October 2020

MAINTENANCE AND POWER SAVINGS IN LARGE MULTIPLANE DATA CENTER FABRICS

Ramakrishnan Chokkanathapuram Sundaram

Pascal Thubert

MAINTENANCE AND POWER SAVINGS IN LARGE MULTIPLANE DATA CENTER FABRICS

AUTHORS:
Ramakrishnan Chokkanathapuram Sundaram
Pascal Thubert

ABSTRACT

Within a Data Center (DC) environment network upgrades are often challenging and may consume significant amounts of time and network administrator resources. Additionally, DC networks tend to consume large amounts of energy and dissipate considerable amounts of heat that can be challenging to evacuate in densely populated fabrics. To address these challenges techniques are presented herein that support, possibly among other things, the construction of a network model; the use of Machine Learning (ML) to predict low and high load periods and, in low periods, determining the ratio of resources that may be taken offline; updating the equal-cost multi-path (ECMP) rules in the leaves to avoid selected planes so as to take the full plane spine and super-spine nodes offline; and upgrading one of the super-spine nodes and then one of the spine nodes to ensure that there is always a rollback path in case of a problem. If the upgrading is successful, the techniques may include proceeding to upgrade all of the super-spine nodes and all of the spine nodes.

DETAILED DESCRIPTION

Network upgrades are often painful and consume a significant number of cycles. Among other things, network administrators need to plan for the maintenance windows and often opt for a period of downtime. A facility such as In-Service Software Upgrade (ISSU) provides a mechanism to upgrade a network device with zero data plane down time. While ISSU can change the software version on a device it cannot change the configuration of the device. There are methods like fast reloads that help in changing the configuration of a device – e.g., after reloading the device can come up with a new configuration and the device will be up after a fixed amount of down time. Both ISSU and fast reloads assume certain dimensions regarding the scale of an environment in connection with what can be brought up and running within a certain amount of time.

While ISSU is mostly nondisruptive under certain constraints, since there is some control plane down time, the learning of new flows during this window is not guaranteed. Also, based on different router form factors the ISSU times for a top of rack (ToR) device and an end of row (EoR) device for platform upgrade times can vary. In service upgrades do not change configurations, and configuration changes can have additional down time – e.g., like fast reload where a router can be booted with a new configuration. The down times can depend on the type of configuration that is being pushed.

Additionally, applying a new image to a router typically reboots the routing process. As well, setting a link offline may cause packet drops until the routing has recovered in the network. Aspects of the techniques presented herein load balance around a device (e.g., a router) before it is made offline.

Aspects of the techniques presented herein employ a ML and Artificial Intelligence (AI) approach with software-defined networking (SDN) technology to understand the traffic patterns, the utilization of network components, feature dependency, etc. at different times to predict low utilization time slots for the components (e.g., nodes) in support of software upgrades for various routing planes in a DC fabric. Further aspects of the techniques presented herein employ planned traffic rerouting to approach optimal scenarios to achieve a smooth upgrade with lower or no down time.

DC networks tend to consume large amounts of energy. In turn, DC systems (such as, for example, NX-OS-based systems) operating at high speed dissipate considerable amounts of heat that can be challenging to evacuate in densely populated fabrics. These characteristics arise even when the network experiences lower utilization – e.g., depending upon the time of the day or the day of the week, with special days like Christmas and Black Friday, etc.

Green operations consist of shutting down some routers and routing around the failures. But a basic 'power off' of a router leads to a service disruption until the network converges. Therefore more subtle mechanisms are required. For instance, injecting an overload bit in an Intermediate System to Intermediate System (ISIS) protocol or establishing high costs on the router's links to indicate that the router is not willing to route before it is effectively shut down (e.g., a make before break approach).

Aspects of the techniques presented herein support detecting a window of opportunity where the network is less loaded, within that window reducing the load of a router to nominally zero (0), and taking the router offline either for upgrading or just to save energy.

An additional challenge with the upgrade activity that was described above is a coexistence problem. That is, the new image or the new configuration of an updated router may behave differently from the old version and even though it interacts well with upgraded routers, it may not do so in a brownfield environment comprising legacy routers. Upgrades therefore create an ordering problem where upgrades must be completed in a certain order and enable rollback.

