Technical Disclosure Commons

Defensive Publications Series

January 2020

# NODE AND SERVICE DISCOVERY IN WIRELESS LOCAL AREA NETWORK CONTROLLER CLUSTER ARCHITECTURES USING HYPERLEDGER

Niranjan M. M

Nagaraj Kenchaiah

Vijay Kothamasu

Ramachandra Murthy

# NODE AND SERVICE DISCOVERY IN WIRELESS LOCAL AREA NETWORK CONTROLLER CLUSTER ARCHITECTURES USING HYPERLEDGER

AUTHORS:

Niranjan M M
Nagaraj Kenchaiah
Vijay Kothamasu
Ramachandra Murthy

## ABSTRACT

In the wireless cluster deployments, node discovery is a challenge. Further, it can be more challenging to discover the services running in such deployments. There are existing methods which use "client side discovery" and "server side discovery" that need a centralized "Service Registry" to maintain all available service instances. Presented herein are techniques that provide for the utilization of a private blockchain and HyperLedger in order to discover nodes and their services in wireless cluster deployments. Techniques of this proposal may provide for de-centralizing node and service discovery in wireless cluster deployments without compromising authentication and security aspects. In one example, when a node comes up, it can authenticate itself with a Blockchain provider to be recognized as a legitimate node for a deployment. The node and services associated therewith would be added to the Ledger. The Ledger can be made available to all nodes in the deployment, which allows members to learn the node and service details and further communicate with the respective nodes for various services.

## DETAILED DESCRIPTION

In a traditional application running on physical hardware, the network locations of service instances are relatively static. For example, an application can determine network locations from a configuration file that is occasionally updated. With the advent of cloud hosting and micro-services architectures, service instance network locations are dynamically assigned. Moreover, the set of service instances can change dynamically due to auto-scaling, failures, and/or upgrades. Consequently, applications need to have a more resilient and elaborated service discovery mechanism. In general, there are two main service discovery patterns: "client side discovery" and "server side discovery."

When using client side discovery, a client is responsible for determining the network locations of available service instances and load balancing requests across them. The client queries a "service registry", which is a database of available service instances. The client then uses a load balancing algorithm to select one of the available service instances and makes a request.

The network location of a service instance is registered with the service registry when it starts up. It is removed from the service registry when the instance terminates. The registration of a service instance is typically refreshed periodically using a heartbeat mechanism. Netflix® Open Source Software (OSS) is an example of a client that utilizes client side discovery. Netflix® Eureka is an example of a service registry that provides a Representational State Transfer (REST) Application Programming Interface (API) for managing service instance registration and for querying available instances. Netflix® Ribbon is an Inter Process Communication (IPC) client that works with Eureka to load balance requests across the available service instances.

When using server side discovery, a client makes a request to a service via a load balancer. The load balancer queries the service registry and routes each request to an available service instance. As with client side discovery, service instances are registered and deregistered with the service registry. The Amazon® Web Service (AWS) Elastic Load Balancer (ELB) is an example of a client that utilizes server side discovery. An ELB is commonly used to load balance external traffic from the Internet.

Both of client side discovery and server side discovery utilize a centralized Service Registry, which maintains all available service instances. The service registry is a key part of service discovery. It is a database containing the network locations of service instances. A service registry needs to be highly available and up to date. Clients can cache network locations obtained from the service registry. However, that information may eventually become out of date and clients become may be unable to discover service instances. Consequently, a service registry consists of a cluster of servers that use a replication protocol to maintain consistency.

Such discovery mechanisms are also applicable in wireless cluster deployments in which all nodes and their services need to be registered with a central service registry for other nodes or applications to discover. As noted, existing methods such as "client side

discovery" and "server side discovery" utilize a centralized "Service Registry" to maintain all available service instances. However, in wireless cluster deployments, node discovery is a challenge. Further, it is challenging to discover services running on nodes.

Considering the disadvantages of centrally managed methods, what is needed is a way to de-centralize node and service discovery for the nodes to know all other participants and their services in-order to communicate with them. Further, an authenticated and secure mechanism for communicating these across the nodes in the cluster deployments may also be needed.

