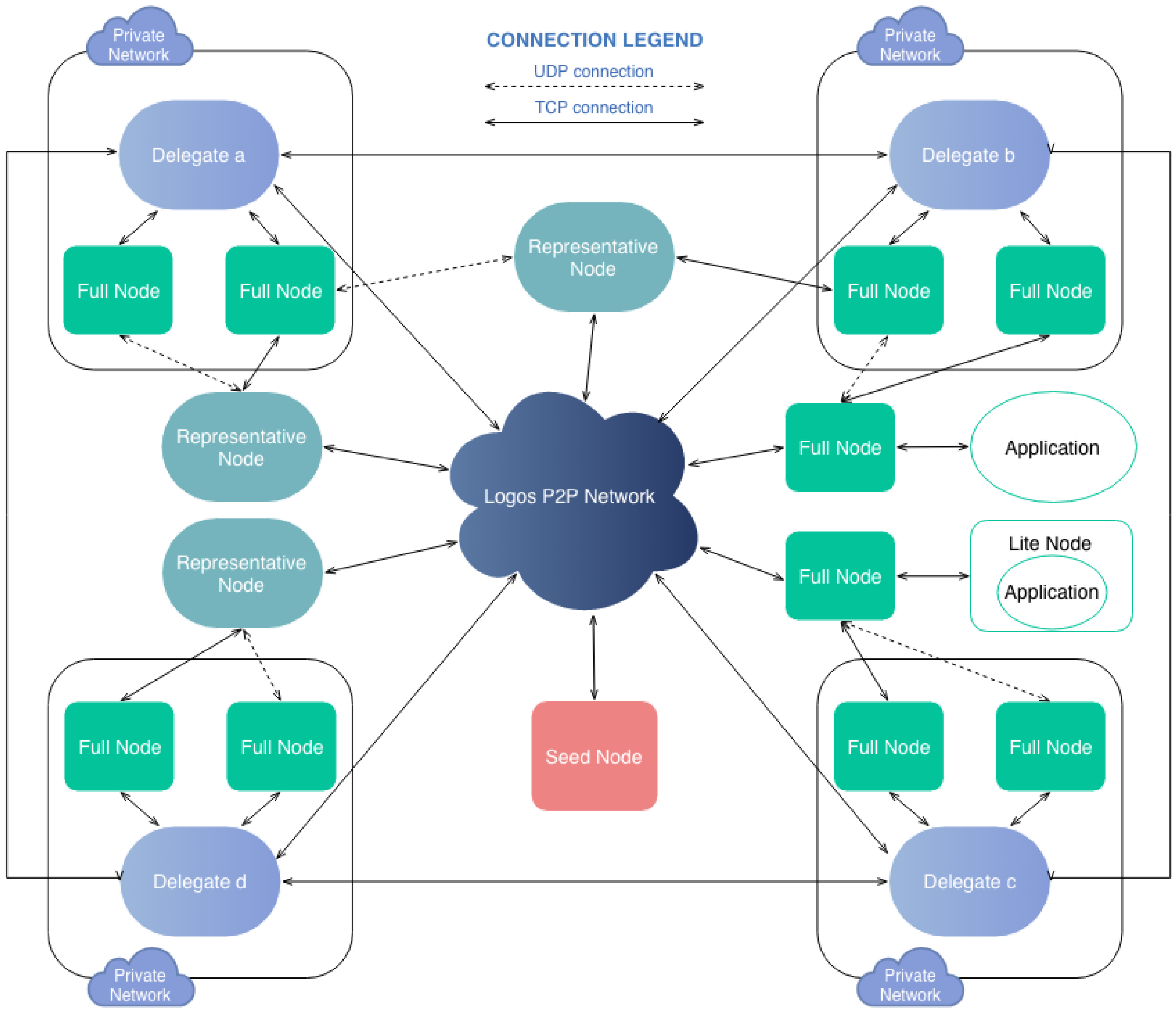
Logos Ecosystem Architecture White paper v0.2

Draft Carl Hua Draft 9/20/2018

The intention of this white paper is to provide a detailed overview of Logos Network's Ecosystem layout - the p2p network architecture. This paper also provides a detailed summary on the type of connection each message category will use. In addition, details on the propagation rules of each message category is also presented. And finally, we will address some of the concerns regarding to network security. The ability to mitigate distributed denial-of-service (DDoS) attacks and load balancing transaction requests among the delegates are two major considerations in designing the Logos ecosystem architecture.

