BGP Control Plane Overhead in Fat-Trees: An Analytical Approach

Leonardo Alberro
Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
lalberro@fing.edu.uy

Eduardo Grampin
Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
grampin@fing.edu.uy

***Abstract*—State-of-the-art Data Center Networks (DCNs) adopt fat-tree topologies based on the original Clos networks due to their non-blocking and constant bisection bandwidth characteristics. Massive Scale Data Center networks comprise thousands of switches, and demand deploying a routing protocol to build Equal Cost Multi-Path connectivity among end nodes. Routing protocol overhead is one of the foremost important parameters to consider in DCN design. Existing approaches to measuring this overhead are based on large emulations or simulations that capture the packets exchanged under certain scenarios and then process them to obtain this measurement. This requires a lot of development effort and may not always be possible at large scales. In this work, using an analytical approach, we analyze the control plane overhead for BGP in the data center. Our proposal manages to measure the overhead injected by BGP in failure and recovery scenarios without the need to develop and deploy large emulations or simulations. The main idea is to use an analytical methodology to obtain mathematical formulas from the expected behavior of BGP. The results show that it is possible to adequately model the load injected by BGP under a set of failure and recovery scenarios. The validation was performed in a simulated environment over the ns-3 simulator.**

***Index Terms*—BGP, fat-tree, control plane overhead**

I. Introduction

Cloud Computing is heavily based on Massive Scale Data Centers (MSDC), which account for hundreds of thousands of physical servers, running myriads of applications over virtualized infrastructure. Such massive amounts of computing instances need a large-scale connectivity infrastructure, often referred to as Data Center Networks (DCNs), which have evolved from typical three-tier hierarchies towards fat-tree-based connectivity fabrics comprising thousands of communication nodes.

Due to the scale of such infrastructures, modern DCNs are Layer 3 multi-path connectivity fabrics, where the choice of routing protocols is a central design problem. There are two families of approaches to managing routing in these infrastructures: centralized, using some variant of Software Defined Networking (SDN), or distributed, using routing protocols. Managing network routing at large scales in MSDC networks is a significant challenge. For this reason, the industry has opted for distributed protocols, adapted to minimize convergence times in case of failures. In this sense, Border Gateway Protocol (BGP) [1] with specific data center extensions has proven to provide scalability and reliability, together with operational efficiency in real-world working environments [2], exhibiting a better performance than alternative proposals such as Link-State IS-IS with flooding reduction [3], and Routing in Fat Trees (RIFT) [4], as measured in our previous work over an emulated environment [5] for large-scale topologies.

Some desirable characteristics of routing protocols are fast convergence and low control plane traffic overhead. The design guides for BGP in the data center, favor fast convergence, but the question concerning protocol "chattiness" remains. Control plane analysis is usually accomplished by formal methods or experimentation using large emulations or simulations. Those extensive emulations or simulations capture packet exchanges in specific scenarios and subsequently analyze them to obtain the desired measurement. However, that approach demands substantial development efforts and may not always be feasible for large-scale deployments. In this work, we use an analytical approach to study the BGP control plane overhead, and we compare it with experiments carried out over the ns-3 simulation environment [6]. This analytical approach was introduced in [7], seeking to validate the software port of the Free Range Routing (FRR) protocol suite to the ns-3 network simulator. A comprehensive analysis of network failures in data centers is presented in [8]. This inspired us to consider that the most stressful events for routing protocols in operational networks are the failure and recovery of network elements.

Considering BGP, its reaction to events is the emission of Update messages (either withdrawals or advertisements of routing prefixes); therefore, counting Updates is needed to measure protocol overhead. In this regard, the main contributions of the present work are i) a methodology for counting Updates and ii) the analysis of protocol overhead under a complete set of use cases, which comprises Leaf, Spine, and Core failure and recovery, validated in realistic scenarios up to 1125 nodes over the ns-3 simulator.

The rest of the article is structured as follows: Section II presents the main aspects of routing in large-scale data centers. Then, in Section III, we present the analysis and results of BGP's behavior under a set of use cases. These results are validated and discussed in Section IV. Finally, in Section V, both the main conclusions and future work are presented.

II. Routing in Large-Scale Data Centers

State-of-the-art DCNs are fat-tree fabrics running a routing protocol that builds multi-path connectivity support for application traffic in the data center. This section will briefly analyze fat-tree characteristics and the BGP routing protocol in this context.

