Peering the Edge: Enabling Low-Latency Interdomain Edge Communication via Collaborative Transmission

Yuxin Wang[†], Haoyu Liu[†], Haohao Song[†], Siyong Huang[†], Shaoxiang Qin[△], Yutong Liu[◇], Qiao Xiang[†*], Linghe Kong[◇], Geng Li[♯], Jiwu Shu[‡†], Xue Liu[△]
[†]Xiamen Key Laboratory of Intelligent Storage and Computing, School of Informatics, Xiamen University, [△]McGill University, [◇]Shanghai Jiao Tong University, [♯]Huawei Techonology, [‡]Minjiang University

***Abstract*—Enabling low-latency end-to-end interdomain communication is critical in edge networks. However, the current network architecture results in unnecessarily long communication paths, leading to high latency between devices. To address this issue, we propose a novel interdomain edge peering framework called Collie. In Collie, edge networks belonging to different network providers collaborate to forward traffic towards destinations, effectively reducing end-to-end communication latency. Importantly, Collie allows network providers to maintain their autonomy in link usage strategy. We also develop a distributed algorithm in Collie that enables edge nodes from different networks to collectively determine optimal routing and traffic assignment, ensuring low-latency delivery while respecting network policies without exposing them. We implement a prototype of Collie and extensively evaluate its performance using real-world topologies. Our results demonstrate that Collie achieves a tight approximation ratio and exhibits scalability in large interdomain edge networks.**

I. Introduction

Edge networks leverage the physical or software-based infrastructure at the network edge to migrate computational resources closer to the data generation or service delivery points. This optimization aims to enhance system performance and provide near-real-time, actionable analysis results. Various network providers have extensively deployed edge networks.

Despite significant advancements , the end-to-end interdomain communication latency is still constrained by their overall architectural design. Specifically, under a typical scenario depicted in Figure 1, the data path from device 1 to device 2 is shown in red. Such an unnecessarily long data path leads to high latency. A recent measurement study [1] identifies the high end-to-end communication latency of inter-backbone-network paths as the primary bottleneck.

Researchers have made efforts to reduce end-to-end interdomain latency in edge networks, their fundamental design principle is to *avoid sending the traffic through the backbone networks*. However, these works lack scalability [2], [3], may not handle dynamic 5G/6G environments effectively [4], [5], or require significant hardware investment and are limited in their capabilities [6], [7].

In this paper, we propose Collie, which can achieve low-latency interdomain end-to-end communication in edge networks at a low cost, using two novel design points:

* Qiao Xiang is the corresponding author.

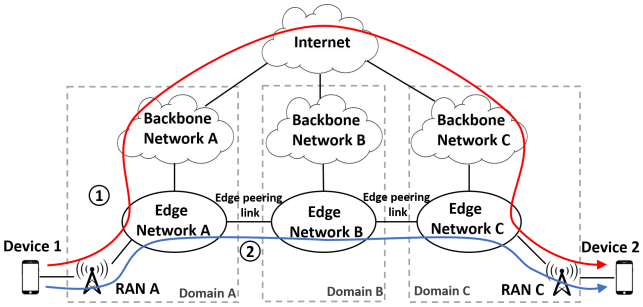


Fig. 1: A motivating example for Collie.

D1: Peering and exchanging traffic at edge networks. (§II). Different edge networks can collaborate to provide low-latency end-to-end paths for all devices, achieving scalability while requiring a minimal investment and social cost. In the scenario depicted in Figure 1, we can add edge peering links between edge networks at common locations (*e.g.*, edge clouds or internet exchange points). By routing traffic through these interconnected edge networks, device 1 can potentially transmit data to device 2 with lower latency, bypassing the backbone networks.

D2: Distributed, collaborative traffic engineering among edge networks (§III). We employ the source-based formulation to formulate the low-latency traffic engineering problem in interdomain edge networks. Then, we introduce a novel distributed algorithm that enables routers (called *edge nodes*) to collaboratively determine routing and traffic assignment along low latency paths, subject to the topologies and policies of all network providers but not exposing them. Compared with our earlier position paper [8], we provided key proof sketches of the core theorems of this distributed algorithm.

Prototype evaluation (§IV). We implement a prototype of Collie and conduct more extensive experiments than that in our position paper [8] to demonstrate the performancec and overhead of Collie. Results show that Collie provides a tight approximation ratio on minimizing the cost function, with small system overhead even in large networks.

II. Overview of Collie

In this section, we present the basic definitions and the workflow of Collie, and then formulate the optimal low-latency routing problem in Collie.

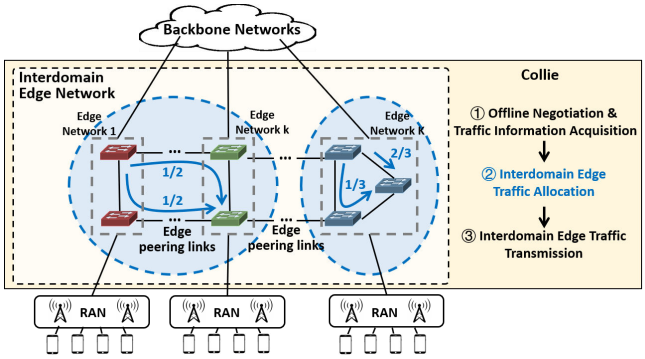


Fig. 2: The architecture and workflow of Collie.

