Toward Low-Latency End-to-End Communication in 5G Using Interdomain Edge Peering

Yuxin Wang[†‡], Qiao Xiang[†‡*], Jiwu Shu[†], Geng Li[°], Linghe Kong[◊],
[†]Xiamen University, [‡]Tan Kah Kee Innovation Laboratory,
[°]Huawei Technologies, [◊]Shanghai Jiao Tong University

Abstract

A core requirement in the 5G network is to support low-latency end-to-end communication. However, in the current network architecture, the communication between devices has to go through unnecessarily long paths involving radio access networks, edge networks and backbone networks, hence could result in high latency. To address this problem, in this paper, we design an interdomain edge peering framework called EdgePeering. Instead of sending traffic to backbone networks, in EdgePeering, edge networks from different network providers use each other to forward traffic toward destinations, and therefore improve the latency of end-to-end communication. Meanwhile, network providers in EdgePeering maintain their autonomy to make policies on which links in their own edge network can be used to forward such traffic. We develop a novel distributed algorithm in EdgePeering, which allows edge nodes of different networks to collaboratively decide the optimal routing and traffic assignment for delivering all traffic along paths with low latency, subject to the policies of all networks but without exposing them. Extensive evaluation using real-world topologies shows that EdgePeering provides a tight approximation ratio, and can scale to large interdomain edge networks.

CCS Concepts

• **Networks** → **Network architectures**; **Network services**; **Routing protocols**.

Keywords

Next-generation network architecture, Interdomain, Distributed algorithm, Traffic engineering

ACM Reference Format:
Yuxin Wang, Qiao Xiang, Jiwu Shu, Geng Li, Linghe Kong. 2022. Toward Low-Latency End-to-End Communication in 5G Using Interdomain Edge Peering . In *ACM SIGCOMM 2022 Workshop on Network-Application Integration (NAI '22), August 22, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3538401.3546600

[*] The corresponding author is Qiao Xiang.

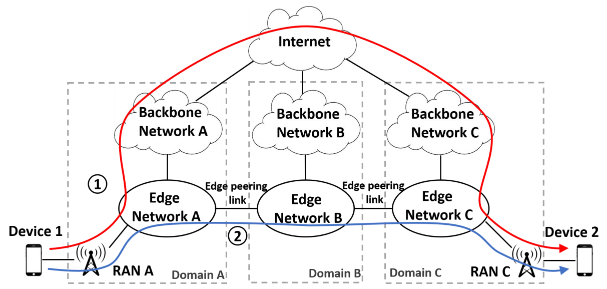


Figure 1: A motivating example for EdgePeering. ①: in the current network architecture, device 1 needs to go through the backbone networks to reach device 2, resulting in potentially high latency. ②: in EdgePeering, different edge networks can collaboratively form interdomain edge paths, potentially reducing the end-to-end latency from device 1 to device 2.

1 Introduction

5G is a key enabler for diverse new use cases, such as autonomous driving [6] , augmented/virtual reality [9] , cloud gaming [33], and ultra-high-resolution video streaming [24]. To this end, many innovations (*e.g.*, 5G New Radio [4], carrier aggregation [8] and dynamic slot scheduling [23]) have been made, including commercial 5G network deployments since 2019 [7, 10].

Despite such progress in both technology innovation and commercial deployment, the end-to-end communication latency in 5G is still constrained by its overall architecture design. Specifically, consider a 5G typical scenario [4] in Figure 1. For device 1 to send data traffic to device 2, the workflow consists of 4 steps: (1.1) device 1 sends the data through its connected radio access network (RAN) to edge network *A*; (1.2) edge network *A* performs packet core functions (*e.g.*, billing) on the received data, and forwards it to the backbone network *A* and Internet; (1.3) the backbone network *C* forward the data to edge network *C*; (1.4) edge network *C* sends the data to device 2 through RAN C. The end-to-end communication latency between two devices is limited by the latency of this unnecessarily long path, and can be very high.

Researchers have made several efforts to reduce the end-to-end latency in cellular networks, and the basic design principle is to *avoid sending the traffic through the backbone networks*. Specifically, some propose to design and use mobile ad hoc network (MANET) routing and scheduling protocols [11, 18, 28] to forward traffic through end devices. However, it has limited scalability, and hence is not suitable for the 5G network. Others propose to use servers in edge networks as coordinators to decide the routing among end devices [22, 32]. However, this separation of control path and data path cannot respond to the highly dynamic environment of 5G in time. Some industry solutions use edge networks for traffic forwarding among devices connected to the same network provider [5, 14], but they require significant hardware investment and only work on data traffic among devices connected to the same network providers.

