Atomcoin and Atom Nodes: The Basis of Atomchain P2P Network

Okpara Okechukwu D.
okpara.net@gmail.com
www.okpara.net
01-05-2022

Abstract:
This paper provides an architectural overview of the maiden release of atomchain, the atom big bang. Distributed computations require agreement between a set of nodes in an atom network. These nodes achieve agreement within a given timeframe via atomchain consensus protocols. The atom network can be used in creating, transferring, and trading of digital assets. The atom network presented in this paper is considered as being almost 100% Byzantine Fault Tolerant.

Keywords: web3, atomchain, atomcoin, Atom Field, Atom Network

Disclosure: The information in this paper is preliminary and is subject to change anytime in the future.

1. Introduction

Atomchain networks are scalable, customizable, secure and decentralized. They can be used as a permissioned/private and/or as a permissionless/public environment for building decentralized applications.

Classical consensus protocols rely on all-to-all communication, hence they can have low confirmation latencies and high throughput – they neither scale to large numbers of participating nodes, nor become robust when node membership changes are highly dynamic. Thus classical protocols has the tendency of making the distributed systems to be permissioned, and mostly statically deployed. On the other hand, atomchain consensus protocols, referred to as “bonds” are based on lightweight network sampling mechanisms. Bonds achieve low latency and high throughput without needing to agree on the precise membership of the nodes. Bonds are created when nodes repeatedly sample the atom network. Bonds of atom fields are regarded as **strong** bonds. Atomic reactor nodes sample molecule nodes repeatedly until convergence (in an atomic molecule node) is reached during normal operation.

In order to support the atom network, developers need the cryptocurrency, atomcoin, to create and run applications. Atomcoin is used to pay for transaction fees and computational services. Atomcoin is pseudonymous in the sense that funds are not tied to real-world entities but rather to atomchain addresses.

Atomcoin is a number that can be stored into wallet or account addresses and can be spent or received as part of transactions or atom generation. In other words, atomcoin is used for atomchain’scoinbase transactions, and users can send atomcoins to other users from their wallets.

Atomcoin comes into existence by the validation of transactions on the Atomchain platform, through an atom creation process called atomic "mining".

Subunits

Multiplier or Precision	Special Name	Note
10^{-9}	atomcoin	Smallest unit
10^{-6}	microoke	
10^{-3}	millioke	
10^{0}	oke	
10^{3}		
10^{6}		
10^{9}	aku	
10^{12}	okeaku	
10^{15}	okechi	Largest unit

Features of atomcoin include the following:

- Atomcoin transactions are recorded and verified on an atomchain.
- There are multiple ways for an individual to obtain atomcoins.
- It can be purchased on an exchange using a fiat currency (under the symbol XAX, or any proposed symbol with prefix, "XA-" such as XATOM, XATC and so on.Note that the nomenclature used here further guarantees conformity with ISO 4217 currency standard.
- It can be transferred to you from another person or entity.
- It can be earned from atomic mining.

In order to hold atomcoin, you must have an Atomchain wallet, which can be downloaded and set up onto a computer, smartphone or other mobile device.

Each Atomchain wallet stores an individual's private key which allows the wallet owner to sign transactions that send atomcoins to other parties.

Atomcoin is a critical component to keeping the Atomchain platform growing and evolving in the digital asset environment.

2. Atomchain Quarks

In Atomchain, the state is made up of 20-bytes(5 words of 32-bits) address objects called atomic "quarks" which can be active or passive. These quarks have state transitions which are direct transfers of value and information between these objects.

An atomchain account normally contains five(5) fields:

1. The **nonce**, a counter used to make sure each transaction can only be processed once.
2. The object's (that is, quark's) current **atomchain balance**.
3. The objects's **quark code**, if present.
4. The object's **storage**, which is empty by default.
5. The object's **nature**, which is not set or null by default.