A. Basic Definitions

Interdomain edge network. Figure 2 gives the architecture of Collie. We consider an interdomain edge network $G = (V, E)$ composed of K edge networks from different network providers, indexed by $k = 1, \ldots, K$. Links in G are directional. A link between two edge nodes $i, j \in V$ represents that they are directly connected. Given a link $e = (i, j)$, we use d_e and d_{ij} interchangably to denote its latency. Each AS k has a set of edge nodes (*i.e.*, routers) denoted as V_k. Each edge node $i \in V$ belongs to one and only one edge network. In other words, for any two edge networks k, k', $V_k \cap V_{k'} = \emptyset$. E_k is the set of links in E whose both ends are in V_k, *i.e.*, $E_k = \{(i, j)|(i, j) \in E, i, j \in V_k\}$. Given a link $e = (i, j) \in E$, if i, j belong to different edge networks, e is called an *edge peering link*. The set of all edge peering links in G is denoted as E_I. We see that $\cup_k V_k = V$ and $(\cup_k E_k) \cup E_I = E$.

For any two different edge nodes $(i, j) \in V \times V$, we use l_{ij} to denote the amount of data traffic node i needs to deliver to j. A node pair (i, j) is called a traffic demand pair if $l_{ij} > 0$. The set of all traffic demand pairs in G is called the traffic demand set, denoted as $S = \{(i, j)|(i, j) \in V \times V, l_{ij} > 0\}$.

Policy-compliant edge sub-networks. In Collie, given a node pair $(i, j) \in S$, its traffic demand l_{ij} can be forwarded along one or more simple paths (*i.e.*, acyclic paths) in G. Such paths may span over multiple edge networks, and are called interdomain edge paths. Given a simple path p, we use $d_p = \sum_{e \in p} d_e$, *i.e.*, the sum of latency of all links in the path, to denote its latency. For each edge network k, there are many simple paths between edge nodes and we denote the latency set as $\{d^k_{peers}\}_{peers,k}$, where the latency of the longest simple path we called *network peer latency* is $d^k_{max} = \max_{peers,k}\{d^k_{peers}\}_{peers,k}$.

Each edge network $G_k = (V_k, E_k)$ has the autonomy to make policies to decide for each link $e \in E_k$, whether it can be used to forward data from i to j, and keeps such policies as private information, similar to other interdomain routing protocols. Given an edge network k, we use edge peer latency d^k_{max} to denote the latency of the longest simple acyclic path among all the edge peer nodes in this network k. And to satisfy the service-level agreement (SLA) such as d_a, it would remove some edges from G_k to get a smaller $d^k_{max} \leq d_a$. We use G^{pol,d_a}_k to denote the policy compliant edge sub-network, where only the traffic pairs in compliant sub-networks can be transmitted. The global network can be denoted as $G^{pol} = (V^{pol}, E^{pol})$ and the policy compliant traffic demand pairs is $S^{pol} = \{(i, j)|(i, j) \in V^{pol} \times V^{pol}, l_{ij} > 0\}$. Under such circumstances, we force the flows to avoid the paths requiring a lot of time.

Optimal Low-Latency Traffic Engineering Problem. Based on the above definitions, we use linear programming to model the global optimal low-latency traffic engineering problem in interdomain edge networks.

We use the maximal traffic load in all edge peering links as the cost function, *i.e.*, $g(\mathbf{x}) = max_{e \in E_I} x_e$, which measures the load balance among edge peering links. We will show later that this is a convex function and discuss the generality of Collie for minimizing other cost functions in Section III-D. We formally define the optimal traffic engineering problem in an interdomain edge network as follows:

***Problem** 1 (Optimal Low-Latency Traffic Engineering Problem in Interdomain Edge Network):* Find the optimal solution to the following optimization problem,

$$\min \quad g(\mathbf{x}) = max_{e \in E_I} x_e, \tag{1}$$

subject to,

$$\sum\nolimits_{i \in V} F_{i,e} \leq x_e, \forall e \in E^{pol}, \tag{2a}$$

$$\sum\nolimits_{e \in E^{pol}} F_{i,e} A_{j,e} \geq l_{ij}, \forall (i, j) \in S^{pol}, \tag{2b}$$

