Prioritizing deployments achieving targeted network performance across a multilayer Pb/s network

Srivatsan Balasubramanian[1], Bodhisattwa Gangopadhyay[2], Vinayak Dangui[1], Satyajeet Singh Ahuja[1], Varun Gupta[1], Grigory Pastukhov[1], Max Noormohammadpour[1], Alexander Nikolaidis[1], Ariyani Copley[1], Xueqi He[1], Jiachuan Tian[1], Jiajia Chen[1], Arash Vakili[1], Chiun Lin Lim[1], Guanqing Yan[1], Anand Gokul[1], Biao Lu[1], and Debottym Mukherjee[1]

[1]*Backbone Engineering, Meta, Menlo Park, CA, USA*
[2]*Backbone Engineering, Meta, London, UK*

***Abstract*—Meta has a large scale backbone infrastructure supporting services with varying QoS requirements. As part of backbone network planning, a capacity plan that differentiates between different classes of services in terms of availability guarantees is generated and scheduled for deployment. Deployment progress is measured traditionally in terms of volumes of capacity deployed. Our work provides insights into the shortcomings of capacity volume driven deployments. We provide a methodology to rank the contribution of each entity pending deployment towards our network performance goals and use this metric to prioritize deployments helping higher classes of services meet their network guarantees earlier in the deployment schedule. By enabling QoS awareness in backbone deployments, we are able to demonstrate a 67% reduction of risk exposure period for high priority services.**

***Index Terms*—multi-layer, prioritization, deployment, planning**

I. Introduction

Meta's services are enabled by the large server farms that are set up in data centers (DC) and Point of Presence (POP) sites. The DCs and POPs are connected over a distributed wide area network. This wide area backbone is an example of a multilayer network enabled by IP/MPLS services carried over a subsea and terrestrial fiber optic mesh network distributed around the world. Designing and planning such a multilayer network is complex in nature considering the various constraints and policies in the IP and optical layer. Traffic in the Meta backbone is categorized into four classes (AF1, AF2, AF3, and AF4, with AF1 being highest priority) to differentiate services with different expectation levels for availability requirements.

During the design phase, a multilayer planning tool is used to generate a network design plan considering various constraints and policies in the IP and optical layer, the different failure models, the traffic growth and ensures that the availability guarantees are met for each service based on their QoS requirements. The result of this network design is a plan of record (POR), wherein each record is a capacity or fiber entity to be deployed. On the operational side, the backbone network uses a centralized network controller to make routing and Traffic Engineering (TE) decisions [1] and utilizes deployed capacity efficiently. The controller implements different path allocation algorithms for different QoS classes [2].

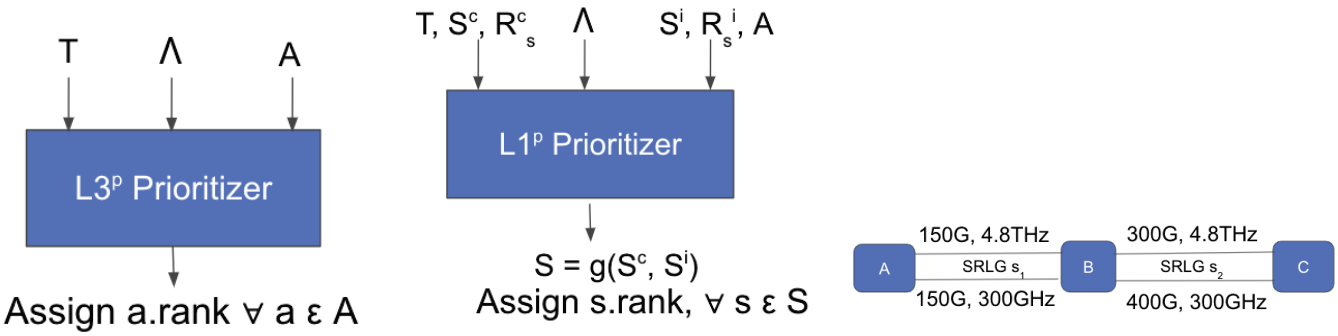
978-3-903176-44-7

Before a QoS differentiated design materializes in the production network to be managed by the TE controller, there are numerous challenges during the network deployment phase. Planned capacity takes months to deploy due to uncertainties in the supply chain which is dependent on various geo-political situations and other unforeseen circumstances (such as covid-19 pandemic). Deployments are traditionally oblivious to QoS requirements. Deployment practices in the industry can range from being a completely adhoc, tedious, non-scalable process to at best being prioritized based on deployed capacity volumes. The implication of capacity delivery without QoS aware prioritization is that it can often lead to situations where lower classes of service get their performance guarantees met during initial phases of deployment at the expense of higher classes of service (CoS). Alternatively, significantly larger portion of the capacity must be deployed for higher CoS to meet the desired guarantees.

