Pointnity Network Technical White Paper

Pointnity Network Provide for the Transfer of Power Without Trust Value Network

Summary

Pointnity Network is a global business-level block chain infrastructure; Pointnity NetworkBased Programming language easily-Calculus is a highly concurrent, parallel and infinitely scalable high block chain business systems, while supporting multi-threaded parallel contract, scheduling and distributed computing resources across heterogeneous systems to interoperate chain. Thus becoming a real sense. A practical system of independent R & D Block Chaining (Text PointnityNetwork abbreviation: Pointnity).

Table of Contents

Chapter 1: Pointnity From Concept to Landing Distance

1.1 Status of Public Block Chain

Whether bit coin, square, or other public Ethernet block chain project, to business logic, a simple technology, are generated for the block only one type of block, chain end to end to form a block chain, thus bringing several universal and unavoidableAnd solve the problem:

- Data bloated: increasing amount of data, to finally reach very large, the current Bitcoin block about 180G, Ethernet Square block has more than 200G, synchronizing take weeks or even months of time, the current solution is the use of light purse, wallet lighter but the problem is that the server provider to request data lighter wallet, lost the meaning of the center to inevitably be a security risk, since there is a synchronization problem for the individual concerned then fast transaction confirmation mechanism block chain system design, have no practical significance becomes.

- Storage bottlenecks: The current design can only achieve the same block chainFull backup storage (unique) data can not be distributed for debris storage, distributed storage can not be achieved on the true meaning, but the data were placed in a lot of places, a copy of the data stored in multiple only the block chain user's hard drive for large-scale applications is critical in terms of storage, inevitably become a bottleneck, because the only flat overlay, storage solutionNever a large-scale industrial applications there is no possibility to migrate to the block chain.

- Audit mechanism is not perfect: the deployment of intelligent interactive contract in the main chain, will become more and more bloated main chain, will be getting low efficiency for real-time computing and distributed computing applications change, while the lack of application or review mechanism said the review mechanism is not perfect, will inevitably affect the safety of the main chain, will ultimately limit the scope of application and development.

- Migration high degree of difficulty: Migration for the industry, enterprise application platform difficult to block chain.

1.2 Brief Characteristics Pointnity

Based on the problems encountered by existing block chain system, we have to rethink and design Pointnity, we are different from the previous public chain system, we know you want to achieve in a relatively decentralized to the environment under the traditional system architecture high-level business application performance is difficult, and we decided to redefine the π-Calculus based on the

basis of the underlying system architecture, rather subversive raise efficiency of existing public block chain technical performance, it is also included concurrent the virtual machine model, formal verification, support for multi-threaded parallel contract, the chain store large data, cross-chain network protocols, network resources in real-time automatic replenishment system (in this section or by Pointnity ecosystem partners to develop) in order to build a high-speed, non-bifurcated, security and stability of contracts intelligent decentralized system large commercial level, there will be a large number of DAPP block chain development based Pointnity public landing a reality, Pointnity the main technical characteristics are as follows:

A characteristic: a monolithic single-threaded completely solve the TPS block chain congestion around 700 million, you can automatically add nodes and fragments, the whole network is expected to peak at around ten million TPS transmission.

Two characteristics: ultra-flux based on π-Calculus underlying virtual machine, will provide a strong concurrent parallel processing system.

Characteristics of three: multi-threaded parallel processing contract, the contract is no longer limited by the traditional line up order processing, it may be the same

Steps to enforce the contract in order to achieve optimal network performance

Four characteristics: the use of stored functions may be implemented on the large side chain file data of the audio, video and the like.

IN 5: Zeng Liang and improve the use of the conventional OSI model to the center of the network, the primary ecological supporting decentralized network computing resources to mobilize real-time systems, providing convenient source within the ecological stability of the whole computer network node. Offline protection of single-node latency issues.

Characteristics of six: support for cross-chain transfer of assets and exchange of foreign currency to support data link types of synchronization.

Characteristics Seven: at the line of self-development to the center of the store application for import and use third-party inquiry

- Introduction and explanation Pointnity infrastructure;
- Introduction Pointnity state of the art;
- for Pointnity provide ecological future roadmap;
- Provide existing application submission window, providing convenient integration into the ecological Pointnity

Mentioned herein technology, functionality and infrastructure Pointnity already undertaken or are being developed, more future plans please refer to the roadmap and the White Paper net official update.

Chapter 2: Pointnity Goal

2.1 Technical Objectives - One of the Top Commercial Grade Block Chain Infrastructure

As of Pointnity V1.0.1 released white paper, Pointnity are moving in the vision and mission of the day and night, in the future, Pointnity will continue to optimize the entire development process, so that developers can more easily use Pointnity development;

Chapter 3: Architecture and Construction Ecosystem

3.1 Safe Reliable Ecological Architecture

Pointnity according to different aspects of the transaction process functions logically divided into four node roles, so that different types of nodes can focus on different types of workload processing.

- Eco-node: hatchDapp projects, provide project incubation services, development Pointnity ecology.
- Consensus node: The Casper consensus mechanisms involved in the consensus.
- Storage nodes: memory node provide full service for all the chain data application (block outer chaining application).
- Trading nodes: provide trade confirmation services for atomic transaction chain (currency currency exchange) cross-chain services.
- Ordinary nodes: ordinary wallet is, a common user node, has sent a query basic functions of trading, exchange and other assets, all of sync blocks;
- Light node: all the sync blocks of data having a transfer, the basic functions of receiving transactions.

3.2 Diversification of Atomic Transactions in the Ecosystem

in The overall ecological Pointnity in circulation will be the economy as a whole retrospective eternal theme, a decentralized world, only proprietary system through certificate (Token) fast free flowing, it will have value in the true sense, rather than a backwater.

Given the diversification of digital asset values, rather than just a simple transaction records, it

isPointnity to build atomic diversified trading system, and ultimately the effect of large circulation.

Pointnity atomic transaction support native assets, through asset and cross-license chain transfer assets to support the direct exchange chain assets, net of fees required atomic transaction.

Future, Pointnity will promote all things digital, all things are assets of ecological objectives, for example, aA hash digital asset in a data storage module, the film rights hashed data stored in a module B, the two sides call a smart contract directly atomic transactions occur asset exchange.

3.3 Distributed Business Ecosystem, Rapid Migration to Internet Block Chain

To achieve the Internet era to era of rapid transformation block chain, and improve the eco-system expansion Pointnity, quick migration, will be particularly important in many public block chain is also polished their technique when, Pointnity the first to realize the possibility of rapid transition, ordinary developers to leverage large data link functionality + Turing-complete high-performance virtual machine interface call + modular way, the traditional Internet also can be developed based on Pointnity, while quick access to previous product Pointnity ecology, not only to achieve its own reforms will enjoy ecological opportunities and dividend brings.

3.4 Commercial-Grade Intelligence of Things to Try to Build Massive User Entry

Pointnity Crystal is an attempt Pointnity future infrastructure hardware R & D, the hardware of intelligent things to try in a short time will not be released, but Pointnity team will continue to deep plowing the field of smart-linked material.

Pointnity in the enterprise clientWill help PointnityCrystal commercial version of the block chain to build the highway, next Pointnity can provide acceleration services to any block chain system, including but not limited to CDN acceleration, node acceleration, chain plate and so on. Meanwhile Dapp development based Pointnity ecology will also enjoy the acceleration service.

At the end user level Pointnity Crystal. We will provide the following services:

Private home security cloud disk: by Pointnity Crystal, may be on Pointnity Crystal, at any location in the world secure freedom of access to their own private files private files stored on a secure terminal equipment (PC, MAC, mobile phone).

Private home media center: by Pointnity Crystal, you can download audio and video files stored on security Pointnity Crystal, anywhere can enjoy the large through a terminal (PC, mobile phones, MAC, smart television).

Unlimited extension of intelligent hardware: Pointnity Crystal is open, shared, interconnected. In the future, there will be more block chain application system login Pointnity, Pointnity Crystal will become full of infinite imagination of intelligent hardware center, assets acquired by Pointnity Crystal will be the exclusive means of fusion of ecological payments.