$$x_e \geq 0, \forall e \in E^{pol}, \tag{2c}$$

$$F_{i,e} \geq 0, \forall i \in V^{pol}, \forall e \in E^{pol}. \tag{2d}$$

In Problem 1, Equation (2a) ensures that for each link e, its traffic load x_e is larger than the sum of the traffic load of each path passing link e. The $F_{i,e}$ denotes the sum of the traffic of the flows from node i. Equation (2b) ensures that for each traffic demand pair (i, j), all its traffic demand l_{ij} is allocated along all paths available. And the $A_{j,e}$ denotes the relationship between node j and link e : $A_{j,e} = 1$ while the link j enters link e, $A_{j,e} = -1$ while the link j leaves link e and $A_{j,e} = 0$ for other cases. Equations (2c)(2d) specify that all x_e and $F_{i,e}$ are non-negative. Compared with the traditional multicommodity flow problem (MCFP), this source-based formulation [9] reduces the number of decision variables from $|V^{pol}|(|V^{pol}|-1)|E^{pol}|$ to $|V^{pol}||E^{pol}|$, substantially reducing the problem scale. It is trival to prove the solution of MCFP is the solution of Problem 1, and for each solution $F_{i,e}$, we could distribute flows evenly across the egress links and we could get the solutions $Z_{i,j,e}$ for flow (i, j), which is the solution of MCFP. Thus, we could get that the Problem 1 and multi-commodity flow modeling are equivalent.

Minimizing $g(\mathbf{x}) = \max_{e \in E_I} x_e$ under linear constraints is a classic convex optimization problem and can be solved efficiently with centralized algorithms. However, centralized algorithms relying on complete knowledge cannot directly solve Problem 1, because it violates the private link usage protection policies. Privacy-preserving techniques like secure multi-party computation would introduce excessive computation and communication overhead, rendering them impractical. To address this, we propose Collie.

B. Workflow of Collie

We briefly present the basic workflow of Collie, in which edge nodes of different edge networks collaboratively compute the optimal routing and traffic assignment decisions for delivering all traffic demand pairs along low-latency paths.

- **Step 1**: all K networks agree on a cost function g for the interdomain edge network through offline negotiation and acquire the following informations: (1) the traffic demand l_{ij} from edge node i to j; (2) the SLA of each edge network k and the sub-graph whose latency of the longest acyclic simple path satisfies the SLA, ensuring only the flows in the policy compliant pairs $S^{pol} = \{(i,j)|(i,j) \in V^{pol} \times V^{pol}, l_{ij} > 0\}$ can be delivered.
- **Step 2**: edge nodes interact with their neighbors to collaboratively execute a novel, distributed algorithm (Section III), deciding the route and traffic assignments for all traffic demand pairs.
- **Step 3**: edge nodes transmit traffic according to the result, such that (1) for each $(i,j) \in S$, its traffic can be delivered along policy-compliant edge sub-networks in free path with satisfying the SLAs, (2) g is minimized, and (3) the policies of edge networks are not exposed.

III. A Distributed Algorithm for Optimal Low-Latency Routing

Our distributed algorithm, inspired by an optimization theory findings [10], solves Problem 1 through three phases: node clustering, local optimization, and aggregation. Algorithm 1 summarizes the key steps. First, edge nodes in G perform a distributed, randomized clustering protocol to form clusters based on latency. Then, within each cluster, the cluster head node constructs and solves a local optimal routing problem and then share the local solution. Finally, each node independently aggregates the received local results to determine the final routing and traffic assignment. We also provide key theorems and the corresponding proof sketches about the property and performance of our distributed algorithm. The full proofs are omitted due to space limit but can be found at [11].

A. Node Clustering

Our clustering protocol leverages the notion of padded decomposition [12], [13]. Specifically, we assume each node i has a unique identifier ID_i that is comparable. Given a node i in G, we use $B(i,d)$ to denote the set of nodes in G^{pol} whom i can reach in a latency no more than d, without considering the policy of any network. Using d_{ij} to represent the shortest latency from i to j, we have $B(i,d) = \{j|j \in V^{pol}, d_{ij} \leq d\}$. We let $D = max(d^k_{max})$, *i.e.*, the largest maximal allowable latency of all edge networks. A padded decomposition can then be defined as follows:

***Definition** 1 (Padded Decomposition):* Given a graph $G = (V,E)$, a (D,ϵ)-padded decomposition, where $\epsilon \in (0,1)$, is a probability measure over the set of graph partitions that (1) for every partition $\mathcal{C}$ that has non-zero probability, and every cluster $C \in \mathcal{C}$, $max_{i,j \in C} d_{ij} \leq O(D \log n/\epsilon)$, and (2) for every node $i \in V$, the probability that all nodes in $B(i,D)$ are in the same cluster is at least $1-\epsilon$, *i.e.*, $Pr(\exists C \in \mathcal{C} : B(i,D) \subset C) \geq 1-\epsilon$.

Algorithm 1: An algorithm that solves Problem 1 distributively through clustering, local optimization, and distributed aggregation of local optimal solutions.