In this paper, we propose EdgePeering, a novel framework that achieves low-latency end-to-end communication in 5G at a low cost when the latency of the backbone network is high, using two novel design points:

Peering and exchanging traffic at edge networks. (§2). Our key observation is that the edge networks of different network providers can collaboratively provide low-latency end-to-end paths among all devices. For example, in the same scenario in Figure 1, by adding edge peering links between edge networks from different providers at common locations (*e.g.*, edge data centers provided by data center providers or internet exchange points) and forwarding traffic through edge networks, device 1 can send data to device 2 with a potentially lower latency, without going through backbone networks. By allowing network providers to use the edge networks of others, we may improve the end-to-end latency of 5G at low investment and social cost, and ensure scalability.

We note that the idea of pooling the resources of multiple networks together to improve the overall network performance is already being practiced in some other contexts. For example, REIN [29] proposes to let two neighboring networks use the infrastructure of each other as a backup to improve the network reliability. Such sharing of resources improves the performance of individual network providers, reduces their investment cost, and also reduces the overall social cost (*e.g.*, less waste of idle resources).

Distributed, collaborative traffic engineering among edge networks (§3). To realize low-latency forwarding using edge networks, the participating network providers first negotiate offline to reach an agreement on a cost function that the network of their edge networks (called an *interdomain edge network*) aims to minimize. Next, a novel, distributed algorithm is developed, which allows routers (called *edge nodes*) from different network providers in the interdomain edge network to collaboratively decide the optimal routing and traffic assignment for delivering traffic in the interdomain edge network along paths with low latency, subject to the topologies and policies of all network providers but without exposing them.

Prototype evaluation (§4). We implement a prototype of EdgePeering, and evaluate its performance using real world topologies. Results show that EdgePeering provides a tight approximation ratio on minimizing the agreed cost function, with small messaging and computation overhead even in large networks.

2 Overview of EdgePeering

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

2.1 Architecture and Workflow

Interdomain edge network. Figure 2 gives the architecture of EdgePeering. 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} interchangeably 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 belongs 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 $(\cup_k E_k) \cup E_I = E$.

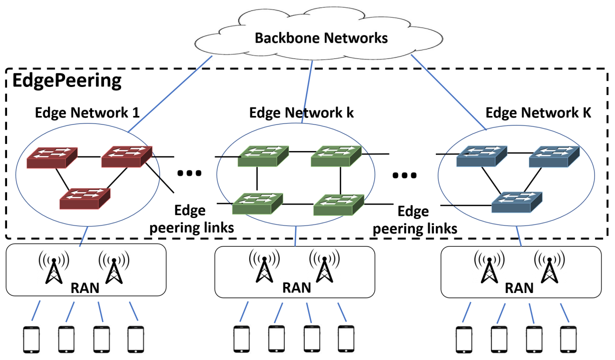


Figure 2: The architecture of EdgePeering.

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 $S = \{(i, j) | (i, j) \in V \times V, l_{ij} > 0\}$.

Workflow of EdgePeering. We briefly present the basic workflow of EdgePeering, in which edge nodes of different networks collaboratively compute the optimal routing and traffic assignment decisions for delivering all traffic demand pairs along low-latency paths. All these steps can be executed in short or long periods as needed.

- **Step 0**: all K networks agree on a cost function g for the interdomain edge network through offline negotiation. This step is similar to that in collaborative science networks [21] and commercial interdomain networks [26, 29, 30];
- **Step 1**: each edge node i estimates l_{ij}, the traffic demand from itself to j, and measures d^B_{ij}, the latency to deliver traffic to j through the backbone networks for all other edge nodes $j \in V/\{i\}$;
- **Step 2**: each edge node i decides $\alpha_{ij} \in (0, 1)$, the maximal latency coefficient, and uses $\alpha_{ij} d^B_{ij}$ as the maximal allowable latency for delivering traffic from i to j;
- **Step 3**: edge nodes execute a distributed algorithm (§3) to decide the route and traffic assignments for all traffic demand pairs, such that (1) for each $(i, j) \in S$, its traffic can be delivered along policy-compliant paths (*i.e.*, paths allowed by network providers to forward interdomain traffic) with latency no higher than $\alpha_{ij} d^B_{ij}$, (2) g is minimized, and (3) the policies of edge networks are not exposed.

2.2 Optimal Low-Latency Routing Problem

In EdgePeering, given a node pair $(i, j) \in S$, its traffic demand l_{ij} can be forwarded along one or more simple paths in G. Such paths may span over multiple edge networks, and are called interdomain edge paths. Given a 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.

Each edge network 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 (*e.g.*, [27, 31]). Given a path p connecting i and j, it is called a *policy-compliant* interdomain edge path for traffic demand pair (i, j) if and only if for each link e in p, the corresponding network's policy allows e to be used to forward traffic from i to j. We use $\mathcal{P}_{ij}^{pol}$ to denote the set of all policy-compliant interdomain edge paths of traffic demand pair (i, j).