The challenges that were described above are normally approached as a network problem. In a traditional routing world one may have, for example, a main path (e.g., Sender Policy Framework (SPF)), Traffic Engineering (TE) paths (e.g., SDN), and a Free Range Routing (FRR) path (e.g., Topology-Independent Loop-Free Alternate (TI-LFA)). Either of the above described methods may be used to route around a router to place it offline. However, because of the coexistence problem this is not sufficient, as there is still a problem of scheduling and rolling back which requires special attention from an operator to avoid stalled nodes in the network.

DCs differ from a classical interior gateway protocol (IGP) environment. Accordingly, aspects of the techniques presented herein leverage that difference to support both new upgrade mechanisms and green operations.

On one hand, a DC is a world of huge ECMP network routing strategies, where all of the leaf-to-leaf paths are basically equivalent and usable. Thus in a DC world the problem is not necessarily addressable as a routing change, but rather as a forwarding change whereby the ECMP operation is altered to avoid some routers.

On the other hand, large DC fabrics are organized in planes. As one example, consider for instance Facebook's design where each plane may be depicted as a different color, as illustrated in Figure 1, below.

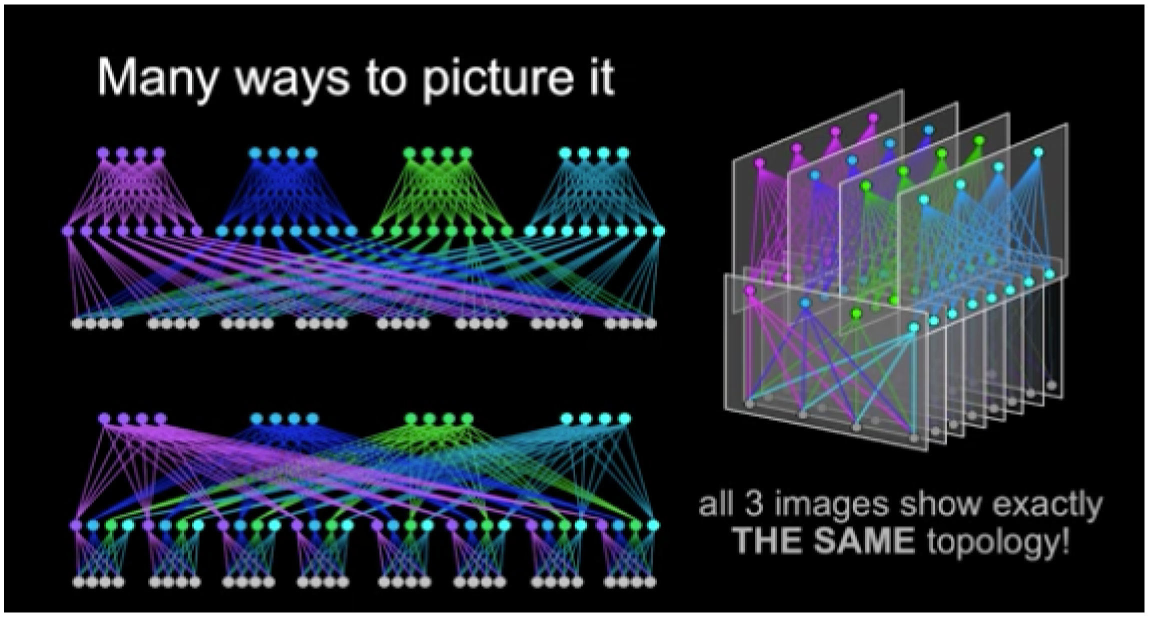

Figure 1: Exemplary Facebook Design

From all of the above several fundamental observations include, possibly among other things, that:

1. Traffic is always leaf-in/leaf-out, the leaves being the nodes at the bottom of the schemas.
2. The ingress leaf selects the plane.
3. All nodes above the leaf level belong to a given plane (i.e., they have one single color whereas the leaves are white, combining all of the colors).
4. Routing happens within a plane all the way from ingress to egress.

From these observations, aspects of the techniques presented herein leverage the partition of the cloud network into planes to selectively put to sleep and then upgrade a full plane.

