Probabilistic Analysis of Overload-Free Property for Critical Traffic

Zhiyun Tang*, Yahui Li*, Ke Ruan†, Han Zhang*, Yongqing Zhu†, Yingjun Ye†, Xia Yin*, Jilong Wang*
*Tsinghua University, China {Email: tzy22@mails.tsinghua.edu.cn, liyahu@tsinghua.edu.cn}
†Research Institute of China Telecom Corporation, China {Email: yeyj5@chinatelecom.cn}

***Abstract*—Network structures are sophisticated and, hence, vulnerable to errors. The link failures and traffic load fluctuations lead to complexity in network states. Different failure scenarios can result in varying network traffic distribution patterns. Meanwhile, the load on links within the same failure scenario dynamically changes with fluctuations in traffic. Network administrators are particularly concerned about whether links along the paths traversed by critical traffic are overload-free guaranteed when link failures occur. Yet, no attention was ever paid to overload-free property analysis for critical traffic. We propose Offaela, an efficient and accurate probabilistic analysis framework that verifies an overload-free property for critical traffic. We prudently formulate the problem and prove its computational hardness, then storm this fortification by proffering a failure scenario merging algorithm and adopting a randomized approximation method. Evaluations on real networks show that Offaela outperforms the state-of-the-art solution by 4.83x and can provide availability analysis assistance in practical scenarios such as verifying the impact of traffic demand scale growth on the network's overload-free property and identifying vulnerable failure scenarios.**

I. Introduction

As the intranet becomes more extensively integrated into various sectors such as business, healthcare, education, etc, and serves tens of billions of customers concurrently, ensuring customer service quality has increasingly become a focal point for internet service providers (ISPs). Providers often enhance customer service quality by improving network reliability and availability and tailoring network solutions for unique customers [1][2]. When customers are deciding from which provider they wish to get service, network availability often becomes a crucial reference point.

For network administrators, certain critical customer traffic is of paramount importance due to commercial requirements. Quite a few network administrators have assigned priorities on traffic demands [3]. Administrators are obligated to ensure that the overload-free property on the links along the paths traversed by critical traffic is verified to the fullest extent. It is significant to quantitatively analyze the availability of critical customer traffic for ISPs. In recent years, numerous efforts [4]-[13] have been made with the aim of assessing network availability to assist administrators in operating the network. Data plane verification tools like Anteater [4] and VeriFlow [5] proposed a Boolean variable to assess network traffic reachability, but only single-flow traffic is taken into consideration, and simply classifying the problem as a yes-or-no question is extremely inadequate. Control plane verification tools like Batfish [6] and Minesweeper [7] detect reachability property violations proactively by deriving data plane models from configuration files. These works are neither competent for simulating traffic load distribution nor for probabilistically analyzing failure scenarios. Jingubang [8] analyzes traffic load property using a traffic distribution graph. However, it cannot probabilistically reason about failures. Netdice [12] improved upon these works by evaluating availability under probability failure scenarios; however, Netdice's solution is not practically multi-flow-adaptive. Pita [13] probabilistically analyses traffic load property, yet its brute-force method for failure scenarios limits its applicability to tiny networks. Besides, no known work is capable of assessing whether links along the paths traversed by critical traffic are overload-free guaranteed when link failures occur. In this paper, we propose an probabilistic analysis framework that verifies an overload-free property for critical traffic in ISP's backbone networks.

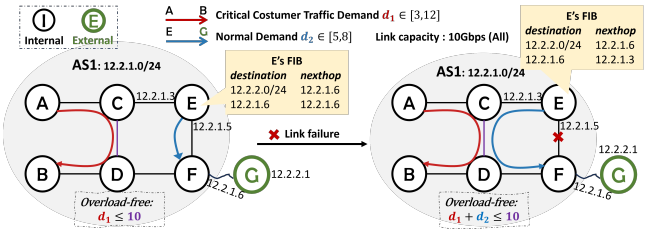

Fig. 1: Traffic load distribution under different failure scenarios

Verifying overload-free property for critical traffic is sophisticated due to diverse failure scenarios and dynamic traffic demands. As exemplified in Fig. 1, suppose there are two traffic demands in the network: d_1 is the critical customer traffic demand, and d_2 is a normal demand. Assuming that all link capacities are 10 Gbps. traffics A→B and E→G do not share any link when no failure happens. However, when one of the links fails, traffic E→G is rerouted and thus shares a common link with traffic A→B. In this case, determining whether critical customer traffic experiences congestion necessitates the extensive consideration of traffic demands d_1 and d_2.

The toy example briefly clarifies our overload-free property verification problem. We face a calculation challenge in this verification process. While verifying the overload-free property on a specific single traffic pattern is relatively simple, for works on network simulations and emulations can transform this problem into a tractable one without effort, verifying the property over all possible traffic patterns, whose

union is a high dimensional continuous space, is arduous. To conquer this intractability, we resort to a stochastic approximation method [15] based on Multiphase Markov Chain Monte Carlo (Multiphase MCMC) to acquire an approximate solution without loss of accuracy compared to the state-of-the-art estimation solutions. Although the stochastic method can be successfully applied to theoretically overcome the intractability difficulty by reducing the problem into one that is applicable to this method, it still takes tens to thousands of minutes to quantitatively analyze a network's overload-free property; hence, the computational hardness persistently perplexes us.

Our key idea is based on the observation that the forwarding behavior of some flows remains consistent in many failure scenarios, leading to similarities in the overload-free property across these scenarios. This fact provides an insight into merging analysis processes in different failure scenarios, allowing for a substantial reduction in sampling and computation processes. We recognize the potential to optimize the analysis process of overload-free property based on the specific characteristics of routing protocols, enabling rapid and accurate identification of all optimizable scenarios.