Initialization: $\lambda \leftarrow \frac{\epsilon(1-\epsilon)}{(2-\epsilon)(1+\epsilon)}$, $T \leftarrow \frac{24(1-\frac{\epsilon}{2})(1+\epsilon)\ln|E|}{\epsilon^2}$;
Construct T partitions sampled from (D,λ)-padded decompositions;
foreach $t \leftarrow 1,\ldots,T$ **do**
  foreach *cluster* $C \in \mathcal{C}_t$ **do**
    The cluster head of C solves its Problem 2 and populates the solution $(\mathbf{x}^{C,t}, \mathbf{F}^{C,t})$ to all nodes in C;
foreach $i \in V^{pol}$ **do**
  foreach $e = (i,j) \in E^{pol}$ **do**
    $T_{i,j} \leftarrow \{t|\exists C \in \mathcal{C}_t : i,j \in C\}$;
    $\tilde{x}_e \leftarrow \frac{1+\epsilon}{T} \sum_{t \in T_{i,j}} x_e^{C_{i,t},t}$;
  $T_i \leftarrow \{t|\exists C \in \mathcal{C}_t : B(i,D) \subseteq C\}$;
  foreach $e \in E^{pol}$ **do**
    $\tilde{F}_{i,e} = \frac{1}{|T_i|} \sum_{t \in T_i} F_{i,e}^{C_{i,t},t}$;
foreach $(i,j) \in S^{pol}$ **do**
  foreach $e_v = (i,j) \in E^{pol}$ **do**
    $Z_{i,j,e_v} = l_{i,j}(\frac{\tilde{F}_{i,e_v}}{\sum_v \tilde{F}_{i,e_v}})$, where links $\{e_v\}$ enter node v.

The clustering protocol consists of two phases: advertising and selection. First, in the advertising phase, each edge node i broadcasts its identifier ID_i to all other edge nodes in $B(i,r_i)$, where the broadcast radius r_i is defined as $min(z_i, r\ln|V| + D)$. Here, $r = \frac{2D}{\epsilon}$, and z_i is independently sampled from a distribution with the following function

$$p(z_i) = \frac{|V|}{|V|-1} \cdot \frac{e^{-z_i/r}}{r}. \tag{3}$$

Second, in the selection phase, from all the IDs it receives, each edge node i selects node j with the smallest ID (*e.g.*, ordered by ASCII value) as its cluster head.

The clusters computed by our clustering protocol have the following property:

***Theorem** 1:* Given $G = (V,E)$ and $\epsilon \in (0,1)$ clustering protocol yields a partition of G sampled from a (D,ϵ)-padded decomposition partition.

Proof sketch. Condition (1) of Definition 1 is directly satisfied by the assignment of r_i. For any node $i \in V$, $Pr(Q(\tau))$, the probability that if $B(i,D)$ does not belong to the first $\tau-1$ clusters, it is also not in any of the remaining clusters, is no larger than $\frac{2-\tau}{|V|-1} \cdot \frac{\epsilon}{2}$. As such, condition (2) can be derived with the fact that $Pr(Q(0)) \leq \epsilon$.

B. Local Optimization

In this phase, we construct local convex programming problems for the clusters computed by the cluster protocol in the previous phase, and have the cluster heads solve them. Specifically, given a cluster C, let E_C be the set of links whose

both ends are in C, *i.e.*, $E_C = \{(i,j)|(i,j) \in E^{pol}, i,j \in C\}$ and S_C be the set of all traffic demand pairs (i,j) such that all nodes of $B(i,D)$ is fully contained in the cluster C, *i.e.*, $S_C = \{(i,j)|(i,j) \in S^{pol}, B(i,D) \subset C\}$. We construct a local optimal routing problem for C, which optimizes the same objective function as the global convex programming in Problem 1, with four differences: (1) only decide the routing and traffic assignments for traffic demand pairs in S_C; (2) only decide the traffic assignments for links in E_C; (3) set the traffic assignment for links in $E^{pol} - E_C$ to zero, and (4) only links in E_C contribute to the topology relationship A^C. Formally, this problem is defined as follows.

***Problem** 2 (Local Optimal Routing Problem):* Given a cluster C in an interdomain edge network G, find the optimal solution to the following optimization problem:

$$\min \quad g(\mathbf{x}) = max_{e \in E_I} x_e \tag{4}$$

subject to,

$$\sum\nolimits_{i \in C} F_{i,e} \leq x_e, \forall e \in E_C, \tag{5a}$$

$$\sum\nolimits_{e \in E_C} F_{i,e} A^C_{j,e} \geq l_{i,j}, \forall (i,j) \in S_C \tag{5b}$$

$$x_e \geq 0, \forall e \in E_C, \tag{5c}$$

$$x_e = 0, \forall e \in E^{pol} - E_C, \tag{5d}$$

$$F_{i,e} \geq 0, \forall i \in V_C, \forall e \in E_C, \tag{5e}$$

$$F_{i,e} = 0, \forall i \in V^{pol} - V_C, \forall e \in E^{pol} - E_C. \tag{5f}$$

The scale of Problem 2 is smaller than Problem 1, but constructing it remains an issue. To address this, we draw inspiration from on-demand routing protocols like AODV [2] and design a recursive query process for each head node to collect necessary information from other nodes in an abstract way. Details of this process are omitted due to the space limit.

We use $(\mathbf{x}^*, \mathbf{F}^*)$ to denote the optimal solution to Problem 1. For a given cluster C, we denote the optimal solution to its Problem 2 as $(\tilde{\mathbf{x}}^C, \tilde{\mathbf{F}}^C)$, and this optimal solution has the following property:

***Theorem** 2:* Define $\mathbf{x}^{*,C} = [x^{*,C}_e]_{e \in E}$ such that $x^{*,C}_e = x^*_e$ if $e \in E_C$, and 0 otherwise, $\mathbf{F}^{*,C} = [F^{*,C}_{i,e}]_{i \in V, e \in E}$ such that $F^{*,C}_{i,e} = F^*_{i,e}$ if $i \in C$ and $e \in E_C$, and 0 otherwise. Then $g(\tilde{\mathbf{x}}^C) \leq g(\mathbf{x}^{*,C})$.