For each traffic demand pair $(i, j) \in S$, we use $\mathcal{P}_{ij}$ to denote the set of all policy-compliant interdomain edge paths in G that has an end-to-end latency lower than or equal to its maximal allowable latency, *i.e.*, $\mathcal{P}_{ij} = \{p | p \in \mathcal{P}_{ij}^{pol}, d_p \leq \alpha_{ij} d_{ij}^B\}$. For each $p \in \mathcal{P}_{ij}$, we use y_p to denote the amount of traffic demand pair (i, j) to be delivered along p. For each link $e \in E$ in the interdomain edge network, we use x_e to denote the total amount of traffic to be delivered along e. For presentation simplicity, we also introduce vector notations $\mathbf{x} = [x_e]_{e \in E}$ and $\mathbf{y} = [y_p]_{p \in \cup_{(i,j) \in S} \mathcal{P}_{ij}}$.

For presentation simplicity, we use the maximal traffic load of all edge peering links as the cost function, *i.e.*, $g(\mathbf{x}) = max_{e \in E_i} x_e$, We believe it is reasonable for EdgePeering, because it measures the fairness (*i.e.*, traffic load balance) among cross-domain traffic forwarding. We will discuss more general cost functions later.

With these notations, we formally define the following problem in EdgePeering:

Problem 1 (Optimal Low-Latency Routing Problem). *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,

$$\sum_{(i,j) \in S} \sum_{p \in \mathcal{P}_{ij}:\, e \in p} y_p \leq x_e, \forall e \in E, \tag{2a}$$

$$\sum_{p \in \mathcal{P}_{ij}} y_p \geq l_{ij}, \forall i, j \in S, \tag{2b}$$

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

$$y_p \geq 0, \forall (i, j) \in S, \forall p \in \mathcal{P}_{ij}, \tag{2d}$$

where each edge network keeps its policy on what links can be used to forward traffic for traffic demand pair (i, j) private.

Equation (2a) ensures that for each link e, the sum of the traffic load of each path passing link e does not exceed the amount of assigned traffic load of link e. Equation (2b) ensures that for each traffic demand pair (i, j), all its traffic demand l_{ij} is allocated along policy-compliant paths with latency no higher than $\alpha_{ij} d_{ij}^B$. Note the $\geq$ and $\leq$ in these two constraints can be replaced by $=$ without affecting the optimal solution. Equations (2c)(2d) specify that all x_es and y_ps are non-negative.

The fundamental challenge of Problem 1 is that there is no global view of any policy-compliant path due to the privacy policies of edge networks. As such, we next design a novel distributed algorithm to solve it.

3 A Distributed Algorithm for Optimal Low-Latency Routing

The key steps of our distributed algorithm are summarized in Algorithm 1. Specifically, it solves Problem 1 in three phase: node clustering, local optimization, and aggregation. First, edge nodes in G execute a distributed, randomized clustering protocol to form a partition of clusters multiple times, based on the latencies among them, *i.e.*, nodes in the same cluster are not too far from each other in terms of latency. Second, in each partition, the head node of each cluster constructs and solves a local optimal routing problem without violating the private information of each network, and sends the corresponding local result to the nodes in the same cluster. In the end, each node independently take a weighted average of the local results it receives to get the final routing and traffic assignment.

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

```
1  Initialization: $\lambda \leftarrow \frac{\epsilon(1-\epsilon)}{(2-\epsilon)(1+\epsilon)}$, $T \leftarrow \frac{24(1-\frac{\epsilon}{2})(1+\epsilon)\ln|E|}{\epsilon^2}$;
2  // Phase 1: Node Clustering
3  Construct $T$ partitions sampled from $(D, \lambda)$-padded
     decompositions ;
4  // Phase 2: Local Optimization
5  foreach $t \leftarrow 1, \ldots, T$ do
6      foreach cluster $C \in C_t$ do
7          The cluster head of $C$ solves its Problem 2 and populates
             the solution $(\mathbf{x}^{C,t}, \mathbf{y}^{C,t})$ to all nodes in $C$;
8  // Phase 3: Aggregation
9  foreach $i \in V$ do
10     foreach $e = (i, j) \in E$ do
11         $T_{i,j} \leftarrow \{t | \exists C \in C_t : i, j \in C\}$;
12         $\tilde{x}_e \leftarrow \frac{1+\epsilon}{T} \sum_{t \in T_{i,j}} x_e^{C_{i,t},t}$;
13     $T_i \leftarrow \{t | \exists C \in C_t : B(i, D) \subseteq C\}$;
14     foreach $p \in \cup \mathcal{P}_{ij}$ that passes i do
15         $\tilde{y}_p = \frac{1}{|T_i|} \sum_{t \in T_i} y_p^{C_{i,t},t}$;
```

3.1 Node Clustering

In the clustering phase, we execute a clustering protocol T times (Line 3) in concurrent to construct T partitions of the complete interdomain edge network G. Our goal is to partition G into smaller clusters in each partition, where nodes in each cluster are not too far in G in terms of latency.