The above gives us a glimpse of a more delicate constructive framework for verifying network availability of critical customer traffic, Offaela. In Section II, we formally formulate the problem. In Section III, we elaborate on our solution, Offaela, and how it effectively solves this problem. In Section IV, we evaluate Offaela's performance.

We make the following contributions.

- To the best of our knowledge, we are the first to propose a probabilistic verification of a critical-traffic-related property, namely CCT-overload-free availability.
- We propose an efficient and accurate probabilistic analysis framework, Offaela, to calculate the value of CCT-overload-free availability. We proffer a new concept *Critical Endnode Matrix* with prudent inference to reduce the computational overhead and adopt a stochastic method based on Multiphase MCMC for approximation.
- We evaluate the efficiency and effectiveness of Offaela by conducting experiments on real networks.

II. PROBLEM DEFINITION

A. Model

Suppose (V, E) is the network topology in which V stands for routers or nodes in the network, and E represents direct IP links. Each link $e \in E$ has a maximum link speed, which we generally call capacity, and a weight number suggests its cost, which is useful when calculating the shortest paths. Let us use $C_e \geq 0$ and $W_e \geq 0$ to represent the capacity and weight, respectively. D is the set of traffic flows. For each demand $d \in D$ we have an upper bound u_d, correspondingly a lower bound l_d, and a continuous random variable x_d uniformly distributed in range $[l_d, u_d]$ which is the traffic volume of demand d. H is a given set of Critical-Customer-Traffic demands, $H \subseteq D$. X is the traffic volume set, for each $x_d \in X, d \in D$, x_d is the traffic volume of demand d.

B. Problem Formulation

Now we begin to give some definitions related to Critical-Customer-Traffic, CCT for short.

Definition 1. ***CCT-related links*** Υ_f.

Given a failure scenario $f \subseteq E$, which is exactly the set of failed links under this failure scenario. Υ_f is defined as:

$$\Upsilon_f = \{e \in E | \ \exists h \in H, \ s.t. \ \gamma_{h,e}^f = 1\} \tag{1}$$

Here $\gamma_{h,e}^f$ is an indicator that is equal to 1 when traffic demand h traverses through link e under failure scenario f and 0 otherwise.

Definition 2. ***CCT-related demands*** Θ_f.

Given a failure scenario $f \subseteq E$, Θ_f is defined as:

$$\Theta_f = \{d \in D | \ \exists v \in \Upsilon_f, \ s.t. \ \gamma_{d,v}^f = 1\} \tag{2}$$

Θ_f is the set of demands that at least share one common link with one of the critical demands in H under scenario f.

Definition 3. ***CCT-overload-free Lebesgue function*** $\varphi_f(x)$.

Given a failure scenario $f \subseteq E$. x is an input vector of demands $x = [x_{d_1}, x_{d_2}, ..., x_{d_{|\Theta_f|}}]$, where $d_i \in \Theta_f$ for i in $\{1, 2, ..., |\Theta_f|\}$ and $x_{d_i} \in [l_{d_i}, u_{d_i}]$. $\varphi_f(x)$ is defined as:

$$\varphi_f(x) = \begin{cases} 1, & \sum_{d \in \Theta_f} x_d \cdot \gamma_{d,v}^f \leq C_v, \forall v \in \Upsilon_f \\ 0, & otherwise \end{cases} \tag{3}$$

Definition 4. ***CCT-overload-free probability*** ρ_f.

Given a failure scenario $f \subseteq E$, ρ_f is defined as:

$$\rho_f = \frac{\int_{R^{|\Theta_f|}} \varphi_f(x) dx}{\prod_{d \in \Theta_f} (u_d - l_d)} \tag{4}$$

Here $R^{|\Theta_f|}$ is $|\Theta_f|$-dimensional real number set. ρ_f is the ratio of traffic loads that cause no congestion on CCT-related links to the total variation range of traffic loads.

Definition 5. ***CCT-overload-free availability*** Ω.

Given a set of failure scenarios F, where $\sum_{f \in F} Pr(f) = 1$. Ω is defined as follows:

$$\Omega = \sum_{f \in F} \rho_f Pr(f) \tag{5}$$

Failure scenario set F and its corresponding failure probabilities could either be given manually or generated by specifying the maximum number of link failures.

C. Challenges and Solutions

Due to the **complexity and diversity of network states**, the flows sharing links with critical flows vary under different failure scenarios. This leads to a significant amount of intricate computations. We propose the concept of ***Critical Endnode Matrix***, which can assist in optimizing this process in III.B.

Furthermore, the **multitude of failure scenarios** complicates the computation. Estimating reliability in each failure scenario then multiplying it by the probability of the failure scenario and summing over all scenarios is computationally

intensive. Although the forwarding behavior consistency lights our way of saving computation costs, the locating of similar failure scenarios is not straight-forward: one needs to compute every link that any traffic demand traverses on all failure scenarios and then compare the forwarding behavior of demands between every pair of failure scenarios. On the contrary, ***Critical Endnode Matrix*** can spare us the trouble. By following the approach described in Section III.C, we **merge failure scenarios**, reducing the number of reliability estimations required. Experiment results show that this leads to a significant improvement in computation speed, up to 4.83x.

The third challenge falls on the calculation process of Ω value, which is defined in II.B; we face a computational challenge since computing the value of CCT-overload-free probability ρ_f is a **#P-hard problem**. This conclusion is trivial for φ_f is a boolean formula and integrating it over $R^{|\Theta_f|}$ is a #SAT problem. We solve this challenge by **reducing this problem into the hyper-polyhedron volume computation problem**, which has practical polynomial time approximation solutions with less than 10% loss of accuracy [15] in the method of Multiphase MCMC.

