Oyster Shell Whitepaper Rev. 0.1b

February 2018

Bruno Block

bruno@oyster.ws

oysterprotocol.com

Introduction

The internet as we know it is constantly threatened by governments worldwide concerning censorship, regulation, and spying. Internet users have resorted to calling their elected officials to try and remedy the problem, despite them being the last place to apply a remedy. Internet Service Providers (ISPs) are collaborators in the degradation of the connectivity that we depend on.

Oyster is a revolutionary protocol that resets the assumptions of the current internet paradigm and solves them with a new comprehensive platform. Oyster's primary operation is based off of the storage, retention and retrieval of static data. The Pearl (PRL) token enables such data storage and is the first asset to be pegged to a known market value without a reserve or a centralized guarantor. The operation of the core Oyster network inadvertently creates a latency-optimized meshnet of nodes, which becomes the perfect environment for fostering decentralized communications. The Shell (SHL) token enables transmission of data across such a meshnet.

This technical document describes the functionality and operation of the web node topology part of the Oyster network. For a more comprehensive understanding of the Oyster network, one should read the original Oyster Whitepaper.

Recursive Triad Peering

Web nodes are resource-light networked devices, typically they are the browsers of website visitors. Web nodes trade valuable information with each other in exchange for proof of work performed on the tangle. Web nodes keep track of other web nodes' addresses so that they can communicate with each other peer-to-peer (typically via WebRTC). For decentralized application (dapp) rendering data packets traverse the meshnet from one web node to another. To accomplish packet navigation across the meshnet web nodes must assemble in mutually agreed upon zones. Such zones become points of reference for data packets to efficiently traverse the meshnet. Zones exist in recursive levels, for each layer a web node is peered with two other web nodes therefore forming a triad. Per recursive level, a web node can exist in a maximum of three zones. Web nodes are enticed to belong to as many reputable zones as possible because it increases their chances of earning SHL.

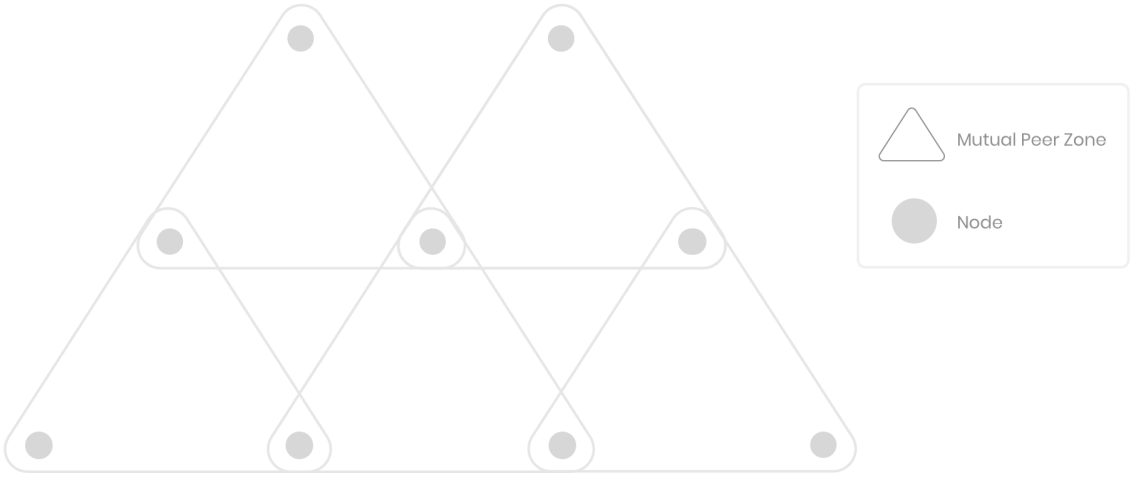


Fig. 1 - Web nodes assemble in triad formations

The following is the procedure for a web node to initiate the formation of a triad zone:

-1. The web node (A) selects the lowest possible recursive level that isn't fully saturated (less than three zones).

-2. The web node (A) sends a triad formation proposal to its neighbors (B) which do not currently belong to a mutual zone with said web node (A).

-3. If the neighboring nodes (B) accept the triad formation proposal they digitally sign the proposal and pass it to a second round of neighbors (C). Neighboring nodes (B + C) accept or reject a proposal based on the zone saturation of the specified recursive level.