Elements of particular interest and note within the techniques that are presented herein are discussed below.

A first element of aspects of the techniques presented herein supports the construction of a network model to predict flow patterns in a DC fabric. One activity that may take place during the construction of a network model is topology discovery.

The network model will learn the link connectivity, capacity, etc. of the routers and will build a topological view. The topology will also learn the various routing planes. The samples are chosen such that they reflect peaks of the various dimensions used. Data samples of link usages at different times, node usages, CPU consumption, and flow patterns may be sampled and learnt using telemetry data and are classified per routing planes in the fabric.

The model is also able to predict the flow occurrences and flow patterns based on learning methods. The model will also predict the set of flows that are more likely to be impacted on a per routing plane down. The flow patterns of long flows are learnt by transporting the Elephant flow trap tables in telemetry. The model will also look at port queue utilization and congestion, the routing patterns, routing tables, etc. and track low utilization times for each node and calculate the effect of node removal or a routing change.

A table may be constructed for each plane based on timestamps and a list of routers with various parameters such as, for example, congestion, bandwidth, multicast states, links, etc. as different dimensions. Minimized windows on a router or load at various times may be matched to that needed for the upgrade. Additionally, high priority flows that may be present on the router during the window may be predicted.

Another activity that may take place during the construction of a network model comprises the development of a range of measurements. Such measurements may include, for example:

1. Plane and node utilization. A value that combines various relations such as, for example, CPU usage, route states, unicast flows, multicast floes, congestion on the links, and link utilization in a per plane way. In addition to arriving at a specific number, also determining how each of these various dimensions contribute to the node utilization or plane load. For a node utilization value one may identify how each factor will contribute to that and how the nodes contribute to the utilization of a plane. As well, the load or utilization of the routing plane may be calculated.
2. Global utilization. A value that indicates overall network utilization or load. As well, calculating how each routing plane influences the overall global load. A time series for the global utilization may be used to predict when the next minimum will occur for global utilization. For a global minimum one can predict how the different planes contribute and which planes have minimums.

One can also predict per-node minimums and how they will impact a routing plane.

3. A prediction of how each plane contributes to the network load.
4. A prediction of how each factor contributes to the node utilization (such as, for example, multicast, unicast, congestion, etc.). Based on each, policies may be proposed to switch planes.
5. Application of policies to temporarily move selective flows to a different plane.

The model may also learn the software version that is running on various nodes, the new software version to which it needs to be upgraded, any known problems, the hardware models of devices like a ToR or an EoR and the approximate reload time, kernel load time, link up times based on different types of port speeds, small form-factor pluggable (SFP) ports, etc. The model may learn about the upgrade constraints of different routing protocols, such as Open Shortest Path First (OSPF), Border Gateway Protocol (BGP), multicast, etc. A leaning engine knows the best outcome scale scenarios of each routing protocol, and the effects of higher scales affecting protocol learning behavior of routes, after, for example, a network node restart. The learning engine predicts when such scenarios may exist in the network based on a combination of predictive rerouting to another plane and utilization factors and flow patterns as mentioned previously and calculates the best times to perform a nondisruptive upgrades. The identified time slots may then be used to schedule a nondisruptive upgrade.

The model may also learn both multicast and unicast flow patterns. Based on multicast/unicast usages and multicast trees, the model can predict the upgrade of nodes that are not part of multicast trees and selectively move unicast/multicast flows to another plane.

A second element of aspects of the techniques presented herein encompasses predicting low utilization times for the planes. For example, the learning engine based on the model will predict time slots (e.g., 10:00PM to 11:00PM) as a best possible upgrade time for a plane Green from software version X to software version Y based on the plane utilization, load, etc. The engine may be built with decision trees. The various constraints are identified to get the node into the best outcome state and polices are installed in the network based on the predictions from the learning engine.

Predicting low utilization time slots, as described above, may leverage, possibly among other things, information about upcoming events or streams (such as a sporting event, etc.), techniques such as time series, etc.

In connection with predicting low utilization time slots, as described above, when balancing around policy is installed the result may be observed before the router is really made offline and rebooted. If the balancing around policy causes a situation that is not the one expected (e.g., in terms of latency, congestion, and/or packet drops) the upgrade may be delayed and the normal policy behavior can be immediately restored (rolled back) to limit any disruption.