III. Methodology

We provide an overview of our solution in III.A, then introduce two insights that can help accelerate the computation process in III.B. We state our principle of merging failure scenarios in III.C and present the implementation in III.D. Due to space limitations, we published the proofs of all theorems presented in this section on GitHub [1].

A. Overview

We take network topology (V, E), link capacity set $\{C_e|e \in E\}$, weight set $\{W_e|e \in E\}$ and a demand set D as input, as well as the lower and upper bounds of each demand $\{l_d|d \in D\}$, $\{u_d|d \in D\}$. Concurrently we have a given set of failure scenarios F with corresponding probability $\{Pr(f)|f \in F\}$.

To compute the value of CCT-overload-free availability Ω, we need to know to which links critical customer traffic was distributed under every failure scenario, as well as every traffic that traverses through those links and how their load is balanced. In other words, we are required to compute the CCT-related links Υ_f, demands Θ_f, and the indicator function set $\gamma^f_{h_i,e_j}, \forall h_i \in \Upsilon_f, \forall e_j \in \Theta_f$ defined in II.B. Instead of brute-force, we design a more efficient algorithm to compute them by first introducing *Critical Endnode Matrix* and two insights in III.B.

Next we approximate the value of CCT-overload-free probability ρ_f in polynomial time by means of Multiphase MCMC. Once we know the value probability ρ_f for all failure scenarios $f \in F$, we can calculate the availability Ω by definition; thus, the ultimate goal is achieved. The calculation process of CCT-overload-free probability ρ_f in some failure scenarios can be merged under certain conditions. Since this process contains mountains of random points sampling and validating, which contribute to the majority of the runtime, merging failure scenarios can avoid recalculations of CCT-overload-free probabilities, which take up massive computation time. We propose an efficient method by first demonstrating under what conditions scenarios can be merged in III.C, then put this conclusion into practical implementation in III.D.

[1] https://github.com/salazar1117/Offaela

B. Critical Endnode Matrix

In this section we introduce *Critical Endnode Matrix*, the key idea of this work. We first give the definition of *Critical Endnode Matrix* in Definition 8, then introduce two insights in Theorem 1 and 2 to demonstrate how *Critical Endnode Matrix* efficiently assist us in speeding up our availability analysis.

Definition 6. ***node pair hash function*** ψ
Suppose $V = \{v_1, v_2, ..., v_{|V|}\}$, $\psi : V \times V \to \mathbb{N}^+$:

$$\psi(v_i, v_j) = (i-1) \times |V| + j, \quad \forall i, j \in \{1, 2, ..., |V|\}$$

Node that ψ is a reversible function and its input is ordered.

Definition 7. ***demand-CCT hash function*** $\{\Psi^h_f\}$
Given failure scenario f and a CCT demand $h \in H$, define $\Psi^h_f : D \to \mathbb{N}^+$:

$$\Psi^h_f(d) = \begin{cases} \psi(g_f(d)), & \mathcal{L}^f_d \cap \mathcal{L}^f_h \neq \emptyset, \\ -1, & otherwise \end{cases}, \forall d \in D$$

Here $g_f(d)$ is a function that outputs an ordered pair of nodes which are sequentially the first and last node in $\mathcal{N}^h_f$ that d passes by. $\mathcal{N}^h_f$ is the set of nodes demand h goes through. $\mathcal{L}^f_d$ is the set of links d traverses through under scenario f.

Definition 8. ***Critical Endnode Matrix*** ST
Given the conditions in Definition 7, define ST matrix:

$$ST[i][j] = \Psi^h_f(d_{i,j}), \quad \forall v_i, v_j \in V$$

Here $d_{i,j}$ represents the traffic demand from v_i to v_j.

By definition, *Critical Endnode Matrix* logs the first and last node in $\mathcal{N}_f$ that each demand d passes by. We introduce this new concept that can help unravel the mergeable scenarios by analyzing costs on links. Our solution is adaptive to standard ISP backbone networks because most IGP routing protocols such as OSPF, RIP and ISIS, select paths basing on cost.

Without loss of generality, we let $|H| = 1$, H being the critical-customer-traffic set. If $|H| > 1$, we only need to make an OR logic of all homologous indicators $\gamma^f_{v,\theta} = \bigvee_{h \in H} \gamma^f_{v,\theta,h}$, where $\gamma^f_{v,\theta,h}$ is the indicator considering only CCT-demand h. Now we propose two theorems.

Theorem 1. *Let $\mathcal{N}_f$ be the set of nodes demands in H traverse through. Let $\mathcal{L}_{a,b}(a, b \in V)$ be the set of links that demand sources from a and ends at b traverses through. $\forall v \in V \backslash \mathcal{N}_f$, define function $\phi : V \backslash \mathcal{N}_f \to \mathcal{N}_f$, $\phi(v)$ is the nearest node in $\mathcal{N}_f$ to v, which means $\sum_{e \in \mathcal{L}_{v,\phi(v)}} W_e \leq \sum_{e \in \mathcal{L}_{v,u}} W_e, \forall u \in \mathcal{N}_f$. Define a family of sets $\{\mathcal{S}_u\}, u \in \mathcal{N}_f$ as follow:*

$$\mathcal{S}_u \triangleq \{v \in V \backslash \mathcal{N}_f | \phi(v) = u\} \tag{6}$$

Then $\forall a, b \in \mathcal{S}_u$, $L_{a,b} \cap \Upsilon_f = \emptyset, \forall u \in \mathcal{N}_f$.

Theorem 1 reveals a potential to accelerate the calculation process of acquiring Θ_f. For two nodes whose nearest neighbor in $\mathcal{N}_f$ are identical, the shortest path between those two nodes shall never share same member with Υ_f.