Pointnity Crystal future or will be Pointnity released the first smart thing associated equipment, in order to open the next line of large user traffic entrance age.

Chapter 4: Status Pointnity Technology

4.1 Pointnity Network of

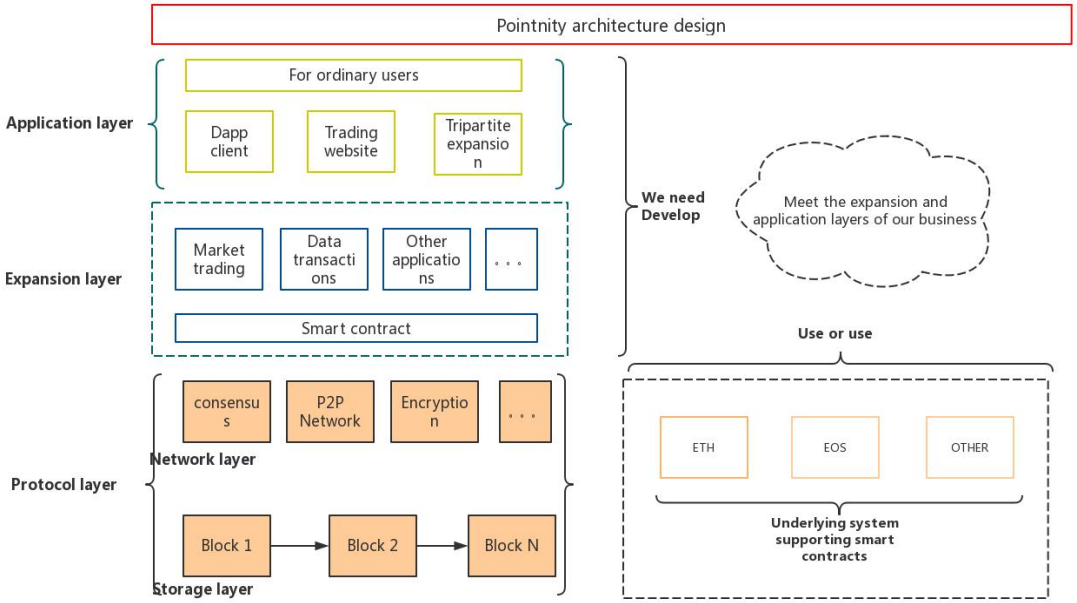

Figure 1

Pointnity architecturePointnity network configuration is divided into the following areas:

- Block chain constituting the core network infrastructure comprising: a system based on the account, the transaction atoms, consensus mechanism, data modules, protocol across the chain. Ensure that the entirePointnity ecological message broadcasting consistency, data security and storage, while ensuring a high degree of autonomy and circulation of the whole economy.

- Block chain network services the middle layer made include: the system call interface, ensure

that any developer can invoke the language of modular functions Pointnity provided by the development, the developers do not need too much common understanding of the block chain technology to large-scale development and application migration, and enjoy the convenience brought about changes in the block chain.

- The highest layer is the application layer: contains The official application developers and future ecological Pointnity provided by self-developed

Dapp, application layer will be the core focus of future Pointnity operations, massive floor application will make eco-Pointnity more healthy and robust development.

4.2 Pointnity Consensus Mechanism

There are currently many different ways to implement the algorithm proof of interest, but the two main principles of equity proof design is based PoS and based on Byzantine fault-tolerant (BFT) of PoS chain.

CAP theoretical computer science to provide more than two-thirds of return does not guarantee the possibility of distributed data systems: availability, consistency, fault-tolerant partition. PoS algorithm based chain tend to choose high availability without selecting a high consistency, because of the high availability means that all transactions can be processed, but to the network at the expense of consistency throughout the state replicate the cost. BFT based on the contrary, will tend to choose high consistency.

In Pointnity consensus mechanism, using a common algorithm (Casper Int) mechanism based on consensus-based chain

4.2.1 Casper Int "Parameters"

Our first parameter is a set of names that will label the senders of protocol messages The senders are understood to be the consensus-forming nodes, called "validators".:

Definition 2.1 (Validator names) A non-empty set:

$$\mathbb{V}$$

Without loss of generality (and in preparation for proof-of-stake), we assign a weight to each validator:. Definition 2.2 (Validator weights) A function:

$$\mathbb{W} : \mathbb{V} \to \mathbb{R}_+$$

The Casper Int protocol states are parametric in a Byzantine-fault-tolerance threshold:

Definition 2.3 (Byzantine-fault-tolerance threshold). A non-negative real number, strictly smaller than the total validator weights

$$0 \leq t < \sum_{\upsilon \in \mathbb{V}} \mathbb{W} \ (\upsilon)$$

The protocols will be guaranteed to have consensus safety tolerating up to "t Byzantine faults", measured by weight.

Different protocols in the Casper Int family can make decisions regarding different values:

Definition 2.4 (Consensus values) A multi-element set:

$$\mathrm{C}$$

In a binary consensus protocol, C = {0,1}, while in a blockchain consensus protocol, C is the set of all possible blockchains.

The final parameter, called "the estimator", relates Casper Int protocol states to sets of consensus values (subsets of C). To give the function signature of this parameter we need to define the Casper Int protocol states, S. However, we will leave this for the protocol definition section, and for now simply write:

Definition 2.5 (Estimator). A function

$$\varepsilon : \Sigma \to \mathrm{P}\left(\mathrm{C}\right) \setminus \varnothing$$

Where P is the powerset function. When the estimator returns a multi-element set, validators will have an opportunity to choose non-deterministically between consensus values. Because the domain of the estimator has not yet been defined, it is not quite proper for us to ask for the estimator as a parameter. However, for ease of presentation we will assume that we have the estimator as a parameter.

Minimal Casper Int protocol states will therefore be determined by the choice of these parameters:

• Validator names V ,

• Validator weights $\mathbb{W}$,
• Fault tolerance threshold t,
• Consensus values $\mathbb{C}$, And
• Estimator ε

4.2.2 Protocol Definition

We will define protocol states (Σ) As certain sets of messages, and messages (M) as certain triples of the form (consensus value, validator name, protocol state) Our target is therefore something that satisfies the following formulas.:

$$\Sigma \subset \mathbb{P}_{finite}(M) \tag{1}$$

$$M \subset \mathbb{C} \times \mathbb{V} \times \Sigma \tag{2}$$

We will call the consensus value the "estimate" of a message, the validator name the "sender", and the protocol state the "justification", and define these corresponding convenience functions:

Definition 2.6 (Estimate, Sender, Justification).

$$Estimate : M \to \mathbb{C} \tag{3}$$

$$Sender : M \to \mathbb{V} \tag{4}$$

$$Justi\ fication : M \to \Sigma \tag{5}$$

$$Estimate((c, \upsilon, \sigma)) := c \tag{6}$$

$$Sender((c, \upsilon, \sigma)) := \upsilon \tag{7}$$

$$Justi\ fication((c, \upsilon, \sigma)) := \sigma \tag{8}$$

We can construct finite protocol states that satisfy these equations by using the empty set of messages as the base case:

Definition 2.7 (Preliminary protocol states, S, protocol messages M).

$$\Sigma^0 := \{\varnothing\} \tag{9}$$

$$M^n := \{m \in \times \times \mathbb{C} : Estimate(m) \in \varepsilon(Justification(m))\} \tag{10}$$

$$\Sigma^n := \left\{\sigma \in \mathrm{P}_{finite}\left(M^{n-1}\right) : m \in \sigma \Rightarrow Justification\left(m\right) \subseteq \sigma\right\} \ \ for\ n > 0 \tag{11}$$

$$M := \bigcup_{i=0}^{\infty} M^i \tag{12}$$

$$\Sigma := \bigcup_{i=0}^{\infty} \Sigma^i \tag{13}$$

The type signature of the estimator is $\varepsilon : \Sigma \to \mathrm{P}\left(\mathrm{C}\right) \setminus \varnothing$, Causing possible concern due to our application of ε to $Justi\,fication\left(m\right) \in \Sigma^n \neq \Sigma$. However we note that $\Sigma^n \subset \Sigma$, Meaning that ε It is defined over Σ^n.

