November 2021

AUTONOMOUS COLLECTION DETECTION AND REMEDIATION DECISIONS BASED ON LOCAL MODELS AND LOCALLY SOURCED DATA

Dave Zacks

Jane (Zizhen) Gao

Nagendra Kumar Nainar

Carlos M. Pignataro

Dmitry Goloubew

AUTONOMOUS COLLECTION DETECTION AND REMEDIATION DECISIONS BASED ON LOCAL MODELS AND LOCALLY SOURCED DATA

AUTHORS:
Dave Zacks
Jane (Zizhen) Gao
Nagendra Kumar Nainar
Carlos M. Pignataro
Dmitry Goloubew

ABSTRACT

In context of distributed monitoring and anomaly detection, when a networking device performs anomaly detection based on local data, such as when a remote controller is not reachable during network convergence or other network issues. Anomaly relevance improves if telemetry data used for anomaly detection comes not only from a local device, but also from the device's immediate surroundings (e.g., physical neighbors, protocol peers, redundancy units, etc.). Presented herein are techniques through which a device can reach its own and nearby telemetry sources in a manner that may follow an effective network topology and configuration. Thus, techniques herein may enable the design of intelligent autonomous agents that can operate beyond the scope of a host (and can integrate nearby information to make smarter assessments) but below the network scale and, hence, are capable of scaling well in order to sample data more quickly and merge data more accurately.

DETAILED DESCRIPTION

Anomaly detection performed in the context of distributed monitoring typically involves a networking device performing anomaly detection based on local data, such as when a remote controller is not reachable during network convergence or other network issues. Anomaly relevance improves if telemetry data used for anomaly detection comes not only from local device, but also from the device's immediate surroundings (e.g., physical neighbors, protocol peers, redundancy units, etc.).

Such anomaly detection involving a device's immediate surroundings provides for the ability to align anomaly detection with many entities that commonly used in network design as building blocks, such as peers (e.g., as utilized via Border Gateway Protocol

(BGP)), neighbors (e.g., as utilized via Link Layer Discovery Protocol (LLDP)), active/standby pairs (e.g., as utilized via Hot Standby Router Protocol (HSRP), Virtual Router Redundancy Protocol (VRRP), Gateway Load Balancing Protocol (GLBP), Link Aggregation Control Protocol (LACP), etc.), and/or other objects that may be addressed using operational data available at device run time (e.g., interface names, protocol peers, table lookup entries, etc.) and bound by logical operators. Thus, expanding beyond a device-centric model allows for the ability to observe anomalies that may not otherwise be visible (e.g., not at the device level due to a lack of complete data, nor at the network level due to more abstract data processing). Additionally, anomalies aligned to entire blocks are easier to integrate with existing operational practices and systems.

Anomaly detection involving a device's immediate surroundings also permits telemetry data to reflect runtime changes in the network. For example, if a neighbor or active standby role changes, received telemetry data can reflect the change (e.g., the change itself can be seen from changes in measurement attributes such as 'source', etc.). This simplifies the development of telemetry data processing pipelines and applications since the data specification can be made in relative terms (e.g., neighbor_of_interface_x) and can remain static, yet accurately reflect the exact network operational configuration/topology at the time of data collection.

Additionally, the local nature of connectivity allows for the ability to pull telemetry data from proximate devices even when routing is down or impaired and also potentially during periods of network instabilities (short- or long-lived). Further, if a particular network device is not reachable locally (e.g., due to link issues), another common neighbor can forward telemetry as a proxy. Thus, the dynamic nature of such mechanisms enables flexible data collection, such that collected data can change as function of previous observations and other stimuli.

This proposal defines techniques through which a device can reach its own and nearby telemetry sources in a manner that can follow an effective network topology and configuration. In other words, the data specification will remain static while data remains accurate and follows changes in the network. Such techniques may facilitate closed-loop telemetry processing applications that adapt and collect different data over time. In some instances, the techniques proposed herein may extend the notation to dynamically resolve

a 2nd stage address (e.g., a YANG (Yet Another Next Generation) path, a Simple Network Management Protocol (SNMP) Object Identifier (OID), or a Command Line Interface (CLI) can be dynamically generated).

Figure 1, below, illustrates an example telemetry collection architecture through techniques of this proposal may be illustrated in which the scope of telemetry collection can be viewed from the perspective of an autonomous behavior unit (ABU) of a router, 'R_{d1}'.