Theorem 2. *$\forall d \in \Theta_f$, $a_d, b_d \in \mathcal{N}_f$ being the first and last node in $\mathcal{N}_f$ that d passes by, then $\mathcal{L}_d^f \cap \Upsilon_f = L_{a_d,b_d}$. Here $\mathcal{N}_f$ and L_{a_d,b_d} is defined as in Theorem 1. $\mathcal{L}_d^f$ is the set of links d traverses through under scenario f.*

Theorem 2 indicates that to calculate matrix Γ, instead of enumerating every link each demand traverses through and verifying whether it belongs to Υ_f, we only need to know the first and last node in $\mathcal{N}_f$ that each demand $d \in D$ passes by, which is recorded in *Critical Endnode Matrix* ST.

C. Failure Scenario Mergence

If $\exists f_1, f_2$ with same relating CCT-related demands and links, and $\gamma_{\theta_j,v_i}^{f_1} = \gamma_{\theta_j,v_i}^{f_2}, \forall v_i \in \Upsilon_f, \forall \theta_j \in \Theta_f$, then the computation phase of CCT-overload-free probability ρ_f for these two failure scenarios can be merged. This observation can lead to a theorem as follows:

Theorem 3. *For $f_1, f_2 \in F$, if $\Psi_{f_1}^h = \Psi_{f_2}^h, \forall h \in H$, then $\rho_{f_1} = \rho_{f_2}$.*

Theorem 3 unveils a fact that, for two given failure scenarios. If both the first and last node in $\mathcal{L}_h$ that d, $\forall h \in H, \forall d \in D$ passes by in the two failure scenarios are the same, $\mathcal{L}_h$ being the set of nodes h goes by, then the CCT-overload-free availability values of the two failure scenario are equal. Thus, the value is required to be calculated only once, saving a lot of calculation labor from avoidance of recomputing.

D. Implementation

Algorithm 1 illustrates the overall process of Offaela. G stands for the weighted adjacency matrix of the current topology (under failure scenario f). For each f, we compute the *Critical Endnode Matrix* ST. Each element $ST[i][j]$ representing the hash value of the pair of nodes that are the first and last node in $\mathcal{N}_f$ that demand from node v_i to node v_j passes by as defined in III.B, where $\mathcal{N}_f$ is the set of nodes the critical demands pass by in f. If ST is equivalent to a *Critical Endnode Matrix* of another failure scenario f', then $\rho_f = \rho_{f'}$, thus we can skip the random approximation phase for f. Otherwise, we finish the procedure of MCMC for f in line 21 and record the value of *Critical Endnode Matrix* as well as ρ_f.

Algorithm 2 shows the calculation process of *Critical Endnode Matrix* ST. Matrix NT is the output matrix of Dijkstra algorithm. $NT[i][j]$ stands for the next hop of node traffic demand from v_i to v_j passes by. Matrix DS is another output matrix of Dijkstra algorithm with each value meaning the distance between the corresponding node pair, equivalent to ∞ if the demand between them is unreachable. For each node v_i, we find node v_{j_i}, such that v_{j_i} is the node closest to v_i in $\mathcal{N}_f$. Then for each node v in $\mathcal{N}_f$, we have a set of nodes whose nearest node in $\mathcal{N}_f$ is v. With Theorem 1,

Algorithm 1: Offaela: Computing availability Ω

input : network topology $(V = \{v_1, v_2, ..., v_{|V|}\}, E)$ with weight and capacity for each link, set of Critical-Customer-traffic demands $H = \{h_1, h_2, ..., h_{|H|}\}$, failure scenarios F with probability $Pr(f)$ for $f \in F$
output : Approximation of availability Ω

1 $\Omega \leftarrow 0$, $P \leftarrow \emptyset$; // P is a set of arrays
2 **for** $f \in F$ **do**
3 Initialize Q as an empty array;
4 $flag \leftarrow 0$; // Q is an array of matrices
5 $G \leftarrow$ weighted adjacency matrix of $(V, E \backslash f)$;
6 **for** $i := 1$ **to** $|H|$ **do**
7 **if** h_i *is unreachable in* $(V, E \backslash f)$ **then**
8 $flag \leftarrow 1$; **break**;
9 Calculate $\mathcal{L}_h$, the set of nodes h_i passes by;
10 $K \leftarrow \{j | v_j \in \mathcal{L}_h\}$;
11 $ST \leftarrow$ Calculate_ST(G, h_i, K); ▷ Algorithm 2
12 Add ST into array Q;
13 **if** $\{flag = 1\}$: // $\rho_f = 0$ if any critical customer
14 **continue**; // demand is unreachable
15 **if** $Q \in P$ **then**
16 $\Omega \leftarrow \Omega + \zeta_Q \cdot Pr(f)$; **continue**;
17 Calculate demand set Θ_f and matrix Γ;
18 $\rho_f \leftarrow$ Multiphase_MCMC(Θ_f, Γ); // see appendix C[1]
19 $\Omega \leftarrow \Omega + \rho_f \cdot Pr(f)$;
20 $\zeta_Q \leftarrow \rho_f$; Add Q into P;

Algorithm 2: Calculate_ST

input : G, h, K
output : ST

1 $NT, DS \leftarrow$ Dijkstra(G); $m \leftarrow |K|$;
2 Generate a zero matrix ST of size $n \times n$;
3 Generate an array M of m empty sets;
4 **for** $i = 0$ **to** $n - 1$ **do**
5 $j \leftarrow j_i$, ***s.t.*** $DS[i][j_i] = min\{DS[i][j] | j \in K\}$;
6 Add i into set $M[j]$; // Find v_i's closest node in $\mathcal{N}_f$
7 **for** $i, j = 0$ **to** $n - 1$ **do**
8 **if** $\exists r$, ***s.t.*** $i, j \in M[r]$ **then**
9 $ST[i][j] \leftarrow -1$; // Demand from v_i to v_j is irrelevant
10 **for** $i, j = 0$ **to** $n - 1$ **do**
11 **if** $ST[i][j] = 0$ **then**
12 DFS_critical_pair(K, i, j); ▷ Algorithm 3