Note that we explicitly restricted protocol messages to only have consensus values from the estimator, allowing validators to make a choice between values of the consensus when the the estimator returns a multi-element set (but not otherwise).

We also restricted the definition of protocol states so that we could have the following definition of protocol state transitions:

Definition 2.8 (Protocol state transitions $\to$).

$$\to : \Sigma \times \Sigma \to \left\{True, False\right\} \tag{14}$$

$$\sigma_1 \to \sigma_2 :\Leftrightarrow \sigma_1 \subseteq \sigma_2 \tag{15}$$

We will write this as $\sigma \to \sigma'$ and we will write $\sigma \not\to \sigma'$ when there is not a state transition from σ to σ'.

We now pivot towards defining the protocol states for the Casper Int protocol with fault tolerance threshold t, $\Sigma_t \subset \Sigma$. But first, we need to talk about equivocation (a kind of Byzantine fault).

Equivocation faults are a Byzantine behaviour that is fundamentally enough to cause consensus failure in any consensus protocol. Casper Int protocols are guaranteed to be safe as long as not more than t weight of validators are equivocating (and therefore Byzantine).

Definition 2.9 (Equivocating messages).

$$\cdot \perp \cdot : M \times M \to \{True, False\} \tag{16}$$

$$m_1 \perp m_2 :\Leftrightarrow Sender(m_1) = Sender(m_2) \wedge m_1 \neq m_2 \tag{17}$$

$$\wedge m_1 \notin Justification(m_2) \wedge m_2 \notin Justification(m1) \tag{18}$$

Equivocating messages are distinct messages from the same validator that do not include each other in their justifications.

The equivocating validators in a protocol state are simply the validators who have equivocating messages in that state:

Definition 2.10 (Equivocating validators).

$$E : \Sigma \to \mathbb{P}(\mathbb{V}) \tag{19}$$

$$E(\sigma) := \{\upsilon \in \mathbb{V} : \exists m_1 \in \sigma, \exists m_2 \in \sigma, m_1 \perp m_2 \wedge Sender(m_1) = \upsilon\} \tag{20}$$

Equivocation corresponds to behaviour that is not "single threaded". We will eventually see a lemma showing that non-equivocating validators have at most one latest message.

Instead of considering the number of equivocations, we measure equivocations

by weight: Definition 2.11 (Equivocation fault weight).

$$F : \Sigma \to \mathbb{R}_+ \tag{21}$$

$$F(\sigma) = \sum_{\upsilon \in E(\sigma)} \mathbb{W}(\upsilon) \tag{22}$$

Note that F is a monotonic function (ie $\sigma_1 \subseteq \sigma_2 \Rightarrow F(\sigma_1) \leq F(\sigma_2)$).

The Casper Int protocol states will be parametric in an equivocation fault tolerance threshold t, and will use the following protocol states:

Definition 2.12 (Protocol states (with equivocation fault tolerance t)).

$$\Sigma_t = \{\sigma \in \Sigma : F(\sigma) \leq t\} \tag{23}$$

We think of protocol-following nodes as existing in these protocol states (Σ_t), And following the protocol state transitions ($\rightarrow$) To get from one state to another. If a node at state $\sigma \in \Sigma_t$ receives messages m such that $\sigma \bigcup \{m\} \in \Sigma$, Then it will transition to state $\sigma \bigcup \{m\}$, If it does not expose the node to too many equivocation faults (ie, as long as $F(\sigma \bigcup \{m\} \in t)$.

We are now finished the protocol definition and are ready to prove consensus safety for the family of protocols that satisfy this definition (in the context of less than t equivocations (by weight)).

4.2.3 Safety Proof

Our goal is to provide a way for nodes to make consistent decisions even if they receive different messages, as long as there are less than t equivocation faults in their protocol states. We do this in two steps, first by guaranteeing that nodes will have common future protocol states (as long as there are less than t equivocation faults in the union of their protocol states), and then by showing that their decisions on properties of protocol states will be consistent (if they share common future states, which they will if there are not too many faults). Finally, we will show how this result can be leveraged to guarantee the consistency of decisions on properties of consensus values.

4.2.3.1 Guaranteeing Common Futures

We will define the common futures of protocol states in terms of the intersection of their "futures cones":

Definition 3.1 (Futures cone in Σ_t).

$$Futures_t : \Sigma_t \rightarrow \mathbb{P} \rightarrow (\Sigma_t) \tag{24}$$

$$Futures_t(\sigma) := \{\sigma' \in \Sigma_t : \sigma \rightarrow \sigma'\} \tag{25}$$

Note that the futures function is monotonic in protocol state transitions, meaning that the futures cone "shrinks" during protocol execution:

Lemma 1 (Monotonic futures). $\forall \sigma \in \Sigma_t$, $\forall \sigma' \in \Sigma_t$,
(Forwards direction)

$$\sigma' \in Futures_t(\sigma) \Leftrightarrow \sigma \rightarrow \sigma' \tag{26}$$

$$\Leftrightarrow \sigma \to \sigma' \wedge \forall \sigma''' \in Futures_t(\sigma'), \sigma' \to \sigma''' \tag{27}$$

$$\Leftrightarrow \sigma \to \sigma' \wedge \forall \sigma''' \in Futures_t(\sigma'), \sigma' \to \sigma''' \wedge \sigma \to \sigma''' \tag{28}$$

$$\Rightarrow \forall \sigma''' \in Futures_t(\sigma'), \sigma' \to \sigma''' \wedge \sigma \to \sigma''' \tag{29}$$

$$\Rightarrow \forall \sigma''' \in Futures_t(\sigma'), \sigma' \to \sigma''' \tag{30}$$

$$\Leftrightarrow \forall \sigma''' \in Futures_t(\sigma'), \sigma''' \in Futures_t(\sigma) \tag{31}$$

$$\Leftrightarrow Futures_t(\sigma') \subseteq Futures_t(\sigma) \tag{32}$$

(Backwards direction)

$$Futures_t(\sigma') \subseteq Futures_t(\sigma) \Leftrightarrow \forall \sigma''' \in Futures_t(\sigma'), \sigma'' \in Futures_t(\sigma) \tag{33}$$

$$\Leftrightarrow \forall \sigma''' \in Futures_t(\sigma'), \sigma'' \in Futures_t(\sigma) \wedge \sigma' \in Futures_t(\sigma') \tag{34}$$

$$\Rightarrow \sigma' \in Futures_t(\sigma) \tag{35}$$

We will now present our first key theorem, which guarantees that pairs of nodes have common future states if not more than t fault weight is observed in the union of their states:

Theorem 1 (2-party common futures). $\forall \sigma_1 \in \Sigma_t$, $\forall \sigma_2 \in \Sigma_t$,

$$F(\sigma_1 \bigcup \sigma_2) \le t \Rightarrow Futures_t(\sigma_1) \bigcap Futures_t(\sigma_2) \neq \varnothing, \ \forall \sigma' \in \Sigma_t \tag{36}$$

Proof.

$$F(\sigma_1 \bigcup \sigma_2) \le t \Leftrightarrow \sigma_1 \bigcup \sigma_2 \in \Sigma_t \tag{37}$$

$$\Leftrightarrow \sigma_1 \bigcup \sigma_2 \in \Sigma_t \wedge \sigma_1 \to \sigma_1 \bigcup \sigma_2 \wedge \sigma_2 \to \sigma_1 \bigcup \sigma_2 \tag{38}$$

$$\Leftrightarrow \sigma_1 \bigcup \sigma_2 \in \Sigma_t \wedge \sigma_1 \to \sigma_1 \bigcup \sigma_2 \wedge \sigma_1 \bigcup \sigma_2 \in \Sigma_t \wedge \sigma_2 \to \sigma_1 \bigcup \sigma_2 \tag{39}$$

$$\Leftrightarrow \sigma_1 \bigcup \sigma_2 \in Futures_t(\sigma_1) \wedge \sigma_1 \bigcup \sigma_2 \in Futures_t(\sigma_2) \tag{40}$$

$$\Leftrightarrow \sigma_1 \bigcup \sigma_2 \in Futures_t(\sigma_1) \bigcap Futures_t(\sigma_2) \tag{41}$$

$$\Rightarrow Futures_t(\sigma_1) \bigcap Futures_t(\sigma_2) \neq \varnothing \tag{42}$$

Note the analogous guarantee that n nodes have a common future if there are not too many faults in

the union of their states:

Theorem 2 (n-party common futures). $\forall \sigma_i \in \Sigma_t$

$$F\left(\bigcup_{i=1}^{n} \sigma_i\right) \leq t \Rightarrow \bigcap_{i=1}^{n} Futures_t\left(\sigma_i\right) \neq \varnothing \tag{43}$$

Proof. Similar to proof of the 2-argument version.