To this end, we leverage the notion of padded decomposition [15, 20] in metric embedding theories [20]. Specifically, we assign each node i 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 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 $B(i, d) = \{j | j \in V, d_{ij} \leq d\}$. We let $D = max(\alpha_{ij} d_{ij}^B)$, *i.e.*, the largest maximal allowable latency of all traffic demand pairs. A (D, ϵ)-padded decomposition, where $\epsilon \in (0, 1)$, is a probability measure over the set of graph partitions, that for each $i \in V$, the probability that all nodes in $B(i, D)$ are in the same cluster is at least $1 - \epsilon$. That is, the closer ϵ gets to 0, the more likely it is that nodes within a certain distance will be assigned to the same cluster, and vice versa.

Our clustering protocol has two phases: advertising and selection. In the advertising phase, each node i broadcasts its ID_i to all other nodes in $B(i, r_i)$, where the broadcast radius r_i is $min(z_i, r \ln |V| + D)$.

Here, $r = \frac{2D}{\epsilon}$, and z_i is independently sampled from a distribution with a probability density function $p(z_i) = \frac{|V|}{|V|-1} \cdot \frac{e^{-z_i/r}}{r}$.

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. Using metric embedding theories [20], we can prove that for each run of our clustering protocol, the resulting set of clusters is a partition of G sampled from a (D, ϵ)-padded decomposition partition.

3.2 Local Optimization

After constructing T partitions of clusters, for each cluster, we let its cluster head construct and solve a local convex programming problem. Specifically, given a cluster C, let E_C be the set of links whose both ends are in C, *i.e.*, $E_C = \{(i,j)|(i,j) \in E, i, j \in C\}$ 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.*, $S_C = \{(i,j)|(i,j) \in S, B(i,D) \subset C\}$. We construct a local optimal routing problem for C, which optimizes the same objective function as the global convex programming in Problem 1, with three 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, and (3) set the traffic assignment for links in $E - E_C$ to zero. Formally, this problem is defined as follows.

Problem 2 (Local Optimal Routing Problem). *Given a cluster C, find the optimal solution to the following problem:*

$$min \quad g(\mathbf{x}) = max_{e \in E_I} x_e \tag{3}$$

subject to,

$$\sum_{(i,j)\in S_C} \sum_{p \in \mathcal{P}_{ij}:\, e \in p} y_p \le x_e, \ \forall e \in E_C, \tag{4a}$$
$$\sum_{p \in \mathcal{P}_{ij}} y_p \ge l_{ij}, \ \forall i,j \in S_C, \tag{4b}$$
$$x_e \ge 0, \ \forall e \in E_C, \tag{4c}$$
$$x_e = 0, \ \forall e \in E - E_C, \tag{4d}$$
$$y_p \ge 0, \ \forall (i,j) \in S_C, \forall p \in \mathcal{P}_{ij}, \tag{4e}$$
$$y_p = 0, \ \forall (i,j) \in S/S_C, \forall p \in \mathcal{P}_{ij}. \tag{4f}$$

The scale of the Problem 2 is much smaller than Problem 1, however, how to construct is still an issue. Because nodes in the same cluster C are not necessarily in the same edge network, a strawman approach that collects the topology and routing policy of all nodes in C to construct this convex programming model is not feasible. To tackle this, for each partitioned cluster computed by the cluster protocol in the previous phase, we use the latency and routing information of nodes in the same cluster to construct a local convex programming problem at its head, and let the head solve it. Because edge nodes in the same cluster may belong to different networks, we use a recursive query process similar to AODV [11] and DSDV [25] to collect the necessary information in a path-vector abstraction (*e.g.*, the ALTO path vector extension [13, 17]) to construct the local convex programming problem, while allowing networks to keep their topology and policy information private. Details of the query process can be found in [1].

For a given a cluster C, we denote the optimal solution to its Problem 2 as $(\tilde{\mathbf{x}}^C, \tilde{\mathbf{y}}^C)$. After the head of C computes $(\tilde{\mathbf{x}}^C, \tilde{\mathbf{y}}^C)$, it sends this solution to all other nodes in C. We use $(\mathbf{x}^*, \mathbf{y}^*)$ to denote the optimal solution to Problem 1, and prove that:

Theorem 1. *Define* $\mathbf{x}^{*,C} = [x_e^{*,C}]_{e \in E}$ *such that* $x_e^{*,C} = x_e^*$ *if* $e \in E_C$, *and 0 otherwise. Then* $g(\tilde{\mathbf{x}}^C) \le g(\mathbf{x}^{*,C})$.

We prove it by showing that $(\mathbf{x}^{*,C}, [y_p^*]_p)$ is a feasible solution to Problem 2. Details can be found in [1].

3.3 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 y_p.

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 $\tilde{y}_p$ for all paths passing i in a similar way (Line 15). The vector $(\tilde{\mathbf{x}} = [\tilde{x}_e]_e, \tilde{\mathbf{y}} = [\tilde{y}_p]_p)$ is then the solution Algorithm 1 computes for the global routing problem (*i.e.*, Problem 1).

