# Diamante Net

# INDEX


1. DIAMANTE NETWORK PAGE 3
2. THE SET OF ACTIVITIES IN SCOPE PAGE 4
3. SUMMARY PAGE 6
4. PAYCIRCLE AND CREDIT CIRCLE ON THE DIAMANTE NET PAGE 30
5. LIST OF REFERENCES PAGE 32
6. PEER-REVIEWED BY PAGE 33
7. TEAM 34

# Diamante Net

The following literature is concerned with the Infrastructure setup- Core and Horizon Server Setup, Integration of API, a creation of digital asset, Client App Integration.
The implementation phase will be carried out for the defined activities as mentioned in this document.

# THE SET OF

## ACTIVITIES IN SCOPE

**Infrastructure Setup | Set Up Diamante Blockchain Network Create Native Digital Asset Of the Blockchain Network Set Up Blockchain Network Node, Create Root Account Setting Up API Layer For Blockchain Network Decide Initial Number Of DIAM Coins | Flush Root Account With Initial Number Of Coins Creating Network Performance Dashboard | Dapps On | Creating Asset Performance Dashboard Creating Blockchain Network**

## Overview

Introspecting into deeper aspects of Diamante Network which follows the pseudonymous Blockchain Architecture thus supporting both the privately-held Blockchain ecosystem as well Decentraliez system. A decentralized network consists of peers that can run independent of each other. The power to transmit information is distributed among a network of servers, instead of being driven from one primary source. This means that the Diamante Network is independent of multiple entities and work on a single entity. The idea is to have as many independent servers participate in the network as possible so that the network will still run successfully even if some servers fail. The ledger within Diamante Network records lists of all the balances and transaction in a similar way to that of the traditional ledger. A complete copy of the individual ledger is hosted on each server that runs Diamante Net. Any entity can run a Diamante Net server. The servers all together form a decentralized network, allowing the ledgers to be distributed as much as possible. The server's sync and validate the ledger by consensus mechanism. The servers communicate and sync with each other to ensure that transactions are valid and get applied successfully to the global ledger. This entire process of coming to a consensus on this network occurs approximately every 3-5 seconds, which is a real-time settlement of the assets. The real- time settlement occurs with any of the assets present on the Blockchain network (Cachin et.al. 2017). The assets can be the Diamante Network Native asset-DIAM, Fiat Currencies, USD, EUR, Cryptocurrencies like BTC, ETH etc. and Central Bank issued cryptocurrencies.

**Anchors do two things:**

- They take the deposit and issue the corresponding credit to the individual's account address on the Diamante Network ledger.
- One can make a withdrawal by bringing them credit they issued.

One has to trust the anchor to honor their deposits and withdrawals of credit it has issued. Anchors exist in the traditional payment system. For example, to use a wallet, you deposit money in from your bank account, prefunding. The wallet then gives you credit the wallet. You can now send that wallet credit to anyone that trusts the wallet, anyone who trusts the wallet. Someone that received your wallet credit can convert it to fiat money using the wallet by withdrawing it to the bank.

The Anchors play an important role in Diamante Network. Anchors are simply entities that people trust to hold their deposits and issue credits into the Diamante Network for those deposits. They form a bridge between different currencies and the Diamante Network. All money transactions in this network occur in the form of credit issued by anchors.

The Diamante Network ledger is able to store offers that people have made to buy or sell currencies. Offers are public commitments to exchange one type of credit for another at a predetermined rate. The ledger becomes a global marketplace for offers. These offers are defined to what is known as order book. There is an order book for each currency/issuer pair. For instance, if you are wanting to exchange Commerz Bank-EUR for Bitstamp-BTC you should look at the particular order book in the ledger to see what people are buying and selling it for. This allows people to not only buy and sell currencies in a way as the authorized dealers work but also to convert currencies seamlessly during transactions. This network also allows you to send any currency you hold to anyone else in a different currency through the built-in distributed exchange.

If there is no explicit relationship between offers to buy and sell, Diamante Network tries to find offers from the network that will lead a chain of conversions from AED to USD. For example, AED to AUD, AUD to BTC, BTC to XLM, XLM to USD.

**People can receive any currency through an anchor they added. Here are a few possible ways the transaction can happen:**

- The network finds an offer on the internal USD/AED exchange for someone wanting to buy AED for USD and automatically makes the exchange between the two parties.
- Using DIAM as an intermediary currency, Diamante Network will look for offers on the network asking for USD in exchange for DIAM (the native — purely digital — currency). It will simultaneously look for an offer asking for DIAM in exchange for AED. The network makes those exchanges and sends beneficiary the credit.

If there is no explicit relationship between offers to buy and sell, Diamante Network tries to find offers from the network that will lead a chain of conversions from AED to USD. For example, AED to AUD, AUD to BTC, BTC to XLM, XLM to USD.

# SUMMARY

The Diamante Network is the decentralized network which facilitates the transaction on a real- time basis with a visibility on the documentation on a real-time basis. The distributed ledger technology makes the documents sharing more transparent and secured.

The transactions which involved a lot of trusted parties and documents can be transacted Blockchain technology.

## Technical Specifications of the Diamante Network

FBA, a crucial part of Diamante Blockchain is the first provably safe consensus mechanism to enjoy four key properties simultaneously:

**Decentralized control.** Anyone is able to participate and no central authority dictates whose approval is required for consensus (Pires 2017).

**Low latency.** In practice, nodes can reach consensus at timescales humans expect for web or payment transactions—i.e., a few seconds at most.

**Flexible trust.** Users have the freedom to trust any combination of parties they see fit. For example, a small non-profit may play a key role in keeping much larger institutions honest.

**Asymptotic security.** Safety rests on digital signatures and hash families whose parameters can realistically be tuned to protect against adversaries with unimaginably vast computing power.

### Diamante Blockchain Consensus Mechanism

Unlike non-federated Byzantine agreement, federated Byzantine agreement (FBA) addresses the problem of updating replicated state, such as a transaction ledger or certificate tree (Adya et. al. 2012). By agreeing on what updates to apply, nodes avoid contradictory, irreconcilable states. We identify each update by a unique slot from which inter-update dependencies can be inferred. For instance, slots may be consecutively numbered positions in a sequentially applied log. A mandatory to mention here, is the glossary of notations that one might need to go through, before diving into the realm of FBA. The picture below must be referred to:

| | | |
|---|---|---|
| iff | | An abbreviation of "if and only if" |
| $f(x)$ | function | The result of calculating function $f$ on argument $x$ |
| $\bar{a}$ | complement | An overbar connotes the opposite, i.e., $\bar{a}$ is the opposite of $a$. |
| $\langle a_1,\ldots,a_n\rangle$ | tuple | A structure (compound value) with field values $a_1,\ldots,a_n$ |
| $A \wedge B$ | logical and | Both $A$ and $B$ are true. |
| $A \vee B$ | logical or | At least one, possibly both, of $A$ and $B$ are true. |
| $\exists e, C(e)$ | there exists | There is at least one value $e$ for which condition $C(e)$ is true. |
| $\forall e, C(e)$ | for all | $C(e)$ is true of every value $e$. |
| $\{a,b,\ldots\}$ | set | A set containing the listed elements $(a,b,\ldots)$ |
| $\{e \mid C(e)\}$ | set-builder | The set of all elements $e$ for which $C(e)$ is true |
| $\emptyset$ | empty set | The set containing no elements |
| $\lvert S\rvert$ | cardinality | The number of elements in set $S$ |
| $e \in S$ | element of | Element $e$ is a member of set $S$. |
| $A \subseteq B$ | subset | Every member of set $A$ is also a member of set $B$. |
| $A \subsetneq B$ | strict subset | $A \subseteq B$ and $A \neq B$. |
| $2^A$ | powerset | The set of sets containing every possible combination of members of $A$, i.e., $2^A = \{B \mid B \subseteq A\}$ |
| $A \cup B$ | union | The set containing all elements that are members of $A$ or members of $B$, i.e., $A \cup B = \{e \mid e \in A \vee e \in B\}$ |
| $A \cap B$ | intersection | The set containing all elements that are members of both $A$ and $B$, i.e., $A \cap B = \{e \mid e \in A \wedge e \in B\}$ |
| $A \setminus B$ | set difference | The set containing every element of $A$ that is not a member of $B$, i.e., $A \setminus B = \{e \mid e \in A \wedge e \notin B\}$ |
| / | not | Negates a symbol's meaning. E.g., $e \notin A$ means $e \in A$ is false, while $\nexists e, C(e)$ means no $e$ exists such that $C(e)$ is true. |

Fig.1. A Glossary of Terms in FBA

An FBA system runs a consensus protocol that ensures nodes agree on slot contents. A node v can safely apply update x in slot i when it has safely applied updates in all slots upon which i depends and, additionally, it believes all correctly functioning nodes will eventually agree on x for slot i. At this point, we say v has externalized x for slot i. The outside world may react to externalized values in irreversible ways, so a node cannot later change its mind about them.

A challenge for FBA is that malicious parties can join many times and outnumber honest nodes. Hence, traditional majority-based quorums do not work. Instead, FBA determines quorums in a decentralized way, by each node selecting what we call quo- rum slices. The next subsection defines quorums based on slices. The following subsection provides some examples and discussion. Finally, we define the key properties of safety and liveness that a consensus protocol should hope to achieve.

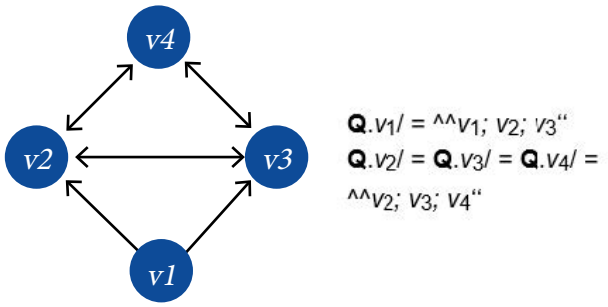



Fig. 2. $v_1$'s quorum slice is not a quorum without $v_4$.

## Quorum slices

In a consensus protocol, nodes exchange messages asserting statements about slots. We assume such assertions cannot be forged, which can be guaranteed if nodes are named by public key and they digitally sign messages. When a node hears a sufficient set of nodes assert a statement, it assumes no functioning node will ever contradict that statement. We call such a sufficient set a quorum slice, or, more concisely, just a slice. To permit progress in the face of node failures, a node may have multiple slices, any one of which is sufficient to convince it of a statement (Stellar.org, 2019). At a high level, then, an FBA system consists of a loose confederation of nodes each of which has chosen one or more slices. More formally:

***Definition (FBAS)***. A federated Byzantine agreement system, or FBAS, is a pair (V, Q) comprising a set of nodes V and a quorum function QV⊆22v \{0} specifying one or more quorum slices for each node, where a node belongs to all of its own quorum slices—i.e., Vv E V, Vq E Q(v), v E q. (Note 2X denotes the power set of X.)

**Definition (quorum).** A set of nodes U ⊆ V in FBAS (V, Q) is a quorum iff U ≠ 0 and U contains a slice for each member—i.e., Vv E U, 3q E Q(v) such that q ⊆ U.

A quorum is a set of nodes sufficient to reach agreement. A quorum slice is the subset of a quorum convincing one particular node of agreement. A quorum slice may be smaller than a quorum. Consider the four-node system in **Figure 2**, where each node has a single slice and arrows point to the other members of that slice. Node v1's slice {v1, v2, v3} is sufficient to convince v1 of a statement. But v2's and v3's slices include v4, meaning neither v2 nor v3 can assert a statement without v4's agreement. Hence, no agreement is possible without v4's participation, and the only quorum including v1 is the set of all nodes {v1, v2, v3, v4}.

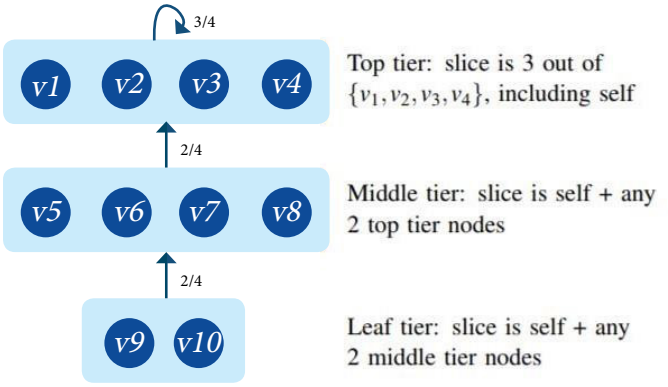



Figure 3: Tiered quorum structure example

Traditional, non-federated Byzantine agreement requires all nodes to accept the same slices, meaning Vv1, v2, Q(v1) = Q(v2). Because every member accepts every slice, traditional systems do not distinguish between slices and quorums. The downside is that membership and quorums must somehow be pre-ordained, precluding open membership and decentralized control. A traditional system, such as PBFT, typically has 3f + 1 nodes, any 2f + 1 of which constitute a quorum. Here f is the maximum number of Byzantine failures—meaning nodes acting arbitrarily— the system can survive (Stolz and Wattenhofer 2016).

FBA, introduced by this paper, generalizes Byzantine agreement to accommodate a greater range of settings. FBA's key innovation is enabling each node v to choose its own quorum slice set Q(v). System-wide quorums thus arise from individual decisions made by each node. Nodes may select slices based on arbitrary criteria such as reputation or financial arrangements. In some settings, no individual node may have complete knowledge of all nodes in the system, yet consensus should still be possible.

## Examples and Discussion

**Figure 3** shows an example of a tiered system in which different nodes have different slice sets, something possible only with FBA. A top tier, comprising v1, … , v4, is structured like a PBFT system with f = 1, meaning it can tolerate one Byzantine failure so long as the other three nodes are reachable and well-behaved. Nodes v5, … , v8 constitute a middle tier and depend not on each other, but rather on the top tier. Only two top tier nodes are required to form a slice for a middle tier node. (The top tier assumes at most one Byzantine failure, so two top tier nodes cannot both fail unless the whole system has failed.) Nodes v9 and v10 are in a leaf tier for which a slice consists of any two middle tier nodes. Note that v9 and v10 may pick disjoint slices such as {v5, v6} and {v7, v8}; nonetheless, both will indirectly depend on the top tier.

In practice, the top tier could consist of anywhere from four to dozens of widely known and trusted financial institutions. As the size of the top tier grows, there may not be exact agreement on its membership, but there will be significant overlap between most parties' notions of top tier. Additionally, one can imagine multiple middle tiers, for instance one for each country or geographic region.

This tiered structure resembles inter-domain network routing. The Internet today is held together by individual peering and transit relationships between pairs of networks. No central authority dictates or arbitrates these arrangements. Yet these pair- wise relationships have sufficed to create a notion of de facto tier one ISPs. Though Internet reachability does suffer from firewalls, transitive reachability is nearly complete—e.g., a firewall might

block The New York Times, but if it allows Google, and Google can reach The New York Times, then The New York Times is transitively reachable. Transitive reachability may be of limited utility for web sites, but it is crucial for consensus; the equivalent example would be Google accepting statements only if The New York Times does.

If we think of quorum slices as analogous to network reachability and quorums as analogous to transitive reachability, then the Internet's near complete transitive reach- ability suggests we can likewise ensure worldwide consensus with FBA. In many ways, consensus is an easier problem than inter-domain routing. While transit consumes re- sources and costs money, slice inclusion merely requires checking digital signatures. Hence, FBA nodes can err on the side of inclusiveness, constructing conservative slices with greater interdependence and redundancy than typically seen in peering and transit arrangements.

Another example not possible with centralized consensus is cyclic dependency structures, such as the one depicted in **Figure 4.** Such a cycle is unlikely to arise intention- ally, but when individual nodes choose their own slices, it is possible for the overall system to end up embedding dependency cycles. The bigger point is that, compared to traditional Byzantine agreement, an FBA protocol must cope with a far wider variety of quorum structures.

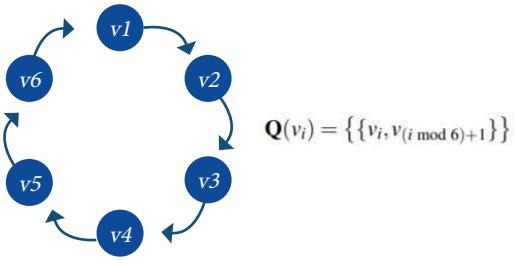



Figure 4: Cyclic quorum structure example

## Safety and liveness

We categorize nodes as either well-behaved or ill-behaved. A well-behaved node chooses sensible quorum slices (discussed further in Section 4.1) and obeys the protocol, including eventually responding to all requests. An ill-behaved node does not. Ill-behaved nodes suffer Byzantine failure, meaning they behave arbitrarily. For instance, an ill- behaved node may be compromised, its owner may have maliciously modified the soft-ware, or it may have crashed.

The goal of Byzantine agreement is to ensure that well-behaved nodes externalize the same values despite the presence of such ill-behaved nodes. There are two parts to this goal. First, we would like to prevent nodes from diverging and externalizing different values for the same slot. Second, we would like to ensure nodes can actually externalize values, as opposed to getting blocked in some dead-end state from which consensus is no longer possible. We introduce the following two terms for these properties:

**Definition (safety)**. A set of nodes in an FBAS enjoy safety if no two of them ever externalize different values for the same slot.

**Definition (liveness)**. A node in an FBAS enjoys liveness if it can externalize new values without the participation of any failed (including ill-behaved) nodes.

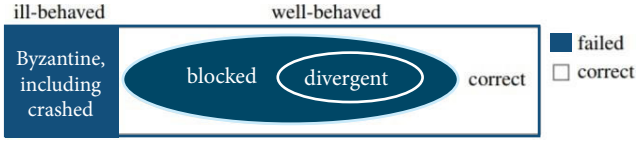



Figure 5: Venn diagram of node failures

We call well-behaved nodes that enjoy both safety and liveness correct. Nodes that are not correct have failed. All ill-behaved nodes have failed, but a well-behaved node can fail, too, by waiting indefinitely for messages from ill-behaved nodes, or, worse, by having its state poisoned by incorrect messages from ill-behaved nodes.

**Figure 5** illustrates the possible kinds of node failure. To the left are Byzantine failures, meaning the ill-behaved nodes. To the right are two kinds of well-behaved but failed nodes. Nodes that lack liveness are termed blocked, while those that lack safety are termed divergent. An attack violating safety is strictly more powerful than one violating only liveness, so we classify divergent nodes as a subset of blocked ones. Our definition of liveness is weak in that it says a node can externalize new values, not that it will. Hence, it admits a state of perpetual preemption in which consensus remains forever possible, yet the network continually thwarts it by delaying or reordering critical messages in just the wrong way. Perpetual preemption is inevitable in a purely asynchronous, deterministic system that survives node failure. Fortunately, preemption is transient. It does not indicate node failure, because the system can recover at any time. Protocols can mitigate the problem through randomness or through realistic assumptions about message latency. Latency assumptions are more practical when one would like to limit execution time or avoid the trusted dealers often required by more efficient Randomized algorithms. Of course, only termination and not safety should depend upon message timing.

## Optimal Resilience

Whether or not nodes enjoy safety and liveness depends on several factors: what quo- rum slices they have chosen, which nodes are ill-behaved, and of course the concrete consensus protocol and network behavior. As is common for asynchronous systems, we assume the network eventually delivers messages between well-behaved nodes, but can otherwise arbitrarily delay or reorder messages.

This section answers the following question: given a specific (V, Q) and particular subset of V that is ill-behaved, what are the best safety and liveness that any federated Byzantine agreement protocol can guarantee regardless of the network? We first discuss quorum intersection, a property without which safety is impossible to guarantee. We then introduce a notion of dispensable sets—sets of failed nodes in spite of which it is possible to guarantee both safety and li

## Quorum Intersection

A protocol can guarantee agreement only if the quorum slices represented by function Q satisfy a validity property we call quorum intersection.

No protocol can guarantee safety in the absence of quorum intersection, since such a configuration can operate as two different FBAS systems that do not exchange any messages. However, even with quorum intersection, safety may be impossible to guarantee in the presence of ill-behaved nodes. Compare Figure 6, in which there are two disjoint quorums, to Figure 7, in which two quorums intersect at a single

node v7, and v7 is ill-behaved. If v7 makes inconsistent statements to the left and right quorums, the effect is equivalent to disjoint quorums. In fact, since ill-behaved nodes contribute nothing to safety, no protocol can guarantee safety without the well-behaved nodes enjoying quorum intersection on their own. After all, in a worst-case scenario for safety, ill-behaved nodes can just always make any possible (contradictory) statement that completes a quorum. Two quorums overlapping only at ill-behaved nodes will again be able to operate like two different FBAS systems thanks to the duplicity of the ill-behaved nodes. In short, FBAS (V, Q) can survive Byzantine failure by a set of nodes B ⊆ V if (V, Q) enjoys quorum intersection after deleting the nodes in B from V and from all slices in Q. More formally:

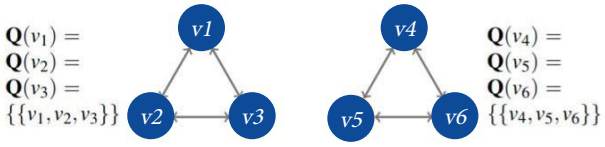



Figure 6: FBAS lacking quorum intersection

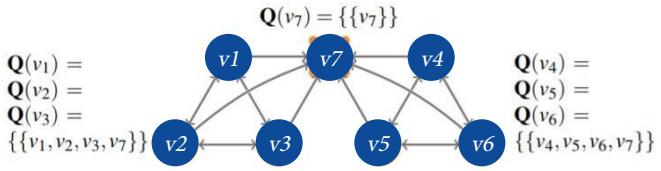



Figure 7: Ill-behaved node $v_7$ can undermine quorum intersection.

**Definition (quorum intersection).** An FBAS enjoys quorum intersection if any two of its quorums share a node—i.e., for all quorums U1 and U2, U1 ∩ U2 ≠ 0.

**Figure 6** illustrates a system lacking quorum intersection, where Q permits two quo- rums, {v1, v2, v3} and {v4, v5, v6}, that do not intersect. Disjoint quorums can independently agree on contradictory statements, undermining system-wide agreement. When many quorums exist, quorum intersection fails if any two do not intersect. For example, the set of all nodes {v1, … , v6} in **Figure 6** is a quorum that intersects the other two, but the system still lacks quorum intersection because the other two do not intersect each other.

**Definition (delete).** If (V, Q) is an FBAS and B ⊆ V is a set of nodes, then to delete B from (V, Q), written (V, Q) B means to compute the modified FBAS (V \ B, QB) where QB (v) = {q \ B I q ∈ Q (v)}.

It is the responsibility of each node v to ensure Q(v) does not violate quorum inter- section. One way to do so is to pick conservative slices that lead to large quorums. Of course, a malicious v may intentionally pick Q(v) to violate quorum intersection. But a malicious v can also lie about the value of Q(v) or ignore Q(v) to make arbitrary assertions. In short, Q(v)'s value is not meaningful when v is ill-behaved. This is why the necessary property for safety—quorum intersection of well-behaved nodes after deleting ill-behaved nodes—is unaffected by the slices of ill-behaved nodes.

Suppose **Figure 6** evolved from a three-node FBAS v1, v2, v3 with quorum intersection to a six-node FBAS without. When v4, v5, v6 join, they maliciously choose slices that violate quorum intersection and no protocol can guarantee safety for V. Fortunately, {v4,v5,v6} deleting

the bad nodes to yield (V, Q) restores quorum intersection, meaning at least {v1, v2, v3} can enjoy safety. Note that deletion is conceptual, for the sake of describing optimal safety. A protocol should guarantee safety for v1, v2, v3 without their needing to know that v4, v5, v6 are ill-behaved.

## Dispensable Sets (D Sets)

We capture the fault tolerance of nodes' slice selections through the notion of a dispensable set or D Set. Informally, the safety and liveness of nodes outside a D Set can be guaranteed regardless of the behavior of nodes inside the D Set (Yin et.al. 2013). Put another way, in an optimally resilient FBAS, if a single D Set encompasses every ill-behaved node, it also contains every failed node, and conversely all nodes outside the D Set are correct. As an example, in a centralized PBFT system with 3f + 1 nodes and quorum size 2f + 1, any f or fewer nodes constitute a D Set. Since PBFT in fact survives up to f Byzantine failures, its robustness is optimal. In the less regular example of **Figure 3,** {v1} is a D Set, since one top tier node can fail without affecting the rest of the system. {v9} is also a D Set because no other node depends on v9 for correctness. {v6, … , v10} is a D Set, because neither v5 nor the top tier depend on any of those five nodes . {v5, v6} is not a D Set, as it is a slice for v9 and v10 and hence, if entirely malicious, can lie to v9 and v10 and convince them of assertions inconsistent with each other or the rest of the system.

**Definition (D Set)**. Let (V, Q) be an FBAS and B $\subseteq$ V be a set of nodes. We say B is a dispensable set, or D Set, if:

- (Quorum intersection despite B) (V,Q) B enjoys quorum intersection, and
- (Quorum availability despite B) Either V \ B is a quorum in (V, Q) or B = V

Quorum availability despite B protects against nodes in B refusing to answer requests and blocking other nodes' progress. Quorum intersection despite B protects against the opposite— nodes in B making contradictory assertions that enable other nodes to externalize inconsistent values for the same slot. Nodes must balance the two threats in slice selection. All else equal, bigger slices lead to bigger quorums with greater overlap, meaning fewer failed node sets B will undermine quorum intersection when deleted. On the other hand, bigger slices are more likely to contain failed nodes, endangering quorum availability.

The smallest D Set containing all ill-behaved nodes may encompass well-behaved nodes as well, reflecting the fact that a sufficiently large set of ill-behaved nodes can cause well- behaved nodes

The D Sets in an FBAS are determined a priori by the quorum function Q. Which nodes are well- and ill-behaved depends on runtime behavior, such as machines getting compromised. The D Sets we care about are those that encompass all ill-behaved nodes, as they help us distinguish nodes that should be guaranteed correct from ones for which such a guarantee is impossible. To this end, we introduce the following terms:

to fail. For instance, in **Figure 3**, the smallest D Set containing v5 and v6 is {v5, v6, v9, v10}. The set of all nodes, V, is always a D Set, as an FBAS (V, Q) vacuously enjoys quorum intersection despite V and, by special case, also enjoys quorum availability despite V. The motivation for the special case is that given sufficiently many ill-behaved nodes, V may be the smallest D Set to contain all ill-behaved ones, indicating a scenario under which no protocol can guarantee anything better than complete system failure.

| | |
|---|---|
| **well-behaved / ill-behaved** | Local property of nodes, independent of other nodes (except for the validity of slice selection). |
| **intact / befouled** | Property of nodes given their quorum slices and a particular set of ill-behaved nodes. Befouled nodes are ill-behaved or depend, possibly indirectly, on too many ill-behaved nodes. |
| **correct / failed** | Property of nodes given their quorum slices, a concrete protocol, and actual network behavior. The goal of a consensus protocol is to guarantee correctness for all intact nodes. |

Fig. 8. Key properties of FBAS nodes

**Definition (intact).** A node v in an FBAS is intact iff there exists a D Set B containing all ill- behaved nodes and such that v ⊆ B.

**Definition (befouled).** A node v in an FBAS is befouled iff it is not intact.

A befouled node v is surrounded by enough failed nodes to block its progress or poi- son its state, even if v itself is well-behaved. No FBAS can guarantee the correctness of a befouled node.

However, an optimal FBAS guarantees that every intact node remains correct. **Figure 8** summarizes the key properties of nodes. The following theorems facilitate analysis by showing that the set of befouled nodes is always a D Set in an FBAS with quorum intersection.

**THEOREM 1.** Let U be a quorum in FBAS (V, Q), let B ⊆ V be a set of nodes, and let U‘ =

U \ B. If U‘≠ ∅ then U t is a quorum in (V, Q)B.

**Proof.** Because U is a quorum, every node v ∈ U has a q ∈ Q(v) such that q ⊆ U. Since U‘ ⊆ U, it follows that every v ∈ U‘ has a q E Q(v) such that q \B ⊆ U‘. Rewriting with deletion notation yields ∀v ∈ U‘, ∃ q ∈ QB (v) such that q ⊆ U‘, which, because U‘ ⊆ V \ B, means that U‘ is a quorum in (V, Q)B.

**THEOREM 2.** If B1 and B2 are D Sets in an FBAS (V, Q) enjoying quorum intersection, then B

= B1 ∩ B2 is a D Set, too.

**Proof.** Let U1 = V \ B1 and U2 = V \ B2. If U1 = θ, then B1 = V and B = B2 (a DSet), so we are done. Similarly, if U2 = θ, then B = B1, and we are done. Otherwise, note that by quorum availability despite D Sets B1 and B2, U1 and U2 are quorums in (V, Q). It follows from the definition that the union of two quorums is also a quorum. Hence V \ B = U1 u U2 is a quorum and we have quorum availability despite B.

We must now show quorum intersection despite B. Let Ua and Ub be any two quorums in (V, Q) B. Let U = U1 ∩ U2 = U2 \ B1. By quorum intersection of (V, Q), U = U1 ∩ U2 ≠ θ. But then by Theorem 1, U = U2 \ B1 must be a quorum in (V, Q) B1. Now consider that Ua \ B1 and Ua\ B2 cannot both be empty, or else Ua \ B1 and Ua \ B2 cannot both be empty or else Ua \ B = Ua would be. Hence, by Theorem 1, either Ua \B1 is a quorum in (⟨V, Q⟩ B) B1 = ⟨V,Q⟩ B1 or Ua \ B2 is a

quorum in ⟨V,Q⟩ B2 or both. In the former case, note that if Ua \ B1 is a quorum in (V, Q) B1, then by quorum intersection of (V, Q) B1, (Ua \ B1) ∩ U ≠ θ; since (Ua \ B1) ∩ U = (Ua \ B1) \ B2, it follows that Ua \ B2 ≠ θ, making Ua \ B2 a quorum in (V, Q) B2. By a similar argument, Ub \ B2 must be a quorum in (V, Q) B2. But then quorum intersection despite B2 tells us that (Ua \ B2) ∩ (Ub \ B2) ≠ θ, which is only possible if Ua ∩ Ub ≠ θ.

**THEOREM 3**. In an FBAS with quorum intersection, the set of befouled nodes is a D Set.

**Proof.** Let Bmin be the intersection of every D Set that contains all ill-behaved nodes. It follows from the definition of intact that a node v is intact if v ∉ Bmin. Thus, Bmin is precisely the set of befouled nodes. By Theorem 2, D Sets are closed under intersection, so Bmin is also a D Set.

### Federated Voting

This section develops a federated voting technique that FBAS nodes can use to agree on a statement. At a high level, the process for agreeing on some statement a involves nodes exchanging two sets of messages. First, nodes vote for a. Then, if the vote was successful, nodes confirm a, effectively holding a second vote on the fact that the first vote succeeded.

From each node's perspective, the two rounds of messages divide agreement on a statement a into three phases: unknown, accepted, and confirmed. Initially, a's status is completely unknown to a node v—a could end up true, false, or even stuck in a permanently indeterminate state. If the first vote succeeds, v may come to accept a. No two intact nodes ever accept contradictory statements, so if v is intact and accepts a, then a cannot be false.

For two reasons, however, v accepting a does not suffice for v to act on a. First, the fact that v accepted a does not mean all intact nodes can; a could be stuck for other nodes. Second, if v is befouled, then accepting a means nothing—a may be false at intact nodes. Yet even if v is befouled—which v does not know—the system may still enjoy quorum intersection of well- behaved nodes, in which case, for optimal safety, v needs greater assurance of a. Holding a second vote addresses both problems. If the second vote succeeds, v moves to the confirmed phase in which it can finally deem a true and act on it.

### Voting with open membership

A correct node in a Byzantine agreement system acts on a statement a only when it knows that other correct nodes will never agree to statements contradicting a. Most protocols employ voting for this purpose. Well-behaved nodes vote for a statement a only if it is valid. Well-behaved nodes also never change their votes. Hence, in centralized Byzantine agreement, it is safe to accept a if a quorum comprising a majority of well-behaved nodes has voted for it. We say a statement is ratified once it has received the necessary votes.

In a federated setting, we must adapt voting to accommodate open membership. One difference is that a quorum no longer

corresponds to a majority of well-behaved nodes. Another implication of open membership is that nodes must discover what constitutes a quorum as part of the voting process. To implement quorum discovery, a protocol should specify Q (v) in all messages from Q (v).

**Definition (vote).** A node v votes for an (abstract) statement a iff

- v asserts a is valid and consistent with all statements v has accepted, and
- v asserts it has never voted against a—i.e., voted for a statement that contradicts a—and v promises never to vote against a in the future.

**Definition (ratify).** A quorum Ua ratifies a statement a iff every member of Ua votes for a. A node v ratifies a iff v is a member of a quorum Ua that ratifies a.

**THEOREM 4.** Two contradictory statements a and a- cannot both be ratified in an FBAS that enjoys quorum intersection and contains no ill-behaved nodes.

**Proof.** By contradiction. Suppose quorum U1 ratifies a and quorum U2 ratifies a- By quorum intersection, ∃v ∈ U1 ∩ U2. Such a v must have illegally voted for both a and a-, violating the assumption of no ill-behaved nodes.

**THEOREM 5.** Let (V, Q) be an FBAS enjoying quorum intersection despite B, and suppose B contains all ill-behaved nodes. Let v1 and v2 be two nodes not in B. Let a and a- be contradictory statements. If v1 ratifies a then v2 cannot ratify a-.

**Proof.** By contradiction. Suppose v1 ratifies a and v2 ratifies a-. By definition, there must exist a quorum U1 containing v1 that ratified a and quorum U2 containing v2 that ratified a. By Theorem 1, since U1 \ ≠ θand U2 \ B ≠ θ, both must be quorums in (V, Q)B, meaning they ratified a and a- respectively in (V, Q)B. But (V, Q)B enjoys quorum intersection and has no ill-behaved nodes, so Theorem 4 tells us a and a- cannot both be ratified.

**THEOREM 6.** Two intact nodes in an FBAS with quorum intersection cannot ratify contradictory statements.

**Proof.** Let B be the set of befouled nodes. By Theorem 3, B is a D Set. By the definition of D Set, (V, Q) enjoys quorum intersection despite B. By Theorem 5, two nodes not in B cannot ratify contradictory statements.

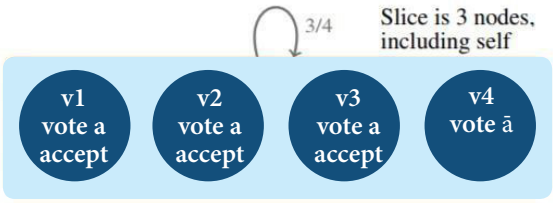



Figure 9: $v_4$ voted for $\bar{a}$, which contradicts ratified statement $a$.

# Blocking Sets

In centralized consensus, liveness is an all-or-nothing property of the system. Either a unanimously well-behaved quorum exists, or else ill-behaved nodes can prevent the rest of the system from accepting new statements. In FBA, by contrast, liveness may differ across nodes. For instance, in the tiered quorum example of **Figure 3**, if

middle tier nodes v6, v7, v8 crash, the leaf tier will be blocked while the top tier and node v5 will continue to enjoy liveness.

An FBA protocol can guarantee liveness to a node v only if Q(v) contains at least one quorum slice comprising only correct nodes. A set B of failed nodes can violate this property if B contains at least one member of each of v's slices. We term such a set B v-blocking, because it has the power to block progress by v.

**Definition (v-blocking).** Let v E V be a node in FBAS (V, Q). A set $B \subseteq V$ is v-blocking iff it overlaps every one of v's slices—i.e., $\forall q \in Q(v), q \cap B \neq \theta$.

**THEOREM 7.** Let $B \subseteq V$ be a set of nodes in FBAS (V, Q). (V, Q) enjoys quorum availability despite B iff B is not v-blocking for any $v \in V \setminus B$.

**Proof.** "$\forall v \in V \setminus B$, B is not v-blocking" is equivalent to "$\forall v \in V \setminus B, \exists q \in Q(v)$ such that $q \subseteq V \setminus B$." By the definition of quorum, the latter holds iff $V \setminus B$ is a quorum or $B = V$, the exact definition of quorum availability despite B.

As a corollary, the D Set of befouled nodes is not v-blocking for any intact v.

## Accepting Statements

When an intact node v learns that it has ratified a statement, Theorem 6 tells v that other intact nodes will not ratify contradictory statements. This condition is sufficient for v to accept a, but we cannot make it necessary. Ratifying a statement requires voting for it, and some nodes may have voted for contradictory statements. In **Figure 9,** for example, v4 votes for a- before learning that the other three nodes ratified the contradictory statement a. Though v4 cannot now vote for a, we would still like it to accept a to be consistent with the other nodes.

A key insight is that if a node v is intact, then no v-blocking set B can consist entirely of befouled nodes. Now suppose B is a v-blocking set and every member of B claims to accept statement a. If v is intact, at least one member of B must be, too. The intact member will not lie about accepting a; hence, a is true and v can accept it. Of course, if v is befouled, then a might not be true. But a befouled node can accept anything and vacuously not affect the correctness of intact nodes.

**Definition (accept).** An FBAS node v accepts a statement a iff it has never accepted a statement contradicting a and it determines that either

- There exists a quorum U such that v U and each member of U either voted for a or claims to accept a, or
- Each member of a v-blocking set claims to accept a.

Though a well-behaved node cannot vote for contradictory statements, condition 2 above allows a node to vote for one statement and later accept a contradictory one.

**THEOREM 8.** Two intact nodes in an FBAS that enjoys quorum intersection cannot accept contradictory statements.

**Proof.** Let ⟨ V, Q ⟩be an FBAS with quorum intersection and let B be its D Set of befouled nodes (which exists by Theorem 3). Suppose an intact node accepts statement a. Let v be the first intact node to accept a. At the point

v accepts a, only befouled nodes in B can claim to accept it. Since by the corollary to Theorem 7, B cannot be v-blocking, it must be that v accepted a through condition 1. Thus, v identified a quorum U such that every node claimed to vote for or accept a, and since v is the first intact node to accept a it must mean all nodes in U \B voted for a. In other words v ratified a in ⟨ V, Q ⟩B.

Generalizing, any statement accepted by an intact node in ⟨ V, Q ⟩must be ratified in ⟨ V, Q ⟩ B. Because B is a D Set, ⟨ V, Q ⟩ B enjoys quorum intersection. Because additionally B contains all ill-behaved nodes, Theorem 4 rules out ratification of contradictory statements.

**Safety**. Consider an FBAS (V, Q) in which the only quorum is unanimous consent—i.e., ∀v, Q(v) = {V}. This ought to be a conservative choice for safety—don't do anything unless everyone agrees. Yet since every node is v-blocking for every v, any node can single-handedly convince any other node to accept arbitrary statements.

The problem is that accepted statements are only safe among intact nodes. But, the only condition necessary to guarantee safety is quorum intersection of well-behaved nodes, which might hold even in the case that some well- behaved nodes are befouled. In particular, when Q(v) = {V}, the only D Sets are θ and V, meaning any node failure befouls the whole system. By contrast, quorum intersection holds despite every B ⊆ V.

**Liveness.** Another limitation of accepted statements is that other intact nodes may be unable to accept them. This possibility makes reliance on accepted statements problematic for liveness. If a node proceeds to act on a statement because it accepted the statement, other nodes could be unable to proceed in a similar fashion.

Consider **Figure 10a,** in which node v3 crashes after helping v1 ratify and accept statement a. Though v1 accepts a, v2 and v4 cannot. In particular, from v2's perspective, the situation depicted is indistinguishable from **Figure 10b,** in which v3 voted for a- and is well-behaved but slow to respond, while v1 is ill-behaved and sent v3 a vote for a- (thereby causing v3 to accept a-) while illegally also sending v2 a vote for a.

To support a protocol-level notion of liveness in cases like **Figure 10a,** v1 needs a way to ensure every other intact node can eventually accept a before v1 acts on a. Once this is the case, it makes sense to say the system agrees on a.

**Definition (agree).** An FBAS (V, Q) agrees on a statement a iff, regardless of what subsequently transpires, once sufficient messages are delivered and processed, every intact node will accept a.

### Comparison to Centralized Voting

To understand why the above issues arise in federated voting, consider a centralized Byzantine agreement system of N nodes with quorum size T. Such a system enjoys quorum availability with fL = N – T or fewer node failures. Since any two quorums share at least 2T –N nodes, quorum intersection of well-behaved nodes holds up to fS = 2T – N – 1 Byzantine failures.

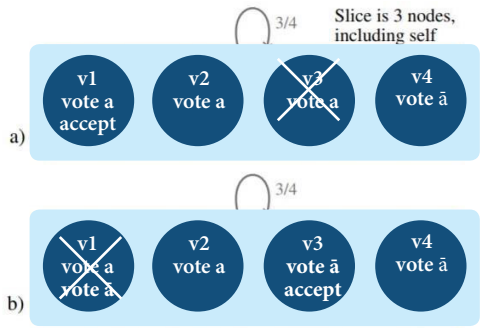



Figure 10: Scenarios indistinguishable to $v_2$ when $v_2$ does not see bold messages

Theorem 7, B cannot be v-blocking, it must be that v accepted a through condition 1. Thus, v identified a quorum U such that every node claimed to vote for or accept a, and since v is the first intact node to accept a it must mean all nodes in U \B voted for a. In other words v ratified a in ⟨ V, Q ⟩B.

Generalizing, any statement accepted by an intact node in ⟨ V, Q ⟩must be ratified in ⟨ V, Q ⟩ B. Because B is a D Set, ⟨ V, Q ⟩ B enjoys quorum intersection. Because additionally B contains all ill-behaved nodes, Theorem 4 rules out ratification of contradictory statements.

**Safety**. Consider an FBAS (V, Q) in which the only quorum is unanimous consent—i.e., ∀v, Q(v) = {V}. This ought to be a conservative choice for safety—don't do anything unless everyone agrees. Yet since every node is v-blocking for every v, any node can single-handedly convince any other node to accept arbitrary statements.

The problem is that accepted statements are only safe among intact nodes. But, the only condition necessary to guarantee safety is quorum intersection of well-behaved nodes, which might hold even in the case that some well- behaved nodes are befouled. In particular, when Q(v) = {V}, the only D Sets are θ and V, meaning any node failure befouls the whole system. By contrast, quorum intersection holds despite every B ⊆ V.

**Liveness.** Another limitation of accepted statements is that other intact nodes may be unable to accept them. This possibility makes reliance on accepted statements problematic for liveness. If a node proceeds to act on a statement because it accepted the statement, other nodes could be unable to proceed in a similar fashion.

Consider **Figure 10a,** in which node v3 crashes after helping v1 ratify and accept statement a. Though v1 accepts a, v2 and v4 cannot. In particular, from v2's perspective, the situation depicted is indistinguishable from **Figure 10b,** in which v3 voted for a- and is well-behaved but slow to respond, while v1 is ill-behaved and sent v3 a vote for a- (thereby causing v3 to accept a-) while illegally also sending v2 a vote for a.

To support a protocol-level notion of liveness in cases like **Figure 10a,** v1 needs a way to ensure every other intact node can eventually accept a before v1 acts on a. Once this is the case, it makes sense to say the system agrees on a.

**Definition (agree).** An FBAS (V, Q) agrees on a statement a iff, regardless of what subsequently transpires, once sufficient messages are delivered and processed, every intact node will accept a.

### Comparison to Centralized Voting

To understand why the above issues arise in federated voting, consider a centralized Byzantine agreement system of N nodes with quorum size T. Such a system enjoys quorum availability with fL = N – T or fewer node failures. Since any two quorums share at least 2T –N nodes, quorum intersection of well-behaved nodes holds up to fS = 2T – N – 1 Byzantine failures.

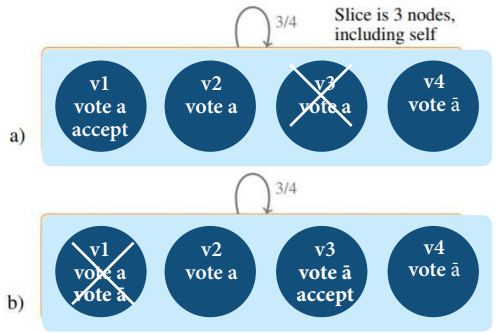



Figure 10: Scenarios indistinguishable to $v_2$ when $v_2$ does not see bold messages

Centralized Byzantine agreement systems typically set N = 3f + 1 and T = 2f + 1 to yield fL = fS = f, the equilibrium point at which safety and liveness have the same fault tolerance. If safety is more important than liveness, some protocols increase T so that fS > fL . In FBA, because quorums arise organically, systems are unlikely to find themselves at equilibrium, making it far more important to protect safety in the absence of liveness.

Now consider a centralized system in which, because of node failure and contradictory votes, some node v cannot ratify statement a that was ratified by other nodes. If v hears fS + 1 nodes claim a was ratified, v knows that either one of them is well- behaved or all safety guarantees have collapsed. Either way, v can act on a with no loss of safety. The FBA equivalent would be to hear from a set B where B, if deleted, undermines quorum intersection of well-behaved nodes. Identifying such a B is hard for three reasons: one, quorums are discovered dynamically; two, ill- behaved nodes may lie about slices; and three, v does not know which nodes are well-behaved. Instead, we defined federated voting to accept a when a v-blocking set does. The v-blocking property has the advantage of being easily checkable, but is equivalent to hearing from fL + 1 nodes in a centralized system when we really want fS + 1.

To guarantee agreement among all well-behaved nodes in a centralized system, one merely needs fL + fS + 1 nodes to acknowledge that a statement was ratified. If more than fL of them fail, we do not expect liveness anyway. If fL or fewer fail, then we know fS + 1 nodes remain willing to attest to ratification, which will in turn convince all other well-behaved nodes. The reliance on fS has no easy analogue in the FBA model. Interestingly, however, fL + fS + 1 = T, the quorum size, suggesting a similar approach might work with a more complex justification.

Put another way, at some point nodes need to believe a statement strongly enough to depend on its truth for safety. A centralized system offers two ways to reach this point for a statement a: ratify a first-hand, or reason backwards from fS + 1 nodes claiming a was ratified, figuring safety is hopeless if they have all lied. FBA lacks the latter approach; the only tool it has for safety among well-behaved nodes is first-hand ratification. Since nodes still need a way to overcome votes against ratified statements, we introduced a notion of accepting, but it provides a weaker consistency guarantee limited to intact nodes.

## Statement Confirmation

Both limitations of accepted statements stem from complications when a set of intact nodes S votes against a statement a that is nonetheless ratified. Particularly in light of FBA's non-uniform quorums, S may prevent some intact node from ever ratifying v. To provide v a means of accepting a despite votes against it, the definition of accept has a second criterion based on v-blocking sets. But the second criterion is weaker than ratification, offering no guarantees to befouled nodes that enjoy quorum intersection.

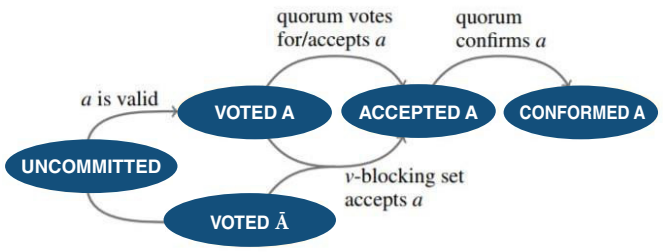



Figure 11: Possible states of an accepted statement $a$ at a single node $v$

Now suppose a statement a has the property that no intact node ever votes against it. Then we have no need to accept a and can instead insist that nodes directly ratify a before acting on it. We call such statements irrefutable.

**Definition (irrefutable).** A statement a is irrefutable in an FBAS if no intact node can ever vote against it.

Theorem 8 tells us that two intact nodes cannot accept contradictory statements. Thus, while some intact nodes may vote against a statement a that was accepted by an intact node, the statement an intact node accepted a is irrefutable. This suggests holding a second vote to ratify the fact that an intact node accepted a.

**Definition (confirm).** A quorum Ua in an FBAS confirms a statement a iff ∀v ∈ Ua, v claims to accept a. A node confirms a iff it is in such a quorum.

Nodes express that they have accepted statement a by stating "accept (a)," an abbreviation of the statement, "An intact node accepted a." To confirm a means to ratify accept (a). A well- behaved node v can vote for accept (a) only after accepting a, as v cannot assume any particular other nodes are intact. If v itself is befouled, accept (a) might be false, in which case voting for it may cost v liveness, but a befouled node has no guarantee of liveness anyway.

**THEOREM 9.** Let (V, Q) be an FBAS enjoying quorum intersection despite B, and suppose B contains all ill-behaved nodes. Let v1 and v2 be two nodes not in B. Let a and a- be contradictory statements. If v1 confirms a, then v2 cannot confirm a-.

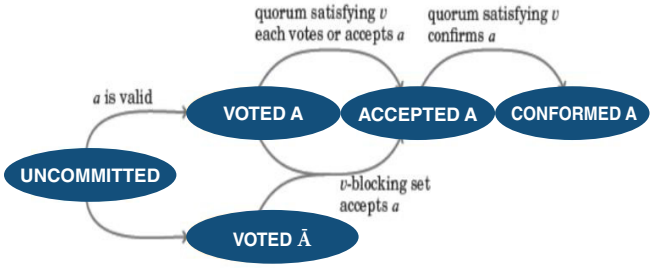



Fig. 11. Possible states of an accepted statement *a* at a single node *v*

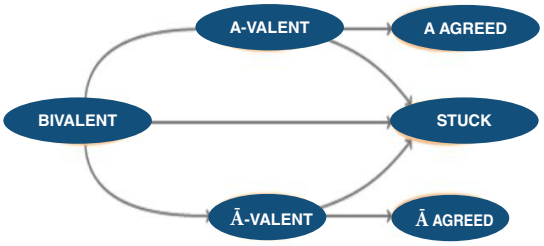



Fig. 12. Possible system-wide status of a statement *a*

**PROOF.** First note that accept (a) contradicts accept (a-)—no well-behaved node can vote for both. Note further that v1 must ratify accept (a) to confirm a. By Theorem 5, v2 cannot ratify accept (a-) and hence cannot confirm a-

**THEOREM 10.** Let B be the set of befouled nodes in an FBAS (V, Q) with quorum intersection. Let U be a quorum containing an intact node (U <J B), and let S be any set such that U⊆ S ⊆ V. Let S+ = S\B be the set of intact nodes in S, and let S– = (V\S)\B be the set of intact nodes not in S. Either S– = θ, or ∃v ∈ S– such that S+ is v-blocking.

**Proof:**

If S+ is v-blocking for some v ∈ S–, then we are done. Otherwise, we must show S– = θ. If S+ is not v-blocking for any v ∈ S–, then by Theorem 7 either S– = θ or S– is a quorum in (V,

Q) B. In the former case we are done, while in the latter we get a contradiction: By Theorem

1 U \ B is a quorum in (V, Q) B. Since B is a D Set, as proven by Theorem 3, (V, Q) B must enjoy quorum intersection, meaning $S- \cap (U \setminus B) \neq \theta$. This is impossible, since $(U \setminus B) \subseteq S$ and $S- \cap S = \theta$.

**THEOREM 11.** If an intact node in an FBAS (V, Q) with quorum intersection confirms a statement a, then, whatever subsequently transpires, once sufficient messages are delivered and processed, every intact node will accept and confirm a.

**Proof.** Let B be the D Set of befouled nodes and let $U \nsubseteq B$ be the quorum through which an intact node confirmed a. Let nodes in U \ B broadcast accept (a). By definition, any node v, regardless of how it has voted, accepts a after receiving accept (a) from a v-blocking set. Hence, these messages may convince additional nodes to accept a. Let these additional nodes in turn broadcast accept (a) until a point is reached at which, regardless of future communication, no further intact nodes can ever accept a. At this point let S be the set of nodes that claim to accept a (where $U \subseteq S$), let S+ be the set of intact nodes in S, and let S– be the set of intact nodes not in S. S+ cannot be v blocking for any node in S–, or else more nodes could come to accept a. By Theorem 10, then, S– = 0, meaning every intact node has accepted a.

**Figure 11** summarizes the paths an intact node v can take to confirm a. Given no knowledge, v might vote for either a or the contradictory a-. If v votes for a-, it cannot later vote for a, but can nonetheless accept a if a v-blocking set accepts it. A subsequent quorum of confirmation messages allows v to confirm a, which by Theorem 11 means the system agrees on a.

**Liveness and Neutralization**

The main challenge of distributed consensus, whether centralized or not, is that a statement can get stuck in a permanently indeterminate state before the system reaches agreement on it. Hence, a protocol must not attempt to ratify externalized values directly. Should the statement “The value of slot i is x” get stuck,

the system will be forever unable to agree on slot i, losing liveness. The solution is to craft the statements in votes carefully. It must be possible to break a stuck statement’s hold on the question we really care about, namely slot contents. We call the process of obsoleting a stuck statement neutralization.

More concretely, **Figure 12** depicts the potential status a statement a can have system- wide. Initially, the system is bivalent, by which we mean there is one sequence of possible events through which all intact nodes will accept a, and another sequence through which all intact nodes will reject a (i.e., accept a statement a- contradicting a). At some point, one of these two outcomes may cease to be possible. If no intact node can ever reject a, we say the system is a valent; conversely, if no intact node can ever accept a, we say the system is a-valent.

To preserve the possibility of consensus, a protocol must ensure that every statement is either irrefutable, and hence cannot get stuck, or neutralizable, and hence cannot block progress if stuck. There are two popular approaches to crafting neutralizable statements: the view-based approach, pioneered by viewstamped replication (Oki and Liskov 1988) and favored by PBFT (Castro and Liskov 1999); and the ballot-based approach, invented by Paxos (Lamport 1998). The ballot-based approach may be harder to understand (Ongaro and Ousterhout 2014). Compounding confusion, people often call view stamped replication "Paxos" or assert that the two algorithms are the same when they are not (Van Renesse, Schiper and Schneider 2015).

At the time an FBAS transitions from bivalent to a-valent, there is a possible out- come in which all intact nodes accept a. However, this might not remain the case. Consider a PBFT- like four-node system {v1, … , v4} in which any three nodes constitute a quorum. If v1 and v2 vote for a, the system becomes a-valent; no three nodes can ratify a contradictory statement. However, if v3 and v4 subsequently vote for a- contradicting a, it also becomes impossible to ratify a. In this case, a's state is permanently indeterminate, or stuck.

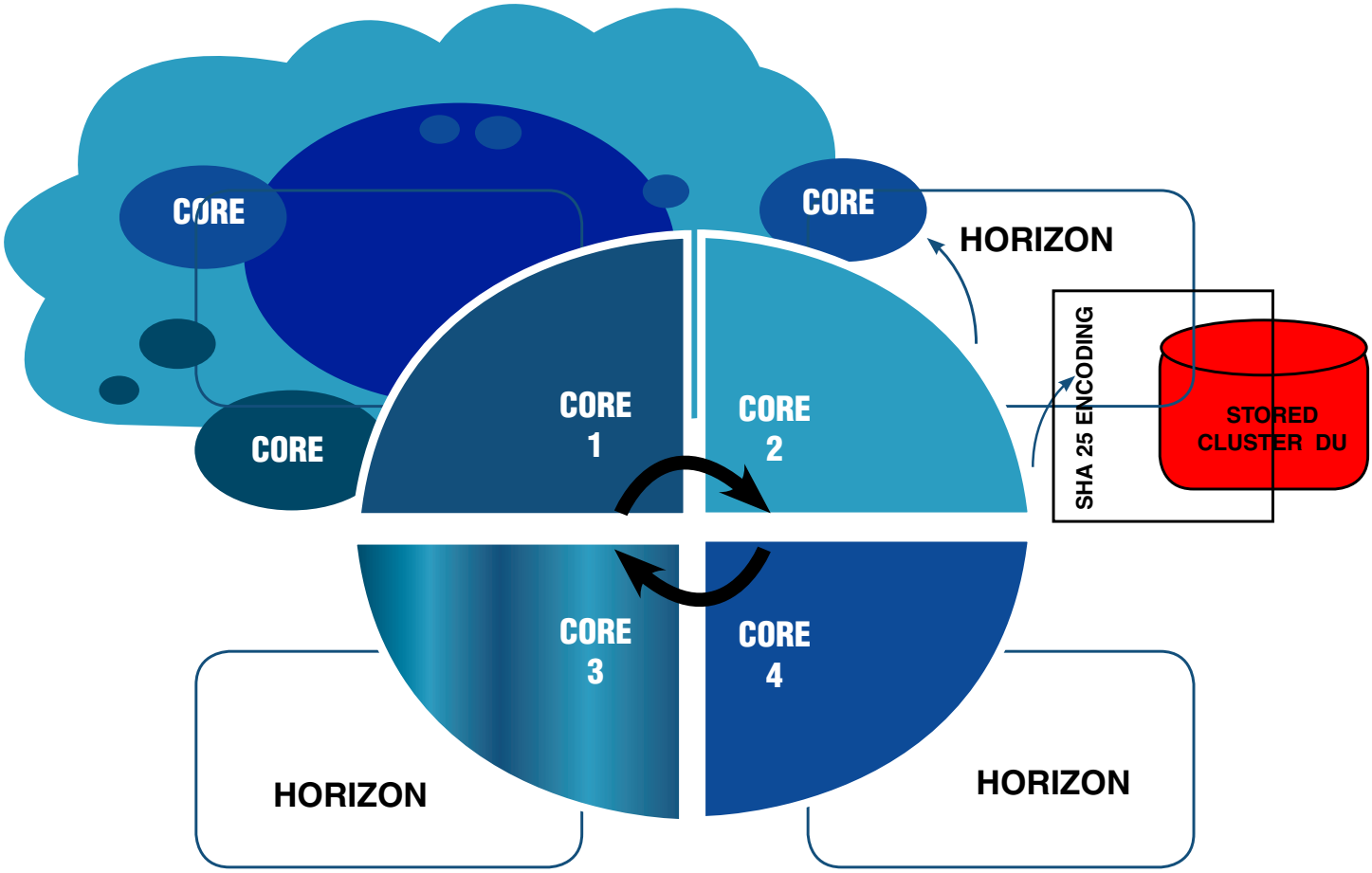
As seen in **Figure 10a,** even once an intact node accepts a, the system may still fail to reach system-wide agreement on a. However, by Theorem 11, once an intact node confirms a, all intact nodes can eventually come to accept it; hence the system has agreed upon a. **Figure 13** summarizes what intact nodes know about the global state of a statement from their own local state.

View-based protocols associate the slots in votes with monotonically increasing view numbers. Should consensus get stuck on the ith slot in view n, nodes recover by agreeing that view n had fewer than i meaningful slots and moving to a higher view number. Ballot-based protocols associate the values in votes with monotonically increasing ballot numbers. Should a ballot get stuck, nodes retry the same slot with a higher ballot, taking care never to select values that would contradict prior stuck ballots. This work takes a ballot-based approach, as doing so makes it easier to do away with the notion of a distinguished primary node or leader. For example, leader behavior can be emulated (Lamport 2011).

| Local state | System-wide status of $a$ |
|---|---|
| uncommitted | unknown (any) |
| voted $a$ | unknown (any) |
| voted $\bar{a}$ | unknown (any) |
| accepted $a$ | stuck, $a$-valent, or $a$ agreed |
| confirmed $a$ | $a$ agreed |

Fig. 13. What an intact node knows about the status of statement $a$

# Blockchain Architecture




# Blockchain Core & Horizon Integration

**Network Details**

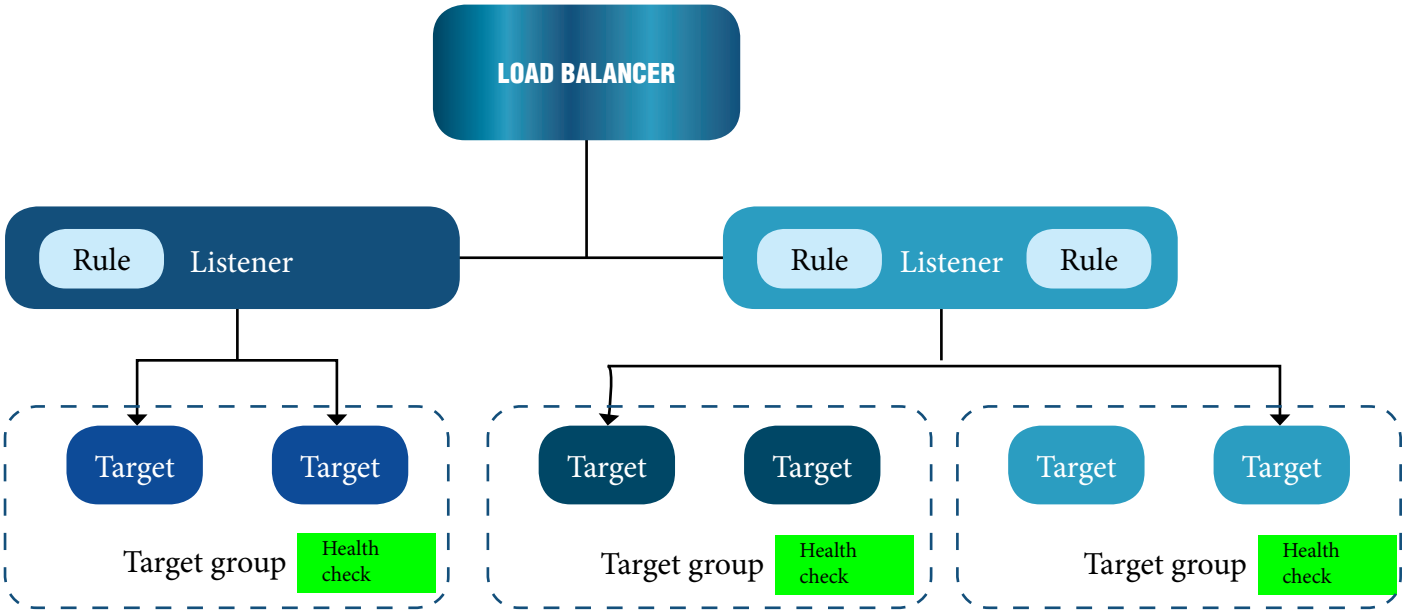
| Features | Details |
|---|---|
| **Transaction Per Second (TPS)** | 3000 |
| **Thread Mechanism** | De-coupled web service & Multithreaded Response supporting 100000 concurrent service executions per second |
| **Scaling Support** | Multiple nodes supporting Horizontal Scaling |
| **Request – Response Caching** | Caching enabled on the request node |
| **Encryption Standard** | AES -256 Encryption support with MD5 for data decryption |
| **Key Management** | Exchange maintains the Private & Public Keys with Oracle in separate instance |
| **Consensus Algorithm** | FBFT (Forward Byzantine Fault Tolerance) Algorithm for consensus |

# Infrastructure Scalability Using Cloud & Load Balancer

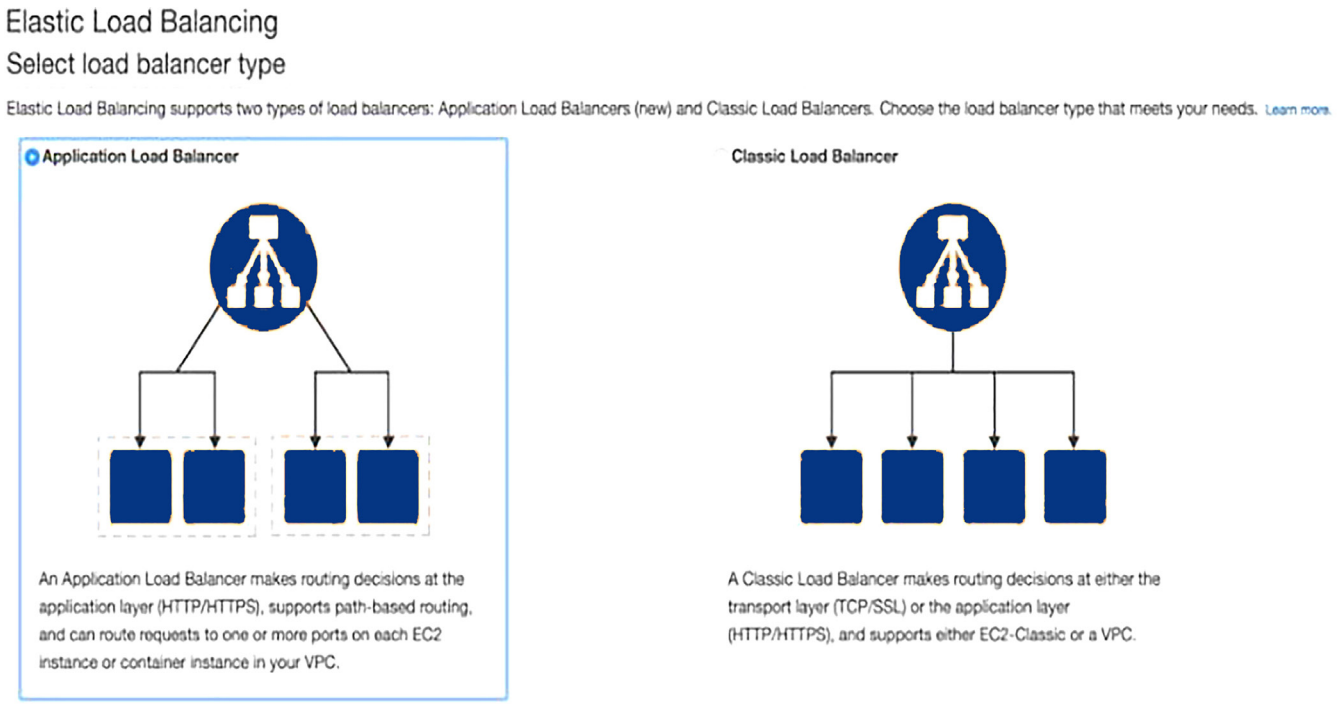
**Key features of Application Load Balancers include:**

- Path-based routing – URL-based routing policies enable using the same ELB URL to route to different micro services
- Multiple ports routing on the same server
- AWS integration – Integrated with many AWS services, such as ECS, IAM, Auto Scaling, and Cloud Formation
- Application monitoring – Improved metrics and health checks for the application

# PROPOSED INFRASTRUCTURE COMPONENT DETAILS



# PROPOSED LOAD BALANCER TYPE



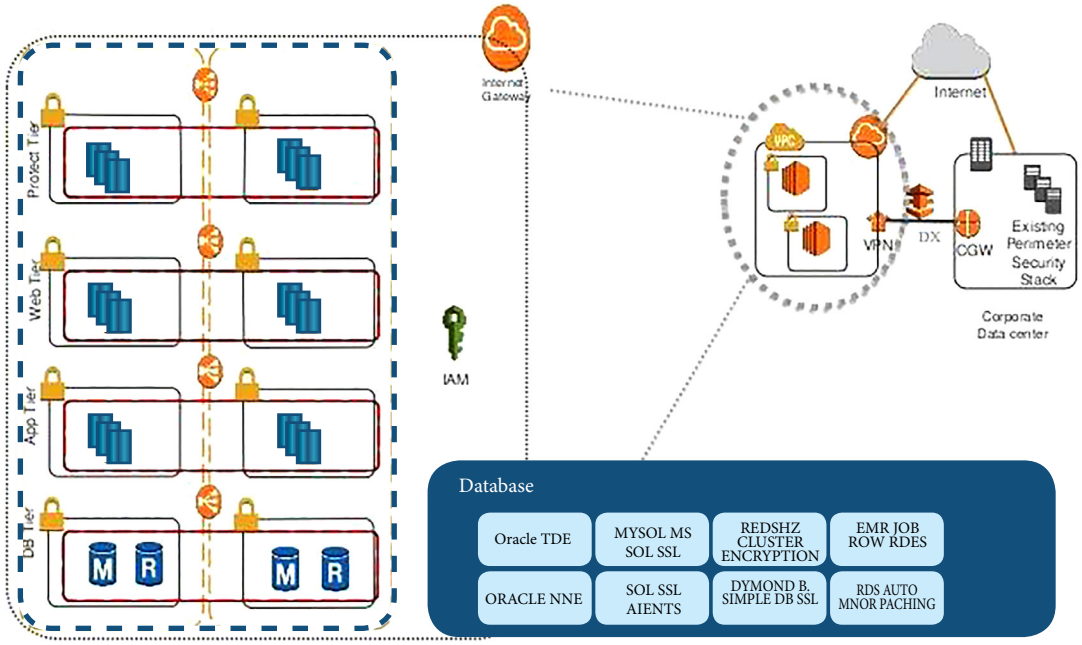

Load Balancer reduces the response latency and can achieve approximately 50,000 TPS. With multiple Load balancer, the TPS can be scaled above 1,50,000 TPS. Considering the below Minimum Hardware and Network specifications –

# SYSTEM REQUIREMENT SPECIFICATIONS

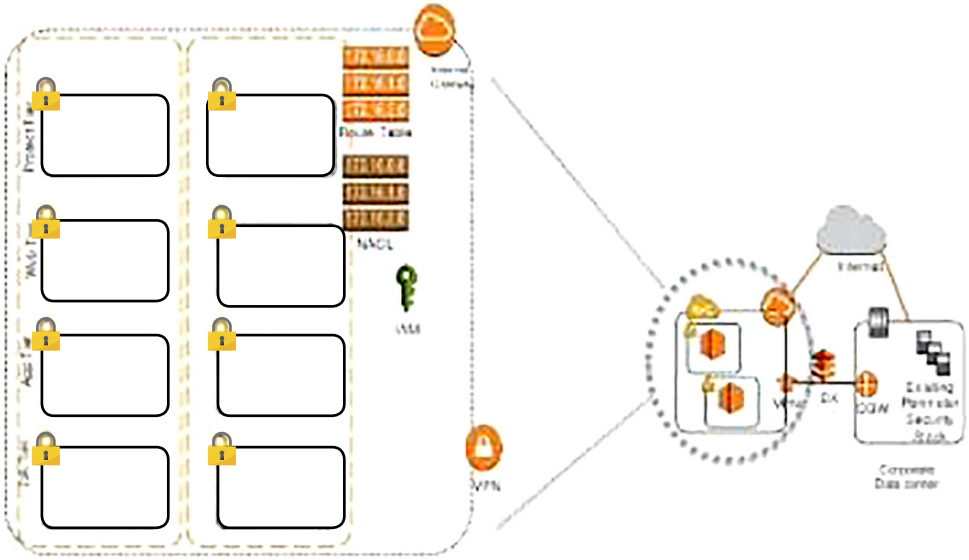
| SEGMENT | MINIMUM REQUIREMENT |
|---|---|
| **CPU** | 4CORES |
| **INTERNAL MEMORY** | 16 GB |
| **CACHE** | 2 GB |
| **BANDWIDTH** | 3.5 Gbps Dedicated EBS bandwidth |
| **OS** | Linux |
| **DATABASE** | Postgress & Oracle (Dev Edition) |

# SECURITY & COMPLIANCE INTEGRATION

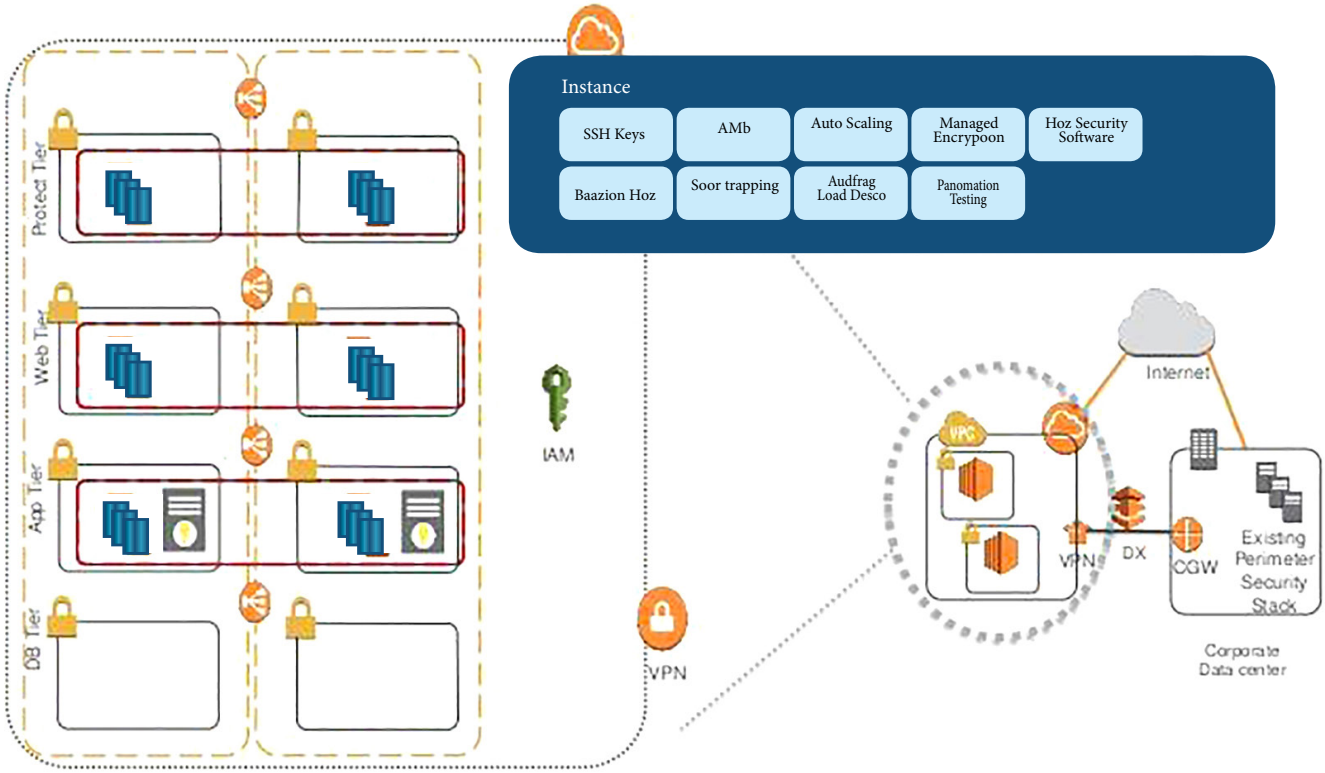
## Database Protection



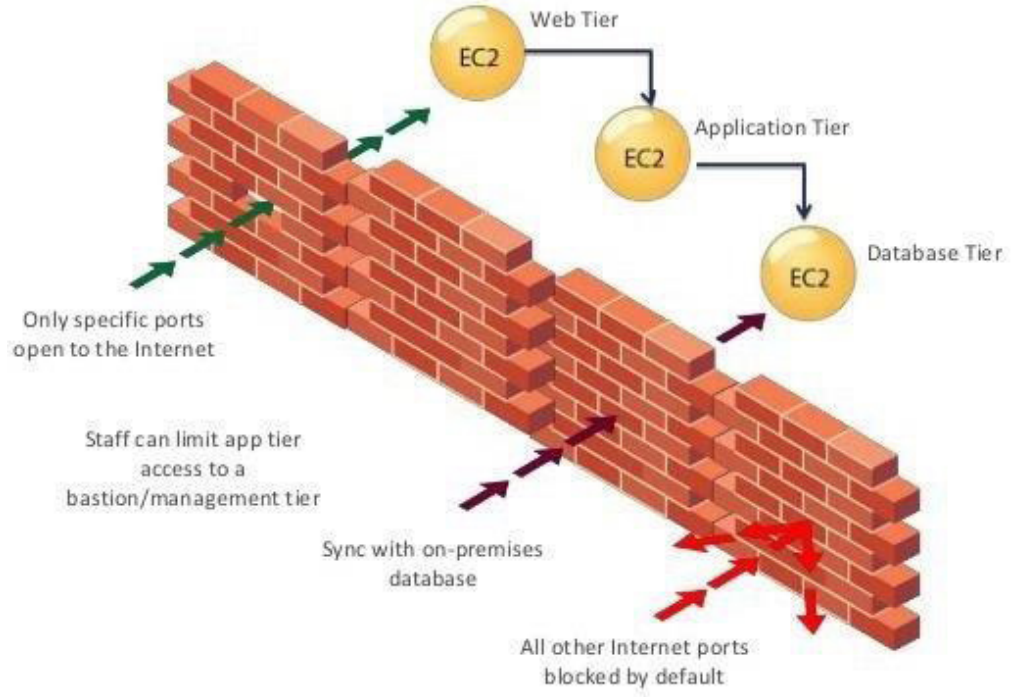

## Network Protection



# Instance Protection




# VPC Security Groups



# PayCircle and CreditCircle on the Diamante Net

To intensify the experience of each and every consortium member, Diamante Blockchain has teamed up with industry experts and technical consultants, and the pioneers in the Blockchain industry. This collaboration will facilitate smooth and flawless functioning within the Consortium using the decentralized ledger, a boon of the Blockchain Network and technology (Holotiuk, Pisani and Moormann 2017).

## PayCircle

### Introduction

The following paragraphs reflect the prospectus for the implementation of the Diamante Blockchain's application, PayCircle. PayCircle is a DeFi payment application built for global businesses and individuals. PayCircle allows companies and individuals to Custody, Send & Receive multi-currency (USD/AUD/CAD/EUR/JPY) Fiat and Digital assets (BTC, ETH, ERC20 based tokens and stable coins like USDT) anytime & anywhere, 24/7 and 365 days using Blockchain (Piazza 2017).

PayCircle
A Diamante Business

### Product Description

PayCircle is a DeFi payment application built for global businesses and individuals using Blockchain.

### Product Benefits

- Utilizing blockchain as technological infrastructure, PayCircle allows relatively speedy and low-cost transaction settlements.
- Ensures enhanced security of the financial contracts and facilitates contract automation utilizing blockchain immutability.
- Forges a bond of reliability through advanced visibility of the transactions while guaranteeing zero payment failure.
- PayCircle users remain in absolute control of their assets without any interference from any intermediary through complete possession of their private keys within the PayCircle ecosystem.

### Key Features of Individual / Business Account

- **Store Digital Assets & Fiat**
  - FDIC insured USD custody; your account is insured up to $5 million.
  - Custody your Digital Assets and Fiat in a single wallet.
- **Send/Receive Digital Assets & Fiat**
  - Send or receive Digital Assets and Fiats to your family and customers globally.
- **Digital Security**
  - PayCircle enables transactions that have absolute digital security from source till settlement.
- **Load Digital Assets and Fiat**
  - Load Fiat directly from your bank account via ACH / Wire / International Wire / Swift / Checks / Credit Cards / Debit Cards and digital assets now from any wallet.
- **Instant Settlement**
  - PayCircle ensures fast and easy transaction settlements across the world with industry best conversion rates.

- **Fraud Protection**
  - PayCircle has a novel network design that guarantees payment value against any fraud attempt, including hack or breach.
- **Open-Source**
  - The payment infrastructure is built on top of an open-source protocol, which is validated by security audits.
- **Global Payments**
  - By enabling PayCircle once, one can start accepting payments in multiple currencies from across the world without having to pay any foreign exchange fees.
- **Cost-effective Platform**
  - There is no hidden or additional fee charged, and the transaction fees are the lowest in the industry.
- **Best Digital Asset Prices**
  - Enables individuals to buy digital assets at the best market price from OTC markets and affiliated exchanges.

### All-in-One Keep-Safe for Digital Assets

PayCircle allows you to take complete control of all your digital assets enabling storage on your device.

- **Multi-Asset Support**
  - The ability to manage BTC, ETH, BCH, LTC, and all your ERC-20 tokens.
- **Secure Storage**
  - All your keys are protected with Secure Enclave and advanced authentication technology.

## CreditCircle

### Introduction

CreditCircle is a DeFi finance product on Diamante Net. It's a decentralized finance application where individuals and businesses can opt for loans and credit at a relatively low-interest rate compared to traditional financing. To opt for loans on CreditCircle, users will have to join the Diamante Consortium platform.

Members of the Diamante Consortium platform will be able to access credit in a number of ways. In the first instance, consortium members will be able to apply for a credit facility directly from Diamante. In addition to Diamante providing credit to consortium members, traditional lenders and credit providers such as banks will be able to access the Diamante platform by paying a platform fee, which will, in turn, allow them to access data and connect with potential borrowers directly.

Diamante would have two financing options, one being a peer-to-peer (p2p) lending platform, and the other is asset-backed financing.

Members' credit ratings will be mapped using high-end technical and financial algorithms that will assist the consortium in approving the line of credit to its members. Before issuing the line of credit to the specific consortium member, diamonds would be used as a means of collateral, which would be secured from the individuals seeking the credit.

Diamante has a concept of asset-backed financing where diamonds would be used as an asset that would act as collateral. For example, if a company is in a cash crunch and requires a credit of $1 million, the company would place $2 million worth of diamonds on to a 3rd party safe deposit vault, which would be valued by the Diamante appraisal team. All these transactions would be recorded via smart contracts on to the Diamantes network, i.e., Diamante Net.

### Product Description

CreditCircle, a corporate finance platform is Diamante Blockchain's product to opt for loans and credit at a relatively low-interest rate.

### Product Benefits

- Opt loans at relatively low-interest rates
- Flexible financing options based on credit ratings
- Instant and easy access to traditional lenders and credit providers
- Smart contract records for safe and secured credits

### Product Features

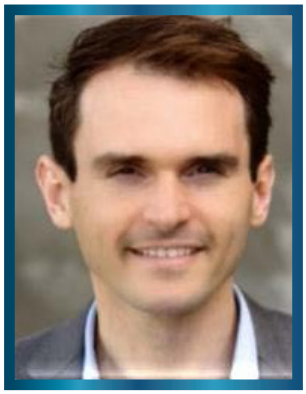
- Users can escrow their stocks, bonds, luxury commodities, and digital assets to avail of financing
- Peer-to-peer (p2p) lending, where individuals and financing institutions from around the world can get access to a large number of customers globally

# LIST OF REFERENCES


- Cachin, C., 2016, July. Architecture of the hyperledger Blockchain fabric. In Workshop on Distributed Cryptocurrencies and Consensus Ledgers (Vol. Cachin, C., Osborne, S.S., Sorniotti, A. and Vukolic, M., 2017. Blockchain and consensus protocols
- Castro, M. and Liskov, B., 1999, February. Practical Byzantine fault tolerance. In OSDI (Vol. 99, pp. 173 186).
- Holotiuk, F., Pisani, F. and Moormann, J., 2017. The impact of Blockchain technology on business models in the payments industry.
- Lamport, L., 1998. The part time parliament. ACM Transactions on Computer Systems (TOCS) TOCS), 16 ( pp.133 169.
- Lamport, L., 2011, September. Brief announcement: Leaderless byzantine paxos. In International Symposium on Distributed Computing (pp. 141 142). Springer, Berlin, Heidelberg.
- Stellar.org. (2019). [online] A vailable at: https://www.stellar.org/papers/stellar consensus protocol.pdf [Accessed 7 Feb. 2019].
- Oki, B.M. and Liskov, B.H., 1988, January. Viewstamped replication: A new primary copy method to support highly available distributed systems. In Proceedings of the seventh annual ACM Symposium on Principles of distributed computing (pp. 8 17). ACM.
- Ongaro, D. and Ousterhout, J.K., 2014, June. In search of an understandable consensus algorithm. In USENIX Annual Technical Conference ( 305 319).
- Piazza, F.S., 2017. Bitcoin and the Blockchain as Possible Corporate Governance Tools: Strengths and Weaknesses. Bocconi Legal Papers, 9 p.125.
- Pires, M.E.B., 2017. Generalized Paxos made Byzantine, Visigoth and Less Complex.
- Stolz, D. and Wattenhofer, R., 2016. Byzantine agreement with median validity. In LIPIcs Leibniz International Proceedings in Informatics (Vol. 46). Schloss Dagstuhl Leibniz Zentrum fuer Informatik.
- Van Renesse, R., Schiper, N. and Schneider, F.B., 2015. Vive la différence: Paxos vs.
- Yin, J., Martin, J.P., Venkataramani, A., Alvisi, L. and Dahlin, M., 2003. Separating agreement from execution for byzantine fault tolerant services. ACM SIGOPS Operating Systems Review, 37 ( pp.253 267.

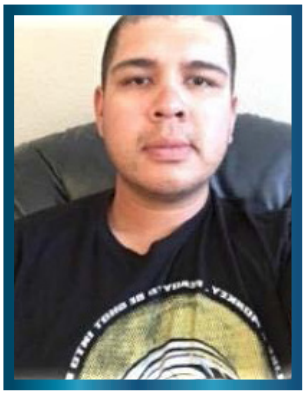
# PEER REVIEWED BY

**Avishek Nag**
Professor at University College Dublin



**Chris Yost-Bremm**
Professor, San Francisco State University

**Blazo Bozovic**
CEO,
Cryptocrypto247.com



**Ezra Salazar**
Bitcoin Manager, Ezcoin

**Samir Kumar Sah**
Founder and CEO, Pritbor

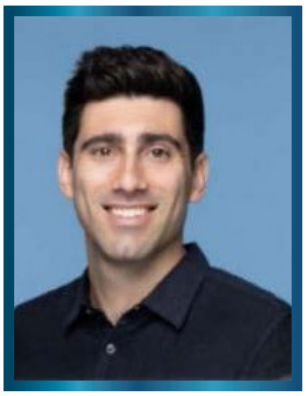
**Lee McKNIGHT**
Professor, Syracuse University

**Alan Donohoe**
CEO, Crypto Chain University

**Michael E Bryant**
CEO, Blockchain Ventures International

**Michael Silberman**
CEO, Traffic Leads Funnels

**Regiane Relva Romano**
Professor & CEO,
Vip-Systems Informática



**Sam Rosenblum**
Director, Coinbase

**Srini Katta**
CEO, Social Chains

**Dave Ozoalor**
Founder, Payrius

**Haris Doumanidis**
Former Professor, MIT

**Matt Vogel**
Founder, Yadacoin.io

**Nurbek Yensebayev**
Founding Partner at
SEZUAL

**Erik A. Jens**
CEO at LuxuryFintech.
com

**Rob Charles**
Founder and CEO,
Goldfingr

# TEAM

## Dinesh Patel
### (Chief Executive Officer)

Dinesh is a business magnet and pioneer with over 25 years of broad involvement in the diamond business. Dinesh has served various organizations around the globe including Israel, India and USA.

## Chirag Jetani
### (Chief Operational Officer)

Chirag an executive managing director of a prestigious Diamond company and early blockchain enthusiast with technical, administrative and authoritative experience working within the diamond Industry.

## Samuel E. Proctor
### (Chief Regulatory Advisor)

CEO of Genesis Block and an expert on the structure of financial systems and the regulation of blockchain technology. Prior to Genesis Block, he was an attorney at Debevoise & Plimpton, one of the leading blockchain and crypto-focused law practices in the U.S.

## Raj Chowdhury
### (Chief Technology Officer)

Managing Director of HashCash Blockchain Consultants and pioneer of the first interbank Trade Finance and Remittance implementation of blockchain technology for India's largest private sector Bank.

## Arijit Biswas
### (Product Manager)

As an experienced domain expert in Asset Management and Capital Market he holds a balance of technical & functional knowledge. He leads the team within the organization to oversee the design of product solution & database modelling for Blockchain based product and services.

## Eric Grollman
### (Project Manager)

Eric Grollman, a Project Manager, has managed data and technology projects at leading financial institutions for nearly 20 years. As a PMP, he led global fintech teams focusing on data integrity and regulatory compliance, e.g., Dodd-Frank, CVA, IHC, and KYC.

## Nikul Godhani
### (Marketing and Business development)

Nikul is a senior business developer, head planner and operational executive with over 14 years of experience in sales, technology, advertising, media buying, and operations.

## Prasanna Lohar
### (Technical & Banking Advisor)

With extensive experience working with DCB Bank as Head of Innovation & Architecture, he advises Diamante on technical & banking innovations. Between 2016-2019, he has bagged many leadership awards - TOP 50 CIO, Top 100 CISO.

## Lalit Choudhary
### (Head of Marketing)

An MBA from India's premier B-School Lalit is the Head of Marketing at Diamante Blockchain. He has expertise in developing, planning, implementing, and managing online and offline marketing campaigns and budgeting for sponsorship & other processes.

www.diamcircle.io