Algorithm 3: DFS_critical_pair

input : K, a, b // Suppose d is the demand from v_a to v_b
output : $ST[a][b]$ // The first and last node in $\mathcal{N}_f$ d passes by

1 **if** $ST[a][b] \neq 0$ **then**
2 ***return*** $ST[a][b]$; // Base case: ST value for d is known
3 **if** $a \in K$ ***and*** $b \in K$ **then**
4 $ST[a][b] \leftarrow \psi(v_a, v_b)$; // Base case: $v_a, v_b \in \mathcal{N}_f$
5 **return** $ST[a][b]$
6 $a_1 \leftarrow NT[a][b]$ if $a \notin K$, otherwise a;
7 $b_1 \leftarrow NT[b][a]$ if $b \notin K$, otherwise b;
8 $tmp \leftarrow$ DFS_critical_pair(K, a_1, b_1); // $ST[a][b] = ST[a_1][b_1]$
9 $ST[a][b] \leftarrow ST[a_1][b] \leftarrow ST[a][b_1] \leftarrow tmp$;
10 $ST[a][b] \leftarrow$ ACLcheck$(a, b, ST[a][b])$;
11 **return** $ST[a][b]$

we can infer the demand between those nodes in the same set would never share common links with critical demands. Hence we give value -1 to the corresponding ST element, marking this demand as an insignificant demand since its load would never affect a critical demand in this case. This saves at least $\frac{(n-m)(n-2m)}{mn(n-1)}$ of calculation burden, m being the total number

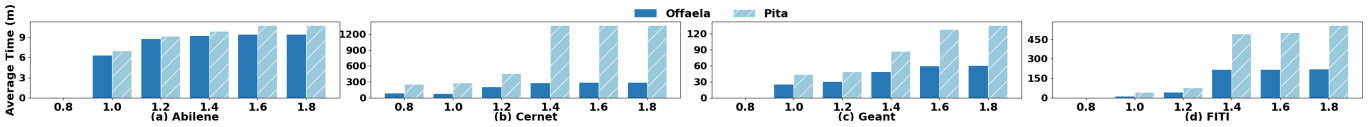

Fig. 2: Average runtime of Offaela and Pita at fixed demand scale upper bound. The x-axis represents the demand scale.

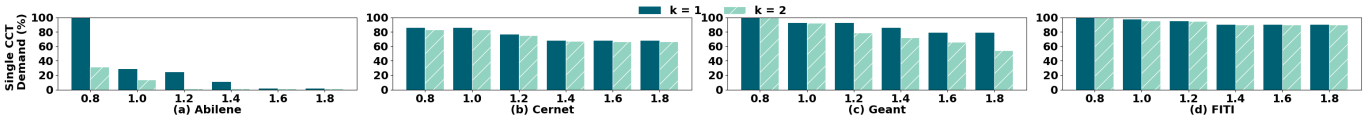

Fig. 3: Percentage of single Critical-Customer-Traffic demand that Offaela runs without MCMC process at fixed demand scale upper bound and k value

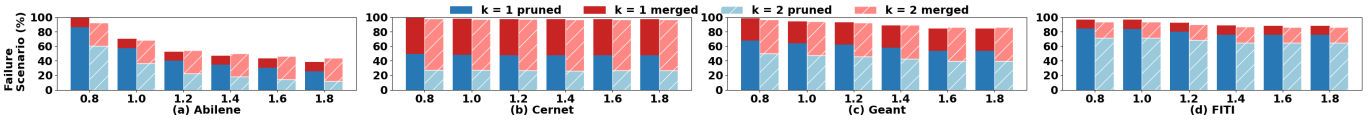

Fig. 4: Percentage of failure scenarios merged and pruned on average at fixed demand scale upper bound and k value

Network	Nodes	Edges	Demands	# Failure scenarios
Abilene	12	15	132	121
Cernet	40	58	1560	1712
Geant	27	38	702	742
FITI	40	44	1560	991

TABLE I: Topologies for evaluation

Demand Scale	0.8		1.0		1.2		1.4		1.6		1.8	
Failure Scenario	k=1	k=2	k=1	k=2	k=1	k=2	k=1	k=2	k=1	k=2	k=1	k=2
Network	**Average Availability (%)**											
Abilene	99.854	99.171	98.607	82.430	98.499	80.132	79.724	70.716	64.996	51.019	62.793	55.821
Cernet	99.776	98.779	99.732	95.948	96.763	95.688	96.718	95.555	96.701	95.614	96.518	95.595
Geant	99.513	96.144	96.965	93.537	95.854	93.211	92.842	88.715	89.492	88.320	89.518	89.561
FITI	99.533	98.484	99.912	97.519	95.380	93.912	94.191	93.514	94.031	92.921	93.910	92.732

TABLE II: Average availability at fixed demand scale upper bound and k value

of nodes critical demands pass by, reducing the computational complexity of this phase from $O(n^2)$ to $O(\frac{m-1}{m}n^2)$. Thus in line 20 of Algorithm 1 Θ_f is the set of demands whose corresponding ST value does not equal to -1, and Γ can be deduced by the ST matrices as explained in section III.B.

