy's long-standing and pioneering work has helped the global community better understand the Internet and how it is used," explained Kathy Brown, President and CEO of the Internet Society, who presented the award.

"In addition to leading the way in the field of Internet measurement and analysis itself, her dedication of resources to ensure widespread access to measurement data has allowed a range of disciplines—from network science and network operations to political science and public policy—to benefit from her efforts."

KSK Rollover Postponed

The *Internet Corporation for Assigned Names and Numbers* (ICANN) recently announced that the plan to change the cryptographic key that helps protect the *Domain Name System* (DNS) is being postponed. Changing the key involves generating a new cryptographic key pair and distributing the new public component to the *Domain Name System Security Extensions* (DNSSEC)-validating resolvers. Based on the estimated number of Internet users who use DNSSEC validating resolvers, an estimated one-in-four global Internet users, or 750 million people, could be affected by the KSK rollover.

The changing or "rolling" of the *Key Signing Keys* (KSK) was originally scheduled to occur on October 11, 2017, but it is being delayed because some recently obtained data shows that a significant number of resolvers used by *Internet Service Providers* (ISPs) and Network Operators are not yet ready for the rollover. The availability of this new data is due to a very recent DNS protocol feature that adds the ability for a resolver to report back to the root servers which keys it has configured. There may be multiple reasons why operators do not have the new key installed in their systems: some may not have their resolver software properly configured and a recently discovered issue in one widely used resolver program appears to not be automatically updating the key as it should, for reasons that are still being explored.

ICANN is reaching out to its community, including its Security and Stability Advisory Committee, the *Regional Internet Registries,* Network Operator Groups and others to help explore and resolve the issues. In the meantime, ICANN believes it prudent to follow its process and to delay the changing of the key rather than run the risk of a significant number of Internet users being adversely affected. ICANN is committed to continuing its education, communication and engagement with the relevant technical organizations to ensure readiness for the key change.

"The security, stability and resiliency of the domain name system is our core mission. We would rather proceed cautiously and reasonably, than continue with the roll on the announced date of 11 October," said Göran Marby, ICANN CEO. "It would be irresponsible to proceed with the roll after we have identified these new issues that could adversely affect its success and could adversely affect the ability of a significant number of end users."

A new date for the Key Roll has not yet been determined. ICANN's Office of the Chief Technology Officer says it is tentatively hoping to reschedule the Key Roll for the first quarter of 2018, but that it will be dependent on more fully understanding the new information and mitigating as many potential failures as possible. ICANN will provide additional information as it becomes available and the new Key Roll date will be announced as appropriate.

For more information, visit:
`https://www.icann.org/resources/pages/ksk-rollover`

Thank You!

Publication of IPJ is made possible by organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol. The following individuals have provided support to IPJ. You can join them by visiting `http://tinyurl.com/IPJ-donate`