-4. If the any of the second rounds of neighbors (C) accept the formation proposal (due to their zone availability) they then digitally sign said proposal and return it to the web node (A) that originated said proposal.

-5. The original web node (A) approves a maximum of one of the returned proposals. The web node (A) can approve the completed proposal based on a first-come-first served basis, or select the web node signatories that have the lowest latency. Web nodes are entitled to adopt their own triad acceptance strategies.

-6. The original web node (A) communicates directly with the two other selected web nodes to announce the confirmation of the newly formed triad. The two other web nodes (B + C, who had previously signed the proposal) confirm the formation of the triad.

Triad zone identities are formed as such:

zone A ID = first5Characters(sha256(web node A ID + web node B ID + web node C ID))

Web node A assumes a leadership role in various scenarios, such as representing the triad in forming a higher recursive level triad and being the active 'gateway node'. For higher recursive levels the same methodology is applied above except triads are used instead of web nodes, and the leader web nodes (A) perform the necessary representations during the process.

zone D ID = first5Characters(sha256(zone A ID + zone B ID + zone C ID))

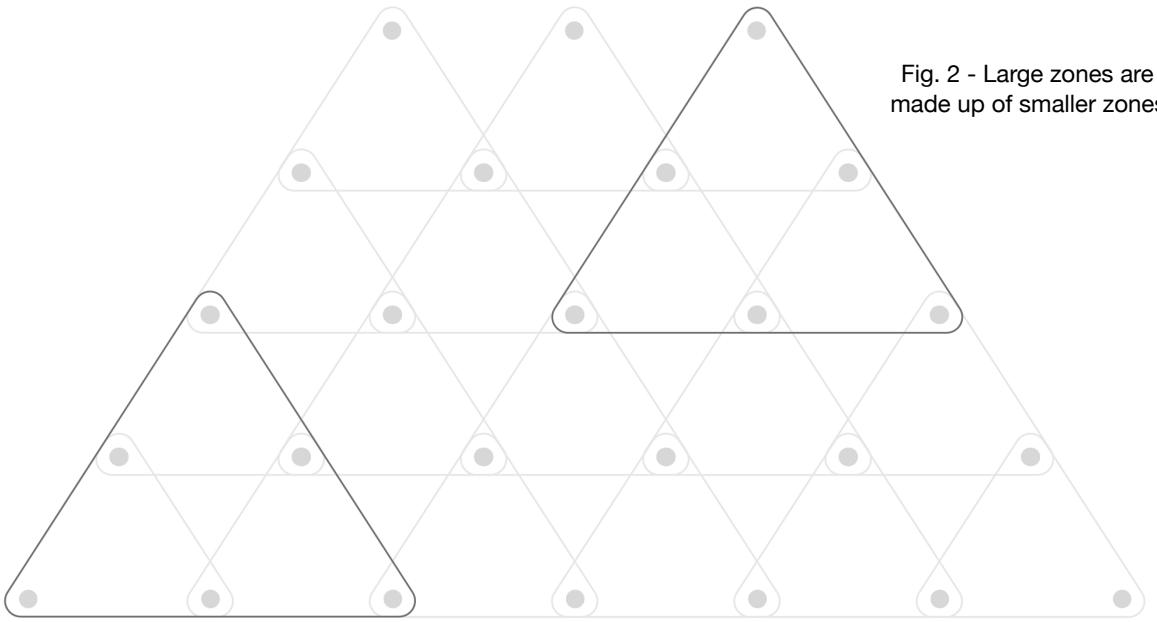


Fig. 2 - Large zones are made up of smaller zones

Web nodes that belong to a zone agree upon an inheritance order, which is the defined A, B and C allocations. All web nodes strive to become the A node (gateway node) since they earn all of the possible SHL revenue. If the A node goes offline; B node becomes A node and C node becomes B node. An announcement for C node availability is then flooded to neighbors of the web nodes that belong to the aforementioned zone. Zones retain the initial identity it formed during its inception, the identity does not alter when the node makeup alters.

Communication Methodology

Data packets are transferred across the web node mesh topology to serve functional decentralized application (dapp) and communication usage. Web nodes learn about neighboring triad zones which helps them build a navigation map of the topology. When a data packet is passed on across the mesh its exposure to various zones are recorded in a cryptographic passport. This passport is used in a similar way to real traveling passports, cryptographic stamps of travelled zones are recorded. This way when a web node receives or passes on a data packet, it can record the contents of the passport and learn about existing zones far away.