A. Fat-tree topology

A fat-tree data center topology is a special case of a Clos non-blocking network [9], with constant bisection bandwidth between every pair of leaves (data center servers). The fat-tree topology idea was originally proposed for supercomputing [10] and was adapted for data center networks [11]. A fat-tree topology can be characterized in terms of the number of Points of Delivery (PoD)s [12]; a PoD is a section of the topology where the leaves are connected. The topology consists of k PoDs with two layers of switches, Leaf and Spine, densely connected to a third layer, the Core switches (also known as Top of Fabric - ToF switches). Each Core switch is connected to every one of the k PoDs. Therefore, in a k PoD fat-tree topology, there are k switches (each with k ports) in each PoD arranged in two layers of $k/2$ switches. The lower layer is for Leaf switches and the upper layer is for Spine switches. Each Leaf is connected to $k/2$ Spines (and vice versa). Notice that there are $k/2$ Leaf switches in each of the k PoDs, adding a total of $k^2/2$ Leaf switches in the topology. This same reasoning can be applied to the Spine switches, obtaining the same result. Finally, there are $(k/2)^2$ Core switches. Adding the numbers of each level of the topology, it can be seen that the total number of switches is $k^2 + (k/2)^2 = 5/4k^2$. The summary of the topological information is shown in Table I. A fat-tree topology with $k = 4$ is shown in Figure 1.

TABLE I
Fat-Tree topology summary [12]

Number of PoDs $= k$	
Core Switches	$(k/2)^2$
Spine Switches	$k^2/2$
Leaf Switches	$k^2/2$
Total Switches	$5k^2/4$
Number of Links	$k^3/2$

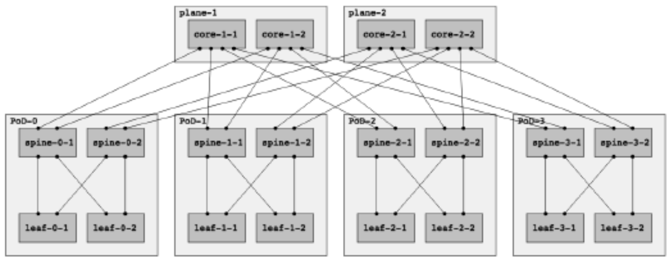


Fig. 1. Fat-tree topology with $k = 4$.

B. BGP in the data center

BGP is the routing protocol of choice for large-scale data centers, among other reasons, for scalability, controlled control plane flooding, and, most importantly, stable and mature implementations. This specific use of BGP is specified in [13], where the operational experience in designing and operating large-scale data centers using BGP as the only routing protocol is reported, together with specific extensions for this particular domain.

For example, in inter-domain routing, the emphasis is often placed on stability rather than immediate notification of changes. Therefore, BGP speakers typically hold off sending notifications about changes for a while; take, for instance, the enormous amount of research on route dampening and BGP timer settings, particularly the Minimum Route Advertisement Interval (MRAI). In contrast, in large data center networks, operators want routing updates to be sent as fast as possible. Regarding best path selection, whereas in the inter-domain routing, BGP speakers build a single best path when a prefix is learned from many different Autonomous Systems (ASes), in the data center networks, multiple path selection is needed to take advantage of fat-tree bandwidth aggregation.

Thus, the specific configuration of BGP for the data center covers the following aspects:

- External BGP (eBGP) is preferred over Internal BGP (iBGP). Although in a data center, the entire network management is under a single administrative domain (and the use of iBGP seems natural), eBGP is preferred since it is simpler to understand, deploy, and configure, especially for multi-path support.
- Only private ASN should be used. The use of public ASNs could, for example, cause ambiguity in tools that attempt to decode the ASNs into meaningful names.
- The 4-byte ASN address space [14] is preferred. In large-scale data centers, it is necessary to have support for more than 1024 ASes (provided by the original 2-byte ASN).
- A practical guideline for ASN assignment should be followed. While it may seem logical to assign a unique ASN to each router, this approach can introduce the *path-hunting* problem. This problem is a variation of the *count-to-infinity* issue encountered in distance vector protocols [15]. In the *path-hunting* problem, routers continuously exchange routing information and update their path costs based on the received information. This process can create a loop where routers keep searching for better paths, resulting in excessive network traffic and instability. To avoid this problem, the practical guideline for ASN assignment in a fat-tree topology is the following: (1) Each Leaf node is assigned a distinct ASN; (2) Spines in the same PoD have the same ASN that is different for each PoD; (3) All ToFs share the same ASN.
- The BGP decision process must only consider the AS_PATH.
- It is worth considering relaxing the strict requirement in BGP that ASNs within the AS_PATH must match ex-

actly. To support Equal Cost Multi-Path (ECMP) routing, it is beneficial to consider a group of routes for a specific destination as equal when they share the same AS_PATH length.

