October 2019

SELF-ORGANIZING MESHES USING DISTRIBUTED LEDGER SYSTEMS

Jeff Venable

Greg Shepherd

Alok Nikhil

Ralf Rantzau

Ijsbrand Wijnands

SELF-ORGANIZING MESHES USING DISTRIBUTED LEDGER SYSTEMS

AUTHORS:
Jeff Venable
Greg Shepherd
Alok Nikhil
Ralf Rantzau
Ijsbrand Wijnands

ABSTRACT

Techniques described herein provide innovation relating to directing traffic through a distributed network. A first node can receive packets that are directed to a second node. The first node can determine that the second node is a member of the distributed network based on information within a distributed ledger stored locally on the first node in which the second node is a member based on fulfilling a condition within a smart contract that manages membership. A tunnel from the first node to the second node can be generated to send the packets to the second node. Previously available blockchain systems do not employ abilities to track products beyond mere custody information.

DETAILED DESCRIPTION

Many traditional storage systems are centralized storage systems. In such storage systems, one or more servers serve as a central repository that stores information. The central repository is accessible to various client devices. The central repository is often managed by a business entity that typically charges a fee to access the central repository. In some instances, there is a transaction fee associated with each transaction. For example, there is often a transaction fee for writing information that pertains to a new transaction, and another transaction fee for accessing information related to an old transaction. As such, centralized storage systems tend to be relatively expensive.

Some centralized storage systems are susceptible to unauthorized data manipulation. For example, in some instances, a malicious actor may gain unauthorized access to a central repository and may surreptitiously change information stored in the central repository. In some scenarios, the unauthorized changes may not be detected. As such, the information stored in a centralized repository is at risk of being inaccurate.

In a distributed ledger environment (blockchain), the nodes (e.g., network routers and other devices) are 'approved' members of a blockchain while 'unapproved' nodes are excluded. In blockchain parlance, node admission is via Proof of Authority in the smart contracts; e.g., the blockchain is hosting a distributed certificate authority that is censorship resistance because its policies and actions are automated via the smart contracts running in the network. Internet Protocol Security (IPSec) mesh overlays can implement secure end-to-end routing and cryptographic message protection. SSL/VPN meshes are also possible in a PKI environment. With the blockchain, novel new addressability mechanisms arise. The concept of building a new blockchain on the fly for a temporary purpose also has appealing applications. Interposing routers (e.g. devices from certain competitors or countries, etc.) are oblivious to these networks, which may be persistent or temporary or intent-driven. Figure 1, below, is a schematic diagram of a distributed ledger environment.