Web nodes keep track of all available zones at various recursive levels to increase their chances for earning SHL. Zones go through popularity cycles; if a zone is recognized in packet passports consistently it will be used more for data transmission, thus further increasing the zone's popularity and reputation.

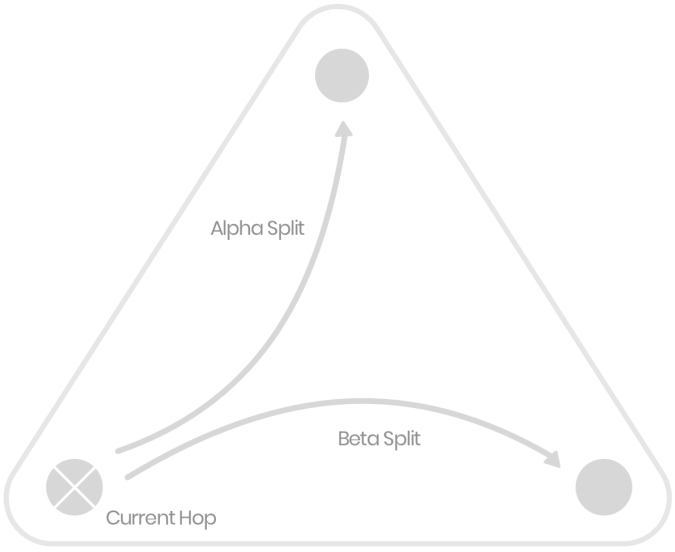


Fig. 3 - Packet traveling within a zone

Packet transmission routes are calculated and chosen during the initiation of a communication session between two web nodes. Therefore intermediary web nodes do not perform any significant calculations, they adhere to pre-calculated routes that accompany a packet as much as possible. If the primary route is no longer applicable (web node topology constantly changes) then the next route defined according to priority is referenced to successfully complete the packet journey.

At any given node hop, marked as 'Current Hop' in Fig. 3, a web node calculates if it should perform an 'Alpha Split' or 'Beta Split'. Alpha indicates passing on the packet to the left web node from the same zone, Beta indicates the right web node from the same zone. Therefore a single web node will only choose between Alpha and Beta, a binary decision, and will not contemplate complex options or outcomes. The aggregate choices between all involved web nodes causes the path selection algorithm to become intelligently complex whilst adhering to the pre-calculated session route.

When a packet travels to a higher recursive level zone, it travels to only the three web nodes at the three tips of the triangle. These three web nodes of any given zone are considered 'gateway nodes'. When that higher recursive level zone is being used by a packet, the packet

does not travel to intermediary web nodes that belong inside that zone. Therefore it is mandatory for gateway nodes of any given zone to know each other's identities for direct peer-to-peer communication. For the lowest recursive zone level, the A web node is the only gateway node since there are only three web nodes in the entire zone. If a gateway node goes offline a web node from the same zone automatically assumes the position. Web nodes within a zone already pre-agree to the order of inheritance. Gateway nodes of a zone receive all the SHL revenue, therefore when they inevitably go offline the other web nodes will be eager to accept the role of being a gateway node.

Web nodes keep track of popular and relevant zones in addition to which zones overlap with each other. All of this information is inferred by witnessing packet passports. The selection algorithm resorts to using low level recursive levels (small zones) near the origination and destination points of the journey, whilst using high level recursive zones (large zones) in the middle of the journey. This is akin to driving in small roads and neighborhoods at the beginning and end of a journey, whilst driving on large highways in the middle of the journey. Small zones grant accuracy at the expense of distance, larger zones grant distance at the expense of accuracy.