Using these configuration guidelines, the fat-tree fabric effectively gets in place multi-path routing information in every FIB; this can be easily checked by solving the k shortest path routing problem for the topology, using, for example, a variant of the Dijkstra's algorithm [16], and comparing with the forwarding entries installed on every node by BGP after convergence.

III. BGP Updates: Analysis and Results

This Section describes the methodology we propose to analyze the behavior of BGP in fat-tree data center topologies that allows us to quantify control plane overhead for a set of use cases. The analysis is strongly based on fat-tree regularity and the well-known behavior of BGP, which permits the count of the expected number of BGP Update messages injected into the network after the occurrence of certain events. Notice that the relevant control plane messages injected by BGP into the network as a consequence of an event are BGP Updates. Other types of BGP messages, exchanged independently of the events (e.g., Keep-alive), are not considered.

Typical data center failures [8] can be used to measure BGP reaction, while recovery tests measure how BGP reacts when the network is back to normal operation. In this work, we consider the following use cases:

- Node Failure: This use case is used to count the number of Updates that BGP exchanges after failures. Such failures can occur in any type of switch within the network topology, including Leaf, Spine, or Core switches.
- Node Recovery: This test case aims to tally the count of Updates exchanged among the switches following the replacement of the faulty switch with a new one. This case also considers Leaf, Spine, or Core nodes.

Packet counting is achieved by analyzing the behavior of BGP after the failure and recovery of a Leaf, Spine, or Core node, expecting to obtain an analytical expression for each case. Taking into account that the whole fat-tree topology can be described by only one parameter (k), the analytical expression should only depend on that k.

Before going into the use cases, let us remark that counting Updates shall take into account a couple of well-known characteristics of BGP. First, BGP is a path vector protocol that suffers from the *path-hunting* problem (i.e., announces partially wrong information under a failure). Second, BGP lacks split horizon; indeed, when a BGP speaker receives an Update from a peer and the new route is installed as the best path after running the BGP decision process, this BGP speaker "reflects" the Update towards the sender, which in turn discards this announcement since it contains its own ASN, and therefore shall be discarded by the loop avoidance mechanism.

Having clarified these aspects, we proceed to consider the use cases. Without losing generality, the analysis assumes that each leaf node announces a unique network prefix.

A. Failure of a Leaf node

When a Leaf node fails, the effect at the routing level is that one prefix in the topology is no longer reachable. Furthermore, the links adjacent to the failed node will no longer be available for the exchange of BGP packets. Considering a chronological analysis of how prefix disaggregation occurs, we should start by noting that the announcements are initiated by the Spine adjacent to the fallen Leaf, which, in the absence of BGP Keepalives messages from it, will assume that the Leaf, and therefore, the prefix reachable through it, is no longer available. In terms of the count of packets exchanged after the failure, this implies that each node in the topology would receive and send through all its interfaces (due to the "reflection" of the Update) a Withdraw containing the prefix that is no longer reachable, resulting in two BGP Updates packets traveling on every link of the topology. The total number of links before the failure corresponds to the expression $k^3/2$ (see Table I). Each Leaf is connected to $k/2$ Spines, so these links will not be considered after failure, leaving a total of $k^3/2 - k/2$ links. Thus, the number of packets exchanged after the failure can be represented by the expression $2 \times (k^3/2 - k/2)$ or equivalently:

$$k^3 - k \tag{1}$$

B. Recovery of a Leaf node

From a global point of view, when a Leaf recovers from a failure it must learn the routes to the prefixes of the entire topology, and, in addition, all the other nodes in the topology must relearn the routes to the new prefix (remember that each Leaf announces a different prefix). To precisely count the number of Update packets exchanged in this scenario, we can divide the topology links into two types: the links adjacent to the recovered node and the rest.