Fabrizio Accatino
Scott Aitken
Antonio Cuñat Alario
Matteo D'Ambrosio
Jens Andersson
Danish Ansari
David Atkins
Jaime Badua
John Bigrow
Axel Boeger
Kevin Breit
Ilia Bromberg
Christophe Brun
Gareth Bryan
Stefan Buckmann
Scott Burleigh
Jon Harald Bøvre
Olivier Cahagne
Roberto Canonico
John Cavanaugh
Lj Cemeras
Dave Chapman
Stefanos Charchalakis
Greg Chisholm
Narelle Clark
Steve Corbató
Brian Courtney
Dave Crocker
Kevin Croes
John Curran
Morgan Davis
Freek Dijkstra
Geert Van Dijk
Richard Dodsworth
Ernesto Doelling
Karlheinz Dölger
Andrew Dul
Holger Durer
Peter Robert Egli
George Ehlers
Peter Eisses
Torbjörn Eklöv
ERNW GmbH
ESdatCo
Steve Esquivel
Mikhail Evstiounin
Paul Ferguson
Gary Ford
Christopher Forsyth
Craig Fox
Tomislav Futivic
Edward Gallagher
Andrew Gallo
Chris Gamboni
Xosé Bravo Garcia
Kevin Gee
Serge Van Ginderachter
Greg Goddard
Octavio Alfageme Gorostiaga
Barry Greene
Martijn Groenleer
Geert Jan de Groot
Gulf Coast Shots
Sheryll de Guzman
Martin Hannigan
John Hardin
Edward Hauser
David Hauweele
Headcrafts SRLS
Robert Hinden
Edward Hotard
Bill Huber
Hagen Hultzsch
Karsten Iwen
Ashford Jaggernauth
David Jaffe
Dennis Jennings
Edward Jennings
Aart Jochem
Jim Johnston
Jonatan Jonasson
Daniel Jones
Gary Jones
Amar Joshi
Merike Kaeo
David Kekar
Shan Ali Khan
Nabeel Khatri
Anthony Klopp
Henry Kluge
Andrew Koch
Carsten Koempe
Alexader Kogan
Antonin Kral
Mathias Körber
John Kristoff
Terje Krogdahl
Bobby Krupczak
Warren Kumari
Darrell Lack
Yan Landriault
Markus Langenmair
Fred Langham
Richard Lamb
Tracy LaQuey Parker
Simon Leinen
Robert Lewis
Sergio Loreti
Guillermo a Loyola
Hannes Lubich
Dan Lynch
Miroslav Madić
Alexis Madriz
Carl Malamud
Michael Malik
Yogesh Mangar
Bill Manning

Harold March
David Martin
Timothy Martin
Gabriel Marroquin
Carles Mateu
Juan Jose Marin Martinez
Ioan Maxim
Miles McCredie
Brian McCullough
Joe McEachern
Carsten Melberg
Kevin Menezes
Bart Jan Menkveld
William Mills
Desiree Miloshevic
Thomas Mino
Mohammad Moghaddas
Charles Monson
Andrea Montefusco
Fernando Montenegro
Soenke Mumm
Tariq Mustafa
Stuart Nadin
Mazdak Rajabi Nasab
Krishna Natarajan
Darryl Newman
Marijana Novakovic
Ovidiu Obersterescu
Mike O'Connor
Carlos Astor Araujo Palmeira
Alexis Panagopoulos
Manuel Uruena Pascual
Ricardo Patara
Dipesh Patel
Alex Parkinson
Craig Partridge
Dan Paynter
Leif-Eric Pedersen
Juan Pena
Chris Perkins
Derrell Piper
Rob Pirnie
Jorge Ivan Pincay Ponce
Blahoslav Popela
Tim Pozar
David Raistrick
Priyan R Rajeevan
Paul Rathbone
Bill Reid
Rodrigo Ribeiro
Justin Richards
Mark Risinger
Ron Rockrohr
Carlos Rodrigues
Lex Van Roon
William Ross
Boudhayan Roychowdhury
Carlos Rubio
RustedMusic
Babak Saberi
George Sadowsky
Scott Sandefur
Sachin Sapkal
Arturas Satkovskis
Phil Scarr
Jeroen Van Ingen Schenau
Carsten Scherb
Roger Schwartz
SeenThere
Scott Seifel
Yury Shefer
Yaron Sheffer
Tj Shumway
Jeffrey Sicuranza
Thorsten Sideboard
Henry Sinnreich
Geoff Sisson
Helge Skrivervik
Darren Sleeth
Bob Smith
Mark Smith
Job Snijders
Ignacio Soto Campos
Peter Spekreijse
Thayumanavan Sridhar
Matthew Stenberg
Adrian Stevens
Clinton Stevens
Viktor Sudakov
Edward-W. Suor
Vincent Surillo
Roman Tarasov
David Theese
Sandro Tumini
Phil Tweedie
Steve Ulrich
Unitek Engineering AG
John Urbanek
Martin Urwaleck
Betsy Vanderpool
Surendran Vangadasalam
Alejandro Vennera
Luca Ventura
Tom Vest
Dario Vitali
Randy Watts
Andrew Webster
Tim Weil
Jd Wegner
Rick Wesson
Peter Whimp
Jurrien Wijlhuizen
Pindar Wong
Bernd Zeimetz

