February 2020

FABRIC AS STORAGE TO AVOID DOWNLOAD FAILURE ON MULTIPLE DEVICES AT BRANCH/DATA CENTERS

Balaji Sundararajan

Ankush Verma

Preety Mordani

Igor Manassypov

Rohan Karamandi

See next page for additional authors

Inventor(s)
Balaji Sundararajan, Ankush Verma, Preety Mordani, Igor Manassypov, Rohan Karamandi, Anand Oswal, Mouli Balasubramanian, and Sireesha Todeti

FABRIC AS STORAGE TO AVOID DOWNLOAD FAILURE ON MULTIPLE DEVICES AT BRANCH/DATA CENTERS

AUTHORS:

Balaji Sundararajan
Ankush Verma
Preety Mordani
Igor Manassypov
Rohan Karamandi
Anand Oswal
Mouli Balasubramanian
Sireesha Todeti

ABSTRACT

Presented herein are techniques to transmit virtual machine (VM) images from a controller to a switch fabric, based on capacity, and to distribute the VM images to a virtual network infrastructure, as needed.

DETAILED DESCRIPTION

Customers commonly experience firmware and virtual machine (VM) download failures due to connectivity issues at, for remote locations (e.g., branches or colocation sites). For example, intermittent connectively problems could interrupt the download. In addition, downloads of images to different devices consume resources separately, which is very inefficient.

Proposed herein are two innovations to address these issues. In regards to the first innovation, while working with customers, we have identified upon numerous issues related to software delivery and, accordingly, have used these experiences to derive a novel delivery mechanism besides regional hub centers for software and virtual machine distribution for branches. In particular, we have focused in on the use of Hypervisor software upgrade images and VNF images.

For Cloud on-ramp for Colo with SD-WAN, there is a need to download many VM images and it is likely that a connectivity issue may occur. The decision to download a image to a server is made based on the difference between the orchestrator and device directly. This path might be slow and does not guarantee connectivity at the time of

downloading. The second innovation proposed herein is to use fabric storage when the connection is idle and use that unused bandwidth on the link to transfer/resume and, in case of link interruptions, download whenever there is connectivity. A Hypervisor server can always obtain these images from switches (i.e., on demand). The need for only one connection may provide a number of advantages, incliding:

1. High Speed of LAN over WAN based software downloads
2. Avoid download failures due to connectivity. We can always resume connection to switch based on caching of partial download can be done on switch.
3. Use unused space on switch rather on orchestrator.

In one example, as shown below in Figure 1, it is assumed that customer 1 has 1400 branches and customer 2 has 2500 sites, creating a massive software distribution management problem. In accordance with the techniques presented herein, these customers would designate a few regional hubs that act as image repositories that would download images automatically. Sites within that region would subsequently auto-download. In addition, extensions may include a peer-to-peer software distribution sharing mechanism for hubs. As such, with the fewest possible controller downloads, it is possible to efficiently manage image delivery to hypervisors or virtual machines that run in branches.