There are two types of objects or quarks ownership in atomchain:

1. Active quarks: externally owned quarks that are controlled by private keys, and
2. Passive quarks: internally owned quarks that are controlled by their **quark code**. A quark code is model consisting of three sets—input, output, and state of the model—and three functions—input transform function, state transition function, and output function. This model provides an improved level of performance and safety for the atomchain since it consists of codes that are in line with current industry performance and security standards.

Active quarks can be associated with both **αokebase** and **Ωokebase** addresses; while, passive quarks can only be associated with **Ωokebase** addresses. (See the atomocore.json file section)

3. Transactions on Atomchain Network

A transaction is a signed data package that stores a message to be sent from an externally owned quark. Transactions contain the following:

- the recipient of the message
- a signature identifying the sender
- the amount of atomcoin to transfer from the sender to the recipient
- an optional data field
- an **energyMAX** value, representing the maximum number of computational steps the transaction execution is allowed to take
- an **energyPRICE** value, representing the fee the sender pays per computational step

The first three are normal fields expected in any cryptocurrency. The data field has no function by default.

In order to prevent accidental or malicious infinite loops or other computational wastage in code, each transaction must set a limit to how many computational steps of code execution it can use. The unit of this computation is "**Atom energy**" (or simply, "**energy**"). By default, there is an incremental fee of **10 energy** for every byte in the transaction data. This fee is necessary so that an attacker will pay proportionately for each resource that they consume, including computation, bandwidth and data storage. The use of both **energyMAX**, and **energyPRICE** fields in Atomchain helps in solving the problem of denial of service attacks through malicious (or bugged) scripts that run forever. Energy in Atomchain is like "gas" in Ethereum blockchain.

As an example, if for a particular transaction, the energy (used by the transaction) limit was 21,000, and the energy price was 21 Oke (pronounced as, "oak", is the lowest denomination of atomcoin). Then, 21 Oke * 21,000 will give the actual transaction fees: 0.000441 atomcoins. This transaction fee goes to the atom creator (or miner), that has validated the transaction.

We elucidate a basic operational mechanism thus: a user creates a transaction - with an application running on atomchain VM – in a message and sends it to a validating (or an

atomic creator) node participating in the consensus (or bonding) procedure. The message is then propagated out to other nodes (such as to an element node) in the network. If the user is malicious and uses a multispend (or doublespend) attack, the node randomly selects a small subset of its atomic reactor nodes and queries which of the conflicting transactions the nodes think is the valid one. If the querying element node receives a supermajority response in favour of one transaction, then the element node changes its own response to that transaction. Every molecule node in the network of that atomic (reactor) node repeats this process until that particular atom field comes to consensus on one of the conflicting transactions.

4. Inside atomcore.json State File

atomcore is the start of the atomchain, atom 0; and atomcore.json is the atomcore state file that defines it. It is the default settings/configurations for the atomchain network.

```
{
      “setup”:{
            “atomID”:0,
            “atomHome”:0,
            “atomState”: {
            “ASH”:”…”,
            “DNA”:”…”,
            “PNH”:”…”,
            “SIG”:”0”
            },
            “atomDump”:”…”
      },

      “atomBank”: {
      “address”:{“balance”:”0xffffffffffffffff” },
      “address”:{“balance”:”…” },
      …
      },

   "αokebase": {
         "address1":"balance",
         "address2":"balance"
      },

   "Ωokebase": {
         "address1":"balance",
         "address2":"balance"
      },
      “Ωnonce”:1,
            “timestamp”:”0x0”,
            “okebase”:”0x00…00”,
            “energyMax”:”0x1fffffffffffff”,
            “version”:”0.1.0”,
            “elevMode”:”0x0”,
            “atoms”: { },
            “periods”: {“name”:”atom” },
            …

}
```

atomchain.json custom state file

This file defines the rules for the atomchain itself.In its **setup**/configuration section it contains: **ASH** (Atom creator's node SHA Hash), **PNH** (Previous Nucleus Hash, a 256-bit hash of the entire parent/previous Nucleus. It is a pointer to the encapsulated atom, thus effectively building the **bond of atoms**. PNH of atomcore is set to zero), the **DNA** (Decentralization Number of the Atom), **SIG** (atom's Significance, this is a boolean value to handle signed/unsigned type or parity).