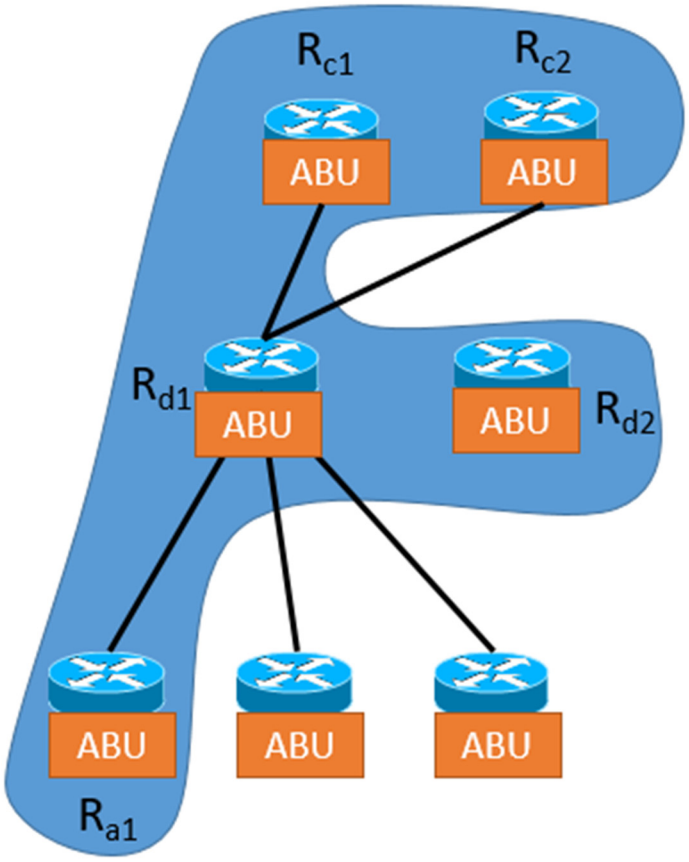

Figure 1: Example Telemetry Collection Architecture

For the example architecture as illustrated in Figure 1, consider that the ABU on R_{d1}, can make decisions not only based on own telemetry, but also telemetry from both upstream (core devices) and one (or all) of the downstream devices. Such decisions can be achieved by defining a 2-stage hierarchical addressing scheme for telemetry. The first stage of the hierarchical addressing scheme is outlined in this proposal, and the second

stage can be realized using any existing or new data addressing scheme, such as YANG, SNMP, or CLI. YANG is discussed with reference to examples provided herein for illustrative purposes only.

In one instance, addressing can be defined as follows:

<anchor>[.<relation1>][…][<relationN>][.property]

For the above definition, an 'anchor' would typically be a hostname or IP address representing a starting point in the taxonomy, a 'relation' would be a path from the anchor to another point in the taxonomy, and a 'property' would be an attribute of the point in in the taxonomy (e.g., a host could have many properties, such as configuration, interfaces, runtime state, etc.)

Since the purpose of this addressing is to specify a location (e.g., IPv4, IPv6, or any other notion of location of the telemetry source in the network), the taxonomy in question would be relatively compact, such that it would typically include interfaces, protocol peers, various tables, etc. through which one or more lookups could be performed to resolve to an address.

Consider an example formatted as:

host.interface.protocol.neighbor

For this example, 'localhost.gigabitEthernet0/1.lldp.neighbor.ip_address' could resolve to an address of an LLDP neighbor of g0/1 on the local device. In another example, 'localhost.gigabitEthernet0/1.broadcast' could resolve to a broadcast out of g0/1, which may be useful for use cases involving a Point-to-Point (P2P) link in which the two sides cannot be assumed to be on the same subnet (e.g., for a fallback/emergency mode configuration).

Consider another example formatted as:

host.lookup.table.entity.field

For this example, 'localhost.lookup.arp_table.gigabitEthernet0/1.1st.ip_address' could resolve to an IP address of the first Address Resolution Protocol (ARP) entry for interface g0/1, which may be useful for P2P links in which no assumption can be made as to whether a remote device supports LLDP. In another example involving BGP, which could be formatted as 'localhost.lookup.bgp.neighbor.as.50.ip_address'. In yet another example, SNMP OID could be used instead of 'relation' and 'property', such as 'host.oid.1.3.6.1.2.1.4.20.1.1'.

The above notations permit an agent-centric view of a network, which provides for the ability to define the entire space of telemetry from the point of view of a certain device (e.g., agent).

In some implementations, this notation may provide for the ability to prepend a YANG path with a relative positional qualifier. For example, 'System-OS-XX-bcm-dpa-npu-stats-oper:stats/nodes/node-stats' could be used to retrieve telemetry objects from a local device, while 'this_host.neighbor.cdp.interface.H1/1/1.System-OS-XX-bcm-dpa-npu-stats-oper:dpa/stats/nodes/node-stats' could be used to collect Network Processing Unit (NPU) statistics from a neighbor - whatever that neighbor is at the time of collection.

Indeed, many devices permit collecting outputs from ‘remote processors’. There are typically two-parts to such collection: 1st from *where* to collect outputs, and 2nd *what* to collect. In case of security appliances and other devices, the answer to the ‘where’ question is often preordained – one has to explicitly specify the location. In some cases, the location could be failover peer, while in others the location could be a rack, slot, processor etc.

A key difference with such collection operations and techniques of this proposal is that ‘where’ can be implicit. In other words, logic does not have to include the exact location of the data being collected. This means that a single artefact of intellectual capital (IC) written using such notation as described herein can work without modification on many networks made of different devices. Another important implication of implicit addressing is that data follows the actual network configuration / topology at the time of collection, which may vary as the network topology itself changes (as it may during a reconvergence event, for example). This permits collecting accurate data from networks

of dynamic operational configurations (e.g., networks with traffic engineering, mesh networks, Delay/Disruption Tolerant Networking (DTN), etc.), and is critical in the highly variable and redundant network topologies that exist today to support mission-critical applications and services.

Further, techniques herein provide a domain specific language (DSL) that provides for the ability to specify data locations anywhere in a network, even if and when that location changes over time. A key point of novelty of such a technique is the ability to collect data without knowing or specifying a single IP address – but rather to learn this dynamically, and be updated as that data may change over the course of network events.

Moreover one doesn't even have to specify 'self' – the device that satisfies the conditions will be the 'self' (the reference for particular execution of particular instance of IC). This affords high expressivity relevant to the domain of network operations and maintenance, providing for the ability to specify conditions such as 'neighbors of a device that has lost >2 BGP neighbors in last 5 minutes'. Such matching can be achieved with head on programming, but it will not be anywhere near real time or scalable (not to mention it would involve hundreds of lines of code to achieve). Thus, the techniques of this proposal enable an entirely new kind of network IC, which can be applied within and improve many service/networking/telecommunications applications and environments (e.g., customer experience, cloud, business critical services, etc.). Additionally, techniques of this proposal do not replicate existing and emerging tools, such as testing/debug tools. If such tools are available near a device, the techniques of this proposal can facilitate interfacing with the tools, and can help further expand the autonomous operation through better, more relevant, and more timely data.

Indeed, YANG and YANG Development Kit (YDK) offer solutions in the area related to specifying the data access (i.e., *what* to collect, as noted above); however, the implicit location addressing is not present in these solutions, making these solutions device-centric. Network insights for a network of 1000 devices is 1000 times the device insights. Network service orchestration and network element drivers are often also reliant on YANG or adaption layers, but, notably, these are controller-level solutions – they are not meant to be operating on a device. The importance of implicit addressing together with local collection stems from the necessity to support cloud-based insight solutions (cloud

L2/L3/L4 solutions) and is a key element of scaling these solutions, while retaining overall simplicity. This entails that solution must be resilient to temporal communication interruptions with graceful degradation in the worst case. The hybrid processing enabled by the architecture of this proposal (involving autonomous processing at the edge, plus decoupled high throughput processing in the cloud) enables the closed-loop remediation approaches without mandating the entire insights stack to be on premise.

In summary, this proposal enables a closed-loop autonomous behavior through which a device can collect telemetry data not only from the device itself, but also from neighbors without assumptions regarding network configuration and topology. Thus, techniques herein give an analytical engine, which is resident on a device, the data to 'see' beyond the source device, even during connectivity issues. This resolves challenges arising from load-balancing and active-standby handing where a local device only sees a fraction of the traffic/signaling/data, and makes switchover situations fully observable by the analytics engine, which can facilitate better error detection and remediation.

Further, the ability to collect data from nearby sources is not reliant of routing to be functional (for adjacent devices) and should therefore be resilient to Layer 3 (L3) connectivity issues. This proposal, thus, enables the design of intelligent agents that can operate beyond the scope of the host (and can integrate nearby information to make smarter assessments) but below the network scale and, hence, scale well (to sample data faster and merge data more accurately) due to the local and limited nature of interactions.

In some instances, retrieval of the data can be made using a mechanism such as GRPC + neighbor address resolution or using pub/sub mechanism (e.g., multicast MQTT, Kafka, etc.), as well as broadcast addressing for P2P. Another use of the mechanism can be to export pod/cluster/access-block/peer-group merged data, such that devices could produce data that could be locally merged and then exported using existing mechanisms such as Model-Driven Telemetry (MDT) subscriptions. Local address (no routing) is another potential side effect in which, for instances involving micro-outages, telemetry data will not be distorted.