We next analyze the performance of 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 $O(|V| \cdot T)$, the number of edge nodes in G times the number of partitions). None of any two 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 $O(|E|)$), and (2) the solutions to all local problems can be populated in parallel.

In addition to low computation complexity, we also prove:

Theorem 2. *Suppose the objective function* $g(\mathbf{x})$ *is convex partitionable[12]. Let* $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ *be the solution computed by Algorithm 1, it is a solution to Problem 1 with high probability* (i.e., $1 - \frac{1}{|E|^2}$), *and has an approximation ratio of* $1 + \epsilon$, i.e., $\frac{g(\tilde{\mathbf{x}})}{g(\mathbf{x}^*)} \le (1 + \epsilon)$.

The proof is omitted due to space limit, and can be found in [1]. This shows the generality of Algorithm 1: it computes a $(1 + \epsilon)$-approximated solution to Problem 1 for any cost function that is convex partitionable. This means that in practice, network providers can negotiate to collaboratively optimize diverse cost functions. For example, not only is $g(\mathbf{x}) = max_{e \in E_I} x_e$ convex partitionable, so are the maximal traffic assignment of any set of links in G, *i.e.*, $g = max_{e \in E_a, E_a \subset E}(x_e)$ and the weighted sum of traffic assignment of any set of links in G, *i.e.*, $g = \sum_{e \in E_a, E_a \subset E}(w_e x_e)$. As such, they can also be optimized by Algorithm 1.

4 Performance Evaluation

We implement a prototype of EdgePeering, and evaluate its performance with extensive experiments using real-world topologies.

We aim to answer the following questions. (1) What is the performance of Algorithm 1 in finding optimal solutions for the optimal low-latency routing problem in interdomain edge networks? (2) Can EdgePeering scale to large networks with low overhead?

4.1 Experiment Settings

Network topology. We use the topology of 243 real networks of various sizes from Topology Zoo [19] and Rocketfuel [3] as the topologies of interdomain edge networks in our experiments. We focus on the scenario where the latency of EdgePeering can be lower than that of the backbone network. In these topologies, the number of nodes ranges from 4 to 96, and the number of links ranges from 4 to 97. When evaluating EdgePeering's scalability later, we add another 10 larger topologies. The largest ones have 127 nodes and 129 links, 113 nodes and 183 edges, respectively.

Experiment parameters. We divide the nodes into 6 ASes for each topology and set the lower bound of latency of each link as 10, and the upper bound as 300. For each topology, we randomly select 10% of the links as edge peering links.

For each topology, we randomly select 80% of node pairs as the set of traffic demand pairs. For each traffic demand pair (i, j), we randomly assign its traffic demand from a uniform distribution between 50 and 100. We use 2 times of its lowest latency in the topology to approximate its maximal allowable latency. To simulate the policies of different network providers, among all paths connecting i and j with a latency smaller than its maximal allowable latency, we randomly select 50% (30%) of them as policy-compliant interdomain edge paths in the topology with ≤ 20 (> 20) nodes.

In our prototype, we use Gurobi [2] as the solver. We conduct experiments with $g = max_{e \in E_I} x_e$ and $g = \sum_{e \in E} x_e$ as the objective functions. We set ϵ to be $\{0.3, 0.5, 0.7, 0.9\}$. For each choice of ϵ, we repeat the experiment for each topology 5 times and measure the average result.

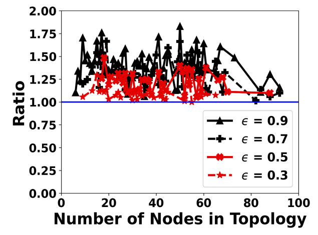
4.2 Results

We only present the results with $g = max_{e \in E_I} x_e$, and omit the results with $g = \sum_{e \in E} x_e$ because they show similar trends in all experiments. We present results on both performance and scalability of EdgePeering.

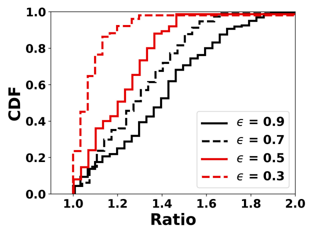
Performance. We study the performance of EdgePeering by comparing the value of the objective function $g(\tilde{\mathbf{x}})$ computed by Algorithm 1 and the actual optimal objective function value $g(\mathbf{x}^*)$ computed by directly solving Problem 1.

Figure 3 shows the ratio of $g(\tilde{\mathbf{x}})$ over $g(\mathbf{x}^*)$ (*i.e.*, the approximation ratio of Algorithm 1). Figure 3a plots the relation between this ratio and the size of topology, and Figure 3b gives the CDF of the ratio in all experiments. We observe that this ratio is bounded by $1 + \epsilon$ and independent from the size of topology. As such, we conclude that EdgePeering has a good performance across various topologies and settings.

