_FABRK

Technical Memorandum v1.0

Delegated Proof of Value (DPoV)

FABRK's approach to integrating network reputation into the consensus algorithm is effectively a hybrid of DPoS and PoI - both stake and value a user generates on the FABRK social networking platform influence that user node's weight in selecting delegate nodes. The Delegated Proof of Value algorithm is similar to the Delegated Proof of Stake (DPoS) approach: community members vote for delegates that secure and manage the blockchain. But unlike DPoS, where token holdings fully determine voting power, Delegated Proof of Value (DPoV) assigns a voting power, v_i, to each individual i, that is a function of the voting power of all users consuming their output, modified by network distance between partners, with an emphasis on nodes who receive requests from distant readers. For example, the value that a user acquires for posting a picture depends on:

1. the number of users requesting the picture (e.g. number of times seen),
2. the value of users requesting the picture (e.g. celebrities vs. spam-bot) and
3. the network distance between those users (e.g. do they belong to the same neighborhood or clique)

All read activity on the platform (e.g. obtaining a signed, one-time URI that grants access to information in a user node's local storage, or to data that user node has stored elsewhere) consumes a baseline amount of FAB. This baseline cost of a read operation allows users to recoup the basic cost of storing and serving data. The baseline cost is calculated and updated periodically by Delegate Nodes. DPoV uses this record of data exchange to reward those not with merely the greatest stake, but those with the greatest positive impact on the platform through the readership of their content (e.g. accessible data).

When content belonging to user i is accessed, the PoV of the user, v^i, is updated. This is followed by an update to the distributed ledger. We consider the network of node interactions in the distributed ledger Λ as a weighted, directed graph where users form the vertices $V(\Lambda)$. A directed edge $e_{ji} \in E(\Lambda)$ has an integer-valued weight corresponding to number of times that user j has accessed the contents of user i. When j first reads i's

content, i.e. $e_{ji} = 0$ at the update step, the PoV update considers the j to i distance as the length of shortest directed path in Λ. If user j frequently accesses the content of user i, then $e_{ji} > 0$ and the asymmetric distance $d(j, i)$ is one. In the case of multiple prior reads, the distance is the reciprocal of the edge weight $1/e_{ji}$.

At a high level, the implementation seeks to account for the following factors in order of importance:

1. PoV prior to read operation, $V_{t-1}{}^{i}$
2. PoV of reader at time of reading, $V_{t-1}{}^{j}$
3. The ledger graph Λ
    a. j to i distance in the ledger graph
    b. Local topology of i in Λ, discussed further in our technical memo

We consider the distributed ledger Λ as a weighted, directed graph where users form the vertices. When the content of user i is accessed, the PoV of the user, v_t^i, is updated. v_t^i corresponds to the weight of vertex i. This is followed by an update to the edges of the distributed ledger. A directed edge $e_{ji} \in E(\Lambda)$ has an integer valued weight as a function of the number of times that user j has accessed the contents of user i. If user j frequently accesses the content of user i, then e_{ji} is heavily weighted and the asymmetric distance $d(j, i)$ is low. In general, the ledger connectivity structure along with previous PoVs are inputs to the update operation:

$$v_t^i = f(\, v_{t-1}{}^{i}\,,\; v_{t-1}{}^{j},\; E(\Lambda_{t-1})\,)$$

The PoV of node i, v_t^i, is updated whenever i's content is read. When the transaction is validated, a state update to Λ reflects the strengthening of edge e_{ji}. This update process is a Markov process in the sense that the most recent Proof of Value state encodes past dynamics, and a forget coefficient α controls the impact of each update.

$$v_t^i \;=\; \alpha v_{t-1}{}^{i} + \;(1-\alpha)\,\varepsilon$$

ε is a small global constant reflecting the baseline value of reading a node's content to the community. The update term is penalized by low distance between reader and content holder in Λ_{t-1}, where frequent reads cause the value of new updates to decay exponentially.

$$\exp(-\lambda/d)$$

Where λ is a decay coefficient determining the severity of exponential decay and d is the j to i distance. j's read count of i's content is stored in e_{ji}. When $e_{ji} = 0$, the length of shortest j-to-i path is used as distance. When $e_{ji} > 0$, the distance continues to decrease as $1/e_{ji}$.

$$d = \begin{cases} \frac{1}{e_{ji}} & e_{ji} > 0 \\ d_{\Lambda}(j,i) & e_{ji} = 0 \end{cases}$$

For most decay coefficients λ, $\exp(-\lambda e_{ji})$ approaches zero after 5 reads. Lastly, the update term scales linearly with the PoV of the reader, v_t^j. We have:

$$v_t^i = \alpha v_{t-1}{}^i + (1-\alpha)\exp(-\lambda/d)\, v_{t-1}{}^j \varepsilon$$

Altogether, this update expression mitigates a range of simple attack vectors via automated creation of new nodes in the network, or by consecutive read operations among a cluster of nodes.The update term is meaningful when i's content is accessed by new readers far away in Λ, or when previous readers have particularly high PoV; this is difficult to achieve without many read operations by novel and distant readers. Moreover, it encourages the creation of compelling content for diverse participants.