Proof sketch. Because $S_C \subset S^{pol}$, $(\mathbf{x}^{*,C}, [F^*_{i,e}]_{i,e})$ satisfy both Equations (5a) and (5b), and is a feasible solution to Problem 2 for cluster C. $(\tilde{\mathbf{x}}^C, \tilde{\mathbf{F}}^C)$ is the optimal solution to the same problem, we have $g(\tilde{\mathbf{x}}^C) \leq g(\mathbf{x}^{*,C})$.

C. Aggregating Local Optimals for Global Optimization

After all cluster heads compute their local optimization problems and send the results to other nodes in the clusters, each node computes the final routing and traffic assignment decisions by taking a weighted average of all its received local optimal solutions for each decision variable x_e and $F_{i,e}$.

Specifically, each node i counts that for each link $(i,j) \in E$, which partitions have a cluster that contains both i and j, denoted as $T_{i,j}$ (Line 11), and which partitions have a cluster that contains the ball $B(i,D)$, denote as T_i (Line 13). Note that for any $(i,j) \in E$, $T_i \subseteq T_{i,j}$. With these results, node i computes $\tilde{x}_e$ for each link $e = (i,j)$ as the average of all local solutions of e in iterations where i, j are in the same cluster, *i.e.*, ,$\tilde{x}_e = \frac{1+\epsilon}{T} \sum_{t \in T_{i,j}} x_e^{C_{i,t},t}$ where $x_e^{C_{i,t},t}$ is the solution of the cluster that contains i in the t-th partition (Line 12). Node i also computes $\tilde{F}_{i,e}$ for all paths passing i in a similar way (Line 15). The vector $(\tilde{\mathbf{x}} = [\tilde{x}_e]_e, \tilde{F}_{i,e} = [F_{i,e}]_p)$ is then the solution Algorithm 1 computes for the global routing problem (*i.e.*, Problem 1).

We conduct analysis on Algorithm 1. First, Algorithm 1 has a low time complexity. Specifically, in Algorithm 1, in total there are a polynomial number of local convex programming problems to solve (*i.e.*, bounded by $O(|V^{pol}| \cdot T)$, the number of edge nodes in G times the number of partitions). None of any two of these local problems is coupled. As such, each can be solved in polynomial time, and in parallel, at the corresponding cluster center. In addition, the population of the optimal solutions to local problems also takes a polynomial time because (1) populating the solution to any local problem is bounded by $O(|E^{pol}|)$), and (2) the solutions to all local problems can be populated in parallel.

Next, we analyze the performance of Algorithm 1. Before we present the main theorem, we show an important property possessed by the objective function $g(\mathbf{x}) = max_{e \in E_I} x_e$.

***Definition** 2 (Convex Partitionable Function):* Given a graph $G = (V,E)$, a function $g : \mathbb{R}^{|E|} \to \mathbb{R}$ is convex partitionable if (1) g is convex; (2) g is non-decreasing, *i.e.*, given $\mathbf{x}, \mathbf{x}'$, if $x_e \leq x'_e$ for all $1 \leq e \leq |E|$, $g(\mathbf{x}) \leq g(\mathbf{x}')$; and (3) for each partition $\mathcal{C} = \{C_1, \ldots, C_l\}$ of nodes V, there exists a non-decreasing function $h_{\mathcal{C}} : \mathbb{R}^l \to \mathbb{R}$, such that $g(\mathbf{x}^\diamond) = h_{\mathcal{C}}(g(\mathbf{x}^{\diamond,C_1}), \ldots, g(\mathbf{x}^{\diamond,C_l}))$, where $\mathbf{x}^\diamond = [x^\diamond_e]_{e \in E}$ where $x^\diamond_e = x_e$ if both ends of e belong to the same cluster in $\mathcal{C}$, and 0 otherwise, and $\mathbf{x}^{\diamond,C_\theta}$, $\theta = 1, \ldots, l$, is defined as $x^{\diamond,C_\theta}_e = x_e$ if both ends of e belong to C_θ, and 0 otherwise.

***Lemma** 1:* $g(\mathbf{x}) = max_{e \in E_I} x_e$ is convex partitionable.

Proof sketch. First, given $\mathbf{x}, \mathbf{x}'$, where $x_e \leq x'_e$ for all $1 \leq e \leq |E|$, because E_I is fixed, $max_{e \in E_I} x_e \leq max_{e \in E_I} x'_e$, *i.e.*, $g(\mathbf{x}) \leq g(\mathbf{x}')$. As such, $max_{e \in E_I} x_e$ is non-decreasing. Second, given $\mathbf{x}, \mathbf{x}'$ and $\lambda \in [0,1]$, $g(\lambda \mathbf{x} + (1-\lambda)\mathbf{x}') = max_{e \in E_I}(\lambda \mathbf{x_e} + (1-\lambda)\mathbf{x'_e}) \leq \lambda max_{e \in E_I} x_e + (1-\lambda) max_{e \in E_I} x'_e \leq \lambda g(\mathbf{x}) + (1-\lambda) g(\mathbf{x}')$. Thus, $max_{e \in E_I} x_e$ is convex. Third, the max function of can meet the requirement.