We make two contributions as part of this paper. **First**, we are not aware of any state-of-the-art, and prior literature that recognizes the role of risk based prioritization in backbone deployments, and we demonstrate how this can be the primary driver for deployment schedules. **Second**, we propose a Mixed-Integer Linear Programming (MILP) framework to provide a QoS based priority signal on every entity to be built as part of POR resulting in a targeted deployment wherein the target is to have higher priority services meet their service guarantees earlier in the deployment schedule before lower priority services. Deployment prioritization is necessary in both L1 and L3 layers. We present in detail the solutions to L1 and L3 prioritization problems followed by risk simulations on production topology to demonstrate that it is possible to reduce risk exposure for high priority services by carefully prioritizing the capacity plan.

II. Benefits of Prioritization

The network planning problem involves three stages: i) fiber/site procurement, ii) optical/site design, and iii) capacity planning. Fiber procurement has the longest lead time and the most risk, and uncertainty related to it should not affect optical planning. Similarly, optical planning is tied to fiber performance and requires site access which translates to an external dependency/risk and should be decoupled from capacity



(a) (b) (c)

Fig. 1: (a) $L3^p$ Prioritization (b) $L1^p$ Prioritization (c) Dependency Mapper example topology

planning to achieve a more predictable delivery. Ideally, optical plans should only use fiber/sites fully delivered and tested and the capacity plans (POR) should only consume optical network previously built and operational. Typical lead times of deployments would be as follows: (a) development of a subsea cable (3-5 years) (b) procurement of existing fiber and deployment of optical network (18-24 months) (c) design and installation of hardware (6-12 months) and (d) capacity design and activation (3-6 months).

During planning phase, we strive to realize POR using only a previously built optical network - and thus avert the external risk of hardware sourcing or vendor fiber installation / testing. We attempt to achieve this by using a conservative demand forecast that results in a network overbuild ahead of time. However, in practice, this does not always go as planned. Network delays may be caused due to i) deferments in fiber or site deliveries, or ii) having to redesign optical links due to stale test data used during initial link budgeting, or iii) unavailability of required power and space for equipment. Alternatively, underestimated demand forecast could require capacity to be turned up even before the optical network is ready. In either of the scenarios (supply coming later than expected, or demand coming earlier than expected), it is important to prioritize POR items which will have immediate benefit on the network to mitigate risk, or unblock future plans. This results in a feedback path where the demand forecast influences capacity priority, and in turn capacity priority influences optical network (fiber) priority.

Under normal scenarios, having a priority for POR helps set up a shared vision across Meta's teams that come together to realize the POR. POR prioritization assumes immense significance when there are unpredicted surges in traffic, disruptions in the supply chain, or in the face of adverse conditions like disasters, storms, pandemics [3] etc. Traditionally, deployment prioritization strategies are based on capacity volumes where bigger capacity chunks are deployed earlier. Since this is agnostic to QoS, it implies that a significant portion of the capacity plan must be deployed for higher priority of services to meet its guarantees. A QoS aware prioritization for backbone deployments targets specific service class and tailors deployment plan in a way that higher priority services meet their service guarantees earlier in the deployment schedule.

In sections to come, we present a MILP framework which teases out the risk mitigation potential of each deployable entity while solving for multiple failure scenarios for different classes of service. The highest priority is given to the entities that enables highest mitigation of risk for the highest classes of service. There are two flavors of deployment prioritization problems addressed in this paper. In the first flavor (called $L3^p$), POR consists of only incremental Layer 3 capacity entities that need to be built (i.e, IP and optical transponders to be lit), and the underlying fiber footprint is already installed (i.e. optical line system is turned up). In the second flavor (called $L1^p$), POR includes both the new fiber entities that is yet to be built (i.e, line system not provisioned yet), and the incremental L3 capacity entities that need to be lit.