Aggregate voting rights are computed at the time a voting transaction is validated. Aggregate voting rights are a function of proof of value, stake, and network topology. For clarity of notation let us denote proof of value v_t^i from earlier by V and coin stake by S. We have the voting power of a user prior to delegate election as

$$g(V, S) = \ln(\beta(S) + 1)V$$

Where $\ln(\cdot)$ is the natural logarithm. We scale the voting power of the user logarithmically with their stake and linearly with their Proof of Value. For large stake, its effect on voting power tapers off at extreme values due to taking its natural log. β tunes the weight of stake impact on voting rights and we will release an impact statement yellowpaper in Q4 2019 on effects of various values for this parameter on simulated behavior on the FABRK test-net.

Role of Social Contribution In Voting Power

While V provides insight into the the community's aggregate interest over time in the data posted by a given node, it fails to account for the fact that content consumption is not merely a simple, unalloyed good. Both fundamental and recent research in the field of network science have shown that authors and disseminators of certain content that is generally understood to harm network stability, such as false rumors, can be identified[1] by their real-time local topology in the network. In addition to the Proof of Value parameter V, we

[1] The Spread of True and False News Online

introduce the network parameter τ representing the node's social contribution to the network. This is computed from interactions and local topology of the node in Λ.

A FABRK community has extensive control over the modular building blocks leading to a quantitative interpretation of social contribution, from low level mechanisms such as determining reputation and importance of a node to high level mechanisms like identifying propagators of true, false, and viral content. Different communities may fork in the ledger to reach a particular consensus regarding the social factor in voting power.

Informed by a node's reputation and interactions in the ledger, delegates may provide algorithms that approximate social characteristics regarding these nodes - from tracing the dissemination of viral content to a few vector nodes, to cataloging particular interaction patterns in the ledger as socially positive or negative. The community may then use community rules to quantify a node's social impact negatively or positively. Throughout, techniques from network analysis and machine learning may be utilized, whose development the FABRK Foundation and community will spearhead.

Should the network reach a consensus definition of social contribution τ, we can imagine it's incorporation having fundamental, structural impact on future revisions of the calculation of voting rights. For example, for a scalar τ, we can relate it directly to stake weight, allowing the community to reduce the voting power of those with large coin stakes and network topologies that are associated with instability as follows:

$$g(\tau, V, S) = \beta(\mu_S - S)^2(\mu_\tau - \tau)^3 V$$

Where μ_τ is the mean of network participations of all nodes in Λ, μ_S is the mean of the relative network stake (to the entire network), and β is an amplifying coefficient that encourages productive behavior for participants with varying degrees of stake and network participation. We first explain the form for this expression then discuss the computation of τ.

The majority of nodes will not be affected by the polynomial product $\beta(\mu_S - S)^2(\mu_\tau - \tau)^3$, which evaluates to one except when a node has extreme stake or network participation values. When a node appears in either tails of the stake/participation distributions, $\beta(\mu_S - S)^2(\mu_\tau - \tau)^3$ modifies the proof of value V according to the following principle. If a node's network participation is in either tail of the population distribution, the node's voting power is amplified or diminished accordingly. However, a node's stake relative to the population stake distribution leads to a refinement. When the node's participation is in the low tail of the population but it's stake is low, this effect is lenient. When the node's participation is in the high tail yet it's stake is high, this effect is less rewarding. This interaction between stake and network participation is controlled by β:

$$\beta = \{0.3 \quad \textit{when} \quad S > \mu_s + \sigma_s,\ \tau > \mu_\tau + \sigma_\tau \quad \textit{or} \quad S < \mu_s - \sigma_s,\ \tau < \mu_\tau - \sigma_\tau$$

1 *elsewhere*

A wide collection of literature[2][3] in social network analysis and social media enables us to reason probabilistically regarding the network value (as opposed to content value, estimated in the Proof of Value computation) of a node. Machine learning techniques further enhance this analysis as embedded representations of the local neighborhood around nodes may be learned. As a group of game theorists, machine learning experts, social network scientists and user experience designers, the FABRK team is uniquely poised to deliver ground-breaking research into the relationship between content and authorship quality in modern, digital social networks, and to explore its application in decentralized systems.

We recognize that τ effectively weights the value of users' actions in the voting power calculation. Different configurations will be more gameable than others. As such, it is incredibly important to both not be defined hastily and to be made open in the future. Our specific approach will be detailed in a future technical memo.