Scalability overhead. To study the scalability of EdgePeering, we analyze its overhead in three key aspects: the number of messages needed for clustering, the number of local problems to be solved, and the scale of these local problems. As noted earlier, we add 10 larger topologies with at most 127 nodes and 129 links, 113 nodes and 183 edges for stress test.

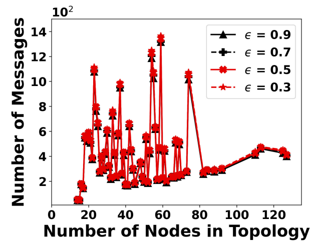


(a) The impact of topology size on $\frac{g(\tilde{\mathbf{x}})}{g(\mathbf{x}^*)}$.

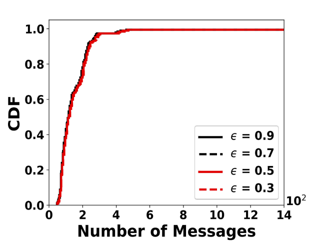


(b) The CDF of $\frac{g(\tilde{\mathbf{x}})}{g(\mathbf{x}^*)}$ in all experiments.

Figure 3: The performance of Algorithm 1 vs. the global optimal solution.



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



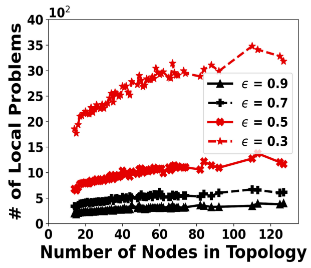
(b) The CDF of the average number of clustering messages of each node.

Figure 4: The average number of clustering messages each node sends.

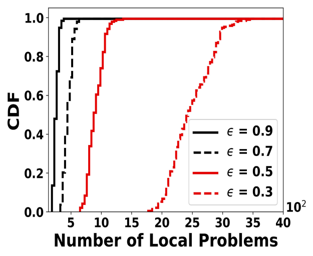
Figure 4 studies the average number of messages each node sends (including originating and forwarding) in the clustering protocol to form a partition. Figure 4a plots this number versus the size of topology, and Figure 4b plots the CDF of the average number of messages. We see that the average number of messages each node sends in general increases as the number of nodes in the topology increases, and is not affected by ϵ. The high peaks in smaller topologies are caused by the richer connectivity (*i.e.*, higher average node degrees) in them. Even in the worst case, each node sends less than 1.4k messages on average to form a cluster, showing that the clustering message overhead of EdgePeering is reasonably low.

Figure 5 studies the number of local convex programming problems in each experiment, which equals the number of clusters in each experiment. Specifically, Figure 5a plots the relation between the number of local problems and the size of topology. We see that this metric increases linearly with topology size, and decreases as ϵ increases. The largest number of local problems needed is less than 3.5k, for a topology with 109 nodes and 200 links, and $\epsilon = 0.3$, because it has more links. Figure 5b further gives the CDF of the number of problems. We see that even when ϵ is slightly large (*e.g.*, > 0.3), 80% of topologies only need to solve less than 1k local problems.

Figure 6 shows the average scale of local problems in all experiments measured as the average number of constraints of each local problem, which roughly equals the average number of variables in a problem. Figure 6a plots the relation between the number of local problems and the size of topology. We see the problem scale increases approximately linearly with topology size, with the largest scale having less than 6k constraints in the largest topology of 127 nodes and 129 links. Figure 6b shows the CDF of the average scale of local problems. It shows that the scale of local problems is less

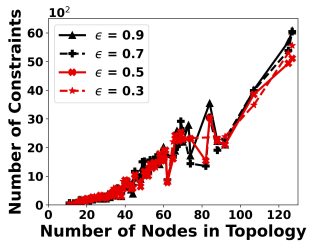


(a) The impact of topology size on the number of local problems.

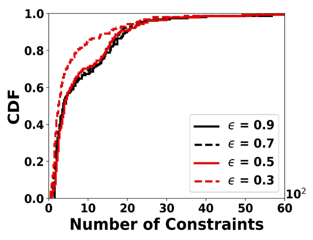


(b) The CDF of the number of local problems in all experiments.

Figure 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.

Figure 6: The scale of local problems in all experiments.

than 2k constraints in 90% of the experiments. Given the polynomial nature of linear programming and the maturity of LP solvers, we conclude that the computation overhead of EdgePeering is also reasonable.

Through these results, we see that the overhead of EdgePeering increases slowly as the network scale increases. As such, we draw the conclusion that EdgePeering has the potential to scale to very large interdomain edge networks.

5 Discussion

Incremental deployment. EdgePeering may seem to be hard to scale, troubleshoot or manage release version across different edge networks. However, EdgePeering does not require full deployment at all networks to reduce the latency of end-to-end communication. It can be deployed incrementally at networks interconnected by edge peering links, and help reduce the communication latency among devices connected to both networks.