Having proved $g(\mathbf{x}) = max_{e \in E_I} x_e$ is convex partitionable, We now present the theorem on Algorithm 1's performance.

***Theorem** 3:* Let $(\tilde{\mathbf{x}}, \tilde{\mathbf{F}})$ be the solution computed by Algorithm 1, it is a solution to Problem 1 with high probability (*i.e.*, $1 - \frac{1}{|E^{pol}|^2}$), and has an approximation ratio of $1+\epsilon$, *i.e.*, $\frac{g(\tilde{\mathbf{x}})}{g(\mathbf{x}^*)} \leq (1+\epsilon)$.

Proof sketch. The feasibility can be easily derived by the aggregation procedure and Definition 1. As for the approximation ratio, by leveraging Lemma 1 and defining $\tilde{x}^t_e = x_e^{C_{i,t},t}$ if $t \in T_{i,j}$, and 0 otherwise, we can have

$$g(\tilde{\mathbf{x}}) \leq \frac{1+\epsilon}{T} \sum_{t=1}^{T} g\Big([\tilde{x}^t_e]_{e=(i,j) \in E}\Big). \tag{6}$$

And we let $\mathcal{C} = \{C_\theta\}_\theta$ be the partition of V at iteration t and $\tilde{\mathbf{x}}^t = [\tilde{x}^t_e]_{e=(i,j) \in E}$, by combining Theorem 2, we can get

$$g(\tilde{\mathbf{x}}^t) = h(g(\tilde{\mathbf{x}}^{t,C_\theta}))_\theta \leq h(g(\mathbf{x}^{*,C_\theta}))_\theta \leq g(\mathbf{x}^*), \tag{7}$$

We plug this result to Equation (6), and get

$$g(\tilde{\mathbf{x}}) \leq \frac{1+\epsilon}{T} \sum_{t=1}^{T} g(\tilde{\mathbf{x}}^t) \leq (1+\epsilon) g(\mathbf{x}^*), \tag{8}$$

completing the approximation ratio proof of Algorithm 1. Combining the feasibility proof and the approximation ratio proof, we have completed the proof of Theorem 3.

D. Discussion

We next discuss several issues regarding the feasibility and practicality of Collie.

Incremental deployment. Collie can reduce end-to-end communication latency without requiring full deployment across all networks. It can be incrementally deployed at interconnected networks via edge peering links, effectively reducing latency for devices connected to both networks. As other networks observe the benefits, they are incentivized to join and deploy Collie.

Support other cost functions. Algorithm 1 in Collie is a generic solution. Our proof of Theorem 3 demonstrates its effectiveness for a wide range of convex partitionable cost functions. This means that network providers can collaboratively optimize various cost functions by negotiating. Examples include maximizing traffic assignment on a set of links in G^{pol} (*i.e.*, $g(\mathbf{x}) = \max_{e \in E_a, E_a \subset E}(x_e)$) and weighted sum of traffic assignment on a set of links in G^{pol} (*i.e.*, $g(\mathbf{x}) = \sum_{e \in E_a, E_a \subset E}(w_e x_e)$). Further exploration of convex partitionable cost functions in interdomain edge networks is left as future work.

IV. Performance Evaluation

We implement a prototype of Collie, and use real-world topologies to evaluate its performance and overhead in finding solutions for the optimal low-latency routing problem in interdomain edge networks.

A. Experiment Settings

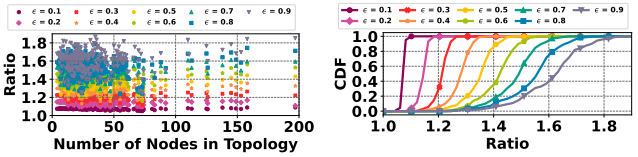
Dataset. We use 249 real network topologies from Topology Zoo [14] with real link delays [15]. In these topologies, the number of nodes ranges from 4 to 197, and the number of links ranges from 4 to 245. We employ twice the latency of the shortest path between edge peer nodes in the network and offer 8 policy-compliant paths generated via breath-first search algorithm. The traffic demands of traffic demand pairs are synthesized using the gravity model [16].

Experiment parameters. We divide the nodes into 2 - 6 ASes for each topology according to its size. In our prototype, we use Gurobi [17] as the solver. We conduct experiments with $g(\mathbf{x}) = max_{e \in E_I} x_e$ and $g(\mathbf{x}) = \sum_{e \in E_I} x_e$ as the cost functions. We set ϵ to 9 different values. And for each choice of ϵ, we repeat the experiment for each topology 5 times and measure the average result.