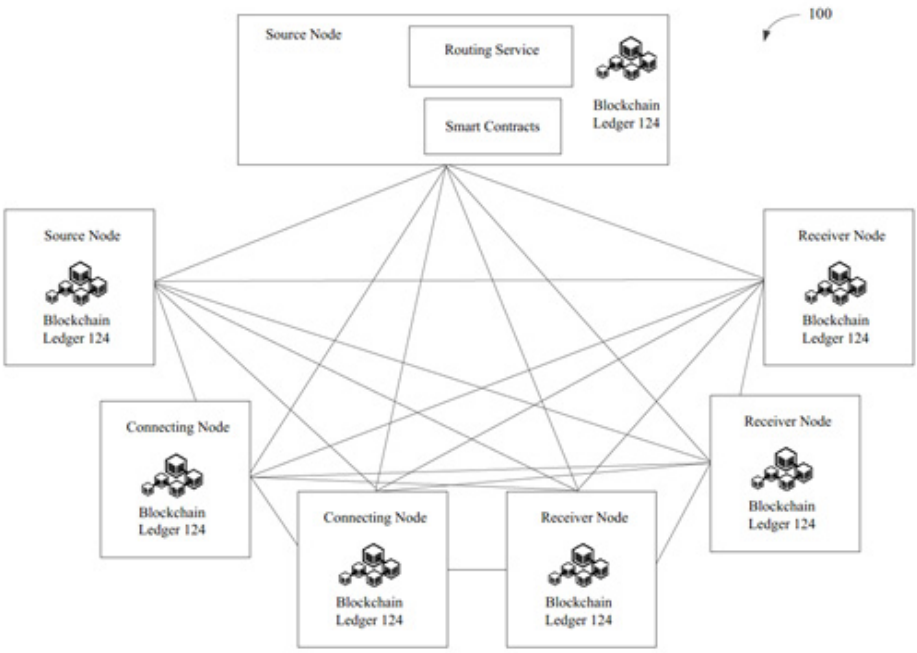

Figure 1

Self-Organizing Mesh

Techniques described herein provide methods and systems to locate (geospatial, travel time, jurisdictional, etc.) member nodes of a blockchain network and map them across all nodes in the blockchain network, with the desirable properties of deterministic, non-interactive algorithms such that all of the nodes arrive at the same state conclusion

(consensus) in parallel. This is achieved through smart-contract transactions where nodes publish enough information as transactions to a blockchain ledger, which is verifiable by all other nodes such that they independently compute the same map with 'pins' placed on it for nodes. This 'registration' data set is signed by each node using its public/private key pair registered in the blockchain (which can be issued and verified by a distributed certificate authority (Dist.CA)).

Such methods and systems can also draw lines between the 'pins' (nodes) in which the lines may represent tunnel/routing relationships in the network (e.g., IPSec tunnels). Tunnel authentication and cryptography will be established using the well-known public/private key pairs owned by each respective node as published in the blockchain ledger (issued and verified by the Dist.CA). Embodiments may include Google Maps®, Wi-Fi®/Bluetooth® signal triangulation, dynamic network telemetry tested over an overlay network, continent, region, locale, number of desired redundant links, etc.

Such methods and systems can include logic in which the network load balances each node with a minimum redundant set of routes (high-availability) and a maximum (graph degree) and adapts in real-time to transient nodes joining and leaving the mesh. Independent copies of this 'network graph' can be maintained at each node, which can be used (in a predictable consensus manner) in the network control plane for updating routes in real-time in the network data plane. General network admission control can be facilitated via blockchain contracts (e.g., smart contracts) and leveraging Dist.CA (equivalent to network identity).

To that end, the distributed ledger environment can include one or more source nodes (e.g., a first source node, a second source node, and a third source node), one or more receiver nodes (e.g., a first receiver node, a second receiver node, and a third receiver node), various connecting nodes, and a distributed ledger copied on all the nodes (e.g., source nodes, receiver nodes, and connecting nodes). Briefly, in various implementations, the connecting nodes provide various communication paths between the source nodes and the receiver nodes, and each node is configured to adjust at least a portion of the communication paths based on their performance. Figure 2, below, illustrates such a distributed ledger environment.