Incremental integration with software-defined networking (SDN). In an interdomain edge network where some edge networks use SDN, EdgePeering can extend the algorithm to let the SDN controllers interact with edge nodes of neighboring networks to sample node clustering, perform local optimization and aggregating the local solutions.

Negotiate offline to solve problems arising from interdomain collaboration. Because there are multiple network providers involved in EdgePeering, such interdomain collaboration may cause safety and regulation issues (*e.g.*, security vulnerabilities, responsibility distribution issues, pricing policies, bandwidth guarantees, leases and architecture adjustments). As such, providers need to pay special attention to the offline negotiation process of EdgePeering and their intradomain traffic forwarding policies. For offline negotiation, providers may refer to the typical customer-peer-provider relationship between ASes and the negotiation process used in MIRO [30], ARROW [26] and REIN [29] as references. For intradomain traffic forwarding policies, providers can apply their internal traffic engineering algorithms to treat their internal traffic with a high priority, and decide which routes have spare capacities for forwarding interdomain traffic in EdgePeering. We leave the full investigation of the negotiation process and the intradomain policies as future work.

6 Related Work

Efforts to reduce the end-to-end latency in cellular networks share a common basic principle: avoid sending the traffic through the backbone networks. Some design MANET protocols [11, 18, 28] to route traffic through end devices but sacrifice scalability. Others introduce servers at edges to coordinate the routing among devices [22, 32]. They cannot respond quickly to highly-dynamic 5G environments. Network providers deploy their own edge networks [5, 14]. But it requires significant investment and cannot reduce the latency among devices connected to different providers. In contrast, EdgePeering lets different edge networks form an interdomain edge network for traffic forwarding, avoiding the scalability issue and reducing the cost.

Multiple systems have been proposed to pool the resources of multiple networks together to improve the overall performance [16, 21, 28, 29]. REIN [29] allows two networks (*e.g.*, AT&T and Sprint) to use the infrastructure of each other as a backup to improve the network reliability of each other. The LHC project [21] uses distributed resources from research agencies all over the world to perform exascale data analytics. To the best of our knowledge, EdgePeering is the first to let edge networks share resources to collaboratively forward end-to-end traffic along low-latency paths.

7 Conclusion

We propose EdgePeering to tackle the low-latency end-to-end communication challenge of 5G. It allows different edge networks to forward traffic collaboratively along low-latency paths. Extensive evaluation shows its feasibility and benefits.

Acknowledgment

We thank all the constructive comments from the anonymous reviewers. We thank Haohao Song for his help during preparation. This work is supported in part by NSFC Award #62172345, Alibaba Innovative Research Award, Open Research Projects of Zhejiang Lab #2022QA0AB05, Future Network Innovation Research Award of Ministry of Education of China #2021FNA02008 and Tan Kah Kee Innovation Laboratory Award #HRTP-2022-34.

References

[1] 2021. Edgepeering technical report. https://www.dropbox.com/sh/rk9964uu1sf0rst/AAATUmGksz9ZPsVzpChV7uQ8a?dl=0.

[2] 2021. The Gurobi Solver. https://www.gurobi.com/.

[3] 2021. Rocketfuel: An ISP topology mapping engine. http://www.cs.washington.edu/research/networking/rocketfuel.

[4] 3GPP. 2019. Release 15. https://www.3gpp.org/release-15.

[5] AT&T. 2021. AT&T Network Edge. https://www.business.att.com/products/att-network-edge.html.

[6] Hamidreza Bagheri, Md Noor-A-Rahim, Zilong Liu, Haeyoung Lee, Dirk Pesch, Klaus Moessner, and Pei Xiao. 2021. 5G NR-V2X: Toward Connected and Cooperative Autonomous Driving. *IEEE Communications Standards Magazine* (2021).

[7] BBC. 2019. China rolls out 'one of the world's largest' 5G networks. https://www.bbc.com/news/business-50258287.

[8] Elias Chavarria-Reyes, Ian F Akyildiz, and Etimad Fadel. 2016. Energy-efficient multi-stream carrier aggregation for heterogeneous networks in 5G wireless systems. *IEEE Transactions on Wireless Communications* 15, 11 (2016), 7432–7443.

[9] Min Chen and Yixue Hao. 2018. Task offloading for mobile edge computing in software defined ultra-dense network. *IEEE Journal on Selected Areas in Communications* 36, 3 (2018), 587–597.

[10] CNN. 2019. Looking for 5G? A list of US cities that have it. https://edition.cnn.com/2019/04/09/tech/5g-network-us-cities/index.html.

[11] Santanu Das, Charles Perkins, and Elizabeth Royer. 2003. Ad hoc on demand distance vector (AODV) routing. *IETF RFC3561, July* 10 (2003).

[12] Michael Dinitz and Yasamin Nazari. 2017. Distributed Distance-Bounded Network Design Through Distributed Convex Programming. In *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20*.