This proposal provides techniques to de-centralize node and service discovery in wireless cluster deployments without compromising authentication and security aspects of such deployments. As deployments may be implemented on public or private clouds, it is important to ensure the participating nodes are authentic and communications are secured.

Techniques of this proposal provide for the utilization of HyperLedger to discover nodes and their services in wireless cluster deployments, which may be useful in cases such as micro-services architecture in which services may run on the remote nodes in a cloud deployment.

HyperLedger, which is developed and maintained by Linux® foundation, is a permissioned blockchain framework that provides privacy and confidentiality required especially for the enterprise deployments and is sometimes referred to as a private blockchain. To implement techniques of this proposal, enterprise deployments would set up a private blockchain to have permissioned network that adds restrictions on who is allowed to participate in the network. Participants will need to obtain an invitation or permission to join. Authenticated participants will have authority to access to permissioned ledger, thereby, allowing for addition of new nodes and/or services. Once a participant has joined the network, the participant will play a role in maintaining the blockchain in a de-centralized manner. HyperLedger can provide this private blockchain functionality for such enterprise wireless cluster deployments.

For such wireless cluster deployments, the Endorser and Consenter functionalities of HyperLedger can be provided by a Cluster Leader node that acts as a Blockchain Provider (BP). Optionally, BP functionality may be provided on a trusted node outside the cluster.

Using HyperLedger provides for the ability to ensure that only authenticated nodes have access to the permissioned ledger. Authentication using secure credentials may not be sufficient, rather it is preferable to verify the trustworthiness and integrity of the nodes before adding them to the Ledger.

To achieve this trustworthiness, an attestation protocol may be utilized, as follows:

1. To attest a given node, the BP retrieves a Public Attestation Identity Key (PK_AIK_NODE) and sends a "nonce" to the target node.
2. The target node answers with a quote that contains the "nonce" and its current Platform Configuration Registers (PCR) values that are stored inside in a local Trusted Platform Module (TPM) of the target node.
3. This quote is signed by a Private Attestation Identity Key (SK_AIK_NODE) of the target node to ensure the integrity of the response.
4. After the receipt of the payload from the target node, the BP compares the "nonce" to check the freshness of the attestation and if this matches the expected value, the BP compares the received PCR values with a library of trusted node configurations.

Note, in some implementations, the TPM functionality and the library of trusted node configurations may not have hardware details such as, for example, in the cases where a node is deployed on a virtual machine or on a cloud.

Figure 1, below, is a diagram illustrating example details that may be associated with cluster node and service discovery for at least one implementation. For simplicity, only one service per node is illustrated in the diagram; however, it is to be understood that a single node may have multiple services running in parallel.

