April 2023

DYNAMIC FUNCTION UPDATES FOR HYBRID CONTROLLERS

Jerome Henry

Robert Barton

Indermeet Gandhi

DYNAMIC FUNCTION UPDATES FOR HYBRID CONTROLLERS

AUTHORS:
Jerome Henry
Robert Barton
Indermeet Gandhi

ABSTRACT

This submission presents a technique to synchronize data displayed in a Cloud network management unit and an on-premise network management unit for an enterprise deployment. In many cases, the available bandwidth for the Cloud network management unit forbids the near-real-time display of data available at an on-premise network management unit. However, by measuring which page may be accessed and the type of data is to be displayed via the page, the technique of this proposal provides for the ability to stream only the amount of data needed for a Cloud administrator to experience real time management, without consuming a large bandwidth with raw data streaming (as would often be the case with on-premise management).

DETAILED DESCRIPTION

With the trend to move/host/manage services via the Cloud, it is becoming more common to see Cloud network management controllers being added to enterprise deployments in addition to on-premise (on-prem) network management controllers. In many cases, network management controllers that were initially designed for on-prem deployments can now be spun off in Cloud environments. In the future, many network equipment providers will gravitate towards the development/implementation of cloud controllers for more and more deployments.

Today, many enterprise customers of network equipment install an on-prem unit or a connector that is tasked to decide which data is to be sent to the Cloud. For example, in a first step, code for an on-prem unit can run remotely on a Cloud device through the use of on-prem translation functionality. The next step is to implement a common code or YANG model (for example on Wi-Fi access points), with management being performed either through an on-prem controller (e.g., a wireless LAN controller (WLC)), through a higher level on-prem controller, a cloud controller, or in the Cloud. Enterprise customers

are likely to own an on-prem controller (e.g., a WLC) and a remote upper-level management platform.

The link geometry is vastly different between on-prem management platforms and their equivalents in the Cloud. For example, considering Wi-Fi as an example, APs can receive signals from a given station a thousand times per second (a Wi-Fi frame duration is commonly 200 to 500 microseconds), and report that information to an on-prem location engine where a station location can be displayed in real time. Sending this type of volume at scale to a Cloud-based controller is unrealistic.

Further, the user experience is vastly different for an on-prem management persona versus a Cloud-based management persona. This alone causes significant resistance to on Cloud-adoption, even for enterprise customers attracted by advantages that Cloud operation can provide. Current technology available in this area is limited in addressing this difference. In most solutions, configuration performed on an on-prem controller is disconnected from the configuration performed on a Cloud version, rather, audit functions performed at intervals or on demand are used to reconcile the differences or the platforms are synchronized at static intervals. However, real-time functions are lost under currently available solutions and management actions are disconnected.

Thus, there is a need for a method to dynamically synchronize on-prem and Cloud controllers, based on the administrator activity and/or the needs of a given enterprise/organization.

This proposal presents a technique to synchronize data in a hybrid setting between a multiplicity of on-prem and Cloud controllers. In particular, the technique of this proposal involves implementing a "SMart" connector (SM) function that operates between a Cloud controller unit and an on-prem controller unit and operates through various steps, as discussed herein, below.

For a first step, the SM monitors the (near real-time) actions of an administrator (admin) on each unit and synchronizes the other unit accordingly. For example, when an admin logs into the local unit, a warning is automatically and instantly displayed in the Cloud unit. When an admin configures an item on the local unit (e.g., clicks ‘validate’, ’okay’, or the like), the SM immediately pushes the configuration change to the Cloud unit. In one embodiment, the difference appears as a "diff" (e.g., on the target page,

showing a warning about the local configuration and its difference to the Cloud configuration). This part is logical and may seem obvious, however, it is needed to support the following steps.