III. L3 Prioritization

L3 adjacencies connect routers in the network and are routed over multiple Shared Risk Link Groups (SRLG). A SRLG is a conduit that has multiple fiber strands called rails. The $L3^p$ design problem can be stated as follows and is described in Figure 1a. Given a set of pending adjacencies A, a traffic matrix T, and a failure set Λ, associate a rank with every adjacency with the objective that when adjacencies are deployed in this sequence, a higher class of service in T is able to meet its guarantees before a lower class of service.

A. $L3^p$ Solution

A greedy solution is to prioritize adjacency with the highest capacity (L3-CAPACITY). In L3-CAPACITY, A is sorted in the descending order of its capacity, the adjacency with the highest capacity is ranked 1, and an increasing number is assigned to each adjacency successively.

The solution that we propose is a MILP based framework called L3-QOS. We rank each pending adjacency by its ability to participate in the mitigation solution for different classes of service under different planning failure scenarios. We prioritize adjacencies that provides higher levels of mitigation for higher classes of service under failure scenarios of higher probability. The MILP formulation takes as input the installed adjacencies (A^i), A, T, Λ, and returns a solution that recommends the minimal subset of A that will mitigate the risk Λ. The demand matrix is converted into a series of matrices, one for each

failure scenario (based on failure policy), since not all classes are protected the same way for each failure.

The pending adjacencies a $\in A$ are modeled in our framework using planning constructs called projects. Our network software, for reasons related to operational simplicity, scale and performance, mandates certain adjacencies to be be paired and brought up together. These adjacency pairings are inferred through policy rules and are mapped into projects. The MILP framework selects projects to mitigate risks, which in turn are mapped back to selected adjacencies to be turned up.

Symbol Description
$\mathbf{A}_\lambda$ set of adjacencies available under a failure $\lambda \in \Lambda$
$\mathbf{c_{u,v}}$ capacity (in gbps) of adjacency (u, v)
$\mathbf{d^\lambda_{i,j,c}}$ demand from source i to destination j of class c under failure λ
$\phi_\mathbf{c}$ weight associated with class of service c
$\delta_\mathbf{k}$ penalty associated with project P_k
$\mathbf{f^\lambda_{i,j,c}}$ flow from source i to destination j of class c under failure λ
$\mathbf{f^\lambda_{i,j,c}(u,v)}$ flow volume from source i to destination j via adjacency $(u, v) \in A_\lambda$ for class c under failure λ
$\mathbf{l_{u,v}}$ indicator variable for an adjacency $(u, v) \in A$
$\mathbf{p_k}$ indicator variable for project P_k

TABLE I: Key notations in problem formulation.

The notation used by the MCF formulation is defined in Table I and the formulation is provided below.

Note that A_λ is the same as A when λ corresponds to the no failure state. When it corresponds to a failure state, the failed adjacencies from A are excluded. Across all failures, total flow is conserved at each node:

The following constraints apply $\forall \;\; d^\lambda_{i,j,c}$

$$\begin{aligned}
&\sum_{\{i,u\}\in A_\lambda} f^\lambda_{i,j,c}(i,u) - \sum_{\{i,u\}\in A_\lambda} f^\lambda_{i,j,c}(u,i) = f^\lambda_{i,j,c} \le d^\lambda_{i,j,c} \\
&\sum_{\{u,j\}\in A_\lambda} f^\lambda_{i,j,c}(u,j) - \sum_{\{j,u\}\in A_\lambda} f^\lambda_{i,j,c}(j,u) = f^\lambda_{i,j,c} \le d^\lambda_{i,j} \\
&\sum_{\substack{\{u,v\}\in A_\lambda,\\ u\neq i, v\neq j}} f^\lambda_{i,j,c}(u,v) - \sum_{\substack{\{u,v\}\in E,\\ u\neq i, v\neq j}} f^\lambda_{i,j,c}(v,u) = 0
\end{aligned} \tag{1}$$

For an adjacency (u, v) $\in$ both A and A_λ capacity on the adjacency is consumed only if the corresponding adjacency is turned up.

$$\begin{aligned}
&\sum_{\{i,j,c\}} f^\lambda_{i,j,c}(u,v) \le c_{u,v} \forall (u,v) \;\; s.t. \;\; (u,v) \in A_\lambda \& (u,v) \in A^i \\
&\sum_{\{i,j,c\}} f^\lambda_{i,j,c}(u,v) \le l_{u,v} * c_{u,v} \; \forall (u,v) \; s.t. \; (u,v) \in A_\lambda, A
\end{aligned} \tag{2}$$

If an adjacency is turned up, the project corresponding to the adjacency is turned up.