B. Results

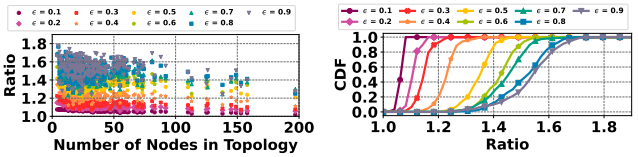
We first present results on the the performance of Collie, and then demonstrate its scalability by analyzing its overhead.

Performance. We study the performance of Collie by comparing the value of the objective function $g(\tilde{\mathbf{x}})$ computed by



(a) The impact of topology size on $\frac{g(\tilde{\mathbf{x}})}{g(\mathbf{x}^*)}$. **(b)** The CDF of $\frac{g(\tilde{\mathbf{x}})}{g(\mathbf{x}^*)}$ in all experiments.

Fig. 3: The performance of Algorithm 1 vs. the global optimal solution using $g(\mathbf{x}) = max_{e \in E_I} x_e$ as the cost function.



(a) The impact of topology size on $\frac{g(\tilde{\mathbf{x}})}{g(\mathbf{x}^*)}$. **(b)** The CDF of $\frac{g(\tilde{\mathbf{x}})}{g(\mathbf{x}^*)}$ in all experiments.

Fig. 4: The performance of Algorithm 1 vs. the global optimal solution using $g(\mathbf{x}) = \sum_{e \in E_I} x_e$ as the cost function.

Algorithm 1 and the actual optimal objective function value $g(\mathbf{x}^*)$ computed by directly solving Problem 1.

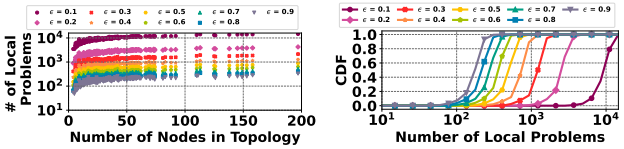
Fig. 3 and Fig. 4 shows the ratio of $g(\tilde{\mathbf{x}})$ over $g(\mathbf{x}^*)$ under two different cost functions respectively (*i.e.*, the approximation ratio of Algorithm 1). We observe that this ratio is bounded by $1+\epsilon$ and independent from the size of topology. As such, we conclude that Collie equipped with Algorithm 1 provides a good approximation performance across various topologies and experiment settings.

Scalability overhead. To study the scalability of Collie, we analyze its overhead in three key aspects: the number and scale of local problems, the number of messages needed for clustering. Due to the fact that these three metrics are independent of the cost function, we combine the results obtained using the two cost functions for presentation.

Fig. 5 studies the number of local convex programming problems in each experiment, which equals to the number of clusters in each experiment. We see that this metric increases slowly with topology size, and decreases rapidly as ϵ increases. The largest number of local problems needed is less than 14.6k, and $\epsilon = 0.1$, because it has more links. For the same topology, when $\epsilon = 0.9$, the number of local problems drops to 331. Even when ϵ is slightly large (*e.g.*, > 0.4), 80% of topologies only need to solve less than 1k local problems.

Fig. 6 shows the average scale of local problems, which is measured as the average number of constraints of each local problem. We see the problem scale increases with topology size, with the largest scale having less than 171k constraints when $\epsilon = 0.1$. It shows that the scale of local problems is less than 10k constraints in 80% of the experiments.

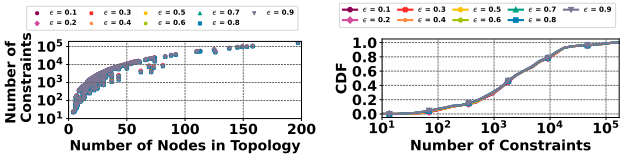
Fig. 7 studies the average number of messages each node sends (including originating and forwarding) in the clustering protocol to form a partition. We see that the average number of messages each node sends in general increases as the number of nodes in the topology increases, and is slightly affected by ϵ. Even in the worst case, each node sends less than



(a) The impact of topology size on the number of local problems.

(b) The CDF of the number of local problems.

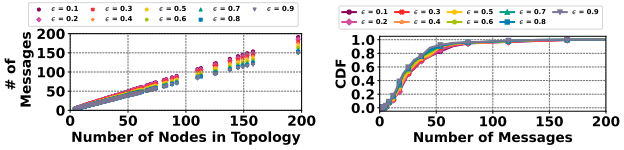
Fig. 5: The number of local convex programming problems each experiment needs to solve.



(a) The impact of topology size on the scale of local problems.

(b) The CDF of the scale of local problems.

Fig. 6: The scale of local problems in all experiments 1.



(a) The impact of topology size on the average number of clustering messages.

(b) The CDF of the average number of clustering messages of each node.

Fig. 7: The average number of messages each node sends.

191 messages on average to form a cluster, showing that the clustering message overhead of Collie is reasonably low.

Through these results, we see that the overhead of Collie increases slowly as the network scale increases. Thus, Collie has the potential to scale to large interdomain edge networks.

V. Related Work

Efforts to reduce the latency of interdomain edge network communication share a common basic principle: bypassing backbone networks for traffic transmission [2]–[7]. However, these approaches suffer from scalability problems, slow response times, high investments, and limited applicability. In contrast, Collie constructs an interdomain edge network for traffic forwarding, mitigating the scalability issue and the potential latency, while reducing the cost of network providers.