Ecosystem Architecture

The diagram below depicts the overall ecosystem architecture:



Per the above diagram, there are various type of nodes in the system:

Delegate Nodes: these nodes are the most heavily loaded nodes in the system as they serve as the validators in the system. They will propagate most of their messages to the rest of the systems similar to the way messages are propagated in Bitcoin. An entity that wishes to run a delegate node can also run one or more instances of full nodes along side of delegate nodes to provide load balancing. More details are explained in the “Logos Network Delegate Node Configuration” section below.

Full Nodes: This is the default configuration for anyone that wishes to participate in the network. These nodes are what we call the traditional “bitcoin like” nodes. They propagate messages via TCP with their peers. They are the fundamental building blocks of Logos Network’s p2p network.

Representative Nodes: these nodes behave similar to full nodes from a message propagation perspective, however they also cast votes to the system. The votes are propagated via Logos Network’s gossip protocol. This type of nodes do not exist as of today - voting is not implemented.

Lite Nodes: these nodes either keep limited account history or no account history at all. They are usually connected to full nodes and query them for information. They do not participate or contribute to the p2p network. Note that certain Lite nodes require additional facilitation to communicate with a full node, and its implementation is usually dependent on the service provider that implements the lite node. Full nodes provide call back features to relay information to the facilitation implementation. One example of such usage is a wallet server that is connected to a full node.

Seed Node: These nodes exist for the sole purpose of providing peer IP address to facilitate the peer discovery.

Logos Network Delegate Node Configuration

The design goal of the delegate node configuration is to provide resilience against various network attacks while minimizing the impact to the actual core protocol, shifting most of the actual mitigation implementation to the under laying IT infrastructure:
The delegate nodes within the system require protections against various types of network attack such as DDoS. The actual protection are realized via IT infrastructure that can differ in implementation. The actual implementation of the IT infrastructure for the private network can vary depending on the entities that run the delegate nodes. For example, one can choose to use a level 4 load balancer to mitigate DDoS in an AWS cloud whereas another entity can choose to have redundancies within its private network. The diagram above depicts a simple delegate node configuration that also has load balancing built in: full nodes that run alongside of the delegate node. In Logos Network, we allow transactions requests to be submitted via UDP in

addition to the conventional TCP propagation (detailed in section below). The UDP transaction requests are routed to the full nodes via the load balancer (not shown in diagram for simplicity purpose) and then sent to the delegate node via batched transaction request over TCP. This is all done by adding mandatory full peers to the delegate nodes via either configuration files or RPC calls. The delegate node also makes direct TCP connections to the other delegates in the network. This reduces the load on the delegate and facilitates further load balancing. Regardless of the actual implementation of the IT infrastructure used for the delegate nodes, they all have to conform to one requirement: **single IP** presentation. This means that to the rest of the network, each delegate only has one IP address that is advertised via the “key adv” message.

Logos Network Message Type

Transaction request: The transaction requests such as send, open, change rep, and other more advanced requests can be done in either UDP or TCP. When TCP is chosen, the request is sent to its peers via broadcast and propagated throughout the network, and eventually it will reach one of the delegate node. For UDP, it is a direct send with lower reliability but much higher speed. A TCP connection here means any TCP connection to its peer. Note that in Logos Network, we enforce a “correct” primary rule by placing a response penalty time on a request that is sent to the wrong delegate. This means that each transaction request has a designated primary processing delegate that is calculated based on the previous transaction request’s hash - if a client sends that request to the wrong delegate, the delegate will not process it immediately, and will insert a timer. Only when the timer expires, that delegate will process the transaction. So in our case, UDP is highly preferred.

Consensus messages: These originate from the delegates. Most of these messages are propagated through the network via TCP with the exception of few.

Bootstrapping: These messages are not propagated through the network and is a point to point communication protocol.

Peer discovery: These messages are sent via TCP, and is a point to point communication protocol.

All other messages: all other messages such as recall request, blacklist and key adv messages are all propagated through TCP.

Logos Network P2P Network Architecture

This section is written with the assumption that the reader has sufficient knowledge with Bitcoin’s p2p subsystem. Several P2P Network design has been analyzed by Logos Network including Bitcoin, Ethereum, Monero, Dash, Cosmos and other affiliated projects. During the analysis, we discovered that Ethereum’s p2p network uses Kademlia to perform peer handling.