$$p_k \ge l_{u,v} \;\; \forall k \quad s.t. \quad u, v \in P_k \tag{3}$$

The objective of the formulation is to minimize the total dropped demand under each failure scenario, and to minimize the projects that need to be turned up to mitigate the risk caused by the failure.

$$\min \sum_{i,j,c,\lambda} \phi_c * (1 - \frac{f^\lambda_{i,j,c}}{d^\lambda_{i,j,c}}) + \sum_k \delta_k * p_k \tag{4}$$

For every failure event $\lambda \in \Lambda$ in the prioritizer policy, we run the above formulation which identifies the subset M_λ from candidate set A that help with the mitigation for the failure λ. For failure λ and no mitigation, suppose, drops in gbps seen in the solution for each demand k is α^λ_k. With M_λ turned on, the drops seen for each demand in the solution again is β^λ_k. A metric called impact credit can be associated with every pending adjacency using the following method.

Suppose c_k is the class of service of demand k, ϕ_{c_k} is the weight we assign to class c_k. The difference in the drops between after and before the mitigation is attributed as the risk mitigation potential that is brought out by the adjacencies in M_λ and is summed over every demand as $\sum_k \phi_{c_k}(\beta^\lambda_k - \alpha^\lambda_k)$. This mitigation potential is apportioned uniformly to every $a \; \varepsilon \; M_\lambda$. Suppose the failure j happens with probability τ_λ, δ^λ_a is the indicator function if an adjacency $a \; \varepsilon \; M_\lambda$, and $|M_\lambda|$ is the number of adjacencies that mitigate failure λ. The impact credit earned by each adjacency $a \in A$ is calculated as $\sum_\lambda \tau_\lambda \frac{\delta^\lambda_l}{|M_\lambda|} \sum_k \phi_{c_k}(\beta^\lambda_k - \alpha^\lambda_k)$. Elements in A are sorted in the descending order of its earned impact credit, the adjacency with the highest impact credit is given the highest priority (priority 1), and an increasing number is assigned to each adjacency successively.

IV. $L1$ Prioritization

Figure 1b describes the $L1^p$ prioritization problem for a given deployment phase. Suppose S^c is the set of all SRLGs and R^c_s is the corresponding end state rail counts $\forall$ s $\in S^c$. Suppose the installed SRLG footprint is S^i, the installed rail counts is $R^i_s \; \forall$ s $\in S^i$ and adjacencies pending deployment is A. The goals of $L1^p$ is as follows: (a) identify $S^i \subseteq S \subseteq S^c$, that need to be built (provisioned) to deliver every adjacency $\in A$, given that some SRLGs and rails may be built ahead of time and may not be required to satisfy the POR at hand. (b) associate a rank with each SRLG $\in S$ so that, when SRLGs are delivered in the order of rank, lower priority services meet their availability guarantees before higher priority services.

A. $L1^p$ Solution

A greedy approach to the problem, called L1-CAPACITY, is described as follows. S is derived as $\{s : R^i_s < R^c_s, \forall s \in$ a.traversed_srlgs, $\forall a \in$A$\}$. Every SRLG s $\in S$ is associated with a metric called pending_capacity computed by summing up the capacities of each adjacency in A that traverses SRLG s. With L1-CAPACITY, S is sorted in the descending order of its pending_capacity, the SRLG with the highest pending_capacity is ranked 1, and an increasing rank is assigned to each SRLG successively.

To solve $L1^p$ problem with QoS awareness, we break it into three simpler sub problems. In the first subproblem, we would like to compute the true risk mitigation potential of every pending adjacency, assuming that all SRLGs are installed. In the second subproblem, we would like to admit as much of the pending adjacencies as possible within S^i, and arrive at the dependency map which describes the list of SRLGs that block each unadmitted adjacency. In the third subproblem, we formulate a MILP that selects SRLGs to be

