March 2020

TRACEABILITY AND TROUBLESHOOTING IN WIRELESS CLUSTER DEPLOYMENTS USING PROVENANCE METADATA AND HYPER LEDGER

Niranjan M. M

Nagaraj Kenchaiah

TRACEABILITY AND TROUBLESHOOTING IN WIRELESS CLUSTER DEPLOYMENTS USING PROVENANCE METADATA AND HYPER LEDGER

AUTHORS:
Niranjan M M
Nagaraj Kenchaiah

ABSTRACT

Techniques are described herein for enhancing traceability and troubleshooting in complex enterprise wireless cluster deployments using provenance metadata and a hyper ledger. State and event information are captured and used to reconstruct/recreate state machines and event diagrams (e.g., using Unified Modeling Language (UML)) which may be directly mapped to the code. The states and events of all Wireless Local Area Network (LAN) Controllers (WLCs) in the cluster are maintained as provenance metadata. Provenance metadata may improve troubleshooting abnormalities/issues caused by an event or state change (positive provenance), and may help in debugging issues caused by missing events (negative provenance). The metadata is maintained as a transaction in the hyper ledger of a private blockchain, which may help in troubleshooting incidents caused by attacks (e.g., repudiation attacks, etc.). The transaction records are signed by the source to provide authenticity of the information that is especially required in the absence of a Trusted Platform Module (TPM).

DETAILED DESCRIPTION

Enterprise wireless clustering deployments comprise clusters/groups of Wireless Area Network (LAN) Controllers (WLCs), intended to provide collaborative services such as load balancing of Access Points (APs), distributed multicast Domain Name System (mDNS) gateways, etc. A cluster may comprise WLCs with respective roles as leader (master) and worker (member). One of the WLCs in the cluster is elected Leader using a consensus algorithm. The leader is the point of contact for all configurations, image management, load balancing (distribution) of APs among worker WLCs, providing management plane activities such as show commands, assurance data, etc.

These cluster deployments may be large (e.g., they may include three to ten, or even more, WLCs), and as such it can be difficult to detect faults therein. Debugging in such

Published by Technical Disclosure Commons, 2020

complex wireless cluster deployments poses great challenges as the state of the system continuously changes due to the occurrences of various events (e.g., worker WLC connects/disconnects to/from the leader WLC, High Availability (HA) switchover of the leader/worker WLC, load balancing of APs based on different criteria such as site tag, Radio Resource Management (RRM) neighbor, round robin methods, etc.) and network issues (e.g., link up/down, port up/down, gateway unreachable, etc.). Sometimes the cluster deployments are distributed in nature (e.g., virtual WLCs, cloud deployment, etc.), and identifying faults in those deployments is particularly complex. The faults are often partial, irregular, and may result in abnormal behavior rather than system failure. Diagnosing a problem in such systems requires collecting relevant information from all WLCs in the cluster and correlating those with the problem.

Today, it is required to manually debug from the logs generated locally and/or from the remote syslog server, even for simple issues (e.g., WLC is not reachable, a worker WLC is not connecting to the leader WLC, APs are not load balanced properly among worker WLCs, etc.). For local debugging, all the WLCs in the cluster (and/or APs) may need to be accessed (e.g., via console, Secure Shell (SSH), Telnet, etc.). For debugging using remote syslog, it may be necessary to sift through every log from all the WLCs configured with the same syslog server. These logs/debugs do not enable troubleshooting abnormalities caused by the absence of events (i.e., missing events, which are events that did not occur).

In other words, conventional (standard) logging methods merely involve logging all generated messages locally and/or with the remote syslog (and/or assurance), which creates debugability issues due to the volume of the logs. It is even more difficult when the same syslog server is used for multiple WLCs, which is the case with clustering deployments where the same global configuration is shared among WLCs of the cluster. Bifurcation of specific information about the state and event of the WLCs from these logs is particularly problematic. Moreover, these logs do not carry information required for troubleshooting abnormalities/issues caused by missing events, or information required for tracing and troubleshooting any incidents resulting from attacks. For example, a repudiation attack is used to modify the authoring information by the attacker in order to log the wrong data to the log files. If this attack takes place, the data stored on the log files

can be considered invalid or misleading. Conventional logging methods lack traceability and troubleshooting tools even though these are required for complex wireless cluster deployments, especially when they are distributed.