For the second step, each page of the management platform is associated (in the SM) with a label (e.g., 'monitor/real time', 'monitor/aggregate', 'configuration/non-real time' etc.). When the Cloud admin connects to a page, the SM reads the label. If the label is real-time, the SM queries the on-prem unit to start streaming data matching that page content to the Cloud unit. The Cloud admin can see, in real-time, data matching the requested page, while other real-time data is not sent to the Cloud. Thus, only what is requested is sent to the Cloud unit. Granularity can be high at this step. For example, the admin can look at a switch and click a particular interface such that only the real-time data for that interface is streamed to the cloud. As the Cloud admin clicks away from the page, the SM informs the on-prem unit and the on-prem unit stops streaming matching data.

Similarly, for a third step, when the Cloud admin connects to a page matching an aggregate label, the SM causes the on-prem unit to immediately upload the matching aggregate data. With the second and third steps, the SM causes the on-prem unit to affect different priority labels to the data sent to the Cloud unit, which can not only enable high-priority items (e.g., alarms) to still be sent in priority, but also enable the on-prem unit to send in near-real time the data matching the aggregate page at which the Cloud admin is (currently) looking. As the Cloud admin uses the page, the same aggregate data, refreshed at the same pace, is seen, as it would be seen on the on-prem unit.

As in the second step, other data sent from the on-prem to the Cloud unit can be provided different priorities accordingly (alarms still provided high priority, followed by the real-time aggregate data update, followed by other data on pages at which the Cloud admin is not looking, etc.).

Naturally, the SM causes the on-prem unit to stream, in real time, any raw data that matches the real-time page at which the Cloud admin is looking, and a hysteresis can be set to limit the volume of data if applicable (e.g., causing a 'real-time' location refresh for the 125 stations on the map to occur once per 250 milliseconds (ms) instead of once per 100 ms on the on-prem unit, until the Cloud admin zooms to a zone for which the volume of data is below a maximum acceptable value). Thus, the refresh pace of the Cloud page

can be modified in the time domain (e.g., once every 250 ms vs 100 ms for the on prem) of the data domain (e.g., new data for each change of 5% or more, instead new data for any change), while still providing the same experience of real-time management for the Cloud admin and the on-prem admin.

In a fourth step, the SM can observe the Cloud admin's management habits and can build a usage profile. In its simplest form, this step can include recording the pages that the admin visits and associating them with a timestamp. Over time, the pages that are visited more often can acquire a higher weight, forming a probabilistic tree of pages to be visited at login time.

More advanced operations can use learning techniques, such as Random Forest, in order to build equivalent structures. In both cases, the outcome can involve the SM constructing a profile for each admin user of the Cloud unit, with a most likely time of connection and a probabilistic view of pages most likely to be visited, with their corresponding order. As an additional source of information, the SM can connect to a Role-based Access Control (RBAC) or Identity Access Management (IAM) system in order to obtain/retrieve access and site privileges of an admin, for example, to determine which pages and sites in the cloud management platform to which the admin has access (e.g., read-only, read-write, no-access) with probabilistic next pages of access in a workflow. Thus, alarms and alerts originating from a certain site may not be sent in priority to the Cloud unit if when no recorded admin of that unit have access privilege to the matching elements.

Using this information, the SM can validate the difference between the pages on the Cloud and the on-prem units in a fifth step, thereby causing the on-prem unit to upload its data at least before the time when the Cloud admin is likely to access the page. This step ensures that the Cloud admin sees the configuration diffs (the same process is operated for the on-prem admin to also see the diffs) as they connect to a matching page, and that the monitor pages are populated with 1) the aggregates and 2) the real time data that the admin would expect from a real time system.

Accordingly, this proposal provides various techniques through which data displayed via a Cloud network management unit can be synchronized with data of an on-premise network management unit. By measuring which page may be accessed and the

type of data that is to be displayed for the page, the techniques herein provide for the ability to stream only the amount of data needed for a Cloud administrator to experience real time management, without consuming a large bandwidth with raw data streaming.