Algorithm 1 DEPENDENCY-MAPPER(A, policy, route_db)

```
1: S_a = {}
2: A^b = list()
3: for adjacency in A.sort(primary_key=a.rank) do
4:     trails = policy.get_trails(adjacency)
5:     for trail in trails do
6:         τ = route_db.get_technology(trail)
7:         data_rate = τ.data_rate()
8:         channel_width = τ.channel_width()
9:         multiples = τ.get_min_channel_increment()
10:        min_width = channel_width * multiples
11:        min_rate = data_rate * multiples
12:        req_bw = trail.admitted_bandwidth + adjacency.bandwidth
13:        min_channels = ⌈ req_bw / min_rate ⌉
14:        spectrum_required = min_width * min_channels
15:        for srlg in adjacency.traversed_srlgs do
16:            if spectrum_required ¿ srlg.residual_spectrum then
17:                S_a[adjacency].add(srlg)
18:                srlg.blocked_adjcancies.add(adjacency)
19:            end if
20:        end for
21:    end for
22:    if S_a[adjacency] then
23:        A^b.append(adjacency)
24:    else
25:        for trail in trails do
26:            trail.admitted_bandwidth += adjacency.bandwidth
27:        end for
28:        for srlg in adjacency.traversed_srlgs do
29:            srlg.residual_spectrum -= spectrum_required
30:        end for
31:    end if
32: end for
```

built iteratively such that the pending adjacencies (identified from second subproblem) is admitted, and the overall risk mitigation potential of the admitted adjacency (as computed from first subproblem) is optimized in each iteration.

The first subproblem is precisely the same as $L3^p$ problem and we use the L3-QOS system described in section III to arrive at the rank of each pending adjacency. The sections to come describe how the second and third problems are solved using the Dependency Mapper and L1-QOS respectively.

1) Dependency Mapper: Given A, S^i, and R_s^i as input, the goal of dependency mapper is to compute the following: (a) the subset of A that can be admitted into S^i and to identify the unadmitted adjacencies (say, A^b) that are blocked on SRLG builds, (b) S required to deliver A^b and (c) $S_a = \{s : \forall s \in S, s \text{ blocks a}\}, \forall a \in A^b$.

Algorithm 1 is used by dependency mapper where an adjacency is realized in the optical network as a sequence of trails (a non-regenerated sequence of SRLGs). Regeneration is driven primarily by policy. Example, trans-continental adjacencies to be regenerated before entering and after existing a subsea segment regardless of optical capabilities is a mandated policy. By applying such a policy, we come up with the trails that constitute each adjacency. For each trail, actual spectrum consumed needs to be estimated. A system called route_db maintains the translation between L3 bandwidth requirement on a trail to spectrum consumed in a SRLG in a heterogenous multi-vendor network deploying different generations of technology. Every SRLG is associated with a residual_spectrum by accounting for spectrum of already installed channels on this SRLG and excluding it from the maximum usable spectrum on the SRLG. Typically, the maximum usable spectrum on a SRLG is set to be 75 % of C-Band to account for spectrum contention and obviates the need to do end to end spectrum assignment as part of planning phase. If there is a SRLG that belongs to the trail and does not have residual spectrum to admit this adjacency, this adjacency is considered blocked on the SRLG. If there is no SRLG that blocks the adjacency, this adjacency is admitted by reserving spectrum on all its traversed SRLGs. At the end of the run of algorithm described in 1, A^b, S, and S_a are available to be used as input to SRLG prioritization.

2) L1-QOS Solution: SRLGs are modeled using the planning construct called Projects for the same reason described in III-A, with the difference that in this context, a project refers to the SRLGs that need to be paired and turned up together. The L1-MIP formulation takes in the various inputs described in Table II, and provides the list of κ projects that are selected. The incentive in the formulation is to maximize weighted sum of mitigation risk potential and admitted capacity, and the penalty is provided as a difficulty level of implementation of the projects that were chosen. The formulation is run, κ projects are selected, each project here is individually ranked, this input is frozen, next κ projects are selected, and this iteration continues until all projects are ranked.