- Links adjacent to the Leaf: When the Leaf node recovers from the failure, it sends an Update to its $k/2$ neighbors (Spines of its PoD) announcing its prefix. In turn, these $k/2$ Spines recognize the prefix as new, and also announce it to all of its neighbors: $k/2$ Leaves to the South and $k/2$ Core to the North. In particular, if we only consider links adjacent to the recovered Leaf, this costs $k/2 \times 2 = k$ packets. On the other hand, the $k/2$ neighbors of the recovered Leaf must announce all the prefixes they know to it, that is, $k^2/2-1$. Finally, the Leaf will recognize these prefixes as new and will announce them to all of its neighbors (the $k/2$ Spines). This adds $(k^2/2 - 1) \times 2 \times k/2$ packets to the above cost, for a total of $(\frac{k^2}{2} - 1) \times k + k$ packets, so far.
- Rest of links: in the rest of the links in the topology, only two packets per link are necessary, that is, each node in the topology will receive and announce the new reachable prefix, for all its links. One way to express this amount, as a function of k, is by subtracting from the total number of links adjacent to the recovered node and multiplying this by the number of packets exchanged on each link. This leaves us with the expression $(\frac{k^3}{2} - \frac{k}{2}) \times 2 = k^3 - k$.

Adding the previous expressions, the polynomial that characterizes the packet overhead for this use case is:

$$\frac{3k^3}{2} - k \tag{2}$$

C. Failure of a Spine node

Due to the location of the Spines in the topology (middle level of the fat tree), it is possible to divide the problem of counting packets after the failure into two sub-problems: inside and outside the PoD of the failure.

Inside the PoD of the failure, each Leaf needs connectivity information for the $k^2/2$ prefixes of the topology; while $k^2/2 - 1$ prefixes are "external", the remaining prefix is directly connected to it. When the failure occurs, in this PoD, $k/2$ Leaf nodes are aware of the failure (connected to the fallen Spine). The BGP processes in each of these Leaf switches will recompute the routes and detect that a potential next hop is missing for each known prefix, indicating that the routes have been updated. Consequently, for each known prefix, they will send an Update with the routes that were updated. In this sense, $k/2$ Leaf nodes send $k^2/2 - 1$ packets (the total number of prefixes in the topology that have lost a next hop) through their $k/2 - 1$ links. Then, the total number of Update packets in the PoD of the failure is equal to $k/2 \times (k/2 - 1) \times (k^2/2 - 1)$.

Outside of the PoD where the failure occurred, the first to notice the fallen Spine are the Cores that were directly connected to it ($k/2$). Due to the characteristics of the topology, these Cores have exactly one link with each PoD. For this reason, after the failure, each of these Cores will no longer have reachability to the prefixes of the PoD of the failure, and therefore, they must send a Withdraw to all their neighbors for the prefixes of such PoD ($k/2$ prefixes to $k - 1$ Spines).

Subsequently, when each Spine receives the mentioned Withdraw, it will notice that reachability has been lost to the announced prefixes, and it will send the corresponding Withdraw through all its interfaces: to the North towards $k/2$ Cores and to the South to $k/2$ Leaves of its PoD. The Cores that receive this will be the same ones that sent the Withdraw since each Spine is connected to ($k/2$) Cores of the same plane. Consequently, they will not update any route and therefore will not send any more packets. On the other hand, the $k/2$ Leaf nodes will receive the Withdraw from the mentioned Spine, they will recalculate their routes and detect that the prefixes received in the Withdraw are no longer reachable through one of their next hops. Then, for each of these $k/2$ prefixes, they will send an Update to all their neighbors ($k/2$ Spines). The Spines that receive these messages will detect their ASN in the AS-PATH of the routes, and therefore, they will not take any action. This is due to the specific ASN numbering, planned to mitigate the effect of the *path-hunting*.

Adding the above expressions, the polynomial:

$$\frac{k^4}{4} - \frac{3k^3}{8} + \frac{5k^2}{4} - k \tag{3}$$

quantifies the injected Updates after the failure of a Spine.

D. Recovery of a Spine node

When the fallen Spine (let's call it S_R) recovers, it receives from its neighbors the announcements of the routes they know. The $k/2$ Leaves connected to it announce the prefixes they know (i.e., $k^2/2$), and on the other hand, the $k/2$ Cores announce the routes they know (to all prefixes except for the $k/2$ of the fallen PoD since they lost reachability during the failure). Let's consider these contributions:

1) Advertisements from the Leaves: S_R receives these advertisements and discards the routes in which its ASN is present in the AS-PATH (all except the prefixes directly connected to its Leaves). Then, it updates its routes and advertises the $k/2$ new routes learned on all its interfaces: north to the $k/2$ Cores and south to the $k/2$ Leaves. This opens two analysis paths: what happens inside the PoD when the Leaves receive the announcements and what happens when these announcements reach the Cores connected to S_R.
   a) Inside the PoD: The leaves connected to S_R receive the announcements of the PoD prefixes. These Leaves update the routes to the other $k/2 - 1$ prefixes of the PoD (since a new next hop is added, that is, a new route) and announce them through all their interfaces to the $k/2$ Spines of the PoD. The Spines receive these announcements and discard them since they are in the AS-PATH of the routes.
   b) From the Cores connected to S_R: After receiving the announcements from S_R, each Core updates the routes towards those prefixes and announces them through all its interfaces, particularly towards the other $k - 1$ Spines connected to each Core. These Spines receive the new $k/2$ routes and announce them through all their interfaces (again towards the Cores and their Leaves). The Leaves that receive these Updates recalculate the routes for the $k/2$ prefixes and advertise them on all their interfaces, that is, back to the Spine that sent them and to the rest of the Spines that are connected to other Cores (from other planes in the topology). The Spines receive this, and since they are on the AS-PATH they take no action.
2) Announcements from the Cores: S_R recalculates the routes to all the prefixes except those below it (since their ASN is in the AS-PATH of the advertisements), that is, $k^2/2 - k/2$, and announces them on all its interfaces, that is, back to the $k/2$ Cores and its $k/2$ Leaves. When these nodes receive the announcements, they do nothing, since their ASN is in their AS-PATH.

To accomplish the packet count, we consider this analysis divided into groups of links with the same behavior:

1) Fault PoD:
   - S_R-Leaves(L) links: $k^2/2$ ($L \rightarrow S_R$) + $k/2$ ($S_R \rightarrow L$) + $k/2 - 1$ ($L \rightarrow S_R$) + $k^2/2 - k/2$ ($S_R \rightarrow L$) + $k^2/2 - k/2$ ($L \rightarrow S_R$) = $3k^2/2 - 1$ packets per

link. There are $k/2$ links of this type, accumulating a total of $(k/2) \times (3k^2/2 - 1) = 3k^3/4 - k/2$ packets.

- Leaf-other-Spine links: only the packets that leave L in the previous point shall be considered, adding a total of $(k/2-1)+(k^2/2-k/2) = k^2/2-1$ packets. There are $(k/2 - 1) \times (k/2)$ (links per Leaf $\times$ number of Leaves) links of this type, accumulating a total of $(k/2-1)(k/2)(k^2/2-1) = k^4/8-k^3/4-k^2/4+k/2$ packets.

2) S_R-Core(C) links: $k/2$ ($S \rightarrow C$) + $k^2/2-k/2$ ($C \rightarrow S$) + $k/2$ ($C \rightarrow S$) + $k^2/2 - k/2$ ($S \rightarrow C$) = k^2 packets. There are $k/2$ links of this type, accumulating a total of $(k^2)(k/2) = k^3/2$ packets.
3) Links of the Core (C) connected to S_R with the rest of its Spines: $k/2$ ($C \rightarrow S$) + $k/2$ ($S \rightarrow C$) = k packets. There are $k-1$ links of this type in $k/2$ Cores, accumulating a total of $(k-1)(k/2)k = k^3/2 - k^2/2$ packets.
4) Links of the Spines (S) of the previous case with their Leaves (L): $k/2$ ($S \rightarrow L$) + $k/2$ ($L \rightarrow S$) = k. There are $k/2$ links of this type per $k-1$ PoDs, accumulating a total of $k \times (k/2) \times (k-1) = k^3/2 - k^2/2$ packets.
5) Links of the Leaves of the previous case with the rest of the Spines (S): $k/2$ ($L \rightarrow S$). There are $k/2-1$ links of this type in $k/2$ Leaves into $k-1$ PoDs, accumulating a total of $= k^4/8 - 3k^3/8 + k^2/4$ packets.

Putting everything together and simplifying we get (4), which quantifies the number of Updates injected into the topology after the recovery of a single Spine node:

$$\frac{k^4}{4} + \frac{13k^3}{8} - k^2 \tag{4}$$

E. Failure of a Core node

In this topology, each Core node is connected once to each PoD, i.e., it has k links with k Spines from different PoDs. When a Core node goes up or down, the k Spines connected to it will be aware of the event.