In addition to conventional logging methods, there are also methods such as assurance/telemetry whereby a large set of data is captured with a series of events and state changes for all the wireless clients along with WLC/AP events. These data are used for network analytics that are difficult to trace and troubleshooting abnormalities/issues caused by missing events and attacks (e.g., repudiation attacks).

There are existing signed logging systems that operate using Trusted Platform Module (TPM) methods. However, these simply enhance authentication and trustworthiness of the logged messages without improving the traceability and troubleshooting of any issue. In other words, these methods do not simplify debugability. Thus, conventional and signed logging method do not incorporate any extra information to troubleshoot abnormalities/issues caused by missing events. Moreover, because large volumes of data and logs are generated by multiple features, debugging using syslog may require an in-depth understanding of WLCs and their features. However, syslog also does not provide information to debug issues caused by missing events, and does not help identify attacks such as repudiation attacks.

The latest system design methodologies are driven by data, state change, and triggered events. Absence of integrity and validity of information may mislead and create an unwanted result. Hence, it is vital to ensure the integrity and validity of the information as well as to track how information has been manipulated through its current state.

Accordingly, in order to minimize the debugability, troubleshoot abnormalities/issues, overcome attacks, etc., presented herein are techniques for collecting state and event information regarding the WLCs in the cluster along with network events (e.g., link and/or port state changes) to enable diagnosing of problems in the wireless cluster deployments. This may be accomplished by plotting state changes against state machine diagrams, and triggered events against event diagrams. State and event information may be securely captured and used to reconstruct/recreate state machines and event diagrams (e.g., using Unified Modeling Language (UML)) which may be directly mapped to the code. Existing tools may be used to convert state machines and event

diagrams to code and vice-versa. These generated state machines and event diagrams may help to map/troubleshoot any abnormalities/issues/attacks caused by missing events/flows. Thus, the techniques described herein may involve troubleshooting abnormalities/issues in the network using states and events before debugging further using syslog/assurance.

Figure 1 below illustrates an example system configured for traceability and troubleshooting in a wireless cluster deployment using provenance metadata and a hyper ledger.

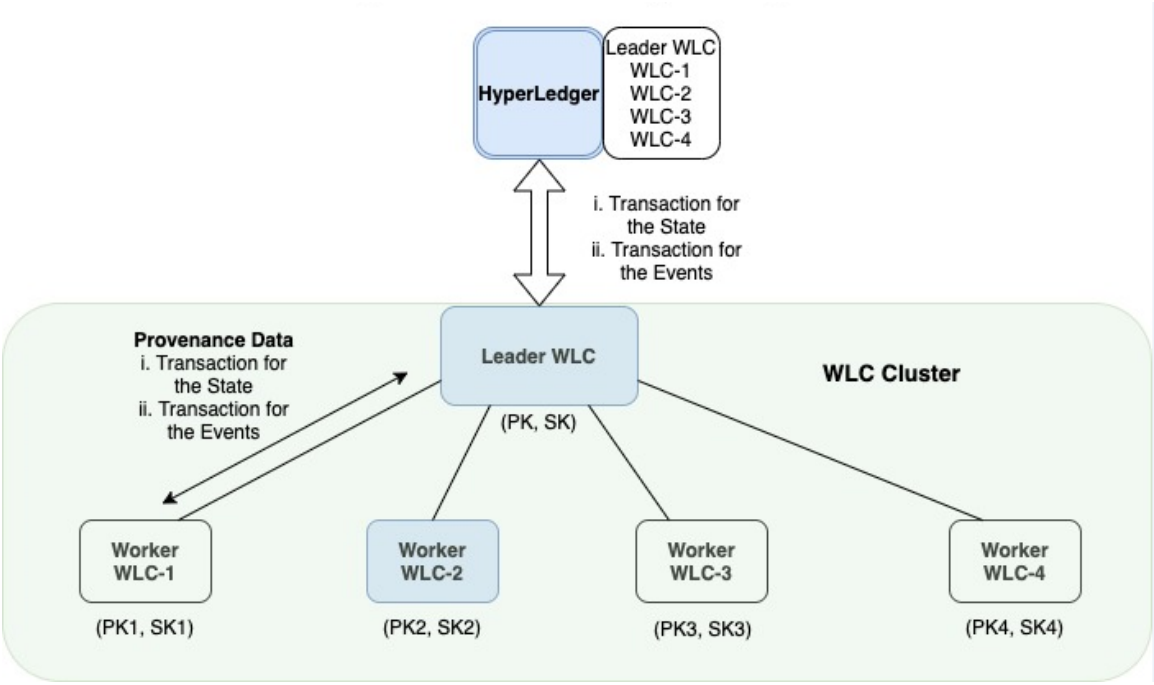

Figure 1

Techniques described herein may utilize the concept of provenance. Provenance relates to mechanisms for caching state and event information related to the whole system, and using that information to track issues whenever problems occur in deployment. Provenance may enable tracing and analyzing problems in complex systems and determining causes of errors and unwanted behavior. Provenance techniques may thus improve traceability and troubleshooting by caching metadata of state changes and events of all WLCs in the cluster. Provenance metadata comprises the history of all WLCs in the cluster, including network events and state changes of all WLCs in the enterprise wireless cluster deployment starting from the beginning state through the current state. All information (e.g., creation time, modified time, etc.) is collected as provenance metadata.

State change metadata (e.g., for the APs) may include information relating to, or statuses such as, booting up, running, active, standby, master capable, leader, worker, follower, joined/disjoined to/from WLC, moved to standalone from connected mode, start/stop of the running services (e.g., cluster services, load balancing service, mDNS service, etc.), etc. Network event metadata may include information relating to WLCs (e.g., HA switchover, link up/down, port up/down, stoppage of service, restart of service, gateway unreachability, etc.), WLCs in a mobility group (e.g., mobility member added/deleted, mobility tunnel up/down, etc.), the cluster (e.g., leader election (consensus algorithm among WLCs), worker WLC connected/disconnected to/from the leader WLC, leader failover, etc.), or the APs (e.g., Control and Provisioning of Wireless APs (CAPWAP) keep-alive messages lost, link up/down, port up/down, etc.).

There are two main types of provenance metadata: positive provenance metadata and negative provenance metadata. Positive provenance metadata ensures the integrity and validity of the information as well as how information has been manipulated through its current state. This ability to explain state changes and the reason(s) behind those changes may enable debugging and diagnosis of various faults. A backtrace may be constructed based on the occurrence of an event or state change of an entity and, if required, the event or set of events that caused the state change or abnormalities may be recursively identified.

For example, WLCs may encounter resource and network failures. Hence, analyzing provenance metadata may be an important mechanism for detecting network failures and monitoring resource malfunctions. Positive provenance metadata may help explain the series of actions that led to the change of an object to its current state from that of its origin. For example, the change could be due to the data, a state, or an event of the WLC in the deployment. Tracking and then analyzing preceding events helps to diagnose the actual reason behind the system failure or security breaches.

Negative provenance metadata uses counterfactual reasoning to identify the conditions for which the missing event would have occurred. This also provides a way to construct a backtrace (e.g., determining reasons explaining the cause of the missing event). The complexity of negative provenance metadata may be greater than that of positive provenance metadata. With positive provenance metadata, a specific chain of events that led to an observed event is known, whereas with negative provenance metadata, all possible

chains of events causing observed event may be considered. In short, positive provenance metadata explains why a state or an event changed or occurred, and negative provenance explains why a state or an event did not change or occur.

For example, the absence of events (e.g., discovery requests not reaching a worker WLC, unavailable events such as socket descriptor of worker WLC not being added to the event loop (evloop), etc.), can be explained by using counterfactual reasoning to identify conditions under which these events could have occurred. In order to find a missing or negative event in a WLC, all possible positive events may be considered from the WLC (Device Under Test (DUT)) and from other WLCs in the cluster that would have resulted in the particular missing event. The reason may be deduced from that information. In this case, a large chain of events needs to be considered.

Although negative events cannot be explained directly with positive provenance, there is a way to construct a similar backtrace for negative events. Instead of explaining how an actual event did occur, as with positive provenance, all the ways in which a missing event could have occurred may be determined, and then the root cause for why each of them did not come to pass may be shown. In short, counterfactual reasoning may be used to recursively generate the explanations, not unlike positive provenance.

Consider an AP, AP1, that initially connects to a leader WLC, WLC1. WLC1 may perform load balancing of AP1 among WLCs in the cluster based on the site tag, RRM neighbors or round robin method, and select a worker WLC, WLC2, for AP1 to join. Visually, AP1 --> WLC1 --> WLC2 --> load balancing, and AP1 --> WLC2 --> Join/Connect. For a CAPWAP discovery request to arrive at WLC2, a request would have had to appear at the WLC1, which did not happen. Such a request could only have to come from AP1 and eventually from WLC1 to WLC2. However, WLC1 would only have sent the request to WLC2 if there had been: (1) an actual CAPWAP discovery request from AP1; (2) a load balancing algorithm on WLC1 considers WLC2 as one of the WLC in the cluster; (3) a load balancing algorithm on WLC1 selects WLC2 for AP1 to connect; or (4) there is communication channel (e.g., tunnel) over User Datagram Protocol (UDP) / Transmission Control Protocol (TCP) / Transport Layer Security (TLS) between WLC1 and WLC2. If conditions (1), (2), and (3) were satisfied, but condition (4) was not (because the link between WLC1 and WLC2 is not stable and disconnect occurred between the

aforementioned events), then it may be determined through positive provenance where the communication channel would have been established. Similar techniques are applicable for finding missing events using positive provenance.

Situations may also arise in which a compromised WLC may provide incorrect information in the absence of a TPM (e.g., attestation/trustworthiness mechanism). In such cases, provenance metadata may be used to track manipulation or tamper-evident properties from other WLCs in the cluster to assist the operator/administrator in detecting the compromised WLC. Thus, provenance metadata may provide useful information to analyze, optimize, and secure any suitable system.

The wireless cluster deployments are more prominent in virtual deployments which utilize virtual WLCs (vWLCs) and cloud deployments. The hyper ledger of the private blockchain may enable use of an authenticated ledger, which is a mandatory requirement for enterprise deployments. Blockchains maintain tamper-proof transactions (i.e., once a transaction is created on the blockchain, it cannot be modified or deleted). As such, blockchains may be used for troubleshooting abnormalities due to attacks (e.g., repudiation attacks, etc.). The hyper ledger may also be used to provide authenticated access to the ledger based on initial registration. Only state and event changes of the WLCs in the cluster are maintained as transactions in the hyper ledger and not the whole syslog. For syslog and assurance, existing signed logging system may be used.

Provenance metadata may be maintained as transactions in the hyper ledger of the private blockchain to improve traceability and troubleshooting in complex enterprise wireless cluster network deployments. All authenticated WLCs of the cluster may have access to the hyper ledger for tracing and troubleshooting network issues at any point in time and on any of the WLCs in the cluster. Metadata for the network events may be customizable by the configuration based on given requirements, thereby minimizing the number of transactions maintained in the hyper ledger. By using the metadata of state changes and available events as transactions in the hyper ledger, the state machine diagram and event diagram may be reconstructed and used to map to the code.

For simplicity, only one leader WLC and one worker WLC are considered in the following example, but it will be appreciated that these techniques may be extended to a larger deployment comprising a cluster of WLCs. The transaction model enhances

traceability and troubleshooting by recording all state changes and network events among WLCs, which helps in capturing the behavior of the entire network. Within a running network, if the network suffers abnormal attacks, the attack process is also logged as a transaction. With these logged transactions of attack trajectories, any future attacks launched on the network may be identified using attack pattern recognition. For enhancing traceability and troubleshooting across WLCs in the cluster, transactions for the WLC states (Transaction_STATE) and events (Transaction_EVENT) may be created and added to the hyper ledger of the private blockchain.

Major state changes of the leader WLC, starting from boot-up, may be captured in the hyper ledger as transactions ("state provenance"). The state of the leader WLC may be defined as STATE_LEADER = (ID_LEADER, ID_state, old_state, new_state, SK_LEADER, PK_LEADER). ID_LEADER may represent the identity of the leader WLC. ID_state may represent the identity of the state, which is maintained as the transaction. old_state may represent the previous state, and new_state may represent the current state. SK_LEADER may represent the private (secret) key of the leader WLC, and PK_LEADER may represent the public key of the leader WLC. SK_LEADER and PK_LEADER may be used for authenticating the information. The corresponding transaction may be defined as Transaction_STATE_LEADER = (ID_WLC_LEADER, ID_state, STATE_LEADER, SIGNATURE_LEADER). SIGNATURE_LEADER may be defined as DS.Signature (SK_LEADER, ID_LEADER, ID_state), and may represent the signature of the transaction for the state change by the leader WLC using its private key.

Major state changes of the worker WLC, starting from boot-up, may be captured in the hyper ledger as transactions ("state provenance"). The state of the worker WLC may be defined as STATE_WORKER = (ID_WORKER, ID_state, old_state, new_state, SK_WORKER, PK_WORKER). ID_WORKER may represent the identity of the worker WLC. SK_WORKER may represent the private (secret) key of the worker WLC, and PK_WORKER may represent the public key of the worker WLC. SK_WORKER and PK_WORKER may be used for authenticating the information. The corresponding transaction may be defined as Transaction_STATE_WORKER = (ID_WORKER, ID_state, STATE_WORKER, SIGNATURE_WORKER). SIGNATURE_WORKER may be

defined as DS.Signature (SK_WORKER, ID_WORKER, ID_state), and may represent the signature of the transaction for the state change by the worker WLC using its private key.

Major events that occurred on the WLCs, starting from the first negotiation, may be captured in the hyper ledger as transactions ("event provenance"). These events may be customized per configuration. An event that occurred on the leader WLC may be defined as EVENT_on_Leader = (ID_LEADER, ID_event, event, SK_LEADER, PK_LEADER). The ID_event may represent the identity of the event, which is maintained as a transaction. The corresponding transaction may be defined as Transaction_EVENT_on_LEADER = (ID_LEADER, ID_event, EVENT_on_LEADER, SIGNATURE_LEADER). SIGNATURE_LEADER may represent the signature of the transaction for the event by the leader WLC using its private key. Similarly, transactions may be created for events on worker WLCs, but signed by the worker WLCs using their private key(s).

Once transactions have been created for the states and events, they may be added to the private blockchain and hence to the hyper ledger. Any of the issues or abnormalities may be troubleshot using this transactional information. To ensure integrity and validity, the WLCs may use a corresponding verification function to verify the signature added to a given transaction by the WLC providing the information to the private blockchain. For example, for event validation, the verification function at the leader WLC may be defined as VER_EVENT_on_LEADER = DS.Verification(PK_LEADER, SIGNATURE_LEADER), and the verification function at the work WLC may be defined as VER_EVENT_on_WORKER = DS.Verification(PK_WORKER, SIGNATURE_WORKER).

Similarly, the verification function for validating the leader WLC may be defined as VER_LEADER = DS.Verification (PK_LEADER, SIGNATURE_LEADER), and the verification function for validating the worker WLC identity may be defined as VER_WORKER = DS.Verification (PK_WORKER, SIGNATURE_WORKER). Along with traceability and troubleshooting, these transactions may help improve accountability by listing out how many events were generated by the leader WLC and the worker WLCs.

The techniques described herein are applicable in many use cases. In cases where enterprise wireless cluster deployments are in a suspicious state and if there are faulty or misbehaving WLCs, provenance information can play a vital role to debug and identify

them. It may also be used to assess the damages that the faulty WLC might have caused to the system. Provenance information can also be used to answer audit questions in order to predict future workloads. Based on such predictions, the performance of the system, as well as the availability of the service, may be identified.

The techniques described herein may be extended to WLCs in a mobility group, APs, etc., by considering the states and events of APs and WLCs in the mobility group. A mobility group may be a group of WLCs that collaborate to provide seamless client roaming. The administrator may configure the level of information (e.g., criticality of state change, priority of events, etc.) required as per the serviceability required. These techniques may be scaled using any suitable mechanisms for blockchain. The use of provenance metadata described herein provides periodic auditing for network behaviors (e.g., network events associated with resulting network states), which helps to enforce the stability and robustness of the enterprise wireless cluster deployment. The traceability and troubleshooting for network issues is also improved. Furthermore, attack pattern recognition may be incorporated to resist future network attacks based on provenance metadata and events.

In summary, techniques are described herein for enhancing traceability and troubleshooting in complex enterprise wireless cluster deployments using provenance metadata and a hyper ledger. State and event information are captured and used to reconstruct/recreate state machines and event diagrams (e.g., using UML) which may be directly mapped to the code. The states and events of all WLCs in the cluster are maintained as provenance metadata. Provenance metadata may improve troubleshooting abnormalities/issues caused by an event or state change (positive provenance), and may help in debugging issues caused by missing events (negative provenance). The metadata is maintained as a transaction in the hyper ledger of a private blockchain, which may help in troubleshooting incidents caused by attacks (e.g., repudiation attacks, etc.). The transaction records are signed by the source to provide authenticity of the information that is especially required in the absence of a TPM.