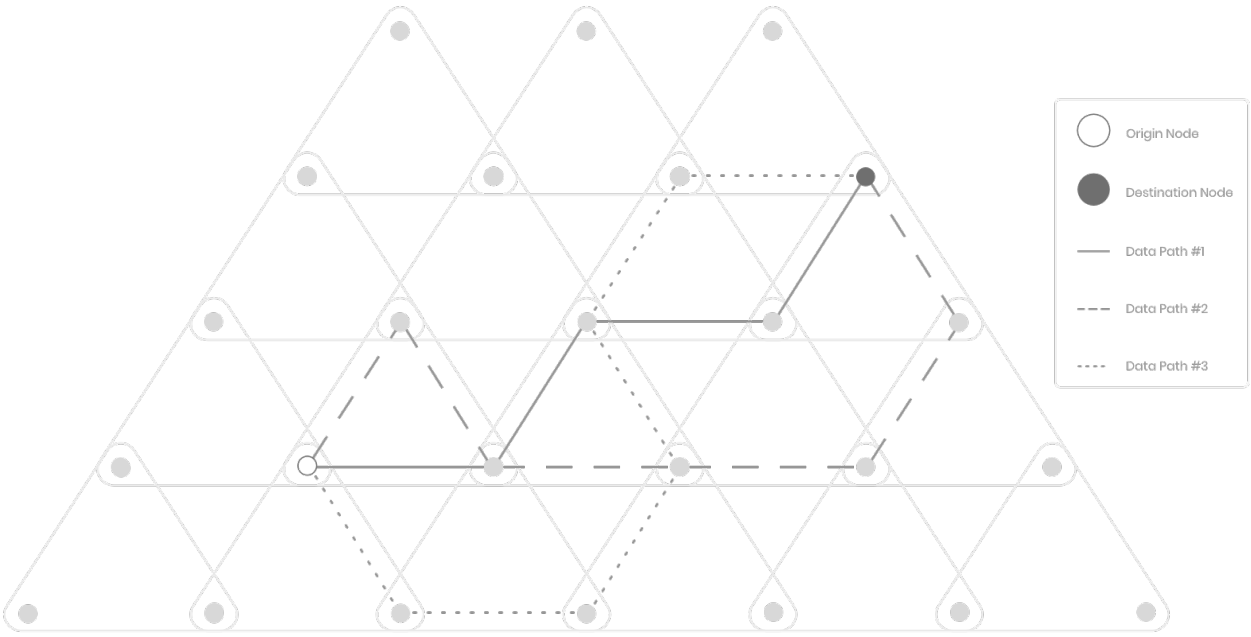
Fig. 4 - A packet traverses zones of various sizes
from the origin node to the destination node

Minimum Viable Resistance Discovery

The meshnet constantly graduates from a state of inefficiency to efficiency. Web nodes constantly monitor and analyze packet passports that pass by them to infer the overall state of the meshnet. Therefore self-awareness of the meshnet structure increases gradually over time.

For dapp communication between two web nodes to occur, a session must be initiated and mutually agreed upon. The origin web node fires several initiation packets to the destination web node using multiple pathways. When the initiation packets are received the destination node reverses the pathway guidance rules and sends a confirmation response back to the origin node. This way both nodes evaluate which pathway has the minimum viable resistance by measuring the time it took for the initiation packets to make a round trip. The top best performing pathways are selected as the routing strategies for the session's packets.

An origin node will send the same packet multiple times over various pathways. This ensures that the destination node will receive the intended packet as early as possible despite the volatility of the web node topology. The destination node will also be able to validate the multiple copies of the packets with each other to prevent an intercept attack.



Pre-Authorized Pathways

When both the origin and destination nodes agree upon several routing strategies (pathways) for packet transmission, an authorization packet is sent along each pathway. Web nodes will only forward payload-capable packets once they have accepted the authorization packet for such a session. The authorization packet contains the seed key to an Ethereum address that contains time-locked SHL. Origin nodes bear the burden of paying the necessary SHL payments. The SHL is locked in a delay-release state using the lock() function of the smart contract by the origin node. An origin node can either send and lock the SHL using its own software, or perform proof of work as defined by a broker node to get the broker node to lock the SHL itself.

The smart contract delay-release lasts as long as the session is intended to last. Gateway nodes validate the authorization packet to confirm that it contains a sufficient amount of SHL in a delay-release state. If the gateway node approves the ‘escrowed’ SHL amount then it informs the zones that it belongs to that the session has been approved for the duration of the delay-release state. Web nodes forward initialization packets without pre-authorization, however initialization packets do not carry any significant payloads and a limited amount of initialization packets are allowed per origin node per hour.

Once the smart contract induced lock delay of the ‘escrowed’ SHL expires, gateway nodes attempt to get broker nodes to claim the SHL on behalf of their website owner. Broker nodes are unwilling to spend ETH for the gas to unlock the SHL, therefore they stipulate a proof of work amount required to complete the transaction. Once the SHL is unlocked, the entire amount is claimed on a first-come-first-served basis. Therefore gateway nodes attempt to pre-pay the broker node with proof of work, so that the broker node will unlock the SHL immediately as soon as the contractually locked state expires. Gateway nodes relay the proof of work burden to other web nodes within the same zone as part of the intra-zone node relationship dynamic.