When a Core node fails, each of the Spines connected to it now has $k/2 - 1$ North links (to other Cores) and $k/2$ South links to their PoD's Leaves. In this context, at the BGP level, each one of these Spines updates the routes of all the prefixes that were reached through the fallen Core, i.e., all the prefixes of the fabric except those of its PoD. This is due to the fact that the fallen Core was a possible next-hop for each of these prefixes. Consequently, after recalculating the routes, these Spines send an Update for each updated prefix ($k^2/2-k/2 =$ the total number of prefixes minus the prefixes in their PoD) for all their interfaces. These announcements are received at North by the Cores, who are in the AS-PATH of the routes received and do not take any action; and at South by the Leaves, who do not take any action either, since the announcements received do not trigger any update of their routes.

In this sense, the k Spines connected to the fallen Core update and announce $k^2/2 - k/2$ routes to their $k-1$ neighbors, accumulating a total of $k(k-1)(k^2/2 - k/2)$ packets. Then, the polynomial:

$$\frac{k^4}{2} - k^3 + \frac{k^2}{2} \tag{5}$$

quantifies the number of Updates injected into the topology by this event.

F. Recovery of a Core node

When the Core recovers from the failure, it will re-establish connections with the corresponding k Spines. These Spines, after establishing the corresponding BGP sessions, announce all the prefixes they know ($k^2/2$). Then, the Core learns these routes and advertises them across all of its interfaces (back to the Spines). The k Spines that receive these announcements update the routes for which they detect a new path, that is, the routes to the prefixes of the rest of the PoDs ($k^2/2 - k/2$), and announce them to all its neighbors: to the north towards $k/2$ Cores and to the south towards the $k/2$ Leaf of its PoD. The $k/2$ Core that receives the announcements will not perform any action since they are in the AS-PATH of the route. In the same way, the $k/2$ Leaves that receive the announcements will not perform any action either, since they already have the paths for the prefixes received through the Spine that sends them the announcements in their routes. Therefore, if we add the expressions described above, we have $k \times (k^2/2) \times 2 + k \times k \times (k^2/2 - k/2)$ packets. Then, the polynomial:

$$\frac{k^4}{2} + \frac{k^3}{2} \tag{6}$$

quantifies the number of Updates injected into the topology by this event.

We close the Section with a summary of the analytical expressions that quantify the messages exchanged in the fat tree in reaction to the different events considered, as shown in Table II below.

TABLE II
SUMMARY OF THE FUNCTIONS OF k THAT QUANTIFY THE AMOUNT OF CONTROL PLANE OVERHEAD INJECTED BY BGP AFTER THE FAILURE OR RECOVERY EVENTS.

Use case	Function $f(k)$
Leaf Failure	$k^3 - k$
Leaf Recovery	$\frac{3k^3}{2} - k$
Spine Failure	$\frac{k^4}{4} - \frac{3k^3}{8} + \frac{5k^2}{4} - k$
Spine Recovery	$\frac{k^4}{4} + \frac{13k^3}{8} - k^2$
Core Failure	$\frac{k^4}{2} - k^3 + \frac{k^2}{2}$
Core Recovery	$\frac{k^4}{2} + \frac{k^3}{2}$

IV. Validation and Discussion

Ideally, a fair validation of the theoretical results should be made against a formally correct implementation of BGP; instead, our validation was carried out running a specific BGP implementation (FRR) under the ns-3 simulator, because, in the first place, it is readily available, and secondly, FRR has important industrial support, and therefore, using this implementation may be considered representative for a real-world scenario. Nevertheless, working code always has bugs, and specifically, a routing protocol implementation is prone to race conditions that may cause the transmission of spurious messages. This approach has been taken in our previous work [7], where a thorough description of the simulation framework can be found.

The validation framework permits the creation of fat-tree topologies as a function of the k parameter, announcing a unique prefix per Leaf, as in our theoretical analysis, permitting the capture of every control packet exchanged during a given experiment. Once completed, it is possible to count the interesting messages, BGP Updates in our case, to compare with the theoretical results.

TABLE III
SUBTRACTION BETWEEN EXPERIMENTAL AND EXPECTED RESULTS FOR EACH EVALUATED k. THE EXPECTED RESULTS ARE THE EVALUATION IN k OF (2), (4) AND (6).

k	Leaf Recovery	Spine Recovery	Core Recovery
4	4	8	8
8	8	16	16
12	12	24	24
16	16	32	32
20	20	40	40
24	24	48	48
28	28	56	56
30	30	60	60

Regarding failure use cases, "Leaf Failure" has already been considered in [7], while the experimental results obtained for the cases of "Core Failure" and "Spine Failure" exactly verify the analytical expression for each of these cases. Thus, if you replace the value of k in formula (5) with the value associated with your fat-tree configuration, you will obtain the number of Updates injected into the network by BGP after the "Core Failure" event. Furthermore, the resulting quantity is identical to what would be obtained by developing and deploying a simulation of the fat tree topology.

