March 2022

METHOD TO EFFICIENTLY UPGRADE MESH ROUTERS IN SOFTWARE-DEFINED WIRELESS MESH NETWORK

NIRANJAN M M

METHOD TO EFFICIENTLY UPGRADE MESH ROUTERS IN SOFTWARE-DEFINED WIRELESS MESH NETWORK

AUTHOR:
NIRANJAN M M

ABSTRACT

Software-Defined Wireless Mesh Network (SDWMN) deployments are considered as a viable option to provide wireless coverage for a vast area, such as community-wide or city-wide. SDWMN is a combination of Software-Defined Networking (SDN) and Wireless Mesh Network (WMN). In other words, it consists of Centralised SDN Controller and Wireless Mesh Network. In SDWMN deployments, SDN Controller manages many (100s and 1000s) of the Mesh Routers (MRs). Upgrading the whole deployment is a challenge and time consuming. There are techniques which provide some sort of optimisation and hence to reduce service downtime. But none of the techniques provide efficient mechanism to upgrade the whole deployment in considerably less time. The techniques presented herein propose method to select group of slave MRs which are already upgraded as Master MRs to upgrade the remaining slave MRs. This method ensures that the load of pre-downloading the image to all the MRs get distributed and shared by all the MRs in the system rather than concentrating the load on the SDN Controller or on any MR thus eliminating the chance of having a single point of failure that can hamper the pre-download process.

DETAILED DESCRIPTION

Software-Defined Wireless Mesh Network (SDWMN) deployments are considered as a viable option to provide wireless coverage for a vast area, such as community-wide or city-wide.

SDWMN is a combination of Software-Defined Networking (SDN) and Wireless Mesh Network (WMN). In other words, it consists of Centralised SDN Controller and Wireless Mesh Network. Wireless Mesh Network (WMN) is a multi-hop radio network whose nodes are IP routers with one or more wireless interfaces, typically based on IEEE 802.11 Wi-Fi technologies. Backbone of a WMN is made up of dedicated wireless nodes called Mesh Routers (MRs). These MRs can be freely organised into any network topology and communicate with each other using protocols such as DSR (Dynamic Source Routing), Optimised Link State Routing (OLSR) etc., to find optimised path and hence to achieve higher performance.

In SDWMN deployments, SDN Controller manages many (100s and 1000s) of the Mesh Routers (MRs). Upgrading the whole deployment is a challenge and time consuming. Upon downloading the image with version on the SDN Controller, MRs are pre-downloaded with the corresponding images from the SDN Controller. It is slow process and time consuming. Its time complexity is "O(n)" where "n" is the number of MRs in the deployments and each MR download image from the SDN Controller directly.

There are techniques which provide some sort of optimisation and hence to reduce service downtime, such as, by upgrading MRs in staged manner based on the Site (Location). But this restricts customer deployments to have bigger sites (i.e., number of MRs per site) thereby forcing customers to split their network deployment into multiple smaller sites. But none of the techniques provide efficient mechanism to upgrade the whole deployment in considerably less time.

The techniques presented herein propose method to select group of slave MRs which are already upgraded as Master MRs to upgrade the remaining slave MRs. This method does not depend on whether the image is downloaded from the SDN Controller or directly from the Cloud Servers. This method is generic enough that it can be incorporated to any of the image upgrade scenarios for the devices in larger deployments. The method proposed is an efficient mechanism to cascade the downloaded image of an MR to its peers within the same Site (Location). The mechanism leverages the existing SDN Controller to MR interaction to orchestrate the downloads amongst the MRs within a Site.

To realise this, let us use a "source queue", an "in-service" tree and the "master" tree. Maintain these for each Site and for each Model.

- The "source queue" contains all the MRs waiting for the download.
- The "in-service" tree contains all the MRs downloading at any point in time.
- The "master" tree contains all the MRs that have already completed the download.

This method begins by adding all the MRs awaiting download into the "source queue". The first "Y" MRs (let us say, 3, to have redundancy, in case one or two MRs fails to download) at the top of the queue downloads the image from the Cloud and once the download completes, they are turned into master MRs and moved into the "master" tree.