periods is for specific network upgrades that will occur in the future at any *atom* (the atom number where the milestone or upgrade occurred). *name* to be used will be adapted from the **Periodic Table of Elements** starting with hydrogen.**atoms** contains the initial list of atomic (reactor) nodes with elevation modes in the format such as*"node@IPaddress:Port":elevMode*

- **setup**contains all the configuration parameters/variables and thresholds that control the atom network's basic operations. The values in the variables need to match the configuration information of any other node this atomcore has to interact with.
- **atomID**protects the network from a **replay attack**– where an authorized attacker node acts as the original sender. It acts like an offset to prevent attackers from deciphering continuous values in the **atom network**.It is a unique identifier that allow only hashed transactions that are signed.
- **okebase or Atom's Base Coins (ABCs)** is the network coinbase account; which is where all atom rewards for the network will be paid. It can be set as an arbitrary address of 160-bits to which all rewards (in atomcoin) collected from the successful mining of this atom have been transferred. Its atomcoin-the native token of Atomchain-is the sum of the atom reward and the transaction fee. The terms "okebase" and "ABCs" can be used interchangeably.
- **energyMax** is the total energy limit for all transactions included in an atom. It defines how large the atom size can be, and it is represented by an hexadecimal string. This limit impacts how much atomchain VM computation can happen within an atom. In short, this is the limit of energy cost per atom.
- **version** is the AtomVM Chainware version.Atomchain codebases will be released using three numeric identifiers, labeled "v.[0-9].[0-9].[0-100]", where thefirst number identifies major releases, the second number identifies minor releases, and the third numberIdentifies patches. The first public release, codenamed **Atomic Big Bang**, is v. 1.0.0.
- **atomBank**contains a number of pre-funded wallet addresses. The first parameter is the address (a 160-bits hexadecimal string. It allows defining a list of pre-filled wallets, needed to handle the "atomcoin pre-sale" period.
- **timestamp** is a scalar value equal to the reasonable output of Unix time() function at the atom's inception/creation.
- **elevMode** is an applied scalar value corresponding to the "reputation or visibility" level of the node during the atomic molecule pool selection process.
- **αokebase** is created similar to the way Bitcoin blockchain's native segregated witness(SegWit) addresses are created. But it starts with the prefix "ac".
- **Ωokebase** is created similar to the way Ethereum blockchain addresses are created.