Regarding recovery use cases, some differences were found. Table III shows the difference between the experimental and the theoretical results for each use case. The first column contains the value of k, while the rest includes the subtraction of the result of evaluating the formula and the experimental results, for the corresponding k. Please note that differences are slight but visibly regular with respect to k. For the case of "Leaf recovery", the difference is exactly k, while for the cases of "Spine recovery" and "Core recovery", the difference is $2k$.

Considering the regularity of the differences, we explored in more depth the packets exchanged in the simulations. In all three cases, we found that the recovered nodes send an empty Update (i.e., without prefixes) to all their neighbors, which reflects it back to the recovered node. Then, if we count the extra Updates, we find that in the case of the recovery of a Spine or Core node (both have k links), there are $2 \times k$ packets, while in the case of the Leaf node (which has $k/2$ links exchanging Updates), there are $2 \times k/2 = k$ packets. These results verify the differences shown in Table III.

Therefore, we can state that the difference found is due to the BGP implementation behavior in the experimental environment. This confirms that the theoretical analysis is correct and valuable since it serves as a lower bound to quantify the control plane overhead.

We could ask ourselves, what is more expensive, a failure or a recovery? Although the orders of the polynomials in Section III give us a hint, it is worth analyzing the issue in detail.

Figure 2 shows the comparison between the cost of a failure and recovery for each level of the topology. In all three scenarios, recovery is more expensive than node failure. Note that the figures show the trend up to $k = 30$, where the topology contains 1125 network nodes (a representative number for a massive scale data center). It is important to note that although the equations for the case of failure and recovery by level are of the same order, the difference may be relevant at the scale of a massive data center. The extra cost of recovery may inspire network managers to plan switch replacement to mitigate the impact carefully.

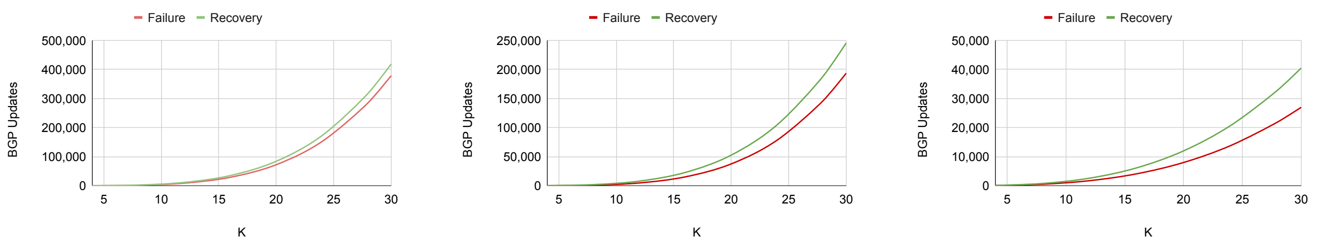
An alternative view of the results is to consider failure and recovery cases separately for all levels of the topology. Figure 3 shows the results for failure cases, which reaffirms the intuition that events at higher levels in the fat tree require more messaging for BGP to re-converge.

Overall, the convergence of BGP in the inter-domain scope has been researched for many years, using both formal and practical approaches. It has been an important issue in the data center scope, and most representative results are condensed in RFC 7938 as practical configuration guidelines [13]. However, there are few efforts to measure the BGP control plane in scale, and, as far as we know, the current article presents the first complete set of failure/recovery use cases theoretical analysis with experimental validation.

V. CONCLUSION AND FUTURE WORK

In this paper, an analytical methodology to count BGP Updates has been presented and validated for representative data center networks in a simulation environment. This methodology can be used as a planning tool to evaluate the control plane overhead for real-world deployments. Furthermore, the formulae developed in this work can be easily adapted to operational environments where each Leaf node announces more than one prefix. Then, using the appropriate k to a given reality, the network planner can obtain the amount of overhead that BGP would inject into the network under failure and recovery events. The same methodology can be applied to other protocols, such as RIFT, obtaining the injected overhead without the need for any infrastructure deployment; this case is particularly interesting since RIFT is under active development.

Our work is based on multi-plane fat-tree networks, which can be described by the unique parameter k, which is both



(a) Core failure and recovery. (b) Spine failure and recovery. (c) Leaf failure and recovery.