Symbol Description
$\mathbf{A^b}$ set of blocked adjacencies. Elements in $\mathbf{A^b}$ are denoted by $a \in \mathbf{A^b}$ - refer section IV-A1
$\mathbf{S}$ set of SRLGs to be built. Elements in $\mathbf{S}$ are denoted by $s \in \mathbf{S}$ - refer section IV-A1
$\mathbf{Proj}$ set of Projects. Elements in $\mathbf{Proj}$ are denoted by $p \in \mathbf{Proj}$
$\mathbf{S}_a = \{s \in \mathbf{S} : s \text{ blocks } a\}$ corresponds to the blocking SRLGs of $a \in \mathbf{A^b}$ (section IV-A1)
$\mathbf{S}_p = \{s \in \mathbf{S} : s \in \text{ project } p\}$ corresponds to SRLGs part of $p \in \mathbf{Proj}$
$\mathbf{Dep} = \{(a, a') \in \mathbf{A^b} \times \mathbf{A^b} : a' \text{ depends on } a\}$ set of dependencies between adjacencies.
$I_a \in \mathbb{R}$ corresponds to impact of adjacency $a \in \mathbf{A^b}$ (section III-A)
$C_a \in \mathbb{R}$ corresponds to the capacity pending on adjacency $a \in \mathbf{A^b}$.
$D_p \in \mathbb{R}$ corresponds to the difficulty level of project $p \in \mathbf{Proj}$.
$\kappa \in \mathbb{N}$ corresponds to the number of projects that we must select.
$u_s \in \{0, 1\}$ variable for each $s \in \mathbf{S}$. Represents if we turn up the corresponding SRLG ($= 1$) or not ($= 0$)
$v_a \in \{0, 1\}$ variable for each $a \in \mathbf{A^b}$. Represents if we turn up the corresponding adjacency ($= 1$) or not ($= 0$)
$w_p \in \{0, 1\}$ variable for each project $p \in \mathbf{Proj}$. Represents if we turn up the corresponding project ($= 1$) or not ($= 0$)

TABLE II: Inputs to the formulation.

Constraints:

All the blocking SRLGs need to be turned up for an adjacency to be unblocked:

$$v_a \leq u_s, \quad \forall a \in \mathbf{A^b}, \forall s \in \mathbf{S}_a.$$

Number of projects turn up is at most κ:

$$\sum_{p \in \mathbf{Proj}} w_p <= \kappa.$$

Selecting project $p \in \mathbf{Proj}$ unblocks SRLG $s \in \mathbf{S}_p$:

$$u_s \geq w_p, \quad \forall p \in \mathbf{Proj}, \forall s \in \mathbf{S}_p.$$

Dependency between adjacencies as described in section III-A. If adjacency a' depends on adjacency a ($(a, a') \in \mathbf{Dep}$), then the only way to turn up a' is having turn up a first:

$$x_{a'} \leq x_a, \quad \forall (a, a') \in \mathbf{Dep}.$$

Objective

$$\max \left\{ \sum_{a \in \mathbf{A^b}} (I_a + C_a) v_a - \sum_{p \in \mathbf{Proj}} D_p w_p \right\}$$

V. Evaluation

The tools and systems presented here were applied to 2021 capacity and SRLG builds in the Meta production topology having 4 cos types and the primary performance metric looked into were availability miss (A_{miss}) which is defined as $max(0, \mathcal{A}_t - \mathcal{A}_a)$, where $\mathcal{A}_t$ is the target guarantee promised for the class of service, and $\mathcal{A}_a$ is achieved availability as reported by Risk Simulation System described in [1].

A. L3-QOS vs L3-CAPACITY Results

Figure 2a shows the normalized cumulative capacity evolution as each pending adjacency is deployed vs the ascending order of rank. With L3-CAPACITY, adjacencies are deployed in the descending order of capacity values observed to be steep, while with L3-QOS, the cumulative capacity evolution curve is observed to be nearly linear. Figure 2b plots the ranks of the pending adjacencies from L3-CAPACITY on the y-axis vs ranks from L3-QOS on the x-axis for the same adjacency. All the points above the diagonal line are considered more important by L3-QOS than L3-CAPACITY, while all the points below are considered less important by L3-QOS than L3-CAPACITY. The graph is divided into four quadrants depending on whether the adjacency has high or low impact and has high or low capacity. The points depict that correlation between impact of an adjacency and its size is not high. The lack of correlation explains the roughly uniform distribution of capacity across ranks and the near linearity for L3-QOS in Figure 2a.

Figure 2c shows the evolution of Amiss of AF1 class of service as capacity deployment progresses and which decreases as more capacity is deployed. With L3-CAPACITY, all AF1 services meet their availability targets when 22% of the capacity is deployed, whereas, with L3-QOS, all AF1 services meet their availability targets much earlier, when only 7% of capacity is installed. The pattern repeats with other cos types as seen in Figure 3. The benefits of meeting availability guarantee with lower capacities becomes further clear in Figure 3c. With L3-CAPACITY, it takes 40 days for AF1, and 108 days for both AF1 and AF2 to meet their respective guarantees. These two targets are achieved with L3-QOS in 13 and 35 days respectively. This is a significant result since it implies that vulnerability of important services (AF1 and AF2) was eliminated by L3-QOS for 73 days - which is a 67% risk reduction over L3-CAPACITY.