Algorithm 3 is a depth-first search (DFS) for efficiently computing the value of ST. Given node pair v_a, v_b, if both nodes belong to $\mathcal{N}_f$, we already have $\Psi(d_{a,b})$ as the ST value. Otherwise, suppose $v_a \notin \mathcal{N}_f$, we define v_{a_1} as the next hop of demand from v_a to v_b, then it is trivial to infer that the first and last node in $\mathcal{N}_f$ that demand from node v_a to node v_b passes by is the same for demand from node v_{a_1} to node v_b. Before returning the value, we perform an access control list (ACL) check on the traffic. If there are corresponding ACL entries on the transit nodes that prevent the traffic from being delivered successfully, we will modify the value of ST based on the actual behavior of the traffic forwarding. For each demand, our algorithm only needs to compute its next hop once; thus, the amortized complexity to compute Γ for each demand d goes down from $O(|\mathcal{L}_d^f|)$ to $O(1)$. $\mathcal{L}_d^f$ is the set of links d travels through. In summary, the computational complexity for each failure scenario is $O(\frac{m-1}{m}n^2)$, while brute-force executes $\sum_{d\in D}\sum_{e\in E}\gamma_{d,e}^f$ times of operations, making the complexity of the latter larger than $O(n^2)$ since at least one link is passed through by each traffic.

IV. Evaluation

In this section, we evaluate the efficiency of our framework, Offaela. Our experimental results focus on the following questions: (i) How efficient is Offaela? (ii) Does Offaela demonstrate scalability? (iii) How does our optimization enhance Offaela's performance? (iv) How can Offaela be utilized to identify vulnerable failure scenarios?

A. Experimental Setting

Setup. We implement Offaela in Python with the open-source implementation of VolEsti [15] as the basis for our hyper-polyhedron volume estimation algorithm. All experiments run on a computer with 48 CPU cores at 3.2GHz and 64GB RAM.

Topologies and demand scaling. We use three network topologies from Topology Zoo [14]: Abilene, Cernet, Geant, and FITI, as shown in TABLE 1. We utilize gravity model [16] to generate traffic matrices such that without any link failure, the resulting link utilization is between 0.6 and 1.8.

Deriving failure scenarios. To evaluate the robustness and tractability of our model, we consider scenarios in which at most 2 links fail simultaneously ($k = 1, 2$), with link failures independent and identically distributed. Presume the failure probability of any link is 0.001 and scale up the probability of each failure scenario until $\sum_{f\in F} Pr(f) = 1$ is satisfied.

B. Verification Performance

We now evaluate the performance of Offaela. For comparison, we implement the state-of-the-art solution, Pita [13].

Running time. We compute CCT-overload-free availability Ω for each single critical traffic at demand scale ranging from 0.6 to 1.8. We report the average running time of Offaela and Pita at fixed demand scale upper bound in Fig. 2. We executed 20, 533, 200 and 705 single CCT demand scenarios

for Abilene, Cernet, Geant and FITI, respectively. Offaela outperforms Pita on all four topologies. On average, with a demand scale upper bound of 1.8, the runtime of Offaela compared to Pita improved by 1.15x, 4.83x, 2.29x, and 2.57x on Abilene, Cernet, Geant, and FITI, respectively, showing the superiority of Offaela over Pita.

In summary, the answer to **the first question** is that **Offaela outperforms state-of-the-art solution** in analyzing overload-free property for critical traffic in the aspect of runtime.

Scalability. To examine the scalability of our solution, we scale the demand up and record the average running time. Fig. 2 elucidates average runtime with respect to the demand scale at $k = 1$. Despite the positive correlation between demand scale and running time, the growth rate of average runtime slows down as the demand scale increases. This phenomenon arises from the fact that most of the evaluation time is allocated to the MCMC process. Moreover, with the increase in demand scale, the rate of growth in the number of failure scenarios susceptible to overload, which leads to the MCMC process, slows down as depicted in Fig. 3. Consequently, our approach exhibits a gradual increase in runtime with demand scale expansion, rendering it suitable for a wide range of usage scenarios and demonstrating high scalability.

In summary, the answer to **the second question** is that Offaela is **adaptive to the scaling up of demand scale**.

C. Offaela Optimization

In this subsection we inspect two main optimizations applied in Offaela and evaluate their effectiveness.

Merge and prune failure scenarios. We illustrate the average percentage of failure scenarios merged or pruned in Fig. 4. Merging indicates the scenarios where the calculation of CCT-overload-free probability ρ_f is bypassed, as elucidated in III.C. Pruning refers to scenarios where maximum traffic load does not cause congestion or minimum traffic does. Both merging and pruning can optimize Offaela's analysis by eliminating the need to recalculate ρ_f.

The percentages of scenarios merged are 31.86%, 70.45%, 46.57% and 21.67% for Abilene, Cernet, Geant and FITI, respectively at $k = 2$. Besides, it can be noticed that as k value and demand scale grows larger, the percentage of scenarios pruned goes down on all topologies except Cernet. Also, the total percentage of pruned and merged failure scenarios exceeds 80% in all traffic schemes and topologies except Abilene. This indicates that many scenarios are spared from the MCMC process, thus saving time.

In conclusion, the answer to **the third question** is that **failure scenario merging and pruning** are significantly improving the performance of Offaela.

D. Analyzing Real Networks

The results of the analysis on real networks are presented in this subsection. We demonstrate how Offaela can be applied to identify vulnerable failure scenarios.

TABLE II records the average availability at fixed demand scale upper bound and k value. When the demand scale is lower than 1.2, the average availabilities of all four topologies are above 80%. We also analyze and illustrate the overload-free probability of Abilene under each single-link failure with an upper demand scale of 1.8 in Fig. 5, considering traffic demand from node 1 to node 5 as the CCT demand. When link 2-6 goes down, the availability becomes 84.747%. The failure of link 2-5(6-7) has a devastating impact on the network; the availability decreased to near zero because the failure of either would make the other link a cut in the topology graph.