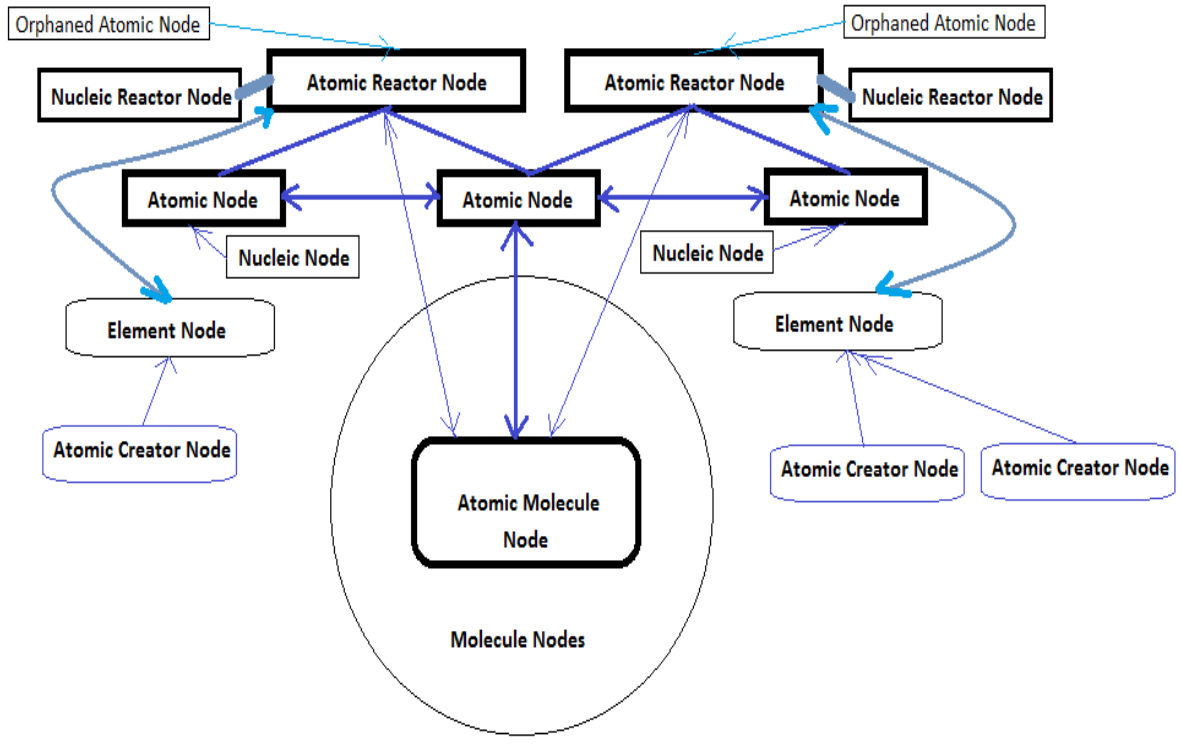
- **Ωnonce:** this is a public viewable nonce which is increased by one every time a transaction is made, in order to prevent the same transaction being submitted more than once (in a multi-spend attack).

5. Atomchain Network a.k.a. Atom Network

Atomchain nodes communicate with one another through the Atomchain P2P network protocol, and by doing so; they guarantee the integrity of the network. Any node that misbehaves or tries to propagate incorrect information is quickly detected by the "honest" nodes and is disconnected from the network. Nucleic reactor and atomic reactor nodes may come with wallets (that is, they have wallet addresses). Listening nodes are publicly accessible, while nonlistening nodes may be hidden nodes, for instance, a node operating behind a firewall. The Atomic nodes are the backbone of the atomchain because they are responsible for storing and distributing copies of the entire atomchain ledger – or database.

The Network ID is usually the same as the atom ID in the atomcore.json state file. It is an integer number that isolates atomchain peer-to-peer networks. Connections between atomchain nodes will occur if and only if both peers use the same Atomcore and network ID; in other words, the atom ID in the Atomcore.json file must be the same for all participating nodes in the network.

Atomchain can be public/permissionless or private/permissioned. An atomchain network is a private or isolated network if the nodes are not connected to the main atomchain network. Atomchain main atom network's atom ID is 1. If you are intending to connect to your private atom network on the internet, it is best to choose a network ID that is not in use.



First Draft of Atomchain Nodes

Atomic Creator Nodes: these nodes create or "**mine**" atoms from their respective transaction pools (transpools). They are also referred to as atom "mining" nodes; similar to "miners" in blockchain.

Atomic Reactor Nodes: these atomic nodes are "**listening**," and publicly visible. It communicates and provides information to any other node that establishes a connection with it. They can either mark invalid atoms (from element nodes) with the "destroy" marker before returning them, or completely discard atoms with invalid transactions. They are basically redistribution points or relays that act both as data sources and as connection bridges for the network.

Atomic Molecule Nodes: this is a node that is connected to most atomic (or atomic reactor) nodes at the point when an atom is to be added to the atomchain. Atomic (and atomic reactor) nodes actively monitor for this node, and accept the atom it provides as the next atomsphere.