Fig. 2. Comparison of expected packets injected into the Fabric after a node failure and recovery.

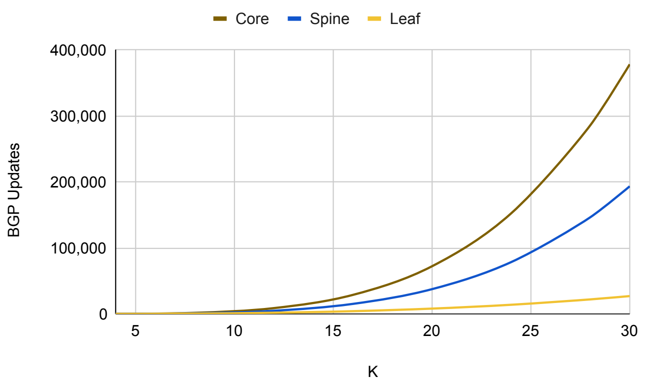


Fig. 3. Expected number of BGP Updates injected into the network after a Spine, Leaf, and Core node failed for different fat-tree configurations.

the number of PoDs and the fan out of each topology switch. A general representation of fat-tree topologies has been introduced in [4], where the topology is described with extra parameters; a generalization of the current work may be attempted to have a general map of BGP control plane overhead for a complete set of fat-tree configurations. Another possible future work is to measure convergence time as an alternative to the Update count presented here. On the other hand, modifications to BGP may be envisioned to reduce convergence time, incorporating, for example, a split-horizon capability. Nevertheless, modifying running (industrial) software is not a simple (nor desirable) task, and this particular modification would possibly demand more memory and CPU resources for network nodes.

Moreover, we can quantify the overhead traffic "in the void", but for that task, we are not considering in the present work two critical parameters: namely i) the payload traffic (i.e., the traffic exchanged by applications), and ii) the frequency of network failures events. Considering these aspects is a line of future work that would help better quantify control plane overhead.

REFERENCES

[1] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," Internet Requests for Comments, RFC Editor, RFC 4271, January 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4271.txt

[2] A. Abhashkumar, K. Subramanian, A. Andreyev, H. Kim, N. K. Salem, J. Yang, P. Lapukhov, A. Akella, and H. Zeng, "Running BGP in data centers at scale," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*. USENIX Association, Apr. 2021, pp. 65–81.

[3] R. White, S. Hegde, and T. Przygienda, "IS-IS Optimal Distributed Flooding for Dense Topologies," Internet Engineering Task Force, Internet-Draft draft-ietf-lsr-distoptflood-01, Jan. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-lsr-distoptflood/01/

[4] T. Przygienda, A. Sharma, P. Thubert, B. Rijsman, D. Afanasiev, and J. Head, "RIFT: Routing in Fat Trees," Internet Engineering Task Force, Internet-Draft draft-ietf-rift-rift-17, Mar. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-rift-rift/17/

[5] T. Caiazzi, M. Scazzariello, L. Alberro, L. Ariemma, A. Castro, E. Grampin, and G. D. Battista, "Sibyl: a framework for evaluating the implementation of routing protocols in fat-trees," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–7.

[6] G. F. Riley and T. R. Henderson, *The ns-3 Network Simulator*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34.

[7] L. Alberro, F. Velázquez, S. Azpiroz, E. Grampin, and M. Richart, "Experimenting with routing protocols in the data center: An ns-3 simulation approach," *Future Internet*, vol. 14, no. 10, 2022.

[8] P. Gill, N. Jain, and N. Nagappan, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 350–361.

[9] C. Clos, "A study of non-blocking switching networks," *The Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, March 1953.

[10] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, Oct 1985.

[11] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, ser. SIGCOMM '08. New York, NY, USA: ACM, 2008, pp. 63–74.

[12] D. Medhi and K. Ramasamy, *Network Routing, Second Edition: Algorithms, Protocols, and Architectures*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017.

[13] P. Lapukhov, A. Premji, and J. Mitchell, "Use of bgp for routing in large-scale data centers," Internet Requests for Comments, RFC Editor, RFC 7938, August 2016.

[14] Q. Vohra and E. Chen, "Bgp support for four-octet as number space," Internet Requests for Comments, RFC Editor, RFC 4893, May 2007.

[15] R. Oliveira, B. Zhang, D. Pei, and L. Zhang, "Quantifying path exploration in the internet," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 445–458, 2009.

[16] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, p. 269–271, dec 1959.