The theorems above guarantee that protocol-following nodes have a common future state if the validators that are equivocating in the union of their states have no more than t weight. Therefore, if Byzantine (and equivocating) validators have no more than t weight, all protocol-following nodes will have a common future state.

We will use this result to ensure the consistency of decisions made by protocol-following nodes, guaranteeing the consensus safety of Casper Int consensus protocols.

Guaranteeing Consistent Decisions

In this section, we show that if nodes have a common future state, then all of the invariant properties of their protocol states are consistent. We guarantee consensus safety by insisting that nodes have decided on the invariant properties of their protocol states. We leverage the consistency of decisions on properties of protocol states to guarantee the consistency of decisions on properties of consensus values.

3.2.1 Guaranteeing Consistent Decisions on Properties of Protocol States

Before we can formally state our consensus safety theorems, we need to define some preliminary notions First, we consider properties of protocol states to be maps from protocol states to True or False.:

Definition 3.2 (Properties of protocol states).

$$P_\Sigma = \{True, False\}^\Sigma \tag{44}$$

We say that a property of a protocol state is called "decided" from some protocol state if it holds for all future protocol states:

Definition 3.3 (Decided properties of protocol states).

$$Decided_{\Sigma,t} : P_\Sigma \times \Sigma \to \{True, False\} \tag{45}$$

$$Decided_{\Sigma,t}(p,\sigma) :\Leftrightarrow \forall \sigma' \in Futures_t(\sigma), p(\sigma') \tag{46}$$

We will now prepare lemmas that we will use to prove consensus safety.

Our first lemma says that if a node is decided on a property at some state, then it will be decided on this property at its future protocol states:

Lemma 2 (Forward consistency). $\forall \sigma \in \Sigma_t,\ \forall \sigma' \in \Sigma_t,\ \forall p \in P_\Sigma,$

$$\sigma' \in Futures_t(\sigma) \Rightarrow \left(Decided_{\Sigma,t}(p,\sigma) \Rightarrow Decided_{\Sigma,t}(p,\sigma')\right) \tag{47}$$

Proof.

$$\sigma' \in Futures_t(\sigma) \wedge Decided_{\Sigma,t}(p,\sigma) \tag{48}$$

$$\Leftrightarrow Futures_t(\sigma') \subseteq Futures_t(\sigma) \wedge Decided_{\Sigma,t}(p,\sigma) \tag{49}$$

$$\Leftrightarrow Futures_t(\sigma') \subseteq Futures_t(\sigma) \wedge \forall \sigma'' \in Futures_t(\sigma), p(\sigma'') \tag{50}$$

$$\Rightarrow \forall \sigma'' \in Futures_t(\sigma'), p(\sigma'') \tag{51}$$

$$\Leftrightarrow Decided_{\Sigma,t} p(\sigma') \tag{52}$$

So now we have

$$\sigma' \in Futures_t(\sigma) \wedge Decided_{\Sigma,t}(p,\sigma) \Rightarrow Decided_{\Sigma,t}(p,\sigma') \tag{53}$$

$$\Rightarrow \sigma' \in Futures_t(\sigma) \Rightarrow \left(Decided_{\Sigma,t}(p,\sigma) \Rightarrow Decided_{\Sigma,t}(p,\sigma')\right) \tag{54}$$

Our second lemma says that if a node has a decided property at some state, then the negation that property must not have been decided at any possible past state.

Lemma 3 (Backwards consistency). $\forall \sigma \in \Sigma_t,\ \forall \sigma' \in \Sigma_t,\ \forall p \in P_\Sigma,$

$$\sigma' \in Futures_t(\sigma) \Rightarrow \left(Decided_{\Sigma,t}(p,\sigma) \Rightarrow \neg Decided_{\Sigma,t}(\neg p,\sigma)\right) \tag{55}$$

Where $\neg$ denotes negation and is defined as expected.

Proof.

$$\sigma' \in Futures_t(\sigma) \wedge Decided_{\Sigma,t}(p,\sigma') \tag{56}$$

$$\Leftrightarrow Futures_t(\sigma') \subseteq Futures_t(\sigma) \wedge Decided_{\Sigma,t}(p,\sigma') \tag{56}$$

$$\Leftrightarrow Futures_t(\sigma') \subseteq Futures_t(\sigma) \wedge \forall \sigma'' \in Futures_t(\sigma'), p(\sigma'') \tag{57}$$

$$\Rightarrow Futures_t(\sigma') \subseteq Futures_t(\sigma) \wedge \exists \sigma'' \in Futures_t(\sigma), p(\sigma'') \tag{58}$$

$$\Rightarrow \exists \sigma'' \in Futures_t(\sigma), p(\sigma'') \tag{59}$$

$$\Rightarrow \neg \forall \sigma'' \in Futures_t(\sigma), \neg p(\sigma'') \tag{60}$$

$$\Leftrightarrow \neg Decided_{\Sigma,t}(\neg p,\sigma) \tag{61}$$

So now we have

$$\sigma' \in Futures_t(\sigma) \wedge Decided_{\Sigma,t}(p,\sigma') \Rightarrow \neg Decided_{\Sigma,t}(\neg p,\sigma) \tag{62}$$

$$\Rightarrow \sigma' \in Futures_t(\sigma) \Rightarrow Decided_{\Sigma,t}(p,\sigma') \Rightarrow \neg Decided_{\Sigma,t}(\neg p,\sigma) \tag{63}$$

We will now use these two lemmas to prove two-party consensus safety. We show this result before showing n-party consensus safety as it is considerably more accessible. Because nodes only decide on properties of protocol states, the only way two nodes can make inconsistent decisions is if one node is decided on a property p, And the other is decided on its negation $\neg p$. Our two-party consensus safety theorem therefore states that two nodes do not decide on p and $\neg p$ if there are not more than t faults:

Theorem 3 (Two-party consensus safety). $\forall \sigma_1 \in \Sigma_t, \forall \sigma_2 \in \Sigma_t, \forall p \in P_\Sigma$,

$$F(\sigma_1 \bigcup \sigma_2) \leq t \Rightarrow \neg\left(Decided_{\Sigma,t}(p,\sigma_1) \wedge Decided_{\Sigma,t}(\neg p,\sigma_2)\right) \tag{64}$$

Proof.

$$F(\sigma_1 \bigcup \sigma_2) \leq t \Rightarrow Futures_t(\sigma_1) \bigcap Futures_t(\sigma_2) \neq 0 \tag{65}$$

$$\Leftrightarrow \exists \sigma \in Futures_t(\sigma_1) \bigcap Futures_t(\sigma_2) \tag{66}$$

$$\Leftrightarrow \exists \sigma \in \Sigma_t, \sigma \in Futures_t(\sigma_1) \wedge \sigma \in Futures_t(\sigma_2) \tag{67}$$

$$\Rightarrow \exists \sigma \in \Sigma_t, \left(Decided_{\Sigma,t}(p,\sigma_1) \Rightarrow Decided_{\Sigma,t}(p,\sigma)\right) \wedge \left(Decided_{\Sigma,t}(p,\sigma) \Rightarrow \neg Decided_{\Sigma,t}(\neg p,\sigma_2)\right)$$