[13] Kai Gao, Qiao Xiang, Xin Wang, Yang Richard Yang, and Jun Bi. 2019. An objective-driven on-demand network abstraction for adaptive applications. *IEEE/ACM Transactions on Networking* 27, 2 (2019), 805–818.

[14] Google. 2021. Google Edge Network. https://peering.google.com/.

[15] Anupam Gupta, Mohammad T Hajiaghayi, and Harald Räcke. 2006. Oblivious network design. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. 970–979.

[16] Chuang Hu, Wei Bao, and Dan Wang. 2018. IoT Communication Sharing: Scenarios, Algorithms and Implementation. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 1556–1564. https://doi.org/10.1109/INFOCOM.2018.8486329

[17] Sabine Randriamasy Y. Richard Yang Jingxuan Zhang Kai Gao, Young Lee. 2022. An ALTO Extension: Path Vector. Experimental RFC. https://datatracker.ietf.org/doc/draft-ietf-alto-path-vector/

[18] Saadallah Kassir, Gustavo de Veciana, Nannan Wang, Xi Wang, and Paparao Palacharla. 2019. Enhancing cellular performance via vehicular-based opportunistic relaying and load balancing. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 91–99.

[19] Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. 2011. The internet topology zoo. *IEEE Journal on Selected Areas in Communications* 29, 9 (2011), 1765–1775.

[20] Robert Krauthgamer, James R Lee, Manor Mendel, and Assaf Naor. 2004. Measured descent: A new embedding method for finite metrics. In *45th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 434–443.

[21] lhc 2021. The Large Hadron Collider (LHC) Experiment. https://home.cern/topics/large-hadron-collider.

[22] Guiyang Luo, Haibo Zhou, Nan Cheng, Quan Yuan, Jinglin Li, Fangchun Yang, and Xuemin Shen. 2021. Software-Defined Cooperative Data Sharing in Edge Computing Assisted 5G-VANET. *IEEE Transactions on Mobile Computing* 20, 3 (2021), 1212–1229. https://doi.org/10.1109/TMC.2019.2953163

[23] Silvio Mandelli, Matthew Andrews, Sem Borst, and Siegfried Klein. 2019. Satisfying network slicing constraints via 5G MAC scheduling. In *IEEE INFOCOM 2019*. IEEE, 2332–2340.

[24] Arvind Narayanan, Eman Ramadan, Jason Carpenter, Qingxu Liu, Yu Liu, Feng Qian, and Zhi-Li Zhang. 2020. A first look at commercial 5G performance on smartphones. In *Proceedings of The Web Conference 2020*. 894–905.

[25] Charles E Perkins and Pravin Bhagwat. 1994. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM SIGCOMM computer communication review* 24, 4 (1994), 234–244.

[26] Simon Peter, Umar Javed, Qiao Zhang, Doug Woos, Thomas Anderson, and Arvind Krishnamurthy. 2014. One tunnel is (often) enough. In *ACM SIGCOMM Computer Communication Review*, Vol. 44. ACM, 99–110.

[27] Yakov Rekhter, Susan Hares, and Dr. Tony Li. 2006. A Border Gateway Protocol 4 (BGP-4). RFC 4271. https://doi.org/10.17487/RFC4271

[28] Daxin Tian, Jianshan Zhou, Min Chen, Zhengguo Sheng, Qiang Ni, and Victor C.M. Leung. 2018. Cooperative Content Transmission for Vehicular Ad Hoc Networks using Robust Optimization. In *IEEE INFOCOM 2018*. 90–98. https://doi.org/10.1109/INFOCOM.2018.8485868

[29] Hao Wang, Yang Richard Yang, Paul H Liu, Jia Wang, Alexandre Gerber, and Albert Greenberg. 2007. Reliability as an interdomain service. *ACM SIGCOMM Computer Communication Review* 37, 4 (2007), 229–240.

[30] Wen Xu and Jennifer Rexford. 2006. Multi-path Interdomain Routing. In *In SIGCOMM*. Citeseer.

[31] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen. 2011. SCION: Scalability, control, and isolation on next-generation networks. In *2011 IEEE Symposium on S & P*. IEEE, 212–227.

[32] Xi Zhang and Qixuan Zhu. 2019. D2D Offloading for Statistical QoS Provisionings Over 5G Multimedia Mobile Wireless Networks. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 82–90. https://doi.org/10.1109/INFOCOM.2019.8737626

[33] Yunfei Zhang, Gang Li, Chunshan Xiong, Yixue Lei, Wei Huang, Yunbo Han, Anwar Walid, Y Richard Yang, and Zhi-Li Zhang. 2020. MoWIE: Toward Systematic, Adaptive Network Information Exposure as an Enabling Technique for Cloud-Based Applications over 5G and Beyond. In *Proceedings of the Workshop on Network Application Integration/CoDesign*. 20–27.