Atomic Nodes: these are servers that host the entire atomchain. They are devices in the network that store and synchronize a copy of the network's entire atomchain history. They also validate atoms and maintain consensus. They are concerned with the general health of the network.

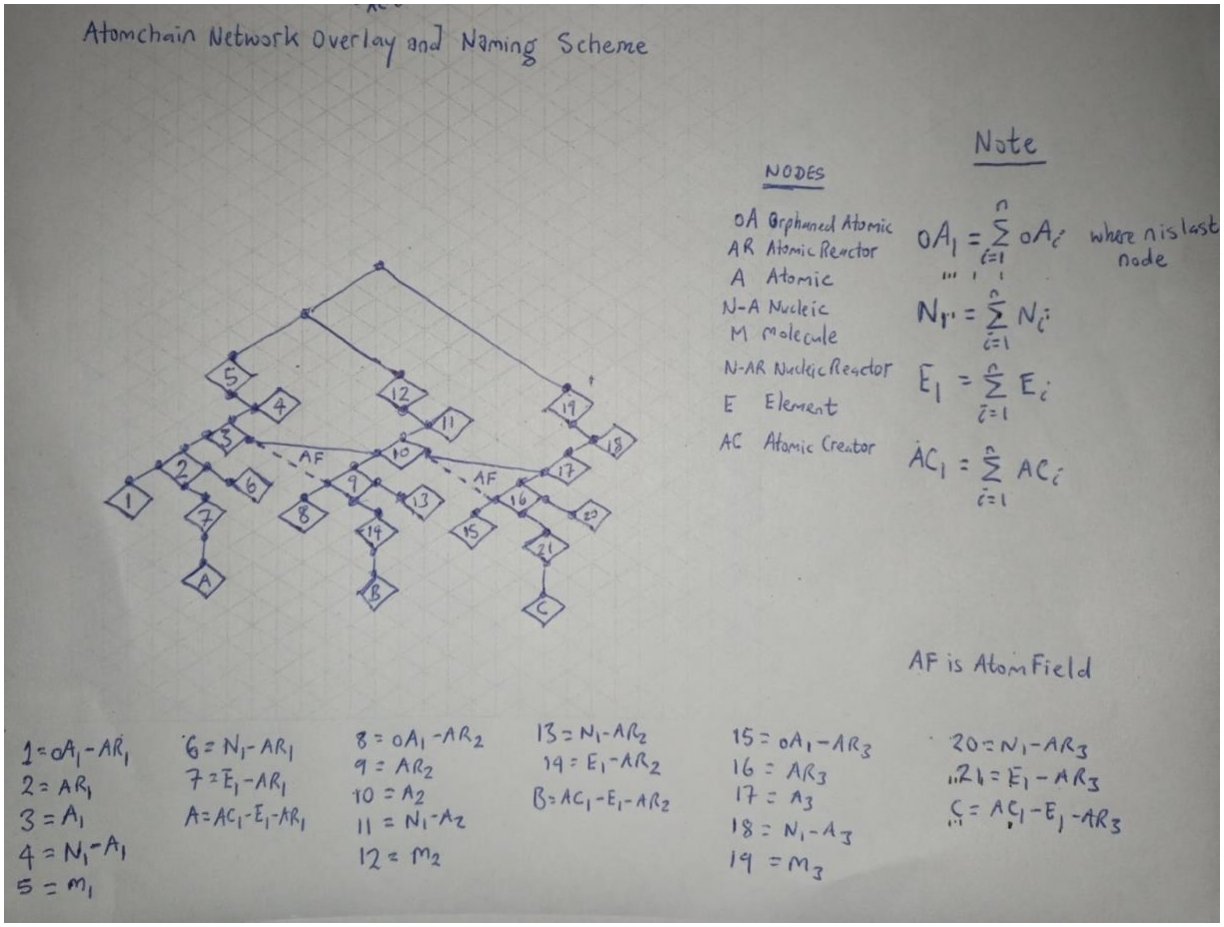
Nucleic Nodes: these nodes contain the nucleus of previous atoms. They depend on parent atomic nodes. These nodes download existing nuclei only, thereby saving users significant download times and storage spaces.