(68)

$$\Rightarrow Decided_{\Sigma,t}\left(p,\sigma_1\right)\Rightarrow\neg Decided_{\Sigma,t}\left(\neg p,\sigma_2\right) \tag{69}$$

$$\Leftrightarrow\neg Decided_{\Sigma,t}\left(p,\sigma_1\right)\vee\neg Decided_{\Sigma,t}\left(\neg p,\sigma_2\right) \tag{70}$$

$$\Leftrightarrow\neg\left(Decided_{\Sigma,t}\left(p,\sigma_1\right)\wedge Decided_{\Sigma,t}\left(\neg p,\sigma_2\right)\right) \tag{71}$$

Note the contrapositive of this theorem, which shows that the existence of nodes at two states with inconsistent decisions implies that there are more than t weight of validators equivocating:

$$\exists p\in P_\Sigma, Decided_{\Sigma,t}\left(p,\sigma_1\right)\wedge Decided_{\Sigma,t}\left(\neg p,\sigma_2\right)\Rightarrow F\left(\sigma_1\bigcup\sigma_2\right)>t \tag{72}$$

Unfortunately, the two-party consensus safety result is not sufficient to guarantee the consistency of decisions made by more than 2 parties. To see this, consider a triple of properties p,q,r such that $p\wedge q\Rightarrow\neg r$. If three nodes decide on p,q and r respectively, then they will have made inconsistent decisions, although they may be consistent pairwise. We therefore need to understand inconsistent decisions in a more general way than we considered for the two-party consensus safety result, where looking at properties and their negations was sufficient .

We will say that properties are "inconsistent" if, for all protocol states, their conjunction is false:
Definition 3.4 (Inconsistency of properties of protocol states).

$$Inconsistent_\Sigma : p\left(P_\Sigma\right)\rightarrow\left\{True,False\right\} \tag{73}$$

$$Inconsistent_\Sigma\left(Q\right):\Leftrightarrow\forall\sigma\in\Sigma,\underset{q\in Q}{\wedge}q\left(\sigma\right)=False \tag{74}$$

Then we will say that properties are "consistent" if they are not inconsistent:

$$\neg Inconsistent_\Sigma\left(Q\right)\Leftrightarrow\neg\left(\forall\sigma\in\Sigma,\underset{q\in Q}{\wedge}q\left(\sigma\right)=False\right) \tag{75}$$

$$\Leftrightarrow\exists\sigma\in\Sigma,\underset{q\in Q}{\wedge}q\left(\sigma\right)=True \tag{76}$$

$$\Leftrightarrow\exists\sigma\in\Sigma,\forall q\in Q,q\left(\sigma\right) \tag{77}$$

Which is to say that a set of properties is consistent if there is at least one protocol state that satisifies each of them:

Definition 3.5 (Consistency of properties of protocol states).

$$Consistent_{\Sigma} : p\left(P_{\Sigma}\right) \to \left\{True, False\right\} \tag{78}$$

$$Consistent_{\Sigma}\left(Q\right) :\Leftrightarrow \forall \sigma \in \Sigma, \forall q \in Q, q\left(\sigma\right) \tag{79}$$

The n-party consensus safety result will show the consistency of all the decided properties in any of the states of our n nodes, if there are not more than t equivocation faults by weight. So, we need to define a function that collects all the decisions from protocol states:

Definition 3.6 (Decisions on properties of protocol states).

$$Decisions_{\Sigma,t} : \Sigma \to \mathrm{p}\left(P_{\Sigma}\right) \tag{80}$$

$$Decisions_{\Sigma,t}\left(\sigma\right) = \left\{p \in P_{\Sigma} : Decisions_{\Sigma,t}\left(p,\sigma\right)\right\} \tag{81}$$

Now, we state the n-party consensus safety result, that the decided properties are consistent (if there are not too many faults):

Theorem 4 (N-party consensus safety for properties of protocol states). $\forall \sigma_i \in \Sigma_t$,

$$F\left(\bigcup_{i=1}^{n} \sigma_i\right) \le t \Rightarrow Consistent_{\Sigma}\left(\bigcup_{i=1}^{n} Decisions_{\Sigma t}\left(\sigma_i\right)\right) \tag{82}$$

Proof.

$$F\left(\bigcup_{i=1}^{n} \sigma_i\right) \le t \Rightarrow \bigcap_{i=1}^{n} Futures_t\left(\sigma_i\right) \neq \emptyset \tag{83}$$

$$\Leftrightarrow \exists \sigma \in \Sigma_t, \sigma \in \bigcap_{i=1}^{n} Futures_t\left(\sigma_i\right) \tag{84}$$

$$\Leftrightarrow \exists \sigma \in \Sigma_t, \underset{i=1}{\overset{n}{\wedge}} \sigma \in Futures_t\left(\sigma_i\right) \tag{85}$$

$$\Leftrightarrow \exists \sigma \in \Sigma_t, \underset{i=1}{\overset{n}{\wedge}} \sigma \in Futures_t\left(\sigma_i\right) \tag{86}$$

$$\wedge \underset{j=1}{\overset{n}{\wedge}} \sigma \in Futures_t\left(\sigma_j\right) \Rightarrow \left(\forall p \in P_{\Sigma}, Decided_{\Sigma t}\left(p,\sigma_j\right) \Rightarrow Decided_{\Sigma t}\left(p,\sigma\right)\right) \tag{87}$$

$$\Rightarrow \exists \sigma \in \Sigma_t, \underset{i=1}{\overset{n}{\wedge}} \forall p \in P_{\Sigma,t}\left(p,\sigma_i\right) \Rightarrow Decided_{\Sigma,t}\left(q,\sigma\right) \tag{88}$$

$$\Rightarrow \exists \sigma \in \Sigma_t, \bigwedge_{i=1}^{n} \forall p \in Decideds_{\Sigma,t}(\sigma_i), Decided_{\Sigma,t}(p,\sigma_i) \Rightarrow Decided_{\Sigma,t}(p,\sigma) \tag{89}$$

$$\Leftrightarrow \exists \sigma \in \Sigma_t, \bigwedge_{i=1}^{n} \forall p \in Decideds_{\Sigma,t}(\sigma_i), Decided_{\Sigma,t}(q,\sigma) \tag{90}$$

$$\Leftrightarrow \exists \sigma \in \Sigma_t, \forall p \in \bigcup_{i=1}^{n} Decideds_{\Sigma,t}(\sigma_i), Decided_{\Sigma,t}(q,\sigma) \tag{91}$$

$$\Rightarrow \exists \sigma \in \Sigma_t, \forall p \in \bigcup_{i=1}^{n} Decideds_{\Sigma,t}(\sigma_i), p(\sigma) \tag{92}$$

$$\Leftrightarrow Consistene_{\Sigma}\left(\bigcup_{i=1}^{n} Decideds_{\Sigma,t}(\sigma_i)\right) \tag{93}$$

Note the contrapositive, that the existence of inconsistent decisions means that there must be more than t weight equiocating:

$$\neg Consistent_{\Sigma}\left(\bigcup_{i=1}^{n} Decideds_{\Sigma,t}(\sigma_i)\right) \Rightarrow F\left(\bigcup_{i=1}^{n} \sigma_i\right) > t \tag{94}$$

This concludes the consensus safety proof for decisions on properties of protocol states. We will now show how we can leverage this consensus safety theorem and the estimator to allow nodes to make consistent decisions on properties of consensus values.

4.2.3.2 Guaranteeing Consistent Decisions on Properties of Consensus Values

We will define properties of consensus values as the maps from consensus values to True or False:

Definition 3.7 (Properties of consensus values).

$$P_C = \left\{ f \in \{True, False\}^C \right\} \tag{95}$$

For every property of consensus values there is a naturally corresponding property of protocol states, which is satisfied (by a protocol state) if the property of consensus holds for all values of the estimator (at that protocol state):

Definition 3.8 (H, naturally corresponding property of protocol states).

$$H : P_C \rightarrow P_{\Sigma} \tag{96}$$

$$H(p) :\Leftrightarrow \lambda_{\sigma}.\forall c \in \varepsilon(\sigma), p(c) \tag{97}$$

For example, if we have an estimator for a binary consensus protocol, which has function signature $\varepsilon : \Sigma \to p(\{0,1\}) \setminus \emptyset$, And the property of consensus values $p \in P_{\{0,1\}}$ with $p(0) = True$ True and $p(1) = False$, Then $H(p)$ is defined to be satisifed by a protocol state σ if $\varepsilon(\sigma) = \{0\}$, But not if $\varepsilon(\sigma) = \{1\}$ or $\varepsilon(\sigma) = \{0,1\}$.