Follow us on Twitter and Facebook

`@protocoljournal` `https://www.facebook.com/newipj`

Call for Papers

The *Internet Protocol Journal* (IPJ) is a quarterly technical publication containing tutorial articles ("What is...?") as well as implementation/operation articles ("How to..."). The journal provides articles about all aspects of Internet technology. IPJ is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. In addition to feature-length articles, IPJ contains technical updates, book reviews, announcements, opinion columns, and letters to the Editor. Topics include but are not limited to:

- Access and infrastructure technologies such as: Wi-Fi, Gigabit Ethernet, SONET, xDSL, cable, fiber optics, satellite, and mobile wireless.
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance.
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, firewalls, troubleshooting, and mapping.
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, cloud computing, and quality of service.
- Application and end-user issues such as: E-mail, Web authoring, server technologies and systems, electronic commerce, and application management.
- Legal, policy, regulatory and governance topics such as: copyright, content control, content liability, settlement charges, resource allocation, and trademark disputes in the context of internetworking.

IPJ will pay a stipend of US$1000 for published, feature-length articles. For further information regarding article submissions, please contact Ole J. Jacobsen, Editor and Publisher. Ole can be reached at `ole@protocoljournal.org` or `olejacobsen@me.com`

The Internet Protocol Journal is published under the "CC BY-NC-ND" Creative Commons Licence. Quotation with attribution encouraged.

This publication is distributed on an "as-is" basis, without warranty of any kind either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This publication could contain technical inaccuracies or typographical errors. Later issues may modify or update information provided in this issue. Neither the publisher nor any contributor shall have any liability to any person for any loss or damage caused directly or indirectly by the information contained herein.

Supporters and Sponsors

Supporters

Internet Society

CISCO

Diamond Sponsors

Google

ICANN

Ruby Sponsor

RIPE NCC
RIPE NETWORK COORDINATION CENTRE

Sapphire Sponsors

Your logo here!

Emerald Sponsors

Afilias · APNIC · APRICOT Asia Pacific Regional Internet Conference on Operational Technologies · CISCO · CNNIC 中国互联网络信息中心 CHINA INTERNET NETWORK INFORMATION CENTER

COMCAST · DE-CIX · EQUINIX · JPRS JAPAN REGISTRY SERVICES · JUNIPer NETWORKS

lacnic · LinkedIn · NSRC Network Startup Resource Center · NTT Communications · TEAM CYMRU

VERISIGN · WIDE PROJECT

Corporate Subscriptions

AFRINIC The Internet Numbers Registry for Africa · amsix amsterdam internet exchange · DNS-OARC · IWL · ISC Internet Systems Consortium

Limelight NETWORKS · netnod · qa cafe · SIDN

For more information about sponsorship, please contact `sponsor@protocoljournal.org`

The Internet Protocol Journal
NMS
535 Brennan Street
San Jose, CA 95131

ADDRESS SERVICE REQUESTED

The Internet Protocol Journal
Ole J. Jacobsen, Editor and Publisher

Editorial Advisory Board

Dr. Vint Cerf, VP and Chief Internet Evangelist
Google Inc, USA

David Conrad, Chief Technology Officer
Internet Corporation for Assigned Names and Numbers

Dr. Steve Crocker, CEO and Co-Founder
Shinkuro, Inc.

Dr. Jon Crowcroft, Marconi Professor of Communications Systems
University of Cambridge, England

Geoff Huston, Chief Scientist
Asia Pacific Network Information Centre, Australia

Dr. Cullen Jennings, Cisco Fellow
Cisco Systems, Inc.

Olaf Kolkman, Chief Internet Technology Officer
The Internet Society

Dr. Jun Murai, Founder, WIDE Project, Dean and Professor
Faculty of Environmental and Information Studies,
Keio University, Japan

Pindar Wong, Chairman and President
Verifi Limited, Hong Kong

The Internet Protocol Journal is published quarterly and supported by the Internet Society and other organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol.

Email: ipj@protocoljournal.org
Web: www.protocoljournal.org

The title "The Internet Protocol Journal" is a trademark of Cisco Systems, Inc. and/or its affiliates ("Cisco"), used under license. All other trademarks mentioned in this document or website are the property of their respective owners.

Printed in the USA on recycled paper.