Nucleic Reactor Nodes: these are listening nucleic nodes that are publicly visible.

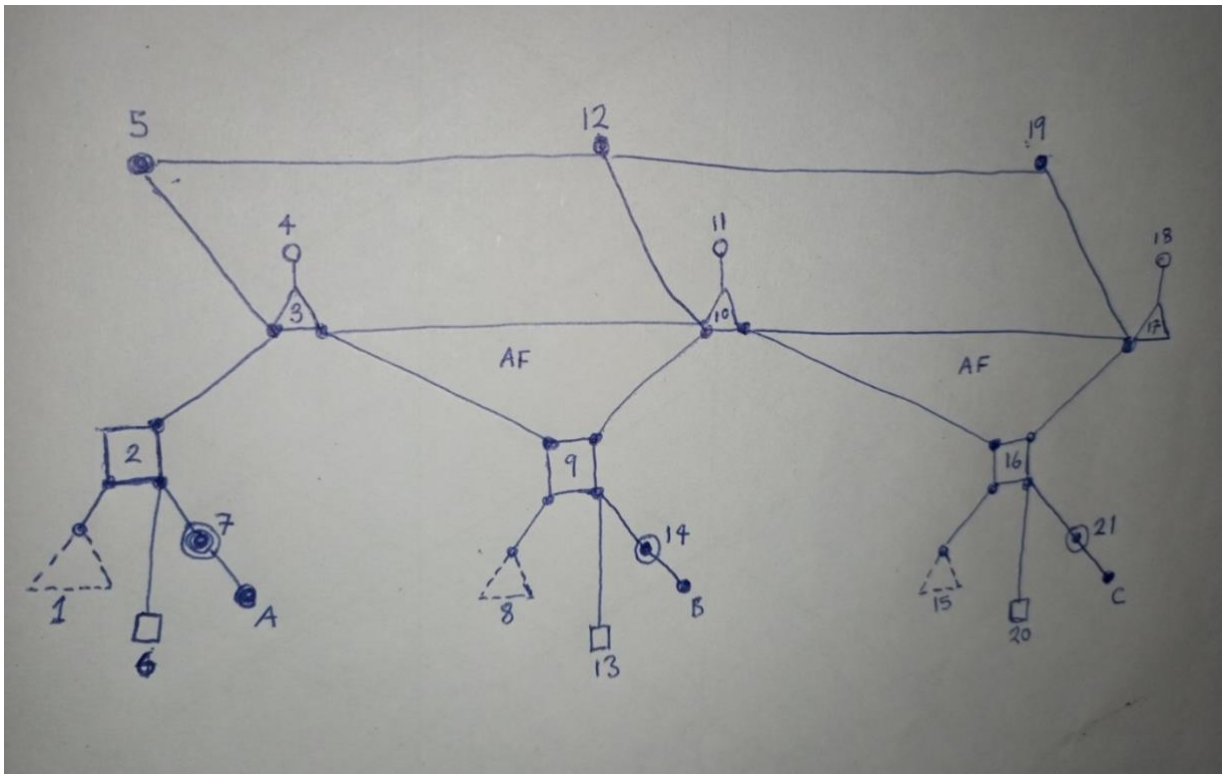
Element Nodes: these are storage servers for atom pools. They receive atoms from atomic creators, sends the atoms to atomic reactors for validation (that is, they query atomic reactor nodes).

Molecule Nodes: these are storage servers for elevation pools.

Orphaned Atomic Nodes: these are mini-atomic nodes that save storage space for its users by "pruning or trimming off" older atoms in the atomchain starting from the atomcore. They first of all download the entire atomchain from an atomic reactor node, then they begin to delete atoms starting from the atomcore until they hold only the most recent transactions (up to a set size limit of the node operator).