We will define the consistency of properties of consensus values in the same way that we defined consistency of properties of protocol states, namely that they are consistent if there is a value of the consensus that satisifes all of them:

Definition 3.9 (Consistency of properties of the consensus).

$$Consistent_C : p(P_C) \to \{Ture, False\} \tag{98}$$

$$Consistent_C(Q) :\Leftrightarrow \exists c \in C, \forall q \in Q, q(c) \tag{99}$$

We will now show an important lemma, that if properties of protocol states that correspond to properties of consensus values are consistent, then these properties of consensus values are also consistent:

Lemma 4. $\forall pi \in P_C$,

$$Consistent_{\Sigma}(\{H(p_1), H(p_2), \cdots, H(p_n)\}) \Rightarrow Consistent_C(\{p_1, p_2, \cdots, p_n\}) \tag{100}$$

Proof.

$$Consistent_{\Sigma}(\{H(p_1), H(p_2), \cdots, H(p_n)\}) \tag{101}$$

$$\Leftrightarrow \exists \sigma \in \Sigma, \forall i \in \{1,2,\cdots,n\}, H(p_i)(\sigma) \tag{102}$$

$$\Leftrightarrow \exists \sigma \in \Sigma, \forall i \in \{1,2,\cdots,n\}, \forall \mathrm{c} \in \varepsilon(\sigma), p_i(c) \tag{103}$$

$$\Leftrightarrow \exists \sigma \in \Sigma, \forall c \in \varepsilon(\sigma), \forall i \in \{1,2,\cdots,n\}, p_i(c) \tag{104}$$

$$\Rightarrow \exists \sigma \in \Sigma, \exists c \in \varepsilon(\sigma), \forall i \in \{1,2,\cdots,n\}, p_i(c) \tag{105}$$

$$\Leftrightarrow \exists c \in C, \forall i \in \{1,2,\cdots,n\}, p_i(c) \tag{106}$$

$$\Leftrightarrow Consistent_C(\{p_1, p_2, \cdots, p_n\}) \tag{107}$$

We will use this lemma to ensure the consistency of properties of consensus values that correspond to decided properties of protocol states (via H) We will call these properties "the decided properties of consensus values".:

Definition 3.10 (Decided on properties of consensus values).

$$Decided_{C,t} : P_C \times \Sigma_t \to \{True, False\} \tag{108}$$

$$Decided_{C,t}(p,\sigma) = Decided_{\Sigma,t}(H(p),\sigma) \tag{109}$$

In addition to deciding on properties of protocol states, nodes can also make decisions on properties of consensus values.

Definition 3.11 (Decisions on properties of consensus values).

$$Decisions_{C,t} : \Sigma_t \to p(P_C) \tag{110}$$

$$Decisions_{C,t}(\sigma) = \{q \in P_C : H(q) \in Decisions_{\Sigma,t}(\sigma)\} \tag{111}$$

We are finally able to give our last consensus safety theorem, which shows that decisions on properties of consensus values are consistent if there are not too many faults:

Theorem 5 (N-party consensus safety for properties of the consensus). $\forall \sigma_i \in \Sigma_t$,

$$F\left(\bigcup_{i=1}^{n} \sigma_i\right) \le t \Rightarrow Consistent_C\left(\bigcup_{i=1}^{n} Decisions_{C\,t}(\sigma_i)\right) \tag{112}$$

Proof.

$$F\left(\bigcup_{i=1}^{n} \sigma_i\right) \le t \tag{113}$$

$$\Rightarrow Consistent_{\Sigma}\left(\bigcup_{i=1}^{n} Decisions_{\Sigma,t}(\sigma_i)\right) \tag{114}$$

$$\Rightarrow Consistent_{\Sigma}\left(\left\{p \in \bigcup_{i=1}^{n} Decisions_{\Sigma,t}\left(\sigma_i\right) : \exists q \in P_C, p = H\left(q\right)\right\}\right) \tag{115}$$

$$\Leftrightarrow Consistent_{\Sigma}\left(\bigcup_{q \in P_C : H\left(q\right) \in \bigcup_{i=1}^{n} Decisions_{\Sigma,t}\left(\sigma_i\right)} \left\{H\left(q\right)\right\}\right) \tag{116}$$

$$\Leftrightarrow Consistent_{C}\left(\left\{q \in P_C : H\left(q\right) \in \bigcup_{i=1}^{n} Decisions_{\Sigma,t}\left(\sigma_i\right)\right\}\right) \tag{117}$$

$$\Leftrightarrow Consistent_{C}\left(\bigcup_{i=1}^{n} Decisions_{C,t}\left(\sigma_i\right)\right) \tag{118}$$

And once again, we note the contrapositive of this theorem, which shows that if there are inconsistent decisions on properties of the consensus, then it must be that more than t weight of validators are Byzantine and equivocating:

$$\neg Consistent_{C}\left(\bigcup_{i=1}^{n} Decisions_{C,t}\left(\sigma_i\right)\right) \Rightarrow F\left(\bigcup_{i=1}^{n} \sigma_i\right) > t \tag{119}$$

This concludes the safety proof for the minimal Casper Int family of protocols.

4.3 Parallel Multi-Chain

Pointnity parallel multi-chain network design :(backbone) common block chain (pendant) private chain (side chain) chain alliance, test chain. And side chains linked by two-way mixed mining technology, tokens can flow between the specific form main and side chains, using UTXO model. Side chain backbone of burden-sharing, most applications run in the side chain.

4.3.1 Backbone

The core Pointnity maintain a token run, decentralized, consensus mechanism using Pointnity working algorithm, each node of the main chain is free to join and exit the network, and to participate in the EU data read and write, the node is running with a flat interconnection topology , there is no network service center end node. Intelligent contracts are the backbone of the core application, but also the automation and intelligence foundation.

4.3.2 Side Chain

Pendant chains and alliances are divided into private chain of private chains and chain alliance is not entirely decentralized, private and alliances chain chain applications should May be provided with a

regional center, application developers can be applications running on a side chain.

Chain Alliance: Alliance each node chain usually has the corresponding entity institutions and organizations are eligible only after authorization by adding or

Exit the network. The composition of the interests of all relevant institutions and organizations of the alliance, to jointly safeguard the healthy operation of the network, with an optimized POS mechanism.

Private chain: the write access of each node chain owned private internal control, read permission is selectively opened as needed. Private chain still have the general structure block chain multi-node operation, applicable to a particular institution internal data management and auditing, which is a consensus mechanism with optimized POS system.

Test chain: Chain test undertake testing before line application development.

4.3.3 Cross-Chain Protocol (Cross-Chain Asset Trading)

To support the transmission of digital assets across the value chain, Pointnity design a protocol across the chainCCINT. For each asset needs across the chain of transmission chain on the target, inPointnity in the corresponding required to issue a pass certificate, as the target assets within Pointnity circulation credentials.