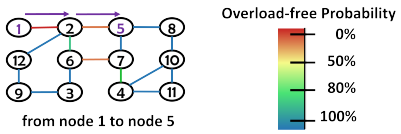

Fig. 5: Overload-free probabilities under each single-link failure in Abilene

Thus, the answer to **the fourth question** is that **Offaela is capable of identifying failure scenarios that are vulnerable to congestion**.

V. Conclusion

We presented Offaela, an efficient and accurate probabilistic analysis tool for verifying the overload-free property of critical traffic in ISPs. We established a refined model and formulated a property verification problem addressed by Offaela. Utilizing a randomized approximation method and a novel inference algorithm, Offaela significantly reduces calculation overhead. This tool proves invaluable for network management and is currently being deployed in the network operation center of China Telecom.

Appendix A

Proof of Thereom 1:

Proof. Without loss of generality, $\forall d \in D$ there exists only one shortest path in the current network topology for any typology can be transformed into a topology with this property, check for more details in Appendix C. If $\exists a, b \in \mathcal{S}_u, u \in \mathcal{N}_f$, s.t. $L_{a,b} \cap \Upsilon_f \neq \emptyset$. Then $\phi(a) = \phi(b)$ holds. Let $\mathcal{Q}_{a,b} = \{q_1, q_2, ..., q_k\}, q_i \in E, i \in \{1, 2, ..., k\}$ be the ordered sequence of links demand from a to b traverses through. $e_1 \triangleq min\{i | q_i \in \Upsilon_f\}$, $e_2 \triangleq max\{i | q_i \in \Upsilon_f\}$. It is trivial that $e_1 > 1$ and $e_2 < k$. By definition we can infer:

$$\sum_{e \in L_{a,\phi(a)}} W_e + \sum_{e \in L_{\phi(a),b}} W_e \leq \sum_{\substack{e \in q_i \\ 1 \leq i \leq e_1 - 1}} W_e + \sum_{\substack{e \in q_k \\ e_2 + 1 \leq j \leq k}} W_e \leq \sum_{e \in L_{a,b}} W_e \tag{7}$$

It is suffice to illustrate that $L_{a,\phi(a)} \cup L_{\phi(a),b}$ contains the shortest path between a and b, thus lead to contradiction for $(L_{a,\phi(a)} \cup L_{\phi(a),b}) \cap \Upsilon_f = \emptyset$ so there exists two different shortest paths between a and b. □

Lemma 1. *$\forall d \in D$, suppose the ordered sequence of links d traverses through under scenario f is $\mathcal{Q}_d^f = \{q_1, q_2, ..., q_k\}, q_i \in E, i \in \{1, 2, ..., k\}$ and the set of nodes*

d traverses through under scenario f is $\mathcal{N}_f$. The $\forall v_1, v_2 \in \mathcal{N}_f, v_1 \neq v_2$, suppose $d' \in D$ is the demand start from v_1 and ends at v_2, then $\mathcal{Q}^f_{d'}$ is a sub-sequence of $\mathcal{Q}^f_d$. That is to say, $\exists 1 \leq i < j \leq k$, s.t. $\mathcal{Q}^f_{d'} = \{q_i, q_{i+1}, ..., q_j\}$.

Proof. By definition we can infer that $\mathcal{Q}^f_{d'}$ is the shortest path between source node v_1 and end node v_2. If v_1 is not an endpoint of d, $\exists i$ s.t. v_1 is the endpoint of q_i and q_{i+1}, else set i to 0 if v_1 is the source point of d and to k if v_1 is the destination point. Similar to this, we obtain a j doing the same to v_2. Suppose $i < j$. Then $\mathcal{Q} = \{q_{i+1}, q_{i+2}, ..., q_j\}$ is a sub-sequence of $\mathcal{Q}^f_d$ starts from v_1 and ends at v_2. If this sub-sequence is identical to $\mathcal{Q}^f_{d'}$, the lemma is proved. If not, we substitute $\mathcal{Q}^f_{d'}$ for this sub-sequence and acquire a new sequence $\widetilde{\mathcal{Q}^f_{d'}}$. According to the condition, we have $\sum_{p \in \mathcal{Q}} W_p > \sum_{p \in \mathcal{Q}^f_{d'}} W_p$. Then we can infer: $\sum_{p \in \mathcal{Q}^f_d} W_p > \sum_{p \in \widetilde{\mathcal{Q}^f_{d'}}} W_p$. Which suggests $\widetilde{\mathcal{Q}^f_{d'}}$ is the shortest path for d, thus lead to contradiction. □

Now we proof Thereom 2.

Proof. Both $L_{a_d,b_d} \subseteq \Upsilon_f$ and $L_{a_d,b_d} \subseteq \mathcal{L}^f_d$ hold in the light of Lemma 1, thus $L_{a_d,b_d} \subseteq (\mathcal{L}^f_d \cap \Upsilon_f)$. Since there is no loop in a shortest path, any node in $\mathcal{L}^f_d$ is passed by only once by demand d. $\forall c \in \mathcal{L}^f_d \cap \Upsilon_f$, if $c \notin L_{a_d,b_d}$, then c is passed by d after it goes through d_a and before d_b. Let $\mathcal{M}_d = \{m_1, m_2, ..., m_l\}$ be the ordered sequence of nodes demand d passes by. Then $\exists i, j, k (1 \leq i < k < j \leq l)$, s.t. $d_a = m_i, d_b = m_j, c = m_k$. Let $\mathcal{R}_{a,b} = \{n_1, n_2, ..., n_s\}$ be the ordered sequence of nodes demand from d_a to d_b passes by. Now replace the slice $\{n_i, n_{i+1}, ..., n_j\}$ in $\mathcal{M}_d$ with sequence $\mathcal{R}_{a,b}$ we have a new node sequence $\widetilde{\mathcal{M}_d}$. Since $c \notin \mathcal{R}_{a,b}$, $\widetilde{\mathcal{M}_d}$ is not identical to $\mathcal{M}_d$. Since there is only one shortest path, the path through ,$R_{a,b}$ costs less than the one through $\{n_i, n_{i+1}, ..., n_j\}$. Thus path through $\widetilde{\mathcal{M}_d}$ costs less than the path through $\mathcal{M}_d$ which contradicts the fact that the latter is the shortest path of d. Thus c belongs to L_{a_d,b_d}, so $\mathcal{L}^f_d \cap \Upsilon_f \subseteq L_{a_d,b_d}$ holds. In conclusion $\mathcal{L}^f_d \cap \Upsilon_f = L_{a_d,b_d}$. □