We consider this to be troublesome - we cannot find a benefit by using DHT to perform peer handling since all nodes replicate the database. Other project's p2p network is relatively new, and not proven. Out of all the projects p2p system, bitcoin's p2p system is the most comprehensive and well documented. Other projects such as Monero and Dash are all Bitcoin forks that utilize the same p2p system and therefore, Logos has chosen to adopt Bitcoin's p2p system at the design level. We are currently porting Bitcoin's p2p system into Logos' code base. Although, we do expect to make some changes, but we will keep majority part of of logic.

In addition, Bitcoin's P2P network isn't tuned for performance - as their block is only once every 10 minutes. Transaction can freely propagate through the network without the issue of increased latency as the block time is significantly greater than propagation time. This is no longer the case for Logos Network - we are expecting transaction to complete within seconds. This means the system cannot use TCP block propagation scheme to submit transaction to the delegates due to its excessive propagation time. Due to the reasons specified here, Logos Network allows transaction requests to be submitted via UDP directly from the client/nodes to the delegates. Consider the TCP p2p network as a conventional p2p network that is similar to Bitcoin's network. While the UDP transaction submission capability is a performance enhancements on top of the TCP network to allow faster transaction submission. Any client can submit transaction directly to delegate node's IP over UDP at any given time - effectively cutting out TCP propagation time.

In order to further increase the consensus validation speed, delegates nodes are directly connected with other delegate nodes to minimize data propagation latency. This way all the messages can be sent rather quickly from one delegate to the other delegates during consensus validation. Logos network still uses TCP network to propagate other data for reliability reasons.

Nodes Type vs Message Type vs Transmission protocol

Various types of nodes are responsible for various message types, and all data are propagated via TCP connection. Note as per the description in previous sections, UDP is only used for transaction submission and it is point to point. In this section, we will list a matrix on how all the messages are mechanized.

There are total of three types of messages not including the RPC calls in the Logos Network Ecosystem:

Transmission Protocol	Definition
TCP Propagation	Message is broadcasted to all of its peers when its either generated or received

TCP Point-to-Point	TCP message that is a point to point communication, no broadcast
UDP Point-to-Point	UDP message that is a point to point communication, no broadcast

In the table below, we show the relationship between message type and transmission protocol

Type of message	**Protocol Type**
Consensus - Pre-prepare	TCP Propagation
Consensus - Prepare	TCP Point-to-Point
Consensus - Prepare reject	TCP Point-to-Point
Consensus - Post-prepare proof	TCP Propagation
Consensus - Commit	TCP Point-to-Point
Consensus - Post-commit proof	TCP Propagation
Request - Transaction request	UDP ptp or TCP Propagation
Request - Transaction reply	TCP Propagation
Batched Request - batched transaction request	TCP Propagation
Bootstrap - bootstrap request	TCP Point-to-Point
Bootstrap - bootstrap response	TCP Point-to-Point
Peer discovery - request	TCP Point-to-Point
Peer discovery - reply	TCP Point-to-Point
Peer discovery - handshake	TCP Point-to-Point
Recall - request	TCP Propagation
Blacklist	TCP Propagation
Key adv	TCP Propagation

Note that an example of a UDP point to point message is when a lite node or a client application sends delegate node IP a transaction request - the full node within the delegate's private network will then propagate the request to the delegate nodes for processing. The client can also decide to send any full node a UDP transaction request message, and the nodes will propagate the message using gossip protocol.

Finally, the table below shows the relationship between message types and node types:

Node Type	Consensus	Request	Batched Request	Bootstrap	Peer discovery	Recall	Blacklist	Key adv	Vote
Full	X	X	X	X	X	X			
Delegate	X	X	X	X		X	X	X	X
Representative	X	X	X	X	X	X			X
Lite		X							

Per the table above, although full nodes support the propagation of the consensus messages, they do not actually participate in the point to point communication of the consensus. Similarly, delegate only accepts transaction requests but do not initiate them.

For invalid transaction requests, the recipient of such request should reject the requests via the requested transmission protocol.

Note that the transaction request's confirmation is the post-commit proof of the batch block, and they are always propagated using TCP propagation.

Summary

This paper attempts to summarize the p2p network architecture as well as the data propagation mechanism in the Logos Network. The details in this paper should be used as input for the p2p network requirement derivation as well as a consideration factor for the actual p2p network, message dispatcher and node initialization's implementation.

TODO:

Work with Dmitry (and Peng) to identify each and every message in the P2P system and how to integrate them into the existing code base.