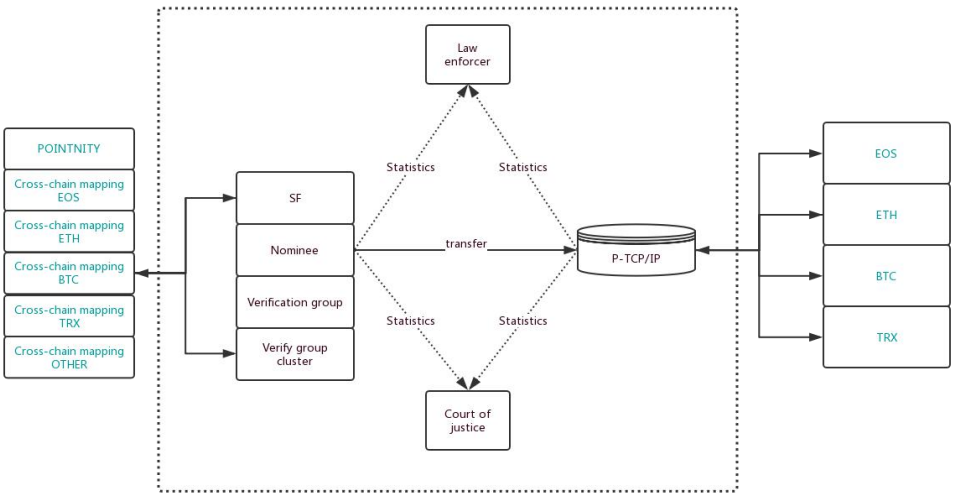

Figure 2

4.4 π-Calculus

4.4.1 Source Language Syntax

```
P :: = send E; TS; ES P Output.
 | Receive E; XS; AS P Input.
 . | Server E; XS; AS P Replicated input
 | Let X:. T = E P Local abstraction
 | If E then P1 else P2 endif Conditional
 | [P1 | P2 | ... | PN] N-ary parallel composition
 | New X P Channel creation.
 | End Null process
XS :: = ε Name list
 | X, XS
TS :: = ε Type list
 | T, TS
ES :: = ε Expression list
 | E, ES
AS :: = ε Typed argument list
 | X: T, AS
```

4.4.2 Expressions

```
E, F :: = X Variables ([a-z _] +)
 | C Channel literals ($ [a-z _] +)
 | I Integers (0 | [1-9] [0-9] *)
 | 'A' Character literals
 | "Abc" String literals
 | True | false Booleans
 | E + F Addition
 | E - F Subtraction
 | E * F Multiplucation
 | E / F Integer division
 | E% F Modulo (remainder after integer division)
 | E == F Equality
 |! E = F Inequality
 | E <F Less-than
 | E <= F Less-or-equal
 | E> F Greater-than
 | E> = F Greater-or-equal
 | E && F Logical and
 | E || F Logical or
 |! E Logical not
```

| {E, F} Tuple constructor
| <- E Left-hand tuple destructor
| -> E Right-hand tuple destructor
| [E1, E2, ..., EN] List literals
| [] Empty list
|? E Is-empty list
| * - E Head of list
| - ** E Tail of list
| E :: F List constructor
| (E) Parenthesised expression

4.4.3 Typse

T, U :: = int Integer
| Bool Boolean
| Char Char
| String String (equivalent to [char])
| [T] List
| {T, U} Pair
| @T Channel
| X Type Variable

4.5 Intelligent Scene Contract

Pointnity intelligent scene contract, which we call intelligent contracts 2.0, sufficient freedom to developers, developers can choose to use their familiar development language, Pointnity contract to provide intelligent interface exchange with chains.

Provide external interaction with the chain of intelligent JSON API provided by the main contract, all available API data exchange method as with the chain.

Additional offers two key ways to ensure that developers can develop highly efficient Dapp. These two methods are: asset creation method, create a data module method. Developers can download Dapp in the development of private beta chain development through the website development can also apply for the corresponding assets and the corresponding data module after the test chain developed, completed chains can be ported to the public.

4.5.1 Things Distributed Data Security

(1) Route reception;

(2) encryptiondistributedstorage;

(3) Redundant data permanently stored;

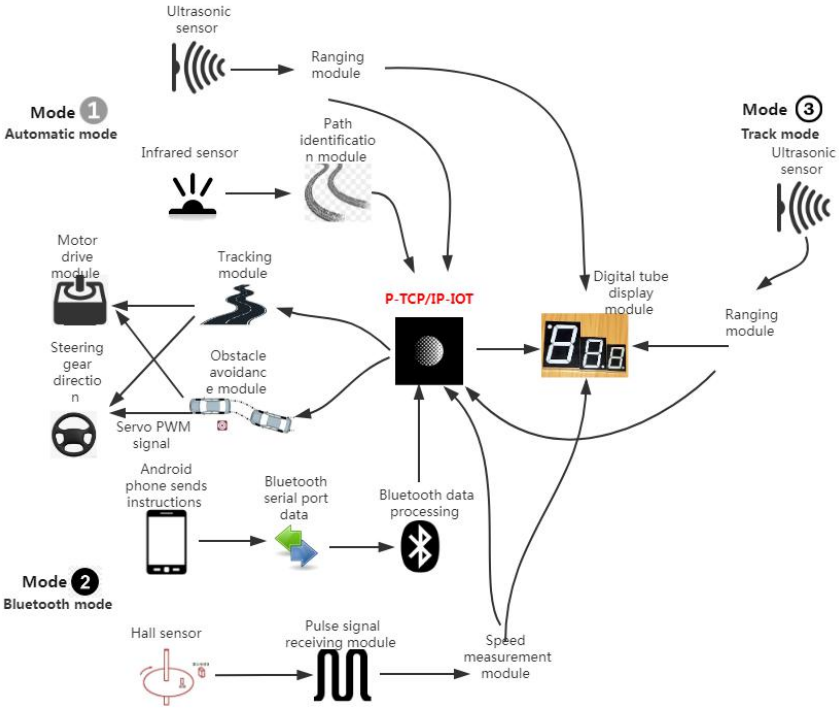

Figure 3

4.5.2 Game Scene Intelligent Contract

For example with cats

(1) Generate kittenCreate an address

Generating corresponding data block: a recording characteristic kitten

(2) Feeding kittens

Currency currency trading, the process recorded in the data module

(3) Breed

Data Module Features overlay data module included in the new cat

4.6 Operating Environment and Development Tools

4.6.1 Operating Environment

PointnityAfter the development is finished,Wide range of mainstream support the current operating

environment, including, but not limited to, a variety of household Windows, Linux, Mac, Android, IOS and other commercial and industrial operating environment. System Requirements:

Operating system environment:

Linux: 64 bit, Debian 7 +, Ubuntu14.04 +, CentOS 7 +
Windows: 64 Wei, Win7 +
Andriod: 7.0+
Hardware environment:
RAM:2G
Disk:3G

4.6.2 Underlying Code

Layer comprised PointnityScalaWriting the implementation, supportPointnity all RPC command, including multiple versions of Windows, Linux, Mac, etc.

4.6.3 Other Dapp

Pointnity support usersTuring completeUniversal language development Dapp, develop your own intelligent scene contract, Pointnity provide standard RPC interfaces and nearly 150 service API for developers to use.

Chapter 5: Pointnity Economic Model

5.1 Pointnity-TOKEN Introduction

Pointnity-TOKEN include PONT available for trading on the chain, compliance and intelligent settlement contracts, PONT is the underlying native currency, any assets of the whole chain of transactions, data exchange need to be consumed PONT.

5.1.1 PONT Native Currency

PONT application scenarios:

- Community incentive: Pointnity with PONT encourage developers and community supporters;
- Voting rights: Pointnity resolution will produce significant results have PONT supporters to vote;
- Store to download fees: Pointnity DAPP STORE download the common currency at DAPP;
- GAS: As fuel transaction or execute contracts on Pointnity intelligent network;

Accounting reward: Pointnity accounting people PONT node can get as a reward from each transaction;

PONT of Access:

- Early supporters useThe ETH Donation Exchange;
- Pointnity transfer between each user;
- participate Pointnity ecological construction, access to incentives;
- Node obtain as the billing person transactions, or performing the contract PONT;
- other methods;

5.2 Pointnity-TOKEN Distribution

5.2.1 PONT Generation and Distribution

- Total PONT50 One hundred million, never issuance
- Creation generation PONT distribution:5% PRE-SALE,Ssle 15%,Team 35%,Cooperation 25%, Community-based and other 20%
- Allocation of funds

2018 is the first year of the outbreak of the public chain, the final outcome will be a key technology maturity, comprehensive level of operational maturity and maturity assessment of ecological, technological maturity Pointnity currently in the leading international status, it will use the funds to expand the eco tilt and operational level.