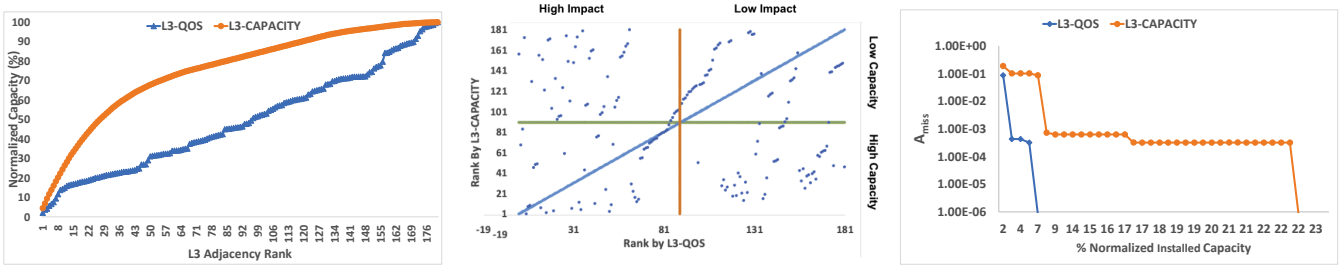
B. L1-QOS vs L1-CAPACITY Results

The results for L1p have same input topology as L3p except that some subset of SRLGs is not installed. For a given value of κ, L1-QOS ranks all pending SRLGs while traversing SRLGs in ascending order. Considering the SRLG of rank r, the adjacencies enabled by all fibers with rank <= r is identified, and impact metric of these adjacencies is summed up, normalized, and plotted (Figure 4a) representing evolution of risk mitigation potential of the topology. For values of $\kappa >$ 1, the difference is very marginal, while κ = 4 demonstrates best results. Figures 4b, 4c and 5a shows the evolution of Amiss of classes of service AF1, AF2, and AF3 respectively as capacity deployment progresses for L1p scenario. With L1-QOS, up to 40% of capacity is required for AF1 services to meet it guarantees, whereas with L1-CAPACITY, 55% is needed. Figure 5b, shows when 25 % of capacity is installed, it selects some capacity entities that reduce risk for AF1 better than L1-QOS (by coincidence), while L1-QOS rapidly progresses towards its target at 40%. Figure 5b shows the evolution for AF4 services, where initially the gain for AF4 is higher for L1-CAPACITY suggesting that it is mitigating risk for AF4, whereas the corresponding values for L3-QOS approach is lower due to its focus on high priority services. However, when 70% of the services are installed, that the two approaches become identical.

Another important point to note is that capacity requirement to meet AF1 guarantees went up from 7 % with L3-QOS to 40 % with L1-QOS This is primarily because the SRLGs required for handling AF1 services are assumed to arrive strictly one after another. Figure 5c shows the % of extra route builds that are required using L1-CAPACITY approach as compared with L1- QOS approach before a class of service reaches its target or end state availability. The excess route builds required varies between 10% and 30%. The primary reason is as follows - there are a few SRLGs with adjacencies traversing it, but there are already sufficient strands in the field to be able to admit this capacity. Since L1-CAPACITY does not perform dependency mapping, we continue to turn up more fibers which could otherwise be deferred to next deployment phase.