Multiple systems have proposed to pool the resources of multiple networks together to improve the overall performance [18], [19]. Some are already deployed with success. Collie enable edge networks to share resources, collaboratively forwarding end-to-end traffic along low-latency paths.

Some efficient algorithms are designed for solving specific linear programming distributively [20], but cannot be used for Problem 1 in Collie. We are inspired by a recent finding in relaxed graph spanner [10] to design a novel, distributed algorithm to solve it with a tight approximation ratio.

VI. Conclusion

We propose Collie to tackle the low-latency interdomain communication challenge in edge networks, which allows different edge networks to collaboratively forward traffic along low-latency paths. Extensive evaluations show that Collie can provide a tight ratio while maintaining low overhead.

Acknowledgment

We are extremely grateful for the anonymous reviewers for their wonderful feedback. Yuxin Wang, Haoyu Liu, Haohao Song, Siyong Huang, and Qiao Xiang are supported in part by the National Key R&D Program of China 2022YFB2901502, NSFC Award 62172345, Open Research Projects of Zhejiang Lab 2022QA0AB05, MOE China 2021FNA02008, and NSF-Fujian-China 2022J01004.

References

[1] H. Zhang, S. Huang, M. Xu, D. Guo, X. Wang, V. C. M. Leung, and W. Wang, "How Far Have Edge Clouds Gone? A Spatial-Temporal Analysis of Edge Network Latency In the Wild," in *IWQoS'23*. IEEE, 2023, pp. 1 – 10.

[2] S. Das, C. Perkins, and E. Royer, "Ad Hoc on Demand Distance Vector (AODV) Routing," *IETF RFC3561, July*, vol. 10, 2003.

[3] S. Kassir, G. de Veciana, N. Wang, X. Wang, and P. Palacharla, "Enhancing Cellular Performance via Vehicular-Based Opportunistic Relaying and Load Balancing," in *INFOCOM'19*. IEEE, 2019, pp. 91–99.

[4] X. Zhang and Q. Zhu, "D2D Offloading for Statistical QoS Provisionings Over 5G Multimedia Mobile Wireless Networks," in *INFOCOM'19*. IEEE, 2019, pp. 82–90.

[5] G. Luo, H. Zhou, N. Cheng, Q. Yuan, J. Li, F. Yang, and X. Shen, "Software-Defined Cooperative Data Sharing in Edge Computing Assisted 5G-VANET," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1212–1229, 2021.

[6] AT&T, "AT&T Network Edge," https://www.business.att.com/products/att-network-edge.html.

[7] Google, "Google Edge Network," https://peering.google.com/.

[8] Y. Wang, Q. Xiang, J. Shu, G. Li, and L. Kong, "Toward Low-Latency End-to-End Communication in 5G Using Interdomain Edge Peering," in *NAI@SIGCOMM'22*. ACM, 2022, pp. 26–32.

[9] P. Yin, S. Diamond, B. Lin, and S. Boyd, "Network Optimization for Unified Packet and Circuit Switched Networks," *Optimization and Engineering*, vol. 21, pp. 159–180, 2020.

[10] M. Dinitz and Y. Nazari, "Distributed Distance-Bounded Network Design Through Distributed Convex Programming," in *OPODIS'17*, 2017, pp. 1 – 19.

[11] Y. Wang, H. Liu, H. Song, S. Huang, S. Qin, Y. Liu, Q. Xiang, L. Kong, G. Li, J. Shu, and X. Liu, "CoLLIE: Collaborative Low-Latency Interdomain Edge Network Communication, Technical Report," https://github.com/sngroup-xmu/EdgePeering-tr.git.

[12] R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor, "Measured Descent: A New Embedding Method for Finite Metrics," in *FOCS'04*. IEEE, 2004, pp. 434–443.

[13] A. Gupta, M. T. Hajiaghayi, and H. Räcke, "Oblivious Network Design," in *SODA'06*. ACM, 2006, pp. 970–979.

[14] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[15] WonderNetwork, "Global Ping Statistics," https://wondernetwork.com/pings, 2024.

[16] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in Measuring Backbone Traffic Variability: Models, Metrics, Measurements and Meaning," in *SIGCOMM'02*. ACM, 2002, pp. 91–92.

[17] "The Gurobi Solver," https://www.gurobi.com/.

[18] C. Hu, W. Bao, and D. Wang, "IoT Communication Sharing: Scenarios, Algorithms and Implementation," in *INFOCOM'18*. IEEE, 2018, pp. 1556–1564.

[19] H. Wang, Y. R. Yang, P. H. Liu, J. Wang, A. Gerber, and A. Greenberg, "Reliability as an Interdomain Service," in *SIGCOMM'07*. ACM, 2007, pp. 229–240.

[20] P. Floréen, M. Hassinen, J. Kaasinen, P. Kaski, T. Musto, and J. Suomela, "Local Approximability of Max-Min and Min-Max Linear Programs," *Theory of Computing Systems*, vol. 49, no. 4, pp. 672–697, 2011.