- product development:50%

Block chain technology for the expansion of the core development team, training, technology and R & D centers and research institutions in the world to open Performance Technology Exhibition public chain cooperation.

- Ecological Development:17%

For Pointnity ecological projects and expansion of fixed investment, were Pointnity public ecological chain application layout, build barriers to competition, improve the overall business value and competitive business ecosystem Pointnity.

- Constituency Operations:5%

Used to promote Activity and community fission rate Pointnity partners.

- Operations Management and Security:3%

Used to attract, retain, motivate chain technology in the block, the block chain community industry has experienced management, technical, marketing, operation and maintenance personnel for project management, operation and maintenance of the project team to establish a strong operational capability, while ensuring international leading items Head technical safety.

- Marketing Promotion:15%

For lifting Pointnity international brand awareness through marketing, advertising, public relations and build brand trust, to attract developers, miners and user value investing.

Chapter 6: Governance of Pointnity

6.1 Foundation(PFUND)

Two major foundations and organizational structure by the Council working group, is responsible for the functions of the technology, operations and management, And jointly safeguard the rhythm of daily operations to ensure the pace of development.

Its functions include the appointment or dismissal of chief executive officer, make important decisions, such as the convening of an emergency meeting is the highest decision-making body of the Foundation's management. Where the following matters, subject to the Council, by secret ballot loading of decisions, each member of the Board has one vote. Council resolved to have to get all members of either the Board ofBy a majority. The Council shall perform the following duties:

(1) Modify Foundation governance structure;

(2) He decided to appoint or dismiss the Chief Executive Officer;

(3) Make important decisions path technology, business models, market direction and so on;

(4) Emergencies, such as the impact of events throughout the community, security software and system upgrade;

(5) Other matters related to major decisions.

Foundation set up a chief executive officer, responsible for the daily management of the project Pointnity, chief executive responsible to the Council, and to exercise the following duties:

(1) He presided over the daily operations management, organization and implementation of resolutions of the Council;

(2) Develop basic management system;

(3)We decided to appoint or dismiss the head of the working group;

(4)The development of open source solutions and the use of funds.

6.2 Community Governance(Pointnity V)

Foundation can set up as many working groups, each responsible for different transaction management, including technology, operations, management,Description Working Group and other projects as follows:

(6) Technical Working Group

Technical Working Group Pointnity development team composed of core developers, decision makers responsible for the direction of research and development, technological development and auditing. In addition, the technical working group in-depth understanding of the community and the industry dynamic and popular sites communicate with participants in the community, and from time to time hold technical seminars.

(7) Operations Working Group

Objective of the Working Group for the operation of community service, responsible for Pointnity technology promotion, marketing, publicity and other applications. Operations Working Group in charge of press conferences, important matters in foreign announcement and answer inquiries and so on. If any adverse events occur, the Working Group will operate as a unified external channels, authorized release response.

(8) Management Team

Personnel Management Working Group responsible for the Foundation, payroll and other administrative matters. Foundation will recruit outstanding management personnel and technical personnel, as a full-time or part-time staff of the Foundation, the Foundation will invite celebrities in various industries as a consultant. All decisions to hire and pay salaries, need to go through all the working groups, agreed to by two or more members of the Council, and ultimately take effect after being signed by the President of the Foundation.

(9) Project Working Group

The project team is responsible for designing the project plan to achieve network operation and application landing, and according to the application of the project, Community-related functions are optimized and adjusted to ensure the healthy development of the network. In the areas of community construction, applied ecology, and cutting-edge science and technology research, the project team is responsible for the establishment and promotion of the project.

(10) Asset Working Group

TheFoundation adopts multiple signatures or other technical means to ensure the security and accuracy of assets. In the principle of openness and transparency, the use of digital assets will be overseen by the Asset Working Group.

6.3 Chain Governance

The PONT holder is the chain owner and manager of the Pointnity network, which implements management by constructing a voting transaction on the Pointnity network.

Since user-initiated transactions or smart contracts take up resources in the blockchain network, users need to pay a quantitative PONT for this. PONT-related parameters are stored in the blockchain, and an adjustment algorithm is used for the community to generate new parameters based on current network development conditions.

Chapter 7: Legal Affairs and Risk Statement

7.1 Legal Structure

The Pointnity Foundation will act as an independent legal entity with the sole responsibility of organizing teams to develop, promote and operate the Pointnity project.

The Pointnity Foundation will accept donations or private placements in a proper manner in accordance with local laws and regulations, and grant them to PONT. Due to legal restrictions on national citizenship or group restrictions, PONT will not conduct public crowdfunding in certain countries or Public recruitment and other activities. PONT is used as a virtual commodity with practical uses, not a security, nor a speculative investment vehicle.

The Pointnity Foundation's revenue will be used by the Pointnity Foundation for technology development, community building, marketing, business collaboration, and financial auditing.

The Pointnity public chain is still likely to be questioned and regulated by the authorities in different countries around the world. In order to comply with and comply with local laws and regulations, the Pointnity public chain may not be able to provide normal services in some areas.

7.2 Risk Warning

Except as expressly stated in this white paper, the Foundation does not have a Pointnity public Pointnity-Token block chain or make any representations or warranties (in particular, its merchantability and fitness for a particular function).Anyone involved in PONT donation / sale and purchase plans are based on its own knowledge of itself and the Token, laws and regulations and the information in this white paper on the project. Under the premise of universality without prejudice to the foregoing, all participants will be accepted as is Pointnity-Token after Pointnity public chain project started, regardless of their technical specifications, parameters, performance or function.

This white paper lists the objectives and content may change, part of the contents of the document may be in the new version as the project progresses White Paper making adjustments or other document, the team will be announced on the website by updating or white paper or other documents peers Style, the updates available to the public.

Foundation not recognized and hereby expressly disclaims responsibility for the following:

(1) Any person who contravenes any of the country's anti-money laundering and financing terrorism or other regulatory requirements at the time of purchase;

(2) Anyone buying PONT violate any statement when the provisions of this white paper, guarante obligations, commitments or other requirements, and can not be used or can not be extracted PONT resulting;

(3) For any reason, PONT's sale plan is terminated;

(4) Pointnity development fails or is terminated, and therefore can not lead to the delivery of or inability to use PONT;

(5) Pointnity development of delayed or deferred, and therefore can not be reached in advance to be put on the agenda due;

(6) Errors, flaws, defects or other problems in the source code;

(7) Pointnity failure, collapse, paralysis, or hard rollback bifurcation;

(8) Pointnity failed to achieve any particular function or is not suitable for any particular purpose;

(9) The use of funds raised by the sale of a real plan;

(10) Failed to timely and complete disclosure about Pointnity development of information:

(11) Any participant leaked, lost or damaged Pointnity private purse;

(12) Breach of third-party distribution platform Pointnity, violation, infringement, collapse, paralysis, terminate or suspend service, fraud, misuse, misconduct, errors, negligence, bankruptcy, liquidation, dissolution or go out of business;

(13) The content of the agreement between any person and third-party distribution platforms and content in this white paper are different, conflict or contradiction;

(14) Anyone Pointnity trading or speculation;

(15) Pointnity any listed trading platform, suspension or delisting;

(16) Pointnity is classified or treated as a currency, securities, commercial paper, negotiable instrument, investment or other thing by any government, quasi-government agency, competent authority or public agency, so that it is prohibited, regulated or legal limit;

(17) Any risk factors disclosed in this white paper and related to the risk factor, resulting in or accompanying

Damage, loss, claim, liability, penalty, cost or other negative effects.

In addition, there are risks that are not mentioned or anticipated by foundations and teams. Foundations and teams are not liable for damages and risks resulting from participation, including but not limited to direct or indirect personal damage, loss of business profits, loss of business information, or other economic losses, to the fullest extent permitted by applicable law.. Participants are required to fully understand the team background, understand the overall framework and ideas of the project, and participate rationally before making participation in the decision-making process.