Now the method selects "X" MRs at the top of the "source queue" (assuming one master MR can service "X" slave MRs at any point in time) and converts them into slave MRs and move them into the "in-service" tree. These slave MRs now pre-download the image from the selected master MR. When a slave MR completes its pre-download, it generates an event to let the SDN Controller know about the same. On receiving this event, the SDN Controller converts any slave MRs that completed the image download into a master MR (thus making it a potential source from where other MRs can download) and moves the MR into the "master" tree.

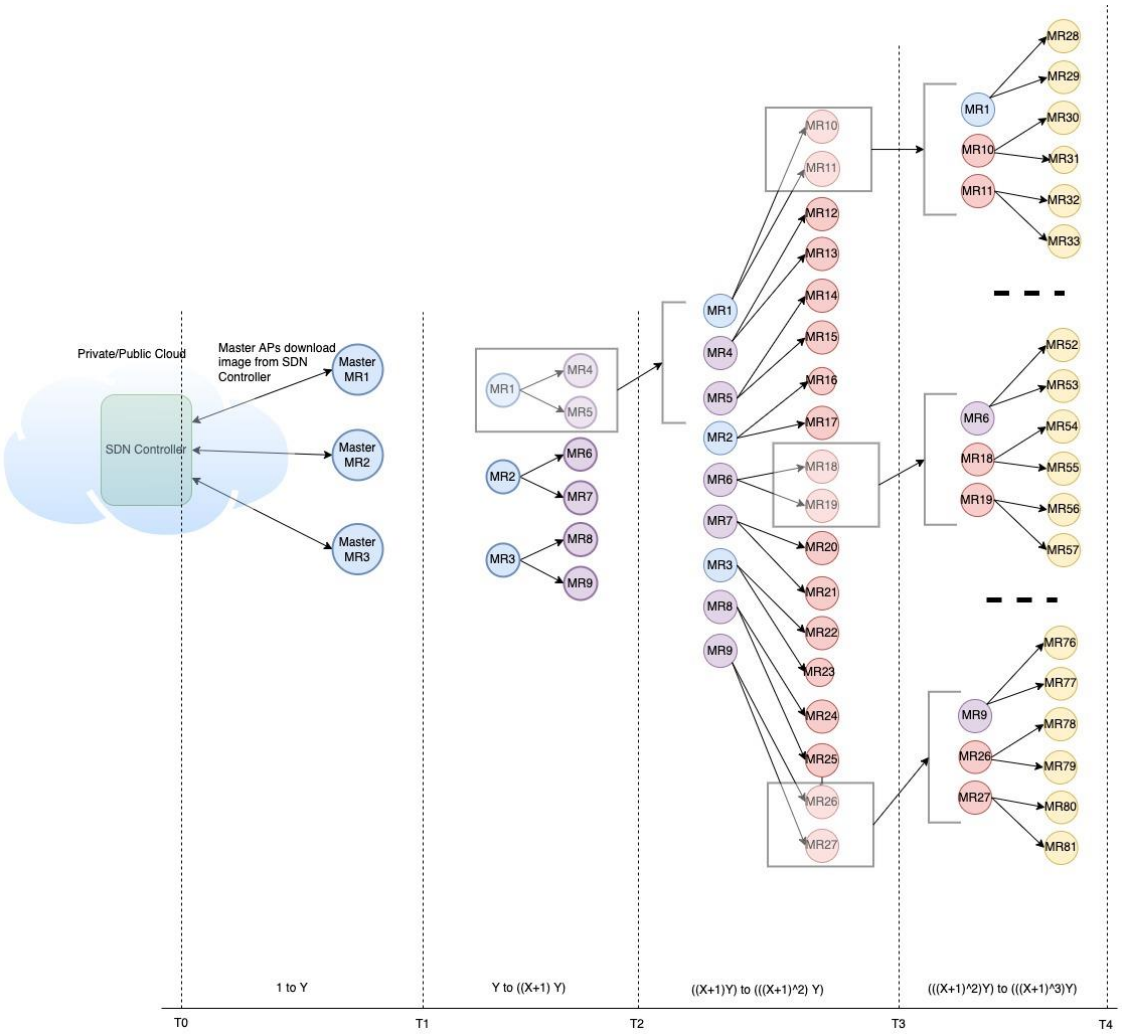
This opens "Y+1" slots from where MRs can pre-download the image. "Y" slots are from the initial download from the Cloud, "1" slot is from the MR that got converted from slave to master.