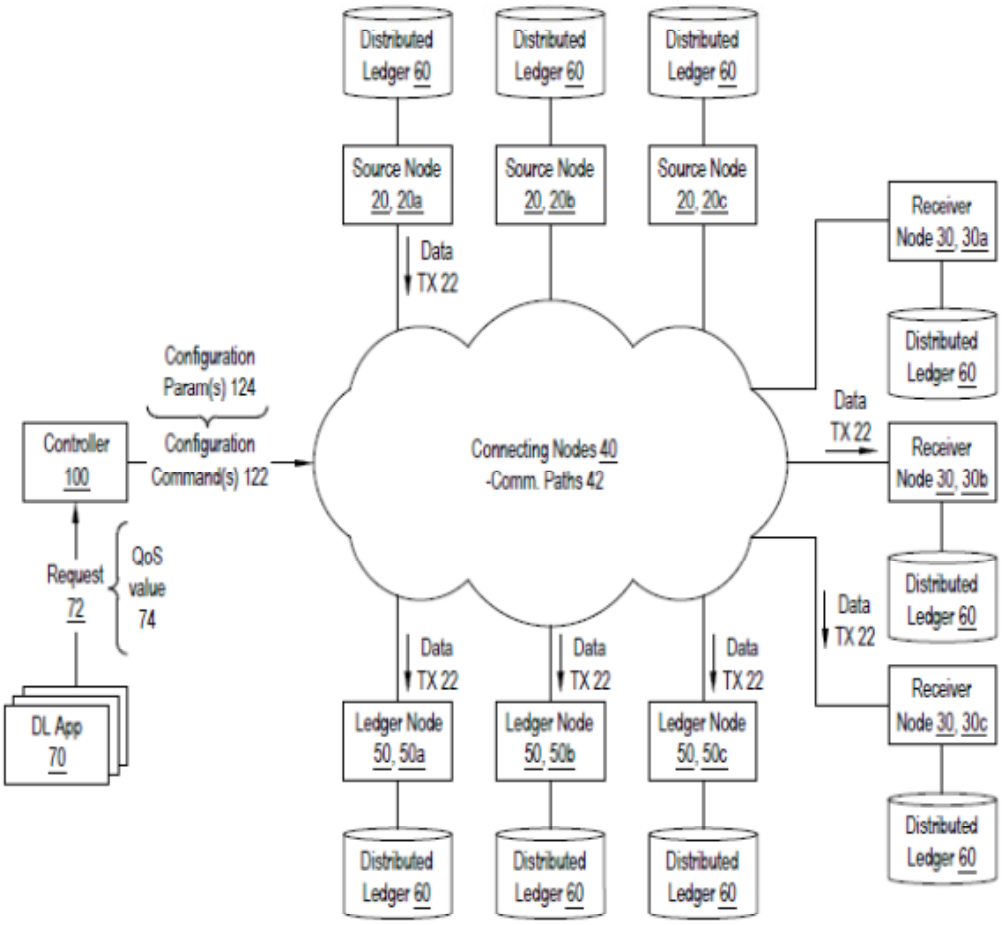

Figure 2

During operation, a source node (e.g., the first source node) can initiate a data transmission. In various implementations, a data transmission indicates (e.g., includes) information related to a transaction. In some implementations, the transaction can be between a source node and one or more receiver nodes (e.g., between the first source node and the second receiver node). In various implementations, the transaction is recorded in the distributed ledger at each node. As such, in various implementations, the source node transmits the data transmission to a receiver node and the ledger records it. The data transmission can include a set of one or more packets, or frames.

The connecting nodes can provide communication paths for the data transmission. In some implementations, the communication paths include one or more routes that the data transmission traverses to reach the receiver node(s). In some implementations, the connecting nodes are connected wirelessly (e.g., via satellite(s), cellular communication,

Wi-Fi®, etc.) in which the communication paths include wireless communication paths. In some implementations, the connecting nodes are connected via wires (e.g., via fiber-optic cables, Ethernet, etc.) in which the communication paths include wired communication paths. More generally, the communication paths can include wired and/or wireless communication paths.

The distributed ledger can be generated and copied across all nodes to be in coordination with each node on the blockchain. In some implementations, the distributed ledger stores transactions. For example, in some implementations, the distributed ledger stores the transaction(s) indicated by the data transmission. As such, the distributed ledger can serve as a record of the transactions that the distributed ledger receives, validates, and/or processes. In some implementations, each node on the blockchain stores a copy (e.g., an instance) of the distributed ledger. As such, in some implementations, there is no need for a centralized ledger. It's also possible to store the information externally and have a cryptographic method which proves the current state of the ledger to the requesting node; e.g., a Zero Knowledge Proof of ledger state.

One or more ledger receiver nodes can receive the data transmission. In some implementations, the various nodes on the blockchain (e.g., source nodes, receiver nodes, and connecting nodes) are added to the blockchain and allowed access to the distributed ledger based on a consensus determination between the existing nodes on the blockchain. For example, a node wishing to join the blockchain can do so in response to receiving permission to join from a threshold number/percentage (e.g., a majority) of the existing nodes (through a consensus mechanism). In some implementations, the existing nodes compete with each other to authorize the new node and store its associated transactions in the distributed ledger.