The Atomchain Network Overlay and Naming Scheme



A simplified visualization of the Atomchain Network Overlay

6. Nodes Configuration File

Peers are called nodes. All peers are supposed to be connected to the atomchain **overlay** network. If a connection or pathway exists from one peer to another, it is a part of this overlay network. The main overlay network connecting atomic reactors and the main atomic nodes is referred to as the **Atom Field**.

A node configuration file should have parameter/variable features such as:

maxtranspool – which should limit the transaction pool in megabytes. Used in reducing memory use on memory-contrained nodes.

maxconnects – which should set the maximum number of nodes from which to accept connections. Lower values implies reduced bandwidth consumption.

transindex – which should maintain an index of all transactions. A complete copy of the atomchain that allows one to programmatically retrieve any transaction by ID.

maxreceivebuffer/maxsendbuffer – which should limit per-connection memory buffer to a given multiple of 1000 bytes. Used on memory-constrained nodes.

minrelaytransfee – which should set the minimum fee rate for transactions to be relayed. This can determine which transactions are to be rejected from the transpool and not relayed.

If a new node or validator attempts to join the atomchain, it will use the atomcore.json file as the starting point in recreating the history of the atomchain in order to synchronize with the existing network (or atom fields).

The following features are expected of atomchain nodes – or **Atom Nodes** for short.

- Each molecule (or atomic creator) node must have at least one account address, if it is to receive atom rewards – from mining.
- Nodes must have an unused listening port (for instance, port 40404) associated with its IP address for communication.
- Nodes must have a data filesystem or directory/folder where the atomchain data will be located or stored.
- Nodes must have an identity according to the atomchain naming scheme.
- Nodes must have an Interprocess Communication (IPC) path, especially for any node running on a machine or on a network it has access to. This is important when attaching a “console” to a running node.

Atomic reactor nodes – or **Atomic Reactors** for short, are powerful nodes that handle all types of requests from nodes querying it. Atomic reactors are not directly linked to one another to avoid network usage payloads or bandwidth wastages. They usually interact with atomic nodes, for instance, they return the IP addresses of all neighbouring atomic nodes having a requested file to the queryingpeer (such as an Element node).

Notes:

- The initial list of atomic (reactor) nodes is configured in the atomcore.json file.
- energyMax limit is set to 1000; and the default energyPrice is set to 26 oke.

ATOMCHAIN TRANSPOOL EXPLAINED

• When a user creates a transaction on atomchain, the unconfirmed transaction is sent to the transpool.

• All atomic (reactor) nodes have access to the transpool. They validate the unconfirmed transactions in the transpool. When a node on atom network receives a new transaction, it verifies the transaction according to the protocol's prevailing validation rules. If a transaction violates the rules (for instance, having an invalid signature or hash), the transaction is not forwarded to other peers, and the transaction is marked with the "destroy marker". On the other hand, if a transaction was successfully validated, it is then added to the transaction sets in the transpool.

• Atomic creators (or miners) select a certain number of validated transactions from the transpool and try to apply the elevation mode consensus. A collection of validated transactions forms an atom. Atom creation rewards consist of "atom reward" (which is by default 10 OKE) and the transaction costs (usually in atomcoins).

• Atomic creators choose which transactions to put in an atom from the transpool according to their fee rate, which is transaction fee divided by the transaction size.

• A transaction with a high fee rate is giving a higher priority than that with a lower fee rate. Prioritizing in this way ensures maximum profitability for the atomic creator. Maximum atom size is 1MB; but transaction sizes vary. If the size of the transpool is very large, say 100MB, it can indicate how busy the atom network is at that moment, and how long one has to wait for the confirmation of a transaction.

• Transactions continuously crisscross the atom network, creating digital footprints that require careful tracking and management to maintain the integrity and reliability of the underlying atomchain.