APPENDIX B

If $\exists d \in D$, s.t. d has two equal-cost shortest paths, suppose $a \in V$ is the source node of d and $b \in V$ is the end node, and the set of nodes directly linked to a is $\mathcal{A}$. Then we could construct a new topology $(V', E'), V' = V \cup \{a'\}, E' = E \cup \{e'\} \cup \mathcal{U}$, where the endpoints of e' are a and a' and $\mathcal{U}$ is a set of links whose endpoints are a' and $n \in \mathcal{A}$. The weight and capacity of those new links are identical to their relevant links in the original topology, only one endpoint is a' instead of a. And we set the weight of link between a' and a be a small number $\xi (\xi < \frac{1}{10} \times min\{W_d | d \in D\})$. Then we reset the weight between a and the second hop of one of the equal-cost shortest paths demand d has to its original value plus $\frac{1}{2}\xi$. Meanwhile for the new demand set D', $\forall q \in D' \backslash D$, if the source point of q is not a' or end point not b, we set $l_q = u_q = 0$. Finally we add some condition between the demand variable of d and d' as required, where d' starts from a' and ends at b. Now that we have a new network model who behave identical to the original model without d having two equal-cost shortest paths, repeat the above process until there are no equal-cost shortest paths anymore.

APPENDIX C

Algorithm 4: Multiphase_MCMC

input : Θ_f, Γ
output : ρ_f

$b \leftarrow [C_{w_1}, C_{w_2}, ..., C_{w_{|\Upsilon_f|}}], w_i \in \Upsilon_f$;
$n \leftarrow |\Theta_f|$; $N \leftarrow 400\ n \log n$;
$P_f \leftarrow \{x | x\Gamma \leq b\}$; // P_f is the polyhedron
Compute Chebyshev ball B(c, r_{min});
Compute circumscribed ball B(c, r_{max});
$\alpha \leftarrow \lfloor n \log r_{min} \rfloor$; $\beta \leftarrow \lceil n \log r_{max} \rceil$;
$Q_\beta \leftarrow P_f \cap B(c, r_{max})$;
Generate a random point $p \in Q_\beta$; $S \leftarrow \{p\}$;
$\rho_f \leftarrow n \frac{\pi^{n/2}}{(\frac{n}{2})!}$; $w \leftarrow \lfloor 10/10 \rfloor$;
for $i := \beta$ **to** $\alpha + 1$ **do**
  $Q_{i-1} \leftarrow P_f \cap B(c, 2^{(i-1)/n})$;
  $count\_prev \leftarrow \#(S)$;
  Remove from S the points not in Q_{i-1};
  $count \leftarrow \#(S)$;
  for $j := 1$ **to** N **do**
    Generate random point $p \in Q_i$;
    if $p \in B(c, 2^{i/n})$ **then**
      $count \leftarrow count + 1$;
      Add p in S;
  $\rho_f \leftarrow \rho_f \cdot \frac{N}{count}$;
$\rho_f \leftarrow \frac{\rho_f}{\prod_{d \in \Theta_f} (u_d - l_d)}$;
return ρ_f;

REFERENCES

[1] Llopis, Joan Meseguer, et al. Minimizing latency of critical traffic through SDN. In NAS, 2016.
[2] Saha N, Bera S, et al: Traffic-aware QoS routing in software-defined IoT[J]. IEEE Transactions on Emerging Topics in Computing, 2018.
[3] Hong, Chi-Yao, et al. Achieving high utilization with software-driven WAN. In SIGCOMM, 2013.
[4] Mai, H., Khurshid, A., et al. Debugging the data plane with anteater. ACM SIGCOMM Computer Communication Review, 2011.
[5] A. Khurshid, X. Zou, et al. Veriflow: Verifying network-wide invariants in real time. In NSDI, 2013.
[6] Fogel, Ari, et al. A general approach to network configuration analysis. In NSDI, 2015.
[7] Beckett, Ryan, et al. A general approach to network configuration verification. In SIGCOMM, 2017.
[8] Li, Ruihan, et al. Reasoning about Network Traffic Load Property at Production Scale. In NSDI, 2024.
[9] Subramanian, Kausik, et al. Detecting network load violations for distributed control planes. In PLDI, 2020.
[10] Carolyn Jane Anderson, Nate Foster, et al. NetKAT: Semantic Foundations for Networks (POPL '14).
[11] Gember-Jacobson, Aaron, et al. Fast control plane analysis using an abstract representation. In SIGCOMM, 2016.
[12] Steffen, Samuel, et al. Probabilistic verification of network configurations. In SIGCOMM, 2020.
[13] Zhang, Y., Xu, H., et al. Probabilistic Analysis of Network Availability. In ICNP, 2022.
[14] "Topology zoo." http://www.topology-zoo.org/.
[15] Chalkis A, Emiris I Z, et al. A practical algorithm for volume estimation based on billiard trajectories and simulated annealing[J]. ACM Journal of Experimental Algorithmics, 2023, 28: 1-34.
[16] M. Roughan, "Simplifying the Synthesis of Internet Traffic Matrices," ACM SIGCOMM Computer Communication Review, 2005.