The consensus mechanism can be implemented through one or more smart contracts which specify whether a node wishing to join the blockchain is allowed to join. This details and implements the network Proof of Authority. In examples, a request is received from a node to join. Various conditions specified within a smart contract is created in accordance with a rule or policy of the smart contract. If the node fulfills conditions in accordance with fulfilling the rule or policy of the smart contract, the node is allowed to join the blockchain. Decentralized status information for the node is updated

when the smart contract has been fulfilled and read access to the decentralized ledger (and its associated status information) is granted to the node. Otherwise, the node is barred from joining the blockchain. Figure 3, below, illustrates an example of nodes being restricted from access.

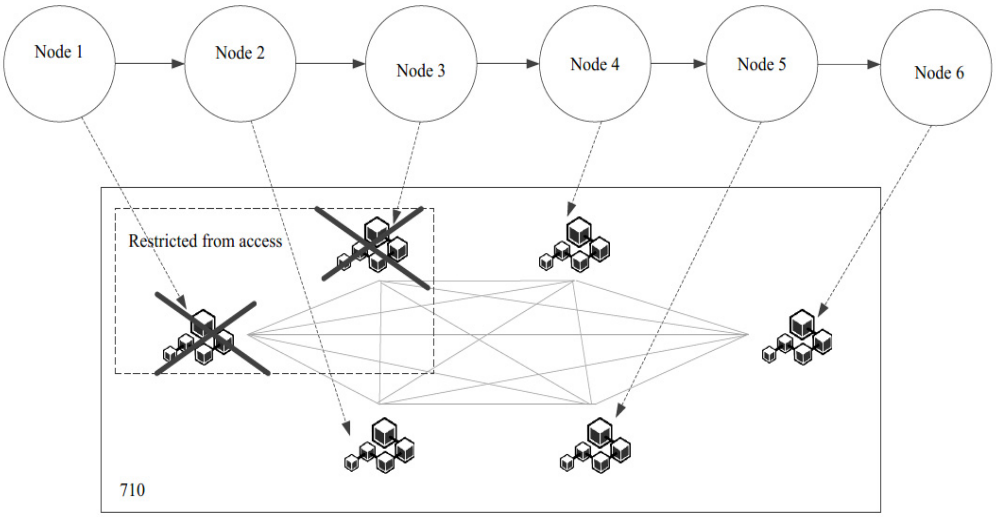

Figure 3

A source node can transmit a request to set up a communication path (e.g., a secure tunnel) to one or more receiver nodes. In some implementations, the request can indicate a quality of service associated with the communication path. In various implementations, the quality of service can indicate: a time duration during which the quality of service is applicable; a latency value that indicates an acceptable level of latency (e.g., an acceptable transmission time) for data transmissions associated with setting up a tunnel between the source node and the receiver node; and/or a priority level associated with data transmissions; combinations thereof; and/or the like.

In some implementations, the blockchain can adjust the performance of at least a portion of the communication path(s) between the source node and the receiver node(s) based on a specified security level. For example, if a node has been barred access to the blockchain because it does not fulfill the smart contracts specifying acceptable nodes, then the communication path will not go through that node (because it will be unsecure). However, if the node has been accepted into the blockchain, then traffic can be routed

through that node, thus ensuring that only trusted nodes on the blockchain handle traffic. In this way, the communication paths remain secure.

For example, nodes that are accepted or unaccepted in the blockchain can be filtered through smart contracts, which control the level or type of traffic that specific nodes on the blockchain can view, access, or modify. For example, the smart contract may specify that a node meets the requisite level of security (for traffic with sensitive or proprietary information) and may specify that another node does not meet security standards and should be restricted from joining the network.