We are currently testing and studying these parameters through simulations on our private testnet and will release specific learnings in a forthcoming yellowpaper, expected Q4 2019. The FABRK contributor community will help refine these factors and the extent of their governance influence. Further modifications will be made by the FABRK Foundation and released as network upgrades.

Noren: Federated Learning Summary and Appraisal Protocol

In order to facilitate a true Federated AI marketplace, developers must be able to confidently exchange value with users for the opportunity to train on their data, and in users must have confidence that their data remains appropriately private and obscure. The FABRK protocol establishes an anonymous summary and appraisal process, as well as a secure aggregation protocol for locally-trained models.

FABRK is augmented by a neural anonymization protocol for distributed ledgers. The Noren protocol serves as an interface between node users and ledger services to accelerate secure, anonymized data transactions. Within the Noren framework, FABRK users may generate encrypted, anonymized summary of ledgered data as means to monetize content and engage with advertising opportunities. In this way, a privacy-aware data commons is formed.

[2] Review on computational trust and reputation models
[3] Reputation and Social Network Analysis in Multi-Agent Systems

FABRK developers with data-centric goals are informed by users engaging in broadcastable AI appraisals of their content.

A node's internal and general containers are cryptographically secure without access to external services by default. By opting into Noren, containers acquire the capability to 1) generate an anonymized statistical summary of the container state for data transactions, and 2) approximate and repute container utility towards various commercial goals. FABRK encourages third party developers to produce alternative tools toward achieving these capabilities and alternative data protocols. Below, we detail the procedures underlying Noren Summary and Noren Appraisal.

FABRK Summary

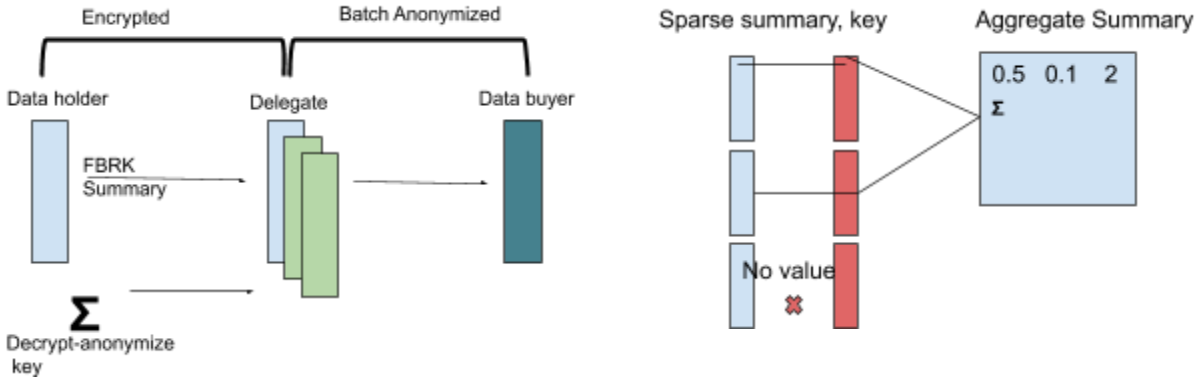
FABRK Summary is an anonymized statistical summary of the container. The container state is statistically encoded into a high dimensional embedding, and in Summary's decrypted state, yields the container's coordinate in this embedding. This embedding representation contains relevant information to be leveraged by applications. The encoding of a container's contents into FABRK Summary is completed via the open source framework Noren Network. The deployment of the framework occurs locally in the container as a blackbox, with inaccessible algorithm parameters. Once computed, the encrypted summary may be stored within the container with semi-restrictive/public permissions, for access by delegates, developers or third party services, in order to be consumed or anonymized.

Users frequently store objects that are high dimensional and complex: image, video, audio and text files are such examples. The Noren framework first structures objects in the container with recognizable formats and appropriate permission settings; it is on these objects that encoding may be attempted. Noren then attempts to encode individual objects through a collection of format-specific machine learning models. Finally, the network outputs a single vector representation for all actionable objects. With additional work in multi-modal learning[4][5][6] and information fusion, we aim to create this representation as versatile and informative as possible toward a wide swath of data-centric projects.

By decoupling encryption and anonymization, specifications of anonymization and execution of anonymization becomes delegatable, whether for computational reasons, or for verification reasons. For most machine learning applications, recent research in distributed machine learning and Federated Learning[7][8] has pointed to the robustness of aggregate high dimensional data as input to machine learning. In a distributed ledger such as FABRK, we achieve anonymization in an aggregated and delegated way.