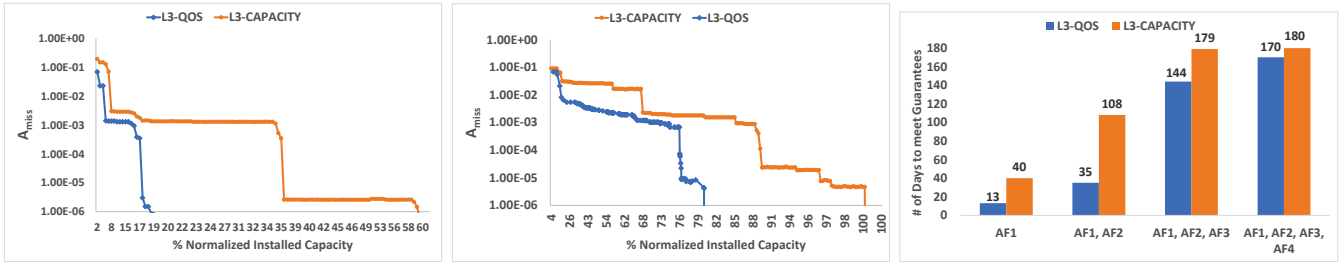
VI. Conclusion

This paper introduces QoS aware $L3^p$ and $L1^p$ prioritizer systems used for sequencing delivery of capacity and SRLG builds at Meta. Through this mechanism, we are able to perform a targeted POR delivery ensuring that higher classes of service meet their availability guarantees faster (67% faster for top two classes) compared with the conventional volumes based deployments, thereby providing the network an improved risk profile. We were able to utilize these tools effectively during the peak of covid-19 times in identifying important routes to build that helped us continue to operate the network at low risk levels despite mobility being severely restricted. We hope that the learning that we shared here will help the broader networking researchers and practitioners in embracing a QoS aware approach for prioritizing deployments, which could prove to be a powerful toolkit in realizing the goal of highly available networked services.



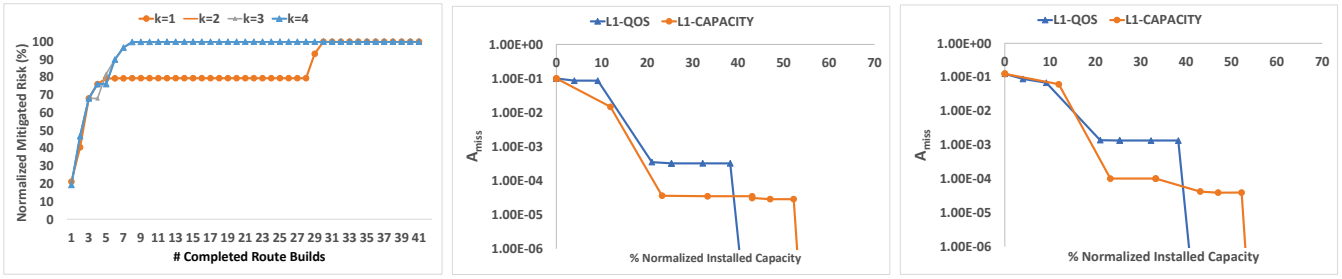
(a) (b) (c)

Fig. 2: (a) Normalized cumulative capacity as a function of adjacency rank (b) L3-CAPACITY rank vs L3-QOS rank (c)A_{miss} as function of normalized cumulative capacity for AF1 demands



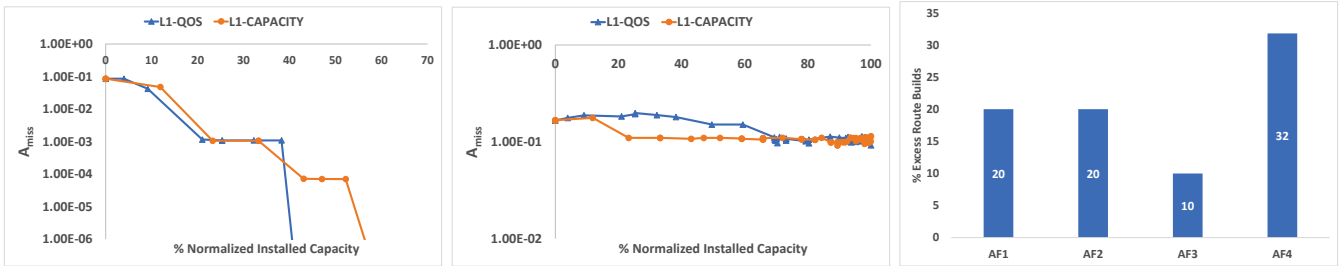
(a) (b) (c)

Fig. 3: A_{miss} as function of normalized cumulative capacity for $L3^p$ (a) AF2 demands (b) AF3 demands (c) Number of days until every demand in a specified set of classes reach their performance guarantees



(a) (b) (c)

Fig. 4: (a) Normalized risk evolution for different values of κ while using L1-QOS. A_{miss} as function of normalized cumulative capacity for $L1^p$ approaches (b) AF1 demands (c) AF2 demands



(a) (b) (c)

Fig. 5: A_{miss} as function of normalized cumulative capacity for (a) AF3 demands (b) AF4 demands. (c) Percentage excess route builds required for different cos types for L1-CAPACITY approach with respect to L1-QOS approach

REFERENCES

[1] Mikel Jimenez Fernandez, Henry Kwok. "Building express backbone: Facebook's new long-haul network", 2017.

[2] Firoiu et al, "Theories and models for internet quality of service.", In Proceedings of the IEEE, 2002.

[3] Zhong et al, A social network under social distancing: Risk-driven backbone management during covid-19 and beyond. USENIX NSDI, 2021.