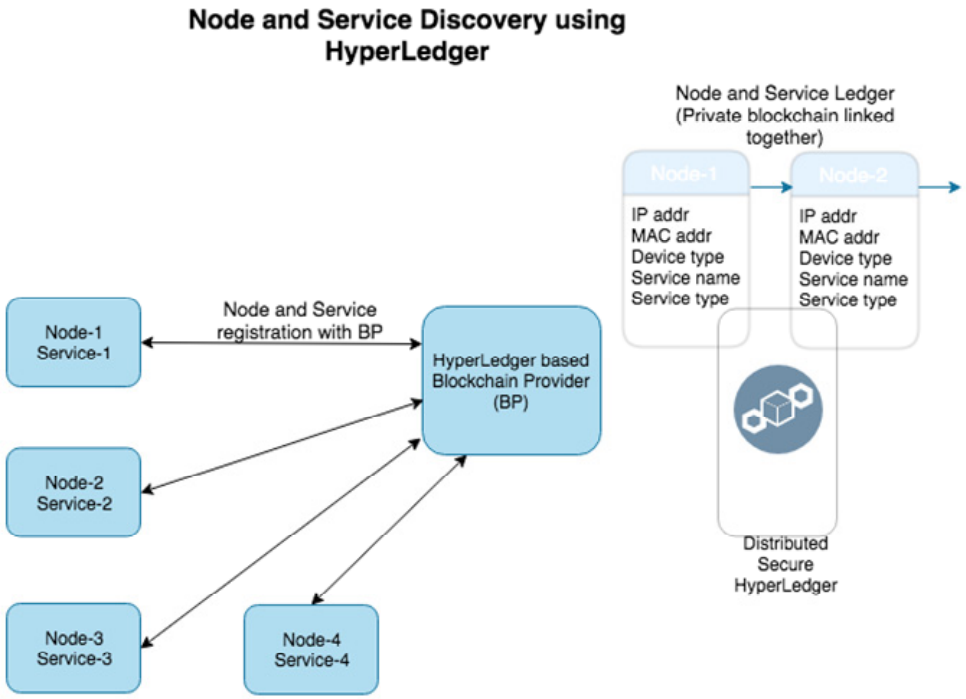


*Figure 1*

To facilitate cluster node and node service discovery for a wireless cluster deployment as illustrated in Figure 1, consider that all the nodes in the wireless cluster deployment are enabled with HyperLedger functionality (e.g., logic/software). The Leader node within the cluster will have the Endorser and Consenter functionalities of HyperLedger and will manage the distributed ledger.

Nodes in the cluster can be authenticated with BP using secure credentials. Along with authentication, trustworthiness of the node is established using attestation protocol as explained above. The nodes register with the BP and are added as legitimate blockchain entities in the ledger. The permissioned ledger is used to allow only authenticated and trusted nodes to participate and share the services. The BP can maintain a list of all registered nodes in the distributed ledger and can help with accounting, lawful intercept, and maintaining immutable records.

In various implementations, Markle tree hash algorithms, such as double Secure Hash Algorithm (SHA) SHA-256, can be used for generation of public keys for enhanced security. Enterprise consensus algorithms such as Proof of elapsed time (PoET) or Practical Byzantine Fault Tolerance (PBFT) can be used to synchronize the database among all nodes in the cluster.

To facilitate service discovery, whenever a service comes up on a respective node, the service can be registered to the blockchain by the respective node as a valid network service along with the service type of the service. It is to be understood that only authenticated and trusted nodes can access the permissioned ledger to update the service.

Each block in the ledger will have the identities of a node and the service(s) that is/are provided by the node. Consider an example involving a wireless clustering environment in which the role of a node can be Leader, Master capable, or Member. If more than one node has the role of Master capable, then the nodes having this role could execute a consensus algorithm to elect a Leader node such that remaining nodes may be considered Member nodes.

Whenever a Leader capable node comes up, it registered with the secure ledger with the role of Leader. The Leader node can provide a load balancing service (to provide load balancing of Access Points across the cluster nodes) and can register this service with the secure HyperLedger. Other nodes would register with secure HyperLedger as Member nodes.

Whenever a Member node seeks to participant in load balancing service, it can analyze the HyperLedger to determine the Leader node information such as Internet Protocol (IP) address, Port number, security type, etc., to connect with the Leader node. Using this information, a Member node can establish a secure connection with the Leader node. The Leader node can also verify the Member node by looking into the distributed ledger before admitting the connection.

For a given deployment, the blockchain may be implemented with a consensus finality property, which provides that once a valid block is appended to the blockchain, the block is never abandoned from the blockchain. For such implementations, a Byzantine Fault Tolerance (BFT)-based blockchain may be utilized. A BFT-based blockchain satisfies the property of consensus finality and can also provide excellent network performance, throughput, etc. for thousands of clients, which is significant for applications involving wireless cluster deployments.

Thus, techniques of this proposal may be utilized in public and private cloud wireless local area network (LAN) controller (WLC) cluster deployments. In some implementations, the techniques can also be adopted in micro-services architectures such

as cloud deployments, in which services may be running on remote nodes. Further, techniques of this proposal may be applicable in generic cluster deployments in which nodes and services can be discovered without utilizing a centralized service registry.

In summary, techniques of this proposal provide for utilization of a private Blockchain and HyperLedger to discover nodes and their services in wireless cluster deployments. The techniques provide for de-centralizing node and service discovery in wireless cluster deployments without compromising authentication and security aspects. Various advantages may be realized by the techniques of this proposal including, but not limited to: providing straightforward, secure, and automated mechanisms for adding and removing services for a deployment; facilitating faster convergence of node and service discovery; providing built-in fail-over mechanisms; and enabling non-static location of services, which can be dynamically changed and automatically learned within a network.