In accordance with aspects of the techniques presented herein digital twin technology may be used to predict the impact of adding or removing nodes, inserting links, etc. This can be used to estimate parameters if they are within limits for proceeding with an upgrade. A simulation may be studied and analyzed before really proceeding with applying the policy to isolate a plane and apply upgrades.

A third element of aspects of the techniques presented herein supports selectively rerouting around a node/switch to alternate planes to meet the constraints for a smooth upgrade. For example, the network policies are injected into the nodes in the plane to reroute the flows to alternate planes. The alternate planes are chosen in such a way that the plane is capable of handling the extra traffic. The policies may be placed in such a way that flows are distributed along multiple planes if that is applicable. This may be done by injecting or modifying the ECMP hash on the leaves.

Note that the policy injection is normally nondisruptive. The policies change the ECMP rules, and there should not be a saturation as a result. As noted previously, the best time for performing the activity is predicted.

The hash rules are designed in a way that the planes to be avoided are not reached in the load balancing operation. Each leaf can be updated separately, and there is no need to synchronize precisely when that happens. As the new rules are installed, the plane to be avoided is used less and less. A split second after the ECMP is activated in the last leaf, all traffic ceases in the plane but the traffic to the nodes in the planes themselves (loopback) to any node (spine/super-spine) in any plane is still reachable by any leaf, but the end-to-end traffic no longer traverses the avoided planes.

A fourth element of aspects of the techniques presented herein supports upgrading nodes in planes alternatively. For example, the planes to be upgraded may be chosen in a fashion so as to have minimal impact. The DC fabrics can be considered to have different routing planes and planes may be chosen alternatively among different planes.

Aspects of the techniques presented herein support upgrading a super-spine node first. That super-spine is still reachable via any spine node its plane. When it is back up, it is reachable, at least as a host, to check that everything is acceptable. Upgrades may then proceed on a spine node, with checks that the new super-spine node and new spine node operate as expected. The upgraded spine node is reachable as a host by any of the leaves in its Pod, while the super-spine node is reachable as a host via any leaf in any other Pod.

If the first pair interacts as expected, then all super-spines and then all spines may be upgraded. Additional items of interest and note in connection with the techniques presented herein include, for example:

1. Bringing back traffic after an upgrade. After an upgrade is completed, for example, policies may be removed to bring back traffic to the upgraded plane.
2. Green methods to reduce speed or bring down ports based on utilization. When low network load times are predicted for a routing plane certain ports or devices can be placed in a low power mode and later woken up to efficiently manage power utilization.
3. Dual-homed considerations for leaves. Times may be identified when the hosts attached to the leaf are globally under loaded in both single-homed and dual-homed cases. In dual-homed cases the leaves are connected to different routing planes. In dual-homed cases when one leaf is reloaded the other leaf can have significant load, so it is better to find low load times for the planes the leaves are connected to. In the case of dual-homed hosts, an upgrade can be considered in an alternating way for the leaves. Under an ISSU environment the forwarding information base (FIB) states are preserved on a router and thus the forwarding can continue in single-homed cases as well.
4. Upgrade workflows may change in various implementations. For example, a customer may request that a network upgrade be scheduled and may issue the request through a Data Center Network Manager (DCNM). The DCNM then

presents an impact analysis and displays windows indicating the best upgrade times for the routers, and predicts an upgrade pattern. The DCNM automatically completes the upgrade by following the method that was described above. The DCNM verifies that all connectivity is acceptable and finally makes the upgraded plane active.

In summary, techniques have been presented that support, possibly among other things:

1) Constructing a network model.
2) Leveraging ML to predict low and high load periods and, in low periods, determining the ratio of resources that may be taken offline.
3) Updating the ECMP rules in the leaves to avoid selected planes so as to take the full plane spine and super-spine nodes offline.
4) Upgrading one of the super-spine nodes and then one of the spine nodes to ensure that there is always a rollback path in case of a problem. If the upgrading is successful, then proceeding to upgrade all of the super-spine nodes and all of the spine nodes.