Web nodes within a zone are eager to perform proof of work assigned by the gateway node, so that they can retain their place in the inheritance line to eventually become the gateway node

themselves. If a web node refuses to do work assigned by the gateway node, a consensus emerges within the zone to disqualify such a node from the line of succession (A, B and C). A gateway node also assigns dapp calculation work to other web nodes within the same zone, whilst the same rules of disqualification apply. A gateway node must resign as being the designated gateway node within a protocol-specified timeframe since having assumed the role. Otherwise a gateway node going offline for a short time interval also removes the gateway node's status and increments the line of succession. Gateway nodes that receive the initial authorization packet claim the full SHL amount on behalf of the website owner that invoked them, contingent on the SHL being successfully claimed via a broker node.

The larger the SHL pre-authorization value is, the higher the chances that the gateway node will accept the session. When the origin node calculates the SHL pre-authorization value, it must consider the expected amount of zones the session packets will traverse. If the SHL value is too low whilst there are too many zones to be traversed, gateway nodes will be hesitant to accept the session since it must compete with too many other gateway nodes to claim the locked SHL.

If a gateway node were to accept the authorization packet (hence benefit from the possibility of claiming the SHL) whilst not forwarding legitimate session packets that ensue, then this defection would reflect in the packet passports that are received by the destination node. Eventually the zone would become gradually forgotten in the collective consciousness of the meshnet as the zones' signature would become vacant in packet passports. Therefore it is economically beneficial for a zone (via its leaders, the gateway nodes) to forward the authorized session packets to promote and propagate its zone signature. This leads to increased chances of earning SHL revenue in the long term.

Decentralized Application Deployment

Origin nodes can propagate specified javascript scripts with their session (during the pre-authorization phase). If the attached SHL amount is high enough, zones will be willing to remember and execute such scripts to facilitate the paid session. Therefore it is more economical for origin nodes to reuse the same established scripts wherever possible, since it would take a significant amount of SHL to convince a gateway node to install a new script. Scripts are identified by the SHA256 hash of the codebase.

Origin nodes can attach execution instructions to session packets, along with the script that is to be used for execution. An origin node will have had to reference a well propagated previously established script, or have had previously propagated its own script with a sufficient SHL fee. If the receiving gateway nodes evaluate that the execution instructions are worth performing considering the SHL pre-authorization amount, the gateway node forwards the execution work to the regular web nodes that belong to the same zone. Web nodes adhere to the execution instructions of the gateway node to remain qualified for eventually becoming the gateway node themselves. Upon forwarding the execution work the gateway node deletes the session packet, so that it does not get forwarded to subsequent zones. This is done to prevent multiple zones from performing the same work.

The origin node will have specified the minimum required redundancy of calculations required for the execution work. Therefore as soon as the session packet with execution instructions reaches a zone with a sufficient web node topology, the session packet is 'consumed' and execution result packets are eventually spawned from the regular web nodes that performed the work. These execution result packets are forwarded to the destination node, therefore continuing where the session packet left off. The destination node then verifies that the execution packets results match each other, to verify that the execution was performed

correctly. An origin node is able to specify that a session packet with execution instructions gets ‘consumed’ by X amount of zones, to further reduce the chances of a rogue calculation being performed.

All the regular web nodes within a zone perform the same execution work specified by the gateway node. They then sign the results with their cryptographic signature and send the results directly for their onward journey towards the destination node. By-standing web nodes in the meshnet validate the execution results by performing the execution themselves voluntarily. The benefit to performing the execution themselves is that they can appropriately validate a productive zone and hence become more aware of where to send execution instructions in the future. There is inherent economic value for web nodes to learn about the zone makeup of the meshnet.

Conclusion

The latency-optimized Oyster meshnet operates due to the economic foundations built by the utility of PRL, yet grants a platform for the SHL token to exercise its own independent utility. Users with basic wireless hardware will be able to convert their device into a web node and seamlessly connect to the meshnet, therefore bypassing centralized infrastructure owned by ISPs and Governments. Thus, the entire Oyster network becomes a massive organic super computer, with PRL (via the tangle) facilitating long term persistent storage (HDD) and SHL (via the meshnet) facilitating volatile memory, instruction execution and data transmission (RAM, CPU, and NIC).