Various communication paths can be provided by the connecting nodes. In an example, as illustrated in Figure 4 below, there are six connecting nodes that form various communication paths. As noted previously, a communication path can provide a route for a data transmission. Consider for the example of Figure 4 that a data transmission from source node 20 would like to reach first receiver node 50a. However, connecting node 40a has been deemed untrustworthy and does not fulfil the smart contract for joining or remaining on the blockchain. The data transmission cannot reach the first receiver node 50 by traveling over communication paths 42a, 42g, and 42i through connecting nodes 40b and 40a. Thus, in this example, the data transmission reaches the first receiver node 50a by traveling over communication paths 42a, 42h, 42k and 42l, and through connecting nodes 40b, 40d, and 40c. Further, the data transmission reaches the second receiver node 50b by traveling over communication paths 42a, 42h, 42k and 42m, and through connecting nodes 40b, 40d, and 40c. Additionally, the data transmission 22 reaches the second receiver node 30b by traveling over communication paths 42a, 42h, 42o, and 42t, and through connecting nodes 40b, 40d and 40e.

In some implementations, the blockchain determines the route of the data transmission 22 over the communication paths 42 and through the connecting nodes 40 based on a function of the security of the nodes. In other words, the blockchain determines which connecting nodes 40 and communication paths 42 are to transport the data transmission 22 based on a function of whether the nodes are trusted within the blockchain network (e.g., allowed to join) or not.

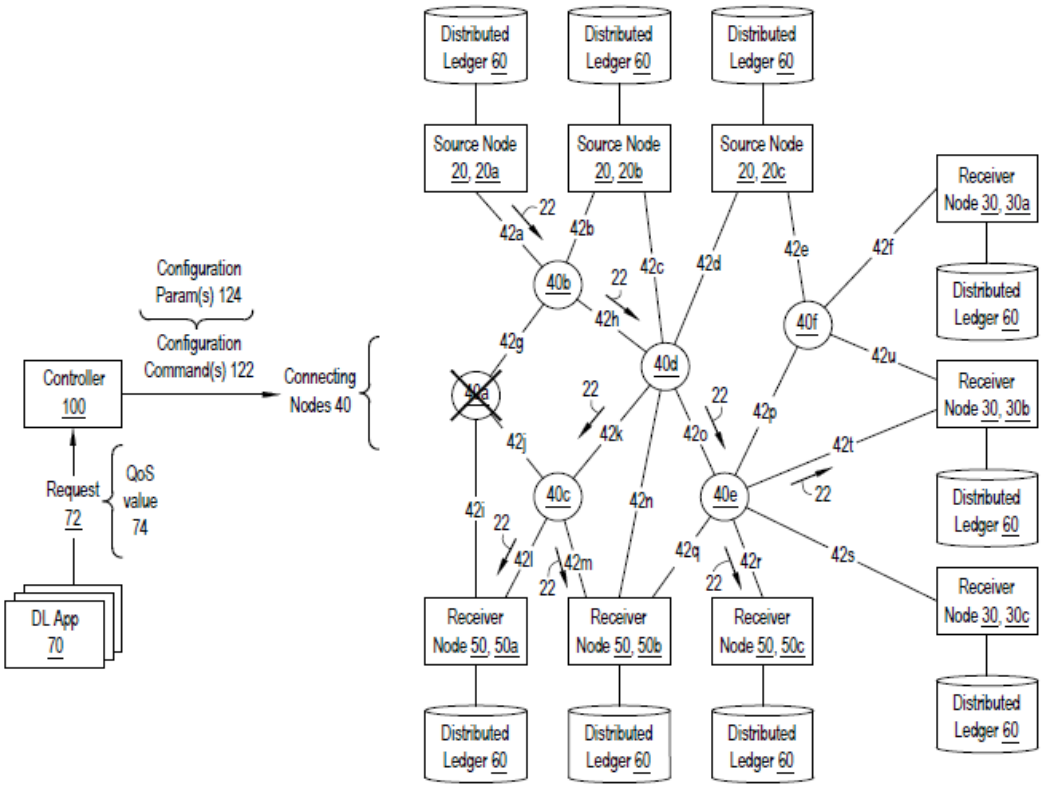

Figure 4

In summary, techniques presented herein provide for the ability to direct traffic through a distributed network. A first node can receive packets that are directed to a second node. The first node can determine that the second node is a member of the distributed network based on information within a distributed ledger stored locally on the first node in which the second node is a member based on fulfilling a condition within a smart contract that manages membership. A tunnel from the first node to the second node can be generated to send the packets to the second node.