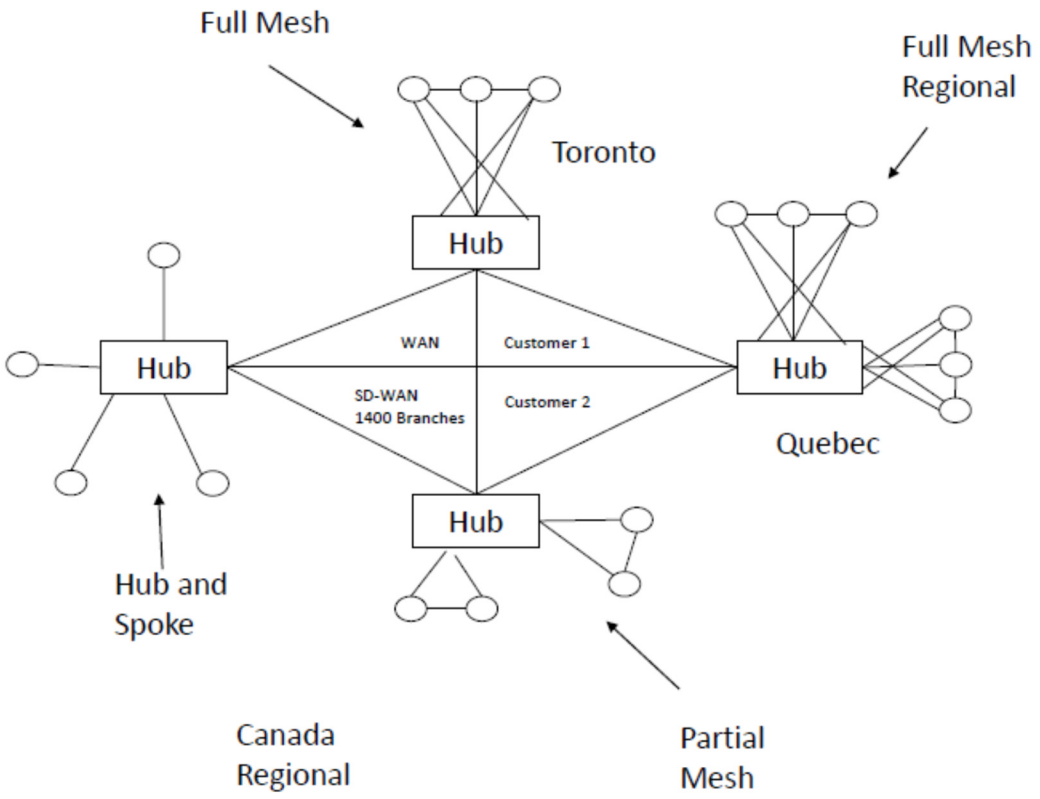

Figure 1

In summary, presented herein are techniques that enable customers to upload Hypervisor images, VM images, etc. into a central repository (e.g., via API or via GUI). The techniques presented then download the Hypervisor Images and VM images into the switch infrastructure. The goal is to download the images into Switch Containers and anytime an upgrade to either the Hypervisors or the VNF is needed, the system can reference the image located in the switch for downloads. Figure 2, below, illustrates a simplified packet walkthrough in accordance with the techniques presented herein, while Figure 3, below, illustrates an VNF packet walkthrough in accordance with the techniques presented herein.

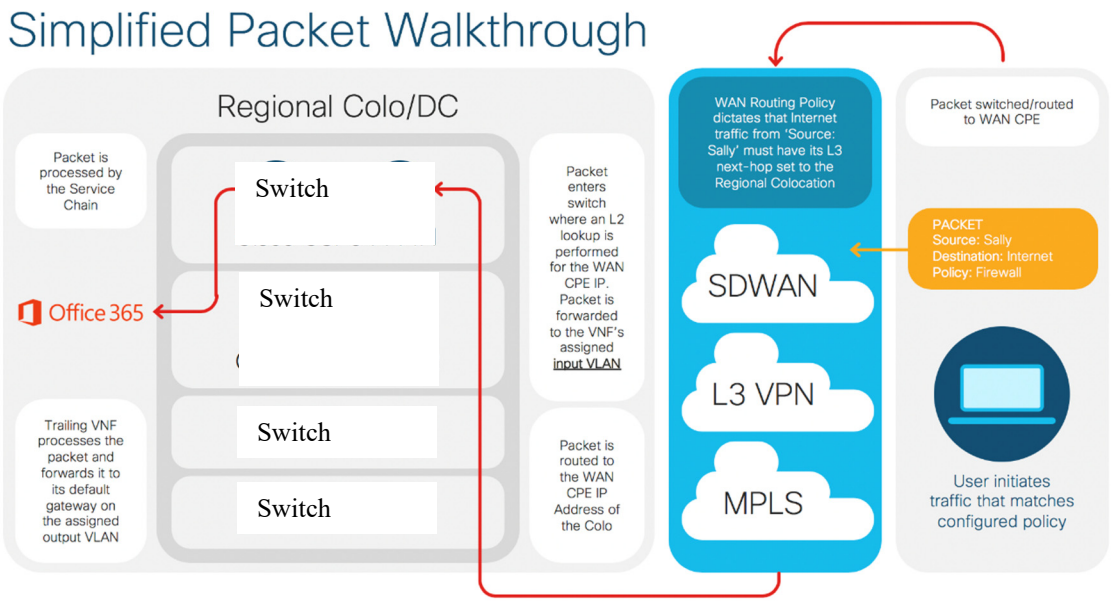

Figure 2

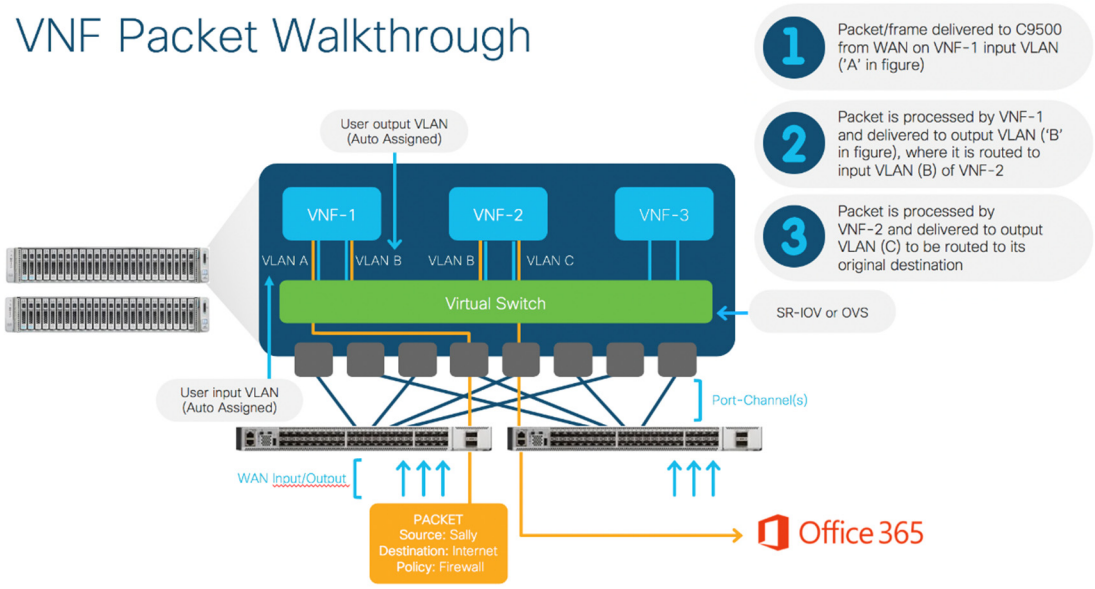

Figure 3