Thus overall "Y*X" MRs at the top of the "source queue" are converted to slave MRs and get moved to the "in-service" tree and begin pre-downloading from their corresponding master MRs. This process goes on until all the MRs in the source queue completes the pre-download. If any MR fails to download the image in the first attempt, it is moved to the end of the source queue. This ensures that it gets to re-attempt the download, but at the same time makes sure that a faulty MR does not block a slot by continuously retrying and blocking other MRs in the queue. This ensures that this method will not go into a deadlock mode if a group of MRs fail to download due to some reason.

On the failed attempt, MR notifies SDN Controller the reason for failure. After "Z" (retry=3) number of failed attempts, the SDN Controller withdraws the MR from the "source queue" and attempts to directly download the image from the Cloud. This method can be further enhanced by factoring in the latency amongst the MRs in making the choice of the master MR.

Thus, instead of having a single source for downloading the image, the proposed method turns all the MRs that already completed the download into a potential source from where other MRs can download the image. This reduces the time complexity of the pre-download to the order of "O(log(N) base (X+1))" where "N" refers to the total number of MRs in the Site and "X" refers to the number of parallel downloads possible, whereas the existing techniques takes linear time "O(N)" to complete the pre-download to all the MRs in the Site. Proposed method relies on events generated by the MRs to guide the overall process into taking the next action so that the SDN Controller is spared from having closely monitor and manage the same. This method achieves an improvement in the time complexity while not loading the performance of the SDN Controller.

The method ensures that the load of pre-downloading the image to all the MRs get distributed and shared by all the MRs in the system rather than concentrating the load on the SDN Controller or on any MR thus eliminating the chance of having a single point of failure that can hamper the pre-download process.
To optimise further, the Master MRs are selected for upgrading Slave MRs based on the RRM neighbour information which is deducible using NDP (Neighbour Discovery Protocol) packets. That is, Slave MRs can communicate the IP address of its neighbours detected over NDP (Neighbour Discovery Protocol) to the SDN Controller, so that the SDN Controller uses this information to select one of these neighbours as a Master MR.
Example: A site with 6000 MRs, 2000 MRs of each model. (Assuming one download takes 1 minute).
Let us consider,
The number of MRs who get the image from the Cloud = Y = 3 (which is initial download factor as explained below with diagram).
The number of MRs who get the image from the Master MR = X = 2 (which is the distribution factor as explained below with diagram).
At time t6, the number of MRs upgraded = ((X+1) ^6 * Y = (2+1) ^6 * 3 = 729 * 3 = 2187.
i.e., With one download taking 1 minute, the proposed method takes "~6 Minutes + Time taken to download from Cloud to MR" to upgrade the total of 2187 MRs. Here we are considering X=2, which is very less, in real deployments, it can take higher value.



Assume "Y" to be initial download factor: The number of MRs who get the image from the Cloud.

Assume "X" to be the distribution factor: The number of MRs who get the image from the Master MR.

At time t0: The system has image at the Cloud, MRs begin downloading the image.
At time t1: The system has downloaded the image to "Y" different MRs.
At time t2: Y -> X * Y + Y = (X + 1) * Y
At time t3: (X+1) * Y -> X * (X+1) * Y + (X+1) * Y = (X+1) * (X+1) * Y = (X+1) ^2 * Y
At time t4: (X+1) ^2 * Y -> X * (X+1) ^2 * Y + (X+1) ^2 * Y = (X+1) ^3 * Y
…..
Since exponentiation happens by a factor of (X+1), the time required for distribution is O(log(X+1) N + T(Y).

The techniques presented herein propose method to efficiently upgrades the SDWMN with-in considerably less time irrespective of the number of MRs in the deployment. This method considers redundancy which is in-built as per design. Moreover, this method does not depend on whether the image is downloaded from the SDN Controller or directly from the Cloud Servers. Additionally, this method is generic enough that it can be incorporated to any of the image upgrade scenarios for the devices in larger deployments.