---

[4] Multimodal Deep Learning
[5] Improved Multimodal Deep Learning with Variation of Information
[6] Learn to Combine Modalities in Multimodal Deep Learning
[7] Federated Learning: Strategies for Improving Communication Efficiency
[8] Communication-Efficient Learning of Deep Networks from Decentralized Data



In most instances, the protocol emits a decryption key that is applicable only after the summary has been anonymized by pre-determined transformations, or the decryption key is itself a decryption-anonymization procedure. We present an example scheme utilizing delegated anonymization and Google's Secure Aggregation Protocol[9], an aggregate anonymization technique. The state of the container summary throughout the procedure is detailed in the left figure.

At the end of its computation, a random seed is utilized to zero random positions of this summary. The resulting sparse matrix has its non-zero values stored as an unordered array, while the random seed serves as the decryption key, or position-to-position map, that recovers the structure of the matrix. The delegated decrypt-and-anonymize heuristic is pictured in the right figure. The decryption key is activated in batch, where iteration over positions of the aggregate matrix yield all non-zero summary values at that position, and is averaged, before moving to the next position. The aggregate summary is then delivered to the data requester.

FABRK Container Appraisal

The FABRK ledger also accelerates data transaction procedures with data content appraisal. Besides reputation given to users and businesses, a developer querying task-relevant data from the FABRK data commons is further informed by user broadcasted appraisals.

Users may opt to establish FABRK Utility Ratings (UR), or similar third party ratings, through a valuation process. The Utility Ratings are a set of attributes that approximates container utility towards various commercial goals. Such goals include demographic sketch for advertisement and data acquisition for machine learning. Utility rating towards a task is determined by a collection of I/O tests and machine learning assessments. The collection of I/O tests verifies the read/access status of objects in the container, as well as their basic properties: file format, file corruption, ill-defined fields, and presence of NaN. For common machine learning applications and for container content that passed the I/O test, the Noren

[9] Practical Secure Aggregation for Privacy-Preserving Machine Learning

framework can conduct machine learning assessments to approximate the utility of small datasets towards that particular application.

The user may foreseeably conduct appraisals for as many machine learning tasks as supported by the Noren framework on its container, choose a subset of utility ratings to broadcast, and engage in data transactions with interested parties. When layered with FABRK Summary, such transactions become secured and anonymized.

Now we provide an example of a common machine learning assessment. Given a container dataset and exemplar dataset, we interpret a wide collection of utility questions as: how closely do the source distributions that generated the container dataset and exemplar dataset, respectively X' and X, resemble each other? If both datasets are sampled from the same distribution, then a developer which purchased access to samples drawn from X' strictly augmented their dataset ($X' = X$). By leveraging statistical inference and information theory, we have quantitative information about the resemblance between a container's content and exemplar content. For image data and machine learning on images, an important assessment is whether the images are naturally occuring. This is an application for which the technique below is well suited.

To empirically estimate the deviation between two source distributions, a number of measures may be used. As an example, the Kullback-Leibler divergence[10] [11] $D_{KL}(P \parallel Q)$ describes the loss of information when an approximation distribution Q is used instead of P, the 'true' distribution in question. A low divergence $D_{KL}(X \parallel X') < \varepsilon$ implies the distributions from which both datasets are sampled from are statistically similar.

However, stored data frequently takes the form of complex, high dimensional objects such as images, sound files, and text files. Instead of computing measures between these objects with exemplar objects directly, techniques from statistical inference are leveraged to extract the most salient factors of variation between X and X', denoted respectively as Z and Z'. In practice, the Noren Network selects an encoder F corresponding to the task, with pre-trained parameters θ. F is often a neural network but not always. This encoder maps both datasets into a common latent space as their latent representations Z and Z'.

$$F_\theta(X) = Z \qquad F_\theta(X') = Z'$$

These latent representations are low dimensional, with mostly information relevant to the task. The Kullback-Leibler divergence is then estimated on these representations by computing the empirical Cumulative Distribution Function or by density estimation[10]. After presenting this divergence information to the user, the user may choose to broadcast this task specific finding.

[10] Kullback-Leibler Divergence Estimation of Continuous Distributions
[11] Estimating divergence functionals and the likelihood ratio by convex risk minimization

A number of challenges in assessing ledgered data include finding few observations within a container. In this case, measures between source distributions may not be reliably estimated. Numerous papers[12][13] have been written to resolve data constraint problems such as these, and more work in the test net phase will shed light on a robust solution. Another challenge is distinguishing genuine data (as opposed to duplicated, or adversarially generated data), where measures between hypothesized distributions may appear very low, but are uninformative towards the specified task. To remedy this, a potential solution is to devise a third set of assessments with more structured assumptions.

[12] Generalizing from a Few Examples: A Survey on Few-Shot Learning
[13